(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) **1-0020328**

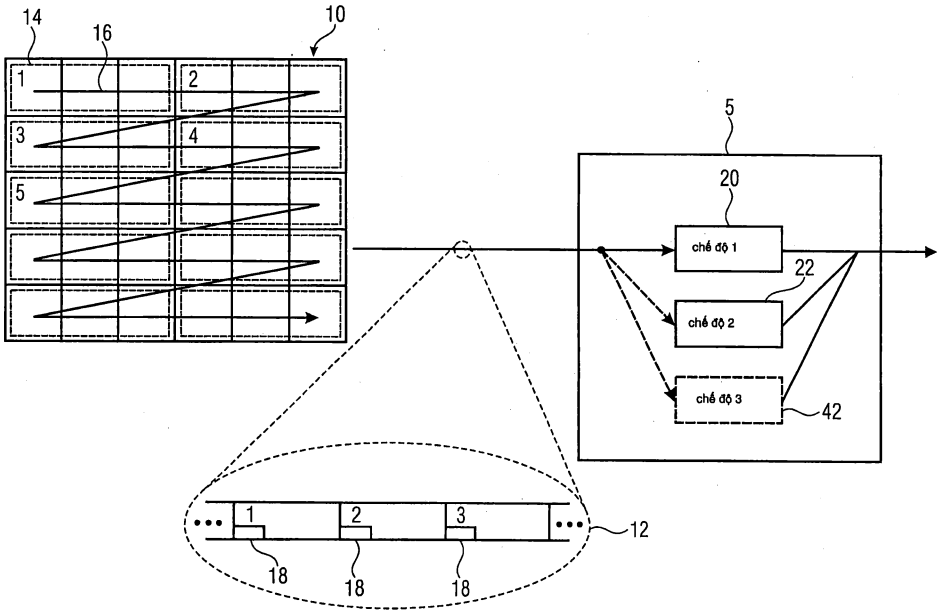
(51)[7] **H04N 7/26**

(13) **B**

(21) 1-2014-03777
(22) 15.04.2013
(86) PCT/EP2013/057798 15.04.2013
(87) WO2013/153226 17.10.2013
(30) 61/624,098 13.04.2012 US
61/666,185 29.06.2012 US
(45) 25.01.2019 370
(43) 27.04.2015 325
(73) GE Video Compression, LLC (US)
8 Southwoods Boulevard, Albany, New York 12211, USA.
(72) SCHIERL, Thomas (DE), GEORGE, Valeri (DE), HENKEL, Anastasia (RU), MARPE, Detlev (DE), GRUENEBERG, Karsten (DE), SKUPIN, Robert (DE)
(74) Công ty Luật TNHH AMBYS Hà Nội (AMBYS HANOI)

(54) BỘ GIẢI MÃ, BỘ MÃ HOÁ VÀ PHƯƠNG PHÁP KHÔI PHỤC VÀ MÃ HOÁ ẢNH

(57) Sáng chế đề cập đến bộ giải mã để khôi phục ảnh, bộ mã hóa để mã hóa ảnh, và phương pháp để giải mã và mã hóa ảnh, trong đó các khái niệm xử lý song song như xử lý song song mặt đầu sóng (WPP), với độ trễ từ đầu đến cuối được giảm đi bằng cách từ bỏ khái niệm lát thông thường mà theo đó các lát được mã hóa hoặc được giải mã hoàn toàn độc lập từ các vùng của bức ảnh nằm phía ngoài lát tương ứng, hoặc ít nhất độc lập với các vùng phía ngoài lát tương ứng do đó mã hóa entropy được quan tâm, cụ thể hướng về các lát của các chế độ khác nhau, cụ thể các lát được gọi là các lát phụ thuộc cho phép các tính liên phụ thuộc vượt qua các biên lát, và các lát khác không phải là lát phụ thuộc, được gọi là các lát thông thường. Kết hợp với một khía cạnh hoặc không, khái niệm xử lý WPP được tạo ra một cách hiệu quả hơn sử dụng các phần cú pháp khởi đầu của lát để định vị các điểm vào WPP.



Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến mã hóa hình ảnh có độ trễ thấp.

Tình trạng kỹ thuật của sáng chế

Các lát entropy (trước đây là các lát trọng số nhẹ) các ô và WPP (WPP là viết tắt của Wavefront Parallel Processing – xử lý song song mặt đầu sóng) được chứa trong các lát kiểu HEVC hiện nay như là các công cụ cho phép song song hóa.

Đối với phép song song hóa của các bộ mã hóa và các bộ giải mã video, sự phân chia mức ảnh có một số thuận lợi so với các phương pháp khác. Trong các bộ mã hóa-giải mã video trước đây, như H.264/AVC [1], các phần hình ảnh chỉ có thể sử dụng với các lát đều có chi phí cao về hiệu suất mã hóa. Để giải mã H.264/AVC song song mở rộng được, cần thiết kết hợp trạng thái song song mức khối macro để khôi phục ảnh và trạng thái song song mức khung để giải mã entropy. Tuy nhiên, phương pháp này cung cấp sự rút gọn bị giới hạn về các thời gian chờ ảnh và sử dụng bộ nhớ cao. Để khắc phục các hạn chế này, các kịch bản phân chia ảnh mới được chứa trong bộ mã hóa-giải mã HEVC. Phiên bản phần mềm tham khảo hiện tại (HM-6) chứa 4 phương pháp khác nhau: các lát đều hoặc chuẩn, các lát entropy, các dòng con xử lý song song mặt đầu sóng (WPP) và các ô. Thường các phần ảnh đó gồm có một tập hợp các đơn vị mã hóa lớn nhất (LCU) (LCU là viết tắt của Largest Coding Units), hoặc diễn đạt đồng nghĩa là các đơn vị cây mã hóa (CTU) (CTU là viết tắt của Coding Tree Unit), như được định ra trong HEVC hoặc thậm chí tập con của chúng.

Fig.1 thể hiện ảnh 898 được đặt làm ví dụ trong một lát đều 900 cho từng hàng 902 của các LCU hoặc các khối macro trong ảnh. Các lát đều hay chuẩn (như được xác định trong H.264 [1]) có bất lợi mã hóa lớn nhất vì chúng phá vỡ sự giải mã entropy và các thuộc tính phụ thuộc dự đoán.

Các lát entropy, kiểu lát, phá vỡ các thuộc tính phụ thuộc giải mã entropy nhưng cho phép dự đoán (và lọc) để vượt qua các đường biên của lát.

Trong WPP, các phần ảnh được chen hàng, và cả sự giải mã và dự đoán entropy được cho phép để sử dụng dữ liệu từ các khối trong các phần khác. Theo cách này, các tổn thất mã hóa được giảm thiểu trong khi đồng thời trạng thái song song mặt đầu sóng có thể được khai thác. Tuy nhiên, việc chen làm phá vỡ quan hệ nhân quả dòng bit vì phần trước đó cần phần kế tiếp để giải mã.

Fig.2 thể hiện ví dụ ảnh 898 được chia thành hai hàng 904a, 904b có các ô phân chia nằm ngang 906. Các ô định ra các biên ngang 908 và biên dọc 910 phân chia bức ảnh 88 thành các cột ô 912a, b, c và các hàng 904 a, b. Tương tự với các lát đều 900, các ô 906 phá vỡ sự giải mã entropy và các thuộc tính phụ thuộc dự đoán, nhưng không yêu cầu đoạn đầu cho mỗi ô.

Đối với mỗi kỹ thuật này, số lượng phần có thể được chọn lựa tự do bởi bộ mã hóa. Nói chung có nhiều phần dẫn đến các sự tổn thất nén cao hơn. Tuy nhiên, trong WPP, việc lan truyền tổn thất là không quá cao và do đó số lượng phần bức ảnh thậm chí có thể được cố định trên mỗi hàng. Điều này cũng dẫn tới một vài thuận lợi. Thứ nhất, đối với dòng bit WPP, quan hệ nhân quả được bảo đảm. Thứ hai, các việc thực hiện của bộ giải mã có thể giả định rằng một lượng nhất định trạng thái song song có thể có sẵn, chúng cũng tăng lên cùng với sự phân giải. Và, cuối cùng, không một sự lựa chọn ngữ cảnh và các tính phụ thuộc dự đoán phải bị phá vỡ khi giải mã trong thứ tự mặt đầu sóng, dẫn đến tổn thất mã hóa thấp tương đối.

Tuy nhiên, cho đến bây giờ tất cả sự mã hóa song song trong các khái niệm biến đổi không cung cấp được hiệu suất nén cao kết hợp với việc giữ độ trễ thấp. Điều này cũng là đúng cho khái niệm WPP. Các lát là các đơn vị nhỏ nhất của sự vận chuyển, trong dòng mã hóa, và một số các dòng con WPP vẫn phải được vận chuyển từng kỳ.

Bản chất kỹ thuật của sáng chế

Theo đó, mục đích của sáng chế là đề xuất khái niệm mã hóa ảnh cho phép giải mã song song theo, ví dụ, xử lý song song mặt đầu sóng, với hiệu suất được tăng lên, như với việc làm giảm thêm độ trễ từ đầu đến cuối hoặc nâng cao hiệu suất mã hóa bằng cách làm giảm chi phí mã hóa đã tiêu dùng.

Mục đích này đạt được bởi đối tượng của các điểm yêu cầu bảo hộ độc lập, cụ thể: bộ giải mã để khôi phục ảnh, bộ mã hóa để mã hóa ảnh, và phương pháp để giải mã và mã hóa ảnh, trong đó:

Bộ giải mã để khôi phục ảnh từ dòng dữ liệu mà bên trong dòng dữ liệu này ảnh được mã hóa trong các đơn vị lát mà ảnh được phân chia thành các đơn vị lát này, trong đó bộ giải mã được tạo cấu hình để giải mã các lát từ dòng dữ liệu theo thứ tự lát và bộ giải mã đáp lại phần phần tử cú pháp nằm trong lát hiện tại của các lát, để giải mã lát hiện tại theo một trong ít nhất hai chế độ.

Bộ mã hóa để mã hóa ảnh thành dòng dữ liệu trong các đơn vị lát mà ảnh được phân chia thành các đơn vị lát này, trong đó bộ mã hóa được tạo cấu hình để mã hóa các lát trong dòng dữ liệu theo thứ tự lát.

Phương pháp để khôi phục ảnh từ dòng dữ liệu mà bên trong dòng dữ liệu này ảnh được mã hóa trong các đơn vị lát mà ảnh được phân chia thành các đơn vị lát này, trong đó phương pháp gồm có giải mã các lát từ dòng dữ liệu theo thứ tự lát và phương pháp đáp ứng lại phần phần tử cú pháp nằm trong lát hiện tại của các lát, để giải mã lát hiện tại theo một trong ít nhất hai chế độ.

Phương pháp để mã hóa ảnh thành dòng dữ liệu trong các đơn vị lát mà ảnh được phân chia thành các đơn vị lát này, trong đó phương pháp gồm có mã hóa các lát thành dòng dữ liệu theo thứ tự lát.

Phát hiện cơ bản của sáng chế đó là các khái niệm xử lý song song như xử lý song song mặt đầu sóng, có thể được nhận thấy với độ trễ từ đầu đến cuối được giảm đi nếu khái niệm lát thông thường mà theo đó các lát được mã hóa hoặc được giải mã độc lập hoàn toàn với các vùng của bức ảnh nằm phía ngoài lát tương ứng, hoặc ít nhất độc lập với các vùng nằm phía ngoài các lát tương ứng cho đến mức mã hóa entropy được quan tâm được từ bỏ các lát của các chế độ khác nhau, cụ thể, các lát được gọi là các lát phụ thuộc cho phép các tính liên phụ thuộc vượt qua các đường biên của lát, và các lát khác không cho phép, được gọi là các lát chuẩn.

Sự phát hiện cơ bản thêm nữa của sáng chế rằng khái niệm xử lý WPP có thể được tạo ra hiệu quả hơn nếu các phần cú pháp khởi đầu của lát được sử dụng để định vị các điểm vào WPP.

Mô tả vắn tắt các hình vẽ

Các phương án ưu tiên của sáng chế được mô tả bên dưới với các hình vẽ, trong đó các phương án thuận lợi là đối tượng của các điểm yêu cầu bảo hộ phụ thuộc.

Fig.1 là hình thể hiện ảnh được chia theo cách ví dụ thành lát đều cho mỗi hàng LCU hoặc khối macro trong ảnh;

Fig.2 là hình thể hiện ảnh được chia theo cách ví dụ thành hai hàng có các ô được phân chia theo chiều ngang;

Fig.3 là hình thể hiện ví dụ phép gán các phần chia đã được mã hóa song song cho lát hoặc đoạn chuyển mạng;

Fig.4 là hình thể hiện giản đồ minh họa sự phân mảnh chung của khung với phương pháp mã hóa ô cho độ trễ từ đầu đến cuối cực tiểu.

Fig.5 là hình thể hiện giản đồ minh họa sự phân mảnh ví dụ của khung với phương pháp mã hóa WPP cho độ trễ từ đầu đến cuối cực tiểu;

Fig.6 là hình thể hiện giản đồ minh họa kịch bản chuyển hội thoại sử dụng các dịch vụ video;

Fig.7 là hình minh họa dạng sơ đồ việc lập thời gian biểu mã hóa có thể có, sự truyền và giải mã cho các ô với các tập con chung với độ trễ từ đầu đến cuối cực tiểu;

Fig.8 là hình minh họa dạng sơ đồ thể hiện thời gian biểu thường đạt được độ trễ từ đầu đến cuối;

Fig.9 minh họa ảnh mẫu có 11x9 khối cây mã hóa mà được chia thành hai lát;

Fig.10 minh họa ảnh mẫu có 13x8 khối cây mã hóa mà được chia thành ba lát;

Fig.11 thể hiện ví dụ cú pháp tập hợp tham số chuỗi;

Fig.12 thể hiện ví dụ cú pháp tập hợp tham số ảnh;

Fig.13 thể hiện ví dụ cú pháp đoạn đầu lát;

Fig.14 minh họa sự phân chia ảnh cho việc xử lý WPP thành lát đều và, cho việc xử lý độ trễ thấp, thành các lát phụ thuộc;

Fig.15 thể hiện ví dụ một phần nằm trong cú pháp tập hợp tham số ảnh;

Fig.16 thể hiện cú pháp đoạn đầu lát có thể có;

Fig.17 minh họa giản lược các tính liên phụ thuộc mã hóa đối với các lát chuẩn (và các lát phụ thuộc);

Fig.18 thể hiện giản đồ so sánh việc mã hóa vận chuyển độ trễ thấp các ô (xử lý song song mặt đầu sóng sử dụng các lát phụ thuộc);

Fig.19 minh họa thời gian biểu minh họa sự mã hóa WPP mẫu với sự truyền có độ trễ thấp đường liên hợp khi sử dụng sự xử lý song song mặt đầu sóng sử dụng các lát phụ thuộc như được thể hiện tại phía bên phải của Fig.18;

Fig.20 thể hiện giản đồ minh họa sự cải thiện mạnh bằng cách sử dụng các lát đều làm thành các neo;

Fig.21 thể hiện phương án khác đối với cú pháp đoạn đầu lát;

Fig.22 thể hiện phương án khác đối với cú pháp tập hợp tham số ảnh;

Fig.23 thể hiện giản đồ minh họa quy trình khởi tạo xác suất biểu tượng cho lát phụ thuộc trong trường hợp bắt đầu tại biên ảnh bên trái;

Fig.24 thể hiện biểu đồ của bộ giải mã;

Fig.25 thể hiện giản lược sơ đồ khối của bộ giải mã cùng với minh họa giản lược việc phân chia ảnh thành các khối và các lát mã hóa;

Fig.26 thể hiện giản lược sơ đồ khối của bộ mã hóa;

Fig.27 thể hiện giản lược ảnh được phân chia thành các lát chuẩn và các lát phụ thuộc, còn được gọi là các đoạn lát.

Fig.28a và 28b thể hiện giản lược ảnh được phân chia thành các lát chuẩn và các lát phụ thuộc, một mặt được còn được gọi là các đoạn lát, và mặt khác là các ô;

Fig.29 thể hiện lưu đồ minh họa quy trình khởi tạo ngữ cảnh sử dụng các lát phụ thuộc.

Fig.30 thể hiện lưu đồ minh họa quy trình lưu trữ ngữ cảnh để sử dụng các lát phụ thuộc; và

Fig.31 thể hiện giản lược các khả năng khác nhau để báo hiệu các điểm vào WPP.

Mô tả chi tiết sáng chế

Dưới đây, bản mô tả bắt đầu với việc mô tả các khái niệm ngày nay để lần lượt cho phép xử lý ảnh song song và mã hóa độ trễ thấp. Các vấn đề xảy ra khi mong muốn có cả hai khả năng được phác thảo. Cụ thể, đi ra khỏi thảo luận dưới đây, khái niệm dòng con WPP như đã được đưa ra cho tới bây giờ bằng cách nào đó vẫn trái ngược với mong muốn có được độ trễ thấp do sự cần thiết truyền các dòng con WPP bằng cách nhóm chúng vào một lát. Các phương án sau đây nêu ra các khái niệm xử lý song song như khái niệm WPP, có thể sử dụng cho các ứng dụng cần phải có dù độ trễ ít hơn bằng cách mở rộng khái niệm lát, cụ thể bằng cách đưa ra loại lát khác, sau đây còn được gọi là các lát phụ thuộc.

Giảm thiểu độ trễ video từ đầu đến cuối từ việc bắt lấy hình ảnh để hiển thị là một trong các mục đích chính trong các ứng dụng như hội thảo video và tương tự.

Chuỗi xử lý tín hiệu để truyền video số gồm có máy quay phim, thiết bị chụp, bộ mã hóa, đóng gói dữ liệu, truyền, bộ tách kênh, bộ giải mã, bộ kết xuất và màn hình hiển thị. Mỗi giai đoạn này góp phần vào độ trễ từ đầu đến cuối bằng cách đệm dữ liệu hình ảnh trước khi truyền theo từng dãy đến giai đoạn kế tiếp.

Một số ứng dụng yêu cầu giảm thiểu độ trễ này, ví dụ, xử lý từ xa các đối tượng trong những vùng nguy hiểm, mà không cần nhìn trực tiếp vào đối tượng được xử lý, hoặc phẫu thuật xâm lấn nhỏ nhất. Thậm chí độ trễ ngắn có thể dẫn đến những khó khăn nghiêm trọng trong việc xử lý thích hợp hoặc thậm chí dẫn đến các lỗi thê thảm.

Trong nhiều trường hợp, toàn bộ khung video được đệm trong giai đoạn xử lý, ví dụ, cho phép xử lý dưới khung. Một số giai đoạn tập hợp lại dữ liệu để tạo ra các gói mà các gói này được chuyển đến giai đoạn kế tiếp. Nói chung, có biên thấp hơn cho độ trễ do các yêu cầu xử lý cục bộ. Điều này được phân tích đối với từng giai đoạn trong sự mô tả chi tiết hơn dưới đây.

Xử lý bên trong máy ảnh không nhất thiết yêu cầu xử lý tín hiệu dưới khung, do đó đem lại độ trễ nhỏ nhất bởi thời gian tích hợp của bộ cảm biến, nó được bao bởi tốc độ khung, và kiểu nào đó chọn lựa bởi nhà sản xuất phần cứng. Đầu ra của máy ảnh thường liên quan đến lệnh quét mà lệnh quét này thường bắt đầu xử lý trong góc trái trên cùng, chuyển qua đến góc phải trên cùng và tiếp tục theo hàng đến góc phải dưới cùng. Tiếp đó, mất khoảng là một khung cho đến khi tất cả dữ liệu được truyền từ bộ cảm biến đến đầu ra máy ảnh.

Thiết bị chụp có thể chuyển dữ liệu máy ảnh ngay sau khi nhận; tuy nhiên nó sẽ thường đệm dữ liệu nào đó và tạo ra các sự truyền từng khối để tối ưu truy cập dữ liệu cho bộ nhớ hoặc bộ lưu trữ. Hơn nữa, sự kết nối giữa máy ảnh/thiết bị chụp và bộ nhớ của máy tính thường làm giới hạn tốc độ bit để chuyển dữ liệu hình ảnh đã chụp đến bộ nhớ để xử lý thêm (mã hóa). Thông thường, các máy ảnh được kết nối qua USB 2.0 hoặc với USB 3.0, sẽ thường bao gồm chuyển một phần dữ liệu hình ảnh đến bộ mã hóa. Điều này làm giới hạn khả năng song song trên phía bộ mã hóa trong các kịch bản độ trễ cực thấp, tức là, bộ mã hóa sẽ thử bắt đầu mã hóa càng sớm càng tốt, khi dữ liệu có sẵn từ máy ảnh, ví dụ trong thứ tự quét mành từ đầu đến cuối của bức ảnh.

Trong bộ mã hóa, có một số mức độ tự do cho phép cân bằng hiệu suất mã hóa, về mặt tốc độ dữ liệu được cần cho độ chính xác video nhất định, để làm giảm độ trễ xử lý.

Bộ mã hóa sử dụng dữ liệu được gửi để dự đoán hình ảnh được mã hóa sau đó. Nói chung, sự khác biệt giữa ảnh thực và dự đoán có thể được mã hóa với ít bit hơn so với được cần thiết mà không cần dự đoán. Các giá trị dự đoán này cần có sẵn ở bộ giải mã, do đó sự dự đoán được dựa trên các phần đã được giải mã trước của cùng hình ảnh (dự đoán dưới khung) hoặc trên các ảnh khác (dự đoán liên khung) mà nó được xử lý sớm hơn. Các tiêu chuẩn mã hóa video Pre-HEVC sử dụng chỉ một phần hình ảnh phía trên hoặc trong cùng một hàng, nhưng phía trái – được mã hóa trước đó – để dự đoán dưới khung, dự đoán vectơ chuyển động và mã hóa entropy (CABAC).

Ngoài việc tối ưu hóa cấu trúc dự đoán, ảnh hưởng của xử lý song song có thể được xem xét. Xử lý song song yêu cầu nhận diện các vùng ảnh mà các vùng ảnh này có thể được xử lý độc lập. Đối với các lý do thực tiễn, các vùng giáp nhau như các

hình chữ nhật nằm ngang hoặc thẳng đứng được chọn mà các hình chữ nhật nằm ngang hoặc thẳng đứng này thường được gọi là các "ô". Trong trường hợp của các ràng buộc độ trễ thấp, các vùng đó nên cho phép mã hóa được song song hóa dữ liệu đến từ thiết bị chụp đến bộ nhớ càng sớm càng tốt. Giả định rằng bộ nhớ quét mành truyền, các phần thẳng đứng của dữ liệu thô có nghĩa là, để bắt đầu mã hóa ngay lập tức. Bên trong các ô này, bộ nhớ quét mành phân chia ảnh thành các phần thẳng đứng (xem hình bên dưới), dưới dự đoán, dự đoán vectơ chuyển động và mã hóa entropy (CABAC) có thể dẫn tới hiệu suất mã hóa hợp lý. Để làm giảm thiểu độ trễ, chỉ một phần của ảnh, bắt đầu từ trên đỉnh, được truyền đến bộ nhớ khung của bộ mã hóa, và xử lý song song nên được bắt đầu trong các ô thẳng đứng.

Cách khác cho phép xử lý song song là sử dụng WPP trong phạm vi một lát chuẩn, lát này được so sánh với các ô, một "hàng" ô được chứa trong một lát. Dữ liệu nằm trong lát đó, có thể cũng được mã hóa song song sử dụng các dòng con WPP, trong phạm vi lát. Sự chia cắt ảnh thành các lát 900 và các ô/các dòng con WPP 914 được thể hiện trong Fig.3/1 tạo ra các ví dụ.

Fig.3 do đó thể hiện sự gán các phần chia đã được mã hóa song song như 906 hoặc 914 vào lát hoặc đoạn chuyển mạng (một gói mạng đơn lẻ hoặc nhiều gói mạng 900).

Sự đóng gói dữ liệu đã được mã hóa thành các đơn vị lớp trừu tượng hóa mạng (Network Abstraction Layer - NAL), như được định rõ trong H.264 hoặc HEVC, trước khi truyền hoặc trong quy trình mã hóa bổ sung đoạn đầu nào đó vào các khối dữ liệu mà cho phép nhận diện mỗi khối và sắp xếp lại các khối, nếu có thể áp dụng được. Trong trường hợp tiêu chuẩn, không có tín hiệu bổ sung nào được yêu cầu, vì thứ tự của các chi tiết mã hóa thường nằm theo thứ tự giải mã, nghĩa là sự gán ẩn vị trí của ô hoặc mảnh mã hóa thông thường được đưa ra.

Nếu việc xử lý song song được xem xét với lớp vận chuyển bổ sung cho sự vận chuyển song song độ trễ thấp, cụ thể, lớp vận chuyển có thể sắp xếp lại các phần chia ảnh cho các ô cho phép truyền độ trễ thấp, có nghĩa gửi các đoạn như được thể hiện trong Fig.4 khi chúng được mã hóa. Các đoạn này có thể cũng là các lát được mã hóa

không đầy đủ, chúng có thể là một tập con của lát, hoặc có thể được chứa trong một lát phụ thuộc.

Trong trường hợp tạo ra các mảnh bổ sung, có một sự thỏa hiệp giữa hiệu suất và độ trễ, hiệu suất sẽ là cao nhất với các khối dữ liệu lớn bởi vì thông tin đoạn đầu bổ sung một lượng byte không đổi, bởi vì các khối dữ liệu lớn của các bộ mã hóa song song sẽ cần được đệm trước khi truyền dẫn. Tổng độ trễ có thể được giảm đi, nếu phép biểu diễn đã được mã hóa của các ô thẳng đứng 906 được phân chia trong một số lượng mảnh 916 mà các mảnh này được truyền đi ngay khi mảnh được mã hóa hoàn toàn. Kích thước của mỗi mảnh có thể được xác định dưới dạng vùng ảnh cố định, như các khối macro, các LCU hoặc dưới dạng dữ liệu lớn nhất như được thể hiện trong Fig.4.

Fig.4, do đó, thể hiện sự phân mảnh thông thường của khung bằng phương pháp mã hóa ô cho độ trễ từ đầu đến cuối nhỏ nhất.

Tương tự, Fig.5 thể hiện sự phân mảnh của khung với phương pháp mã hóa WPP cho độ trễ từ đầu đến cuối nhỏ nhất.

Việc truyền có thể làm tăng thêm độ trễ, ví dụ, nếu khối bổ sung được định hướng áp dụng bước xử lý, như Forward Error Correction mã hóa làm tăng độ mạnh của việc truyền. Bên cạnh đó, các bộ phận mạng (bộ định tuyến, v.v) hoặc đường nối vật lý có thể bổ sung độ trễ, điều này thường được biết đến là độ trễ cho kết nối. Ngoài độ trễ, tốc độ bit truyền xác định, (độ trễ) thời gian để truyền dữ liệu từ Bên a đến Bên b, trong cuộc hội thoại như được thể hiện trên Fig.6 sử dụng các dịch vụ video.

Nếu các khối dữ liệu đã mã hóa được truyền không theo thứ tự, độ trễ sắp xếp lại theo thứ tự phải được xem xét.

Việc giải mã có thể bắt đầu ngay khi đơn vị dữ liệu đến, giả thiết rằng các đơn vị dữ liệu khác phải được giải mã trước khi chúng có mặt.

Trong trường hợp của các ô, không có sự phụ thuộc nào giữa các ô, do đó ô có thể được giải mã ngay lập tức. Nếu các đoạn được tạo ra từ các ô, như các lát riêng biệt trên mỗi đoạn như được thể hiện trong Fig.4, các đoạn có thể được vận chuyển

trực tiếp, ngay khi chúng được mã hóa lần lượt, các LCU hoặc CU được chứa của chúng được mã hóa.

Bộ kết xuất thu thập các đầu ra của động cơ giải mã song song và chuyển ảnh đã được kết hợp theo hàng đến màn hình hiển thị.

Màn hình hiển thị không cần thiết thêm bất kỳ độ trễ nào, nhưng trong thực tế có thể thực hiện xử lý dưới khung nào đó trước khi dữ liệu hình ảnh được hiển thị thực tế. Điều này tùy vào các sự lựa chọn thiết kế bởi nhà sản xuất phần cứng.

Tóm tại, chúng ta có thể tác động đến các giai đoạn mã hóa, đóng gói, truyền dẫn và giải mã để đạt được độ trễ từ đầu đến cuối nhỏ nhất. Nếu chúng ta sử dụng xử lý song song, các ô và mảnh trong các ô, tổng độ trễ có thể được làm giảm một cách đáng kể như được thể hiện trong FIG.7, so với chuỗi xử lý được sử dụng thông thường mà chuỗi xử lý này bổ sung khoảng một độ trễ khung ở mỗi giai đoạn trong các giai đoạn đã nêu như được thể hiện trong Fig.8.

Cụ thể, trong khi Fig.7 thể hiện mã hóa, truyền và giải mã cho các ô với các tập con chung với độ trễ từ đầu đến cuối nhỏ nhất, Fig.8 minh họa độ trễ từ đầu đến cuối đạt được thông thường.

HEVC cho phép sử dụng sự phân chia lát, sự phân chia ô, và theo cách thức sau đây.

Ô: một lượng số nguyên các khối cây cùng xuất hiện trong một cột và một hàng, được sắp xếp liên tiếp theo sự quét mành khối cây của ô. Sự phân chia từng bức ảnh thành các ô là một sự chia nhỏ. Các ô trong bức ảnh được sắp xếp liên tiếp theo sự quét mành ô của bức ảnh. Mặc dù lát chứa các khối cây liếp tiếp theo sự quét mành khối cây của ô, các khối cây này là liên tiếp không cần thiết theo thứ tự quét mành khối cây của bức ảnh.

Lát: một lượng số nguyên các khối cây được sắp xếp liên tiếp theo sự quét mành. Việc phân chia từng bức ảnh thành các lát là sự chia nhỏ. Các địa chỉ khối cây được lấy từ địa chỉ khối cây thứ nhất trong lát (như được thể hiện trong đoạn đầu lát).

Quét mành: ánh xạ mẫu hai chiều hình chữ nhật vào mẫu một chiều sao cho các lối vào thứ nhất trong mẫu một chiều là từ hàng trên cùng thứ nhất của mẫu hai chiều

được quét từ bên trái sang bên phải, tiếp theo tương tự là các hàng thứ hai, thứ ba, v.v., của mẫu (đi xuống phía dưới) mà mỗi hàng được quét từ bên trái sang bên phải.

khối cây: khối NxN của mẫu độ sáng và hai khối tương ứng mẫu sắc độ của bức ảnh có ba mảng mẫu, hoặc khối NxN của mẫu của bức ảnh đơn sắc hoặc bức ảnh được mã hóa sử dụng ba mức độ màu riêng biệt. Sự chia lát thành các khối cây được gọi là sự chia nhỏ.

Sự chia nhỏ: sự chia một tập hợp thành các tập con sao cho mỗi phần tử của một tập hợp là trong chính xác một trong số các tập con.

Cây tứ phân: cây trong đó nút mẹ có thể được chia tách thành bốn nút con. Nút con có thể trở thành nút mẹ đối với một sự chia khác thành bốn nút con.

Sau đây, sự phân chia nhỏ không gian của các bức ảnh, các lát và các ô được giải thích. Cụ thể, phần mô tả dưới đây chỉ ra làm thế nào bức ảnh được phân chia thành các lát, các ô và các khối cây mã hóa. Các bức ảnh được chia ra thành các lát và các ô. Lát là một chuỗi các khối cây mã hóa. Tương tự, ô là một chuỗi các khối cây mã hóa.

Các mẫu này được xử lý trong các đơn vị khối cây mã hóa. Kích thước mảng độ sáng cho mỗi khối cây trong các mẫu theo cả chiều rộng và chiều cao là CtbSize. Chiều rộng và chiều cao của các mảng màu sắc cho mỗi khối cây mã hóa lần lượt là CtbWidthC và CtbHeightC.

Ví dụ, ảnh có thể được chia thành hai lát như được thể hiện trong hình vẽ kế tiếp. Một ví dụ khác, ảnh có thể được chia thành ba ô như được thể hiện trong hình vẽ thứ hai sau đây.

Không giống như các lát, các ô thường có dạng hình chữ nhật và thường chứa một lượng nguyên các khối cây mã hóa theo thứ tự quét mành khối cây mã hóa. Ô có thể gồm các khối cây mã hóa được chứa trong hơn một lát. Tương tự, lát có thể gồm có các khối cây mã hóa được chứa trong hơn một ô.

Fig.9 minh họa ảnh 898 với 11x9 khối cây mã hóa 918 được phân chia thành hai lát 900a,b.

Fig.10 minh họa ảnh với 13x8 khối cây mã hóa 918 được phân chia thành ba ô.

Mỗi mã hóa 898 khối cây 918 được gán báo hiệu phần chia để nhận diện kích thước khối cho phép dưới dự đoán hoặc liên dự đoán và để mã hóa biến đổi. Sự phân chia là sự phân chia cây tứ phân đệ quy. Rễ của cây tứ phân được kết hợp với khối cây mã hóa. Cây tứ phân được chia tách cho đến khi đạt được lá, cây tứ phân được đề cập đến là khối mã hóa. Khối mã hóa là nút rễ của hai cây, cây dự đoán và cây biến đổi.

Cây dự đoán chỉ ra vị trí và kích thước của các khối dự đoán. Các khối dự đoán và dữ liệu dự đoán kết hợp được đề cập đến là đơn vị dự đoán.

Fig.11 thể hiện cú pháp RBSP của tập hợp tham số chuỗi ví dụ.

Cây biến đổi chỉ ra vị trí và kích thước của các khối biến đổi. Các khối biến đổi và dữ liệu biến đổi được kết hợp được đề cập đến là đơn vị biến đổi.

Thông tin chia tách cho độ sáng và sắc độ (luma và chroma) là giống nhau đối với cây dự đoán và có thể hoặc không thể là giống nhau đối với cây biến đổi.

Khối mã hóa, dữ liệu mã hóa kết hợp và dự đoán kết hợp và các đơn vị biến đổi tạo ra với nhau đơn vị mã hóa.

Quy trình chuyển đổi địa chỉ khối cây mã hóa theo thứ tự quét khối cây mã hóa sang thứ tự quét ô có thể là như sau:

Các đầu ra của quy trình là

–mảng CtbAddrTS[ctbAddrRS], với ctbAddrRS nằm trong khoảng từ 0 đến PicHeightInCtbs * PicWidthInCtbs – 1, bao gồm hai giá trị đầu mút.

–mảng TileId[ctbAddrTS], với ctbAddrTS nằm trong khoảng từ 0 đến PicHeightInCtbs * PicWidthInCtbs – 1, bao gồm hai giá trị đầu mút.

Mảng CtbAddrTS[ ] được suy ra như sau:

```
for( ctbAddrRS = 0; ctbAddrRS < PicHeightInCtbs * PicWidthInCtbs; ctbAddrRS++ ) {
    tbX = ctbAddrRS % PicWidthInCtbs
    tbY = ctbAddrRS / PicWidthInCtbs
    for( j = 0; j <= num_tile_columns_minus1; j++ )
        if( tbX < ColBd[ j + 1 ] )
            tileX = j
    for( i = 0; i <= num_tile_rows_minus1; i++ )
        if( tbY < RowBd[ i + 1 ] )
            tileY = i
    CtbAddrTS[ ctbAddrRS ] = ctbAddrRS − tbX
            for( i = 0; i < tileX; i++ )
        ctbAddrTS += RowHeight[ tileY ] * ColumnWidth[ i ]
    CtbAddrTS[ ctbAddrRS ] += ( tbY − RowBd[ tileY ] ) * ColumnWidth[ tileY ] + tbX − ColBd[ tileX ]
}
```

Mảng TileId[ ] được suy ra như sau:

```
for( j = 0, tileId = 0; j <= num_tile_columns_minus1; j++ )
    for( i = 0; i <= num_tile_rows_minus1; i++, tileId++ )
        for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++ )
            for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++ )
                TileId[ CtbAddrTS[ y*PicWidthInCtbs + x ] ] = tileId
```

Cú pháp mẫu, tương ứng được thể hiện trên các Fig.11, 12, và 13, trong đó Fig.12 có cú pháp RBSP của tập hợp tham số ảnh mẫu. Fig.13 thể hiện cú pháp đoạn đầu lát mẫu.

Trong ví dụ cú pháp, các ngữ nghĩa sau đây có thể áp dụng:

entropy_slice_flag bằng 1 chỉ ra rằng giá trị của các phần tử cú pháp đoạn đầu lát không có mặt được suy ra là bằng với giá trị của các phần tử cú pháp đoạn đầu lát trong lát trước đó, trong đó lát trước đó được định ra là lát chứa khối cây mã hóa với sự định vị (SliceCtbAddrRS − 1). entropy_slice_flag sẽ bằng 0 khi SliceCtbAddrRS bằng 0.

tiles_or_entropy_coding_sync_idc bằng 0 chỉ ra rằng có chỉ một ô trong mỗi bức ảnh trong chuỗi video đã được mã hóa, và không có quá trình đồng bộ hóa riêng nào cho các biến ngữ cảnh được cần trước khi giải mã khối cây mã hóa thứ nhất của một hàng các khối cây mã hóa.

tiles_or_entropy_coding_sync_idc bằng 1 chỉ ra rằng có thể là nhiều hơn một ô trong mỗi bức ảnh trong chuỗi video đã được mã hóa, và không có quá trình đồng bộ

hóa riêng nào cho các biến ngữ cảnh được suy ra trước khi giải mã khối cây mã hóa thứ nhất của hàng các khối cây mã hóa.

tiles_or_entropy_coding_sync_idc bằng 2 chỉ ra rằng chỉ có một ô trong từng bức ảnh trong chuỗi video đã được mã hóa, quá trình đồng bộ hóa riêng cho các biến ngữ cảnh được cần trước khi giải mã khối cây mã hóa thứ nhất của hàng gồm các khối cây mã hóa, và quá trình nhớ riêng cho các biến ngữ cảnh được dẫn ra sau khi giải mã hai khối cây mã hóa của hàng gồm các khối cây mã hóa.

Giá trị của tiles_or_entropy_coding_sync_idc sẽ nằm trong khoảng từ 0 đến 2, bao gồm hai giá trị đầu mút.

num_tile_columns_minus1 plus 1 chỉ ra số lượng cột ô phân chia bức ảnh.

num_tile_rows_minus1 plus 1 chỉ ra số lượng hàng ô phân chia bức ảnh.

Khi num_tile_columns_minus1 = 0, num_tile_rows_minus1 sẽ không bằng 0.

Một hoặc cả hai điều kiện sau đây sẽ là thỏa mãn cho mỗi lát và ô:

–Tất cả các khối đã được mã hóa trong lát thuộc về cùng một ô.

–Tất cả các khối đã được mã hóa trong ô thuộc về cùng một lát.

Lưu ý rằng – Trong cùng bức ảnh, có thể có cả hai lát chứa nhiều ô và các ô chứa nhiều lát.

uniform_spacing_flag bằng 1 chỉ ra rằng các biên cột và tương tự là các biên hàng được phân bố đồng đều trên bức ảnh. uniform_spacing_flag bằng 0 chỉ ra rằng các biên cột và tương tự là các biên hàng không được phân bố đồng đều trên bức ảnh nhưng được báo hiệu rõ ràng sử dụng các phần tử cú pháp column_width[i] và row_height[i].

column_width[i] chỉ ra độ rộng của cột ô thứ i trong các đơn vị các khối cây mã hóa.

row_height[i] chỉ ra chiều cao của hàng ô thứ i trong các đơn vị các khối cây mã hóa.

Các giá trị của ColumnWidth[i], chỉ ra rằng độ rộng của cột ô thứ i trong các đơn vị của các khối cây mã hóa, và các giá trị của ColumnWidthInLumaSamples[i], chỉ ra rằng độ rộng của cột ô thứ i trong các đơn vị mẫu độ sáng, được suy ra như sau:

```
for( i = 0; i <= num_tile_columns_minus1; i++ ) {
    if( uniform_spacing_flag )
      ColumnWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 ) –
                         ( i * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 )
    else
      ColumnWidth[ i ] = column_width[ i ]
    ColumnWidthInLumaSamples[ i ] = ColumnWidth[ i ] << Log2CtbSize
}
```

Các giá trị của RowHeight[i], chỉ ra rằng độ cao của hàng ô thứ i trong các đơn vị của các khối cây mã hóa, được suy ra như sau:

```
for( i = 0; i <= num_tile_rows_minus1; i++ )
    if( uniform_spacing_flag )
      RowHeight[ i ]  = ( ( i + 1 ) * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1 ) –
                        ( i * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1)
    else
      RowHeight[ i ] = row_height[ i ]
```

Các giá trị của ColBd[i], chỉ ra vị trí của biên cột bên trái của cột ô thứ i trong các đơn vị của các khối cây mã hóa, được suy ra như sau:

```
for( ColBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
      ColBd[ i + 1 ] = ColBd[ i ] + ColumnWidth[ i ]
```

Các giá trị của RowBd[i], chỉ ra vị trí của biên hàng trên cùng của hàng ô thứ i trong các đơn vị khối cây mã hóa, được suy ra như sau:

```
for( RowBd[ 0 ] = 0, i = 0; i <= num_tile_rows_minus1; i++ )
    RowBd[ i + 1 ] = RowBd[ i ] + RowHeight[ i ]
```

num_substreams_minus1 plus 1 chỉ ra rằng số tập con lớn nhất được chứa trong lát khi tiles_or_entropy_coding_sync_idc bằng 2. Khi không có mặt, giá trị của num_substreams_minus1 được suy ra là bằng 0.

num_entry_point_offsets chỉ ra số lượng các phần tử cú pháp entry_point_offset[i] trong đoạn đầu lát. Khi tiles_or_entropy_coding_sync_idc bằng 1, giá trị của num_entry_point_offsets sẽ nằm trong khoảng từ 0 đến (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) – 1, bao gồm cả hai

giá trị đầu mút. Khi tiles_or_entropy_coding_sync_idc bằng 2, giá trị của num_entry_point_offsets sẽ nằm trong khoảng từ 0 đến num_substreams_minus1, bao gồm cả hai giá trị đầu mút. Khi không có mặt, giá trị của num_entry_point_offsets được suy ra là bằng 0.

ffset_len_minus1 plus 1 chỉ ra độ dài, tính bằng bit, của các phần tử cú pháp entry_point_offset[i].

entry_point_offset[i] chỉ ra độ lệch điểm vào thứ i, tính bằng byte và sẽ được biểu diễn bởi offset_len_minus1 cộng 1 bit. Đơn vị NAL lát đã được mã hóa gồm có num_entry_point_offsets + 1 tập con, với các giá trị chỉ số tập con nằm trong khoảng từ 0 đến num_entry_point_offsets, bao gồm cả hai giá trị đầu mút. Tập con 0 gồm có 0 byte đến entry_point_offset[0] − 1, bao gồm cả hai giá trị đầu mút, đơn vị NAL lát đã được mã hóa, tập con k, với k nằm trong khoảng từ 1 đến num_entry_point_offsets − 1, bao gồm cả hai giá trị đầu mút, gồm có entry_point_offset[k − 1] đến entry_point_offset[k] + entry_point_offset[k − 1] − 1 byte, bao gồm cả hai giá trị đầu mút, của đơn vị NAL lát đã được mã hóa, và tập con cuối cùng (với chỉ số tập con bằng num_entry_point_offsets) gồm có các byte còn lại của đơn vị NAL lát đã được mã hóa.

Lưu ý rằng: Đoạn đầu đơn vị NAL và đoạn đầu lát của đơn vị NAL lát đã được mã hóa luôn được nằm trong tập con 0.

Khi tiles_or_entropy_coding_sync_idc bằng 1 và num_entry_point_offsets là lớn hơn 0, mỗi tập con sẽ chứa tất cả các bit đã được mã hóa của một hoặc nhiều ô đầy đủ, và số lượng tập con sẽ bằng hoặc nhỏ hơn so với số lượng ô trong lát.

Khi tiles_or_entropy_coding_sync_idc bằng 2 và num_entry_point_offsets là lớn hơn 0, tập con k, cho từng giá trị k của tất cả các giá trị k có thể, sẽ chứa tất cả các bit được sử dụng trong quy trình khởi tạo cho con trỏ dòng bit hiện tại k.

Các ngữ nghĩa dữ liệu lát sau đây có thể áp dụng.

end_of_slice_flag bằng 0 chỉ ra khối macro khác là tiếp theo trong lát. end_of_slice_flag bằng 1 chỉ ra sự kết thúc lát và không có khối macro nào nữa tiếp theo.

entry_point_marker_two_3bytes là một chuỗi giá trị cố định của 3 byte bằng 0x000002. Phần tử cú pháp này được gọi là tiền tố đánh dấu vào. Phần tử cú pháp này được gọi là tiền tố đánh dấu vào.

tile_idx_minus_1 chỉ ra TileID trong thứ tự quét mành. Ô thứ nhất trong bức ảnh sẽ có TileID của 0. Giá trị của tile_idx_minus_1 sẽ nằm trong khoảng từ 0 đến (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1.

Quy trình phân tích cú pháp CABAC cho dữ liệu lát có thể là như sau:

Quy trình này cần đến khi phân tích cú pháp các phần tử cú pháp bằng bộ mô tả ae(v).

Các đầu vào cho quy trình này là yêu cầu giá trị của phần tử cú pháp và các giá trị của các phần tử cú pháp đã được phân tích cú pháp trước đó.

Đầu ra của quy trình này là giá trị của phần tử cú pháp.

Khi bắt đầu phân tích cú pháp dữ liệu lát của lát, quá trình khởi tạo của quy trình phân tích cú pháp CABAC được cần đến. Khi tiles_or_entropy_coding_sync_idc bằng 2 và num_substreams_minus1 lớn hơn 0, bảng ánh xạ BitStreamTable với các lối vào num_substreams_minus1 + 1 chỉ ra bảng con trỏ dòng bit để sử dụng cho phép suy ra con trỏ dòng bit hiện tại về sau thu được như sau.

–BitStreamTable[0] được khởi tạo để chứa con trỏ dòng bit.

–Đối với tất cả các chỉ mục i lớn hơn 0 và nhỏ hơn num_substreams_minus1+1, BitStreamTable[i] chứa con trỏ dòng bit cho entry_point_offset[i] byte sau BitStreamTable[i − 1].

Con trỏ dòng bit hiện tại được thiết lập đến BitStreamTable[0].

Địa chỉ khối mã hóa nhỏ nhất của khối cây mã hóa chứa khối lân cận trong không gian T, ctbMinCbAddrT, được suy ra nhờ sử dụng vị trí (x0, y0) của mẫu độ sáng trên cùng-bên trái của khối cây mã hóa hiện tại, ví dụ như sau:

$$x = x0 + 2 << Log2CtbSize - 1$$

$$y = y0 - 1$$

ctbMinCbAddrT = MinCbAddrZS[x >> Log2MinCbSize][y >> Log2MinCbSi ze]

Biến availableFlagT thu được bằng cách dẫn ra quy trình suy ra khả dụng khối mã hóa thích hợp với ctbMinCbAddrT làm đầu vào.

Khi bắt đầu phân tích cú pháp cây mã hóa và tiles_or_entropy_coding_sync_idc bằng 2 và num_substreams_minus1 là lớn hơn 0, dưới đây áp dụng.

– Nếu CtbAddrRS % PicWidthInCtbs bằng 0, sau đây áp dụng.

–Khi availableFlagT bằng 1, quá trình đồng bộ hóa của quy trình phân tích cú pháp CABAC được dẫn ra như được chỉ ra trong mệnh đề phụ "quá trình đồng bộ hóa cho các biến ngữ cảnh".

–Quá trình giải mã cho các quyết định nhị phân trước khi kết thúc được dẫn ra, sau đó là quá trình khởi tạo cho động cơ giải mã số học.

–Con trỏ dòng bit hiện tại được thiết lập để chỉ ra BitStreamTable[i] với chỉ số i thu được như sau.

i = (CtbAddrRS / PicWidthInCtbs) % (num_substreams_minus1 + 1)

–Nói cách khác, nếu CtbAddrRS % PicWidthInCtbs bằng 2, quá trình ghi nhớ của quy trình phân tích cú pháp CABAC được dẫn ra như được chỉ ra trong mệnh đề phụ "quá trình ghi nhớ cho các biến ngữ cảnh".

Quy trình khởi tạo có thể là như sau:

Các đầu ra của quy trình này là các biến bên trong CABAC được khởi tạo.

Các quy trình đặc biệt của nó được dẫn ra khi bắt đầu phân tích cú pháp dữ liệu lát của lát hoặc khi bắt đầu phân tích cú pháp của dữ liệu cây mã hóa và cây mã hóa là cây mã hóa thứ nhất trong ô.

Quá trình ghi nhớ cho các biến ngữ cảnh có thể là như sau:

Các đầu vào của quá trình này là các biến ngữ cảnh CABAC được chỉ dẫn bằng ctxIdx.

Đầu ra của quá trình này là các biến TableStateSync và TableMPSSync chứa các giá trị của các biến m và n được sử dụng trong quá trình khởi tạo của các biến ngữ cảnh mà các biến ngữ cảnh này được gán cho các phần tử cú pháp ngoại trừ end-of-slice flag. Đối với mỗi biến ngữ cảnh, các lối vào tương ứng n và m của các bảng TableStateSync và TableMPSSync được khởi tạo cho pStateIdx và valMPS tương ứng.

Quá trình đồng bộ hóa cho các biến ngữ cảnh có thể là như sau:

Các đầu vào của quy trình này là các biến TableStateSync và TableMPSSync chứa các giá trị của các biến n và m được sử dụng trong quá trình nhớ các biến ngữ cảnh được gán cho các phần tử cú pháp ngoại trừ end-of-slice flag.

Các đầu ra của quy trình này là các biến ngữ cảnh CABAC được biểu thị bởi ctxIdx.

Đối với mỗi biến ngữ cảnh, các biến ngữ cảnh tương ứng pStateIdx và valMPS được khởi tạo cho các lối vào tương ứng n và m của các bảng TableStateSync và TableMPSSync.

Sau đây, bước vận chuyển và mã hóa độ trễ thấp sử dụng WPP được giải thích. Cụ thể, phần thảo luận bên dưới bộc lộ về việc làm thế nào để việc vận chuyển độ trễ thấp như được mô tả trong Fig.7 cũng có thể được áp dụng cho WPP.

Trước hết, quan trọng là tập con của bức ảnh có thể được gửi đi, trước khi hoàn thành toàn bộ bức ảnh. Thông thường, điều này có thể đạt được nhờ sử dụng các lát, như được thể hiện trong Fig.5.

Để làm giảm độ trễ so với các ô, như được thể hiện trong các hình vẽ sau đây, cần áp dụng một dòng con WPP đơn lẻ trên mỗi hàng LCU và còn cho phép truyền riêng biệt mỗi hàng. Để giữ hiệu suất mã hóa cao, các lát trên mỗi hàng/dòng con không thể được sử dụng. Do đó, dưới đây, cái được gọi là lát phụ thuộc như được định ra trong phần kế tiếp được giới thiệu. Lát này, ví dụ, không có tất cả các trường của đoạn đầu lát HEVC đầy đủ, nhưng các trường được sử dụng cho các lát entropy. Hơn nữa, có thể có sự chuyển đổi để ngắt sự gián đoạn của CABAC giữa các hàng. Trong trường hợp WPP, sử dụng ngữ cảnh CABAC (các mũi tên trên Fig.14) và dự đoán các hàng sẽ được cho phép để giữ sự tăng hiệu suất mã hóa của WPP trên các ô.

Cụ thể, Fig.14 minh họa ảnh 10 cho WPP thành lát đều 900 (reg. SL), và, để xử lý độ trễ thấp, thành các lát phụ thuộc (OS) 920.

Hiện tại, tiêu chuẩn HEVC sắp tới đưa ra hai loại phân chia dưới dạng các lát. Có lát đều (chuẩn) và lát entropy. Lát đều là phần ảnh độc lập hoàn toàn ngoại trừ một số thuộc tính phụ thuộc có thể sẵn có do quá trình bộ lọc tách khối trên các biên lát. Lát entropy cũng có tính độc lập nhưng chỉ về mặt mã hóa entropy. Ý tưởng của Fig.14 là để tạo ra khái niệm cắt lát. Do đó, tiêu chuẩn HEVC sắp tới nên đưa ra hai loại lát chung: độc lập (đều) hoặc phụ thuộc. Do đó, loại lát mới, lát phụ thuộc được giới thiệu.

Lát phụ thuộc là lát có các sự phụ thuộc vào lát trước. Các sự phụ thuộc là dữ liệu cụ thể có thể được sử dụng giữa các lát trong quá trình giải mã entropy và/hoặc quy trình khôi phục điểm ảnh.

Trong Fig.14, khái niệm của các lát phụ thuộc được đưa ra làm thí dụ. Bức ảnh bắt đầu, ví dụ, thường với lát đều. Lưu ý rằng, trong khái niệm này trạng thái lát đều thay đổi một chút. Thông thường, trong các tiêu chuẩn như H264/AVC hoặc HEVC, lát đều là phần độc lập hoàn toàn và không có giữ bất kỳ dữ liệu nào sau khi giải mã ngoại trừ một vài dữ liệu cho quy trình của bộ lọc tách khối. Nhưng việc xử lý lát phụ thuộc sắp tới 920 chỉ có thể thực hiện được bằng cách tham chiếu dữ liệu của lát bên trên, ở đây trong hàng thứ nhất: lát đều 900. Để thiết lập điều đó, các lát đều 900 sẽ giữ dữ liệu của hàng CU cuối cùng. Dữ liệu này gồm có:

- dữ liệu động cơ mã hóa CABAC (các trạng thái mô hình ngữ cảnh của một CU mà từ đó quá trình giải mã entropy của lát phụ thuộc có thể được khởi tạo),

- tất cả các phần tử cú pháp đã được giải mã của các CU cho quá trình giải mã CABAC đều của các CU phụ thuộc,

- Dữ liệu dự đoán dưới vectơ và chuyển động.

Do đó mỗi lát phụ thuộc 920 sẽ thực hiện cùng thủ tục – giữ dữ liệu cho lát phụ thuộc sắp tới trong cùng bức ảnh.

Trong thực tế, các bước bổ sung này không phải là vấn đề, bởi vì quá trình giải mã nói chung bị ép buộc thường xuyên để lưu trữ dữ liệu nào đó như các phần tử cú pháp.

Trong các phần bên dưới, các sự thay đổi có thể của cú pháp tiêu chuẩn HEVC được yêu cầu để cho phép khái niệm của các lát phụ thuộc, được xuất hiện.

Fig.5 minh họa các sự thay đổi có thể có về cú pháp RBSP của tập hợp tham số ảnh.

Ngữ nghĩa của tập hợp tham số ảnh cho các lát phụ thuộc có thể là như sau:

dependent_slices_present_flag bằng 1 chỉ ra rằng bức ảnh đang chứa các lát phụ thuộc và quá trình giải mã từng lát (đều hoặc phụ thuộc) sẽ lưu trữ các trạng thái giải mã entropy và dữ liệu dự đoán vectơ chuyển động và dưới vectơ cho lát kế tiếp mà lát kế tiếp này có thể là lát phụ thuộc mà lát phụ thuộc này cũng có thể theo sau lát đều. Lát phụ thuộc sau đây có thể tham chiếu dữ liệu đã được lưu trữ đó.

Fig.16 thể hiện cú pháp slice_header có thể có với các sự thay đổi đối với trạng thái hiện tại của HEVC.

dependent_slice_flag bằng 1 chỉ ra rằng giá trị của các phần tử cú pháp đoạn đầu lát không xuất hiện được suy ra là bằng với giá trị của phần tử cú pháp đoạn đầu lát trong lát bắt đầu (đều), ở đó lát bắt đầu được định nghĩa là lát chứa khối cây mã hóa với vị trí (SliceCtbAddrRS − 1). dependent_slice_flag sẽ bằng 0 khi SliceCtbAddrRS = 0.

no_cabac_reset_flag bằng 1 chỉ ra sự khởi tạo CABAC từ trạng thái đã được lưu của lát đã được giải mã trước (và không với các giá trị ban đầu). Mặt khác, cụ thể, nếu bằng 0, khởi tạo CABAC độc lập với trạng thái bất kỳ của lát đã được giải mã trước đó, cụ thể với các giá trị ban đầu.

last_ctb_cabac_init_flag bằng 1 chỉ ra sự khởi tạo CABAC từ trạng thái đã được lưu của khối cây đã được mã hóa cuối cùng của lát đã được giải mã trước đó (ví dụ, cho các ô luôn bằng 1). Nói cách khác (bằng 0), dữ liệu khởi tạo được tham chiếu từ trạng thái đã lưu của khối cây đã được mã hóa thứ hai của hàng ctb cuối cùng (lân cận) của lát đã được giải mã trước đó, nếu khối cây đã được mã hóa thứ nhất của lát

hiện tại là khối cây đã được mã hóa thứ nhất trong hàng (cụ thể, chế độ WPP), nói cách khác khởi tạo CABAC được thực hiện từ trạng thái đã lưu của khối cây đã được mã hóa cuối cùng của lát đã được giải mã trước đó.

So sánh các lát phụ thuộc và các sơ đồ phân chia khác (thông tin) được cung cấp bên dưới.

Trên Fig.17, sự khác biệt giữa lát chuẩn và lát phụ thuộc được thể hiện.

Mã hóa và sự truyền có thể các dòng con WPP trong các lát phụ thuộc (DS - dependent slice) như được minh họa đối với Fig.18 so sánh mã hóa đối với sự vận chuyển độ trễ thấp các ô (bên trái) và WPP/DS (bên phải). Các dấu chữ thập được vẽ liên tục đậm nét trên Fig.18 thể hiện cùng thời điểm cho hai phương pháp giả định rằng việc mã hóa hàng WPP mất cùng thời gian như mã hóa một ô đơn lẻ. Do các sự phụ thuộc mã hóa, chỉ hàng thứ nhất của WPP là sẵn sàng, sau khi tất cả các ô đã được mã hóa. Nhưng sử dụng phương pháp lát phụ thuộc cho phép phương pháp WPP gửi đi hàng chứ nhất một khi nó được mã hóa. Điều này là khác với các sự gán dòng con sớm, "dòng con" được định ra cho WPP đóng vai trò là phép ghép nối các hàng CU của lát là WPP được giải mã bởi cùng dòng bộ giải mã, cụ thể, cùng lõi/bộ xử lý. Mặc dù, dòng con trên mỗi hàng và mỗi lát entropy cũng là có khả năng trước khi, lát entropy phá vỡ các sự phụ thuộc mã hóa entropy và do đó có hiệu suất mã hóa thấp hơn, cụ thể, mất đi sự tăng hiệu suất WPP.

Ngoài ra, sự khác biệt độ trễ giữa cả hai phương pháp có thể là thực sự thấp, giả định rằng sự truyền như được thể hiện trên Fig.19. Cụ thể, Fig.19 minh họa mã hóa WPP với sự truyền độ trễ thấp được dẫn.

Giả định rằng việc mã hóa hai CU về sau của DS #1.1 trong phương pháp WPP trên Fig.18 không làm mất thời gian lâu hơn so với việc truyền hàng thứ nhất SL #1, không có sự khác biệt nào giữa các ô và WPP trong trường hợp độ trễ thấp. Nhưng hiệu suất mã hóa của WP/DS làm tốt hơn khái niệm ô.

Để làm tăng độ mạnh cho chế độ độ trễ thấp WPP, Fig.20 minh họa sự cải thiện độ mạnh đạt được bằng cách sử dụng các lát đều (RS) (RS là viết tắt của Regular Slices) như các neo. Trong bức ảnh được thể hiện trong Fig.20, lát (đều) (RS) theo sau

là các lát phụ thuộc (DS) (DS là viết tắt của dependent slice). Ở đây, lát (đều) đóng vai trò là mẩu neo để phá vỡ các tính phụ thuộc với các lát có trước, do đó độ mạnh hơn được cung cấp ở điểm gài lát (đều). Về nguyên tắc, dù thế nào cũng không có gì khác với việc gài các lát (đều).

Khái niệm của các lát phụ thuộc cũng có thể được cung cấp sau đây.

Ở đây, Fig.21 thể hiện cú pháp đoạn đầu lát có thể có.

Các ngữ nghĩa đoạn đầu lát là như sau:

dependent_slice_flag bằng 1 chỉ ra rằng giá trị của mỗi phần tử cú pháp đoạn đầu lát không xuất hiện được suy ra là bằng với giá trị của phần tử cú pháp đoạn đầu lát tương ứng trong lát trước chứa khối cây mã hóa mà địa chỉ khối cây mã hóa cho khối cây mã hóa này là SliceCtbAddrRS − 1. Khi không xuất hiện, giá trị của dependent_slice_flag được suy ra là bằng 0. Giá trị dependent_slice_flag sẽ bằng 0 khi SliceCtbAddrRS = 0.

slice_address chỉ ra địa chỉ trong độ phân giải độ chi tiết lát trong đó lát bắt đầu. Độ dài của phần tử cú pháp slice_address là (Ceil(Log2(PicWidthInCtbs * PicHeightInCtbs)) + SliceGranularity) bit.

Biến SliceCtbAddrRS, chỉ ra khối cây mã hóa trong đó lát bắt đầu trong thứ tự quét mành khối cây mã hóa, thu được như sau.

SliceCtbAddrRS = (slice_address >> SliceGranularity)

Biến SliceCbAddrZS, chỉ ra rằng địa chỉ của khối mã hóa thứ nhất trong lát trong độ chi tiết khối mã hóa nhỏ nhất theo thứ tự quét z, thu được như sau.

SliceCbAddrZS = slice_address
<< ((log2_diff_max_min_coding_block_size − SliceGranularity) <<1)

Giải mã lát bắt đầu với đơn vị giải mã lớn nhất có thể, hoặc, nói cách khác, CTU, ở tọa độ khởi đầu lát.

first_slice_in_pic_flag chỉ ra liệu lát là lát thứ nhất của bức ảnh hay không. Nếu first_slice_in_pic_flag = 1, các biến SliceCbAddrZS và SliceCtbAddrRS đều được thiết lập bằng 0 và giải mã bắt đầu với khối cây giải mã thứ nhất trong bức ảnh.

pic_parameter_set_id chỉ ra tập hợp tham số ảnh trong sử dụng. Giá trị pic_parameter_set_id sẽ nằm trong khoảng từ 0 đến 255, bao gồm cả hai giá trị đầu mút.

num_entry_point_offsets chỉ ra số lượng các phần tử cú pháp entry_point_offset[i] trong đoạn đầu lát. Khi tiles_or_entropy_coding_sync_idc bằng 1, giá trị của num_entry_point_offsets sẽ nằm trong khoảng từ 0 đến (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1. Khi tiles_or_entropy_coding_sync_idc bằng 2, giá trị của num_entry_point_offsets sẽ nằm trong khoảng từ 0 đến PicHeightInCtbs − 1. Khi không xuất hiện, giá trị của num_entry_point_offsets được suy ra là bằng 0.

offset_len_minus1 cộng 1 chỉ ra độ dài, theo bit, của các phần tử cú pháp entry_point_offset[i].

entry_point_offset[i] chỉ ra entry_point_offset thứ i, theo byte và sẽ được biểu diễn bởi offset_len_minus1 cộng 1 bit. Dữ liệu lát đã được mã hóa sau khi đoạn đầu lát gồm có num_entry_point_offsets + 1 tập con, với các giá trị chỉ số tập con nằm trong khoảng từ 0 đến num_entry_point_offsets. Tập con 0 gồm có 0 byte đến entry_point_offset[0] − 1, bao gồm cả hai giá trị đầu mút, của dữ liệu lát đã được mã hóa, tập con k, với k nằm trong khoảng từ 1 đến num_entry_point_offsets - 1, bao gồm entry_point_offset[k − 1] byte đến entry_point_offset[k] + entry_point_offset[k − 1] − 1 byte, bao gồm cả hai giá trị đầu mút, của dữ liệu lát đã được mã hóa, và tập con cuối cùng (với chỉ số tập con bằng với num_entry_point_offsets) gồm có các byte còn lại của dữ liệu lát đã được mã hóa.

Khi tiles_or_entropy_coding_sync_idc bằng 1 và num_entry_point_offsets là lớn hơn 0, mỗi tập con sẽ chứa tất cả các bit đã được mã hóa của chính xác một ô, và số lượng tập con (cụ thể, giá trị của num_entry_point_offsets + 1) sẽ bằng hoặc nhỏ hơn so với số lượng ô trong lát.

Lưu ý – Khi tiles_or_entropy_coding_sync_idc bằng 1, mỗi lát phải bao gồm hoặc một tập con của một ô (trong đó trường hợp báo hiệu các điểm vào là không cần thiết) hoặc một số nguyên các ô đầy đủ.

Khi tiles_or_entropy_coding_sync_idc bằng 2 và num_entry_point_offsets là lớn hơn 0, mỗi tập con k với k nằm trong khoảng từ 0 đến num_entry_point_offsets − 1, bao gồm cả hai giá trị đầu mút, sẽ chứa tất cả các bit đã được mã hóa của chính xác một hàng các khối cây mã hóa, tập con cuối cùng (với chỉ số tập con bằng num_entry_point_offsets) sẽ chứa tất cả các bit đã được mã hóa của khối mã hóa còn lại được chứa trong lát, trong đó các khối mã hóa còn lại gồm có chính xác một hàng các khối cây mã hóa hoặc tập con của một hàng các khối cây mã hóa, và số lượng tập con (cụ thể, giá trị của num_entry_point_offsets + 1) sẽ bằng với số lượng hàng của khối cây mã hóa trong lát, trong đó một tập con của một hàng của các khối cây mã hóa trong lát cũng được đếm.

Lưu ý: Khi tiles_or_entropy_coding_sync_idc bằng 2, lát có thể bao gồm một số lượng hàng gồm các khối cây mã hóa và một tập con hàng gồm các khối cây mã hóa. Ví dụ, nếu lát bao gồm 2,5 hàng khối cây mã hóa, số lượng tập con (cụ thể, giá trị của num_entry_point_offsets + 1) sẽ là bằng 3.

Cú pháp RBSP của tập hợp tham số ảnh tương ứng có thể được chọn như được thể hiện trên Fig.22.

Các ngữ nghĩa của RBSP của tập hợp tham số ảnh có thể là như sau:

dependent_slice_enabled_flag = 1 chỉ ra sự có mặt của phần tử cú pháp dependent_slice_flag trong đoạn đầu lát cho các bức ảnh đã được mã hóa tham chiếu đến tập hợp tham số ảnh. dependent_slice_enabled_flag bằng 0 chỉ ra sự vắng mặt của phần tử cú pháp dependent_slice_flag trong đoạn đầu lát cho các bức ảnh đã được mã hóa tham chiếu đến tập hợp tham số ảnh. Khi tiles_or_entropy_coding_sync_idc bằng 3, giá trị dependent_slice_enabled_flag sẽ bằng 1.

tiles_or_entropy_coding_sync_idc bằng 0 chỉ ra rằng sẽ chỉ có một ô trong từng bức ảnh tham chiếu đến tập hợp tham số ảnh, sẽ không có quá trình đồng bộ hóa riêng nào cho các biến ngữ cảnh được dẫn ra trước khi giải mã khối cây mã hóa thứ nhất của hàng các khối cây mã hóa trong từng bức ảnh tham chiếu đến tập hợp tham số ảnh, và các giá trị của cabac_independent_flag và dependent_slice_flag cho các bức ảnh đã mã hóa tham chiếu đến tập hợp tham số ảnh sẽ đều không bằng 1.

Lưu ý, khi cabac_independent_flag và depedent_slice_flag đều bằng 1 cho một lát, lát là lát entropy.

tiles_or_entropy_coding_sync_idc bằng 1 chỉ ra rằng có thể có nhiều hơn một ô trong từng bức ảnh tham chiếu đến tập hợp tham số ảnh, sẽ không có quá trình đồng bộ hóa riêng nào cho các biến ngữ cảnh được dẫn ra trước khi giải mã khối cây mã hóa thứ nhất của một hàng các khối cây mã hóa trong từng bức ảnh tham chiếu đến tập hợp tham số ảnh, và các giá trị của cabac_independent_flag và dependent_slice_flag cho các bức ảnh đã mã hóa đề cập đến tạp hợp tham số ảnh sẽ đều không bằng 1.

tiles_or_entropy_coding_sync_idc bằng 2 chỉ ra rằng sẽ chỉ có một ô trong từng bức ảnh tham chiếu đến tập hợp tham số ảnh, quá trình đồng bộ hóa riêng cho các biến ngữ cảnh sẽ được dẫn ra trước khi giải mã khối cây mã hóa thứ nhất của hàng gồm các khối cây mã hóa trong từng bức ảnh tham chiếu đến tập hợp tham số ảnh, và quy trình nhớ riêng cho các biến ngữ cảnh sẽ được dẫn ra sau khi giải mã hai khối cây mã hóa của hàng gồm các khối cây mã hóa trong từng bức ảnh đề cập đến tập hợp tham số ảnh, và các giá trị của cabac_independent_flag và dependent_slice_flag cho các bức ảnh đã được mã hóa đề cập đến tập hợp tham số ảnh sẽ đều không bằng 1.

tiles_or_entropy_coding_sync_idc = 3 chỉ ra rằng sẽ chỉ có một ô trong từng bức ảnh liên quan đến tập hợp tham số ảnh, sẽ không có quá trình đồng bộ hóa riêng nào cho các biến ngữ cảnh được dẫn ra trước khi giải mã khối cây mã hóa thứ nhất của hàng gồm các khối cây mã hóa trong từng bức ảnh đề cập đến tập hợp tham số ảnh, và các giá trị của cabac_independent_flag và dependent_slice_flag cho các bức ảnh đã được mã hóa đề cập đến tập hợp tham số ảnh có thể đều bằng 1.

Khi dependent_slice_enabled_flag bằng 0, tiles_or_entropy_coding_sync_idc sẽ không bằng 3.

Yêu cầu tương thích dòng bit của nó là giá trị của tiles_or_entropy_coding_sync_idc sẽ là như nhau đối với tất cả các tập hợp tham số ảnh mà các tập hợp tham số này được kích hoạt trong chuỗi video đã được mã hóa.

Đối với mỗi lát tham chiếu đến tập hợp tham số ảnh, khi tiles_or_entropy_coding_sync_idc bằng 2 và khối mã hóa thứ nhất trong lát không

phải là khối mã hóa thứ nhất trong khối cây mã hóa thứ nhất của hàng gồm các khối cây mã hóa, khối mã hóa cuối cùng trong lát sẽ thuộc về cùng hàng các khối cây mã hóa giống như khối mã hóa thứ nhất trong lát.

num_tile_columns_minus1 cộng 1 chỉ ra số lượng cột ô phân chia bức ảnh.

num_tile_rows_minus1 cộng 1 chỉ ra số lượng hàng ô phân chia bức ảnh.

Khi num_tile_columns_minus1 bằng 0, num_tile_rows_minus1 sẽ không bằng 0. uniform_spacing_flag bằng 1 chỉ ra các biên cột và tương tự các biên hàng được phân bố đồng đều trên bức ảnh. uniform_spacing_flag bằng 0 chỉ ra rằng các biên cột và tương tự các biên hàng không được phân bố đều trên bức ảnh nhưng được báo hiệu một cách rõ ràng nhờ sử dụng các phần tử cú pháp column_width[i] và row_height[i].

column_width[i] chỉ ra chiều rộng của cột ô thứ i trong các đơn vị của các khối cây mã hóa.

row_height[i] chỉ ra độ cao của hàng ô thứ i trong các đơn vị của các khối cây mã hóa.

Vectơ colWidth[i] chỉ ra chiều rộng của cột ô thứ i trong các đơn vị của CTB với cột i nằm trong khoảng từ 0 đến num_tile_columns_minus1, bao gồm cả hai giá trị đầu mút.

Vectơ CtbAddrRStoTS[ctbAddrRS] chỉ ra cuộc hội thoại từ địa chỉ CTB theo thứ tự quét mành đến địa chỉ CTB theo thứ tự quét ô với chỉ số ctbAddrRS nằm trong khoảng từ 0 đến (picHeightInCtbs * picWidthInCtbs) − 1, bao gồm cả hai giá trị đầu mút.

Vectơ CtbAddrTStoRS[ctbAddrTS] chỉ ra cuộc hội thoại từ địa chỉ CTB trong thứ tự quét ô đến địa cỉ CTB theo thứ tự quét mành với chỉ số ctbAddrTS nằm trong khoảng từ 0 đến (picHeightInCtbs * picWidthInCtbs) − 1, bao gồm cả hai giá trị đầu mút.

Vectơ TileId[ctbAddrTS] chỉ ra cuộc hội thoại từ địa chỉ CTB trong thứ tự quét ô đến id với ctbAddrTS nằm trong khoảng từ 0 đến (picHeightInCtbs * picWidthInCtbs) − 1, bao gồm cả hai giá trị đầu mút.

Các giá trị của colWidth, CtbAddrRStoTS, CtbAddrTStoRS và TileId thu được bằng cách dẫn ra mành CTB và quá trình hội thoại quét ô với PicHeightInCtbs và PicWidthInCtbs như các đầu vào và đầu ra được gán cho colWidth, CtbAddrRStoTS và TileId.

Các giá trị của ColumnWidthInLumaSamples[i], chỉ ra chiều rộng của cột ô thứ i trong các đơn vị mẫu độ sáng, được thiết lập bằng với colWidth[i] << Log2CtbSize.

Mảng MinCbAddrZS[x][y], chỉ ra cuộc hội thoại từ vị trí (x, y) trong các đơn vị CB nhỏ nhất đến địa chỉ CB nhỏ nhất theo thứ tự quét z với x nằm trong khoảng từ 0 đến picWidthInMinCbs − 1, bao gồm cả hai giá trị đầu mút và y nằm trong khoảng từ 0 đến picHeightInMinCbs − 1, bao gồm cả hai giá trị đầu mút, thu được bằng cách dẫn ra qua trình khởi tạo mảng theo thứ tự quét Z với Log2MinCbSize, Log2CtbSize, PicHeightInCtbs, PicWidthInCtbs, và vectơ CtbAddrRStoTS làm các đầu vào và đầu ra được gán cho MinCbAddrZS.

loop_filter_across_tiles_enabled_flag bằng 1 chỉ ra rằng các phép toán lọc theo vòng được thực hiện qua các biên ô. loop_filter_across_tiles_enabled_flag bằng 0 chỉ ra rằng các phép toán lọc theo vòng không được thực hiện qua các biên ô. Các phép toán lọc theo vòng bao gồm các bộ lọc tách khối, độ lệch thích ứng mẫu, và các phép toán của bộ lọc vòng thích ứng. Khi không xuất hiện, giá trị của loop_filter_across_tiles_enabled_flag được suy ra là bằng 1.

cabac_independent_flag bằng 1 chỉ ra rằng bước giải mã CABAC các khối mã hóa trong lát là độc lập từ bất cứ trạng thái nào của lát đã được giải mã trước đó. cabac_independent_flag bằng 0 chỉ ra rằng bước giải mã CABAC các khối mã hóa trong lát là phụ thuộc vào các trạng thái của lát đã được giải mã trước. Khi không xuất hiện, giá trị của cabac_independent_flag được suy ra là bằng 0.

Quy trình suy ra cho tính khả dụng của khối mã hóa với địa chỉ khối mã hóa nhỏ nhất có thể là như sau:

Các đầu vào cho quy trình này là

–địa chỉ khối mã hóa nhỏ nhất minCbAddrZS theo thứ tự quét z

địa chỉ khối mã hóa nhỏ nhất hiện tại currMinCBAddrZS theo thứ tự quét z

Đầu ra của quy trình này là tính khả dụng của khối mã hóa với địa chỉ khối mã hóa nhỏ nhất cbAddrZS theo thứ tự quét z cbAvailable.

Lưu ý 1 – Nghĩa của tính khả dụng được xác định khi quy trình này được viện dẫn.

Lưu ý 2 – Khối mã hóa bất kỳ, không kể đến kích thước của nó, gắn với địa chỉ khối mã hóa nhỏ nhất, là địa chỉ của khối mã hóa với kích thước khối mã hóa nhỏ nhất theo thứ tự quét z.

–Nếu một hoặc nhiều điều kiện sau đây là đúng, cbAvailable được thiết lập là FALSE.

–minCbAddrZS là nhỏ hơn 0

–minCbAddrZS là lớn hơn currMinCBAddrZS

–khối mã hóa với địa chỉ khối mã hóa nhỏ nhất minCbAddrZS thuộc về lát khác so với khối mã hóa với địa chỉ khối mã hóa nhỏ nhất hiện tại currMinCBAddrZS và dependent_slice_flag của lát chứa khối mã hóa với địa chỉ khối mã hóa nhỏ nhất hiện tại currMinCBAddrZS = 0.

–khối mã hóa với địa chỉ khối mã hóa nhỏ nhất minCbAddrZS được chứa trong ô khác so với khối mã hóa với địa chỉ khối mã hóa nhỏ nhất hiện tại currMinCBAddrZS.

–Nếu không thì, cbAvailable được thiết lập là TRUE.

Quy trình phân tích cú pháp CABAC cho dữ liệu lát có thể là như sau:

Quy trình này được viện dẫn khi phân tích cú pháp các phần tử cú pháp nhất định với bộ mô tả ae(v).

Các đầu vào cho quy trình này là yêu cầu giá trị của phần tử cú pháp và các giá trị của các phần tử cú pháp đã được phân tích cú pháp từ trước.

Đầu ra của quy trình này là giá trị của phần tử cú pháp.

Khi bắt đầu phân tích cú pháp dữ liệu lát của lát, quy trình khởi tạo của quy trình phân tích cú pháp CABAC được dẫn ra.

Fig.23 minh họa về việc làm thế nào vùng lân cận trong không gian T được dùng để dẫn ra quy trình suy ra có sẵn khối cây mã hóa đối với khối cây mã hóa hiện tại (mạng thông tin).

Địa chỉ khối mã hóa nhỏ nhất của khối cây mã hóa chứa khối lân cận trong không gian T (Fig.23), ctbMinCbAddrT, được suy ra nhờ sử dụng vị trí (x0, y0) của mẫu độ sáng trên cùng-bên trái của khối cây mã hóa hiện tại như sau.

$$x = x0 + 2 << Log2CtbSize - 1$$

$$y = y0 - 1$$

$$ctbMinCbAddrT = MinCbAddrZS[\ x >> Log2MinCbSize\ ][\ y >> Log2MinCbSize\ ]$$

Biến availableFlagT được thu bằng cách dẫn quá trình suy ra tính khả dụng khối mã hóa với ctbMinCbAddrT làm đầu vào.

Khi bắt đầu phân tích cú pháp cây mã hóa như đã chỉ ra, các bước theo thứ tự sau đây sử dụng.

Động cơ giải mã số học được bắt đầu như sau.

Nếu CtbAddrRS bằng slice_address, dependent_slice_flag bằng 1 và entropy_coding_reset_flag bằng 0, dưới đây áp dụng.

Quá trình đồng bộ hóa của quy trình phân tích cú pháp CABAC được dẫn ra với TableStateIdxDS và TableMPSValDS làm đầu vào.

Quá trình giải mã cho các quyết định nhị phân trước khi kết thúc được dẫn ra, sau đó là quá trình khởi tạo cho động cơ giải mã số học.

Nếu không thì nếu tiles_or_entropy_coding_sync_idc bằng 2, và CtbAddrRS % PicWidthInCtbs bằng 0, sau đây áp dụng.

Khi availableFlagT bằng 1, quá trình đồng bộ hóa của quy trình phân tích cú pháp CABAC được dẫn ra với TableStateIdxWPP và TableMPSValWPP làm đầu vào.

Quá trình giải mã cho các quyết định nhị phân trước khi kết thúc được dẫn ra, sau đó là quá trình cho động cơ giải mã số học.

Khi cabac_independent_flag bằng 0 và dependent_slice_flag bằng 1, hoặc khi tiles_or_entropy_coding_sync_idc bằng 2, quá trình ghi nhớ được áp dụng như sau.

Khi tiles_or_entropy_coding_sync_idc bằng 2 và CtbAddrRS % PicWidthInCtbs bằng 2, quá trình ghi nhớ của quy trình phân tích cú pháp CABAC được dẫn ra với TableStateIdxWPP và TableMPSValWPP làm đầu ra.

Khi cabac_independent_flag bằng 0, dependent_slice_flag bằng 1, và end_of_slice_flag bằng 1, quá trình ghi nhớ của quy trình phân tích cú pháp CABAC được dẫn ra với TableStateIdxDS và TableMPSValDS làm đầu ra.

Phân tích cú pháp các phần tử cú pháp tiến hành như sau:

Đối với mỗi giá trị được yêu cầu của phần tử cú pháp, phép nhị phân hóa được suy ra.

Phép nhị phân hóa phần tử cú pháp và chuỗi các kí tự nhị phân đã được phân tích cú pháp định ra dòng quá trình giải mã.

Đối với mỗi kí tự nhị phân của quá trình nhị phân hóa phần tử cú pháp, mà được chỉ dẫn bởi biến binIdx, chỉ số ngữ cảnh ctxIdx được suy ra.

Đối với mỗi ctxIdx, quá trình giải mã số học được dẫn ra.

Chuỗi thu được (b0..bbinIdx) của các kí tự nhị phân đã được phân tích cú pháp được so sánh với tập hợp các chuỗi nhị phân được đưa ra bởi quá trình nhị phân hóa sau khi giải mã từng kí tự nhị phân. Khi chuỗi khớp chuỗi kí tự nhị phân trong bộ đã cho, giá trị tương ứng được gán cho phần tử cú pháp.

Trong trường hợp yêu cầu giá trị phần tử cú pháp được xử lý cho phần tử cú pháp pcm-flag và giá trị đã được giải mã của pcm_flag bằng 1, động cơ giải mã được bắt đầu sau khi giải mã bất kỳ pcm_alignment_zero_bit, num_subsequent_pcm, và tất cả pcm_sample_luma và pcm_sample_chroma data.

Do đó, phần mô tả bên trên bộc lộ bộ giải mã như được thể hiện trên Fig.24. Bộ giải mã này được chỉ ra bởi số tham chiếu 5, bộ giải mã khôi phục ảnh 10 từ dòng dữ

liệu 12 mà ảnh 10 được mã hóa trong các đơn vị lát 14 thành dòng dữ liệu này, và ảnh 10 được phân chia thành các đơn vị lát này, trong đó bộ giải mã 5 được tạo cấu hình để giải mã các lát 14 từ dòng dữ liệu 12 theo thứ tự lát 16. Tất nhiên, bộ giải mã 5 không bị giới hạn để giải mã định kỳ các lát 14. Hơn nữa, bộ giải mã 5 có thể sử dụng sự xử lý song song mặt đầu sóng để giải mã các lát 14, cung cấp ảnh 10 phân chia thành các lát 14 là thích hợp để xử lý song song mặt đầu sóng. Theo đó, bộ giải mã 5 có thể, ví dụ, là bộ giải mã có thể giải mã các lát 14 song song theo cách thức so le với việc bắt đầu giải mã các lát 14 bằng cách để ý đến thứ tự lát 16 cho phép xử lý mặt đầu sóng như đã mô tả bên trên và cũng sẽ được mô tả bên dưới.

Bộ giải mã 5 đáp lại phần phần tử cú pháp 18 nằm trong lát hiện tại của các lát 14 để giải mã lát hiện tại theo một trong ít nhất hai chế độ 20 và 22. Theo chế độ thứ nhất trong ít nhất hai chế độ, cụ thể chế độ 20, lát hiện tại được giải mã từ dòng dữ liệu 12 sử dụng bước giải mã entropy thích ứng ngữ cảnh bao gồm suy ra ngữ cảnh vượt qua các biên lát, cụ thể, vượt qua các đường đứt nét trên Fig.24, cụ thể, bằng cách sử dụng thông tin xuất phát từ việc mã hóa/giải mã "các lát có trước theo thứ tự lát 16" khác. Hơn nữa, việc giải mã lát hiện tại từ dòng dữ liệu 12 sử dụng chế độ thứ nhất 20 gồm có cập nhật liên tục các xác suất ký hiệu của bộ mã hóa-bộ giải mã và khởi tạo các xác suất ký hiệu tại lúc bắt đầu giải mã lát hiện tại mà nó phụ thuộc vào các trạng thái của các xác suất ký hiệu được lưu của lát đã được giải mã trước đó. Tính phụ thuộc này được mô tả bên trên, kết hợp với "quá trình đồng bộ hóa cho các biến số bộ mã hóa-bộ giải mã". Cuối cùng, chế độ thứ nhất 20 cũng bao gồm giải mã dự đoán vượt qua các biên lát. Giải mã dự đoán này vượt qua các biên lát có thể, ví dụ, bao gồm dưới dự đoán vượt qua biên lát, cụ thể dự đoán các giá trị mẫu trong lát hiện tại trên cơ sở của các giá trị mẫu đã được khôi phục của "theo thứ tự lát 16", lát có trước, hoặc dự đoán các tham số mã hóa vượt qua các biên lát như dự đoán các vectơ chuyển động, các chế độ dự đoán hoặc tương tự.

Theo chế độ thứ hai 22, bộ giải mã 5 giải mã lát hiện tại, cụ thể, lát hiện tại được giải mã, từ dòng dữ liệu 12 sử dụng giải mã entropy thích ứng ngữ cảnh với giới hạn suy ra các ngữ cảnh để không vượt qua các biên lát. Nếu, ví dụ, khuôn mẫu của các vị trí lân cận được sử dụng để suy ra ngữ cảnh cho phần tử cú pháp nào đó liên quan tới

khối nằm trong lát hiện tại kéo dài đến lát lân cận, do đó việc vượt qua biên lát của lát hiện tại, thuộc tính tương ứng của phần tương ứng của lát lân cận, như giá trị của phần tử cú pháp tương ứng của phần lân cận này của lát lân cận, được thiết lập đến giá trị mặc định để hạn chế các thuộc tính liên phụ thuộc giữa lát hiện tại và các lát lân cận. Trong khi việc cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh có thể diễn ra chỉ khi nó là trường hợp nằm trong chế độ thứ nhất 20, phép khởi tạo các xác suất ký hiệu trong chế độ thứ hai 22 là độc lập với bất kỳ lát đã được giải mã trước đó nào. Hơn nữa, việc giải mã dự đoán được thực hiện với giới hạn giải mã dự đoán để không vượt qua các biên lát.

Để dễ hiểu phần mô tả của Fig.24 và phần mô tả sau đây, tham chiếu đến Fig.25, Fig.25 thể hiện việc thực hiện có thể có của bộ giải mã 5 theo nghĩa cấu trúc hơn so với Fig.24. Giống như trường hợp trong Fig.24, bộ giải mã 5 là bộ giải mã dự đoán sử dụng bước giải mã entropy thích ứng ngữ cảnh để giải mã dòng dữ liệu để thu được, ví dụ, số dư dự đoán và các tham số dự đoán.

Như được thể hiện trên Fig.25, bộ giải mã 5 có thể bao gồm bộ giải mã entropy 24, môđun biến đổi ngược và hủy lượng tử hóa 26, bộ tổ hợp 28 được cung cấp, như được thể hiện trên Fig.25, làm thành bộ cộng và bộ dự đoán 28. Bộ giải mã entropy 24, môđun 26 và bộ cộng 27 được kết nối liên tiếp giữa đầu vào và đầu ra của bộ giải mã 5 theo thứ tự đề cập chúng, và bộ dự đoán 28 được kết nối giữa đầu ra của bộ cộng 28 và còn đầu vào của nó để tạo ra vòng dự đoán cùng với bộ tổ hợp 27. Do đó, bộ giải mã 24 kết nối thêm ở đầu ra của nó với đầu vào tham số mã hóa của bộ dự đoán 28.

Mặc dù Fig.25 cung cấp ấn tượng rằng bộ giải mã giải mã định kỳ bức ảnh hiện tại, bộ giải mã 5 có thể, ví dụ, được cung cấp để giải mã ảnh 10 theo cách song song. Bộ giải mã 5 có thể ví dụ gồm có nhiều lõi mà mỗi lõi hoạt động theo các chi tiết 24-28 trên Fig.25. Tuy nhiên, xử lý song song là tùy ý và bộ giải mã 5 hoạt động định kỳ cũng có thể giải mã dòng dữ liệu vào ở đầu vào của bộ giải mã entropy 24.

Để đạt được hiệu quả về khả năng giải mã song song hoặc định kỳ vừa nêu ra cho ảnh 10 hiện tại, bộ giải mã 5 hoạt động trong các đơn vị khối mã hóa 30 để giải mã ảnh 10. Các khối mã hóa 30, ví dụ, là các khối lá mà các khối cây mã hóa hoặc các khối mã hóa lớn nhất 32 được phân chia thành các khối lá này bằng cách phân chia

nhiều cây đệ quy như phân chia cây tứ phân. Các khối cây mã hóa 32, lần lượt, có thể được sắp đặt cách đều nhau trong các cột và các hàng để tạo ra sự phân chia đều đặn bức ảnh 10 thành các khối cây mã hóa 32 này. Trên Fig.25, các khối cây mã hóa 32 được thể hiện bằng các đường liên tục, trong khi các khối mã hóa 30 được thể hiện bằng các đường nét đứt. Với mục đích minh họa, chỉ một khối cây mã hóa 32 được thể hiện được phân chia thêm thành các khối mã hóa 30, trong khi các khối mã hóa 32 khác được thể hiện không được phân chia thêm để tạo ra trực tiếp khối mã hóa. Dòng dữ liệu 12 có thể gồm có phần cú pháp báo hiệu về việc làm thế nào ảnh 10 được phân chia thành các khối mã hóa 30.

Dòng dữ liệu 12 truyền, cho mỗi khối mã hóa 30, các phần tử cú pháp bộc lộ về việc làm thế nào các môđun 24 đến 28 khôi phục lại nội dung ảnh trong khối mã hóa 30 đó. Ví dụ, các phần tử cú pháp này gồm có:

1) Một cách tùy ý, phân chia dữ liệu còn phân chia thêm nữa khối mã hóa 30 thành các khối dự đoán;

2) Một cách tùy ý, phân chia dữ liệu còn phân chia thêm nữa khối mã hóa 30 thành các khối dư và/hoặc khối biến đổi;

3) Chế độ dự đoán báo hiệu về việc chế độ dự đoán nào được sử dụng để suy ra tín hiệu dự đoán cho khối mã hóa 30, trong đó độ chi tiết mà tại đó chế độ dự đoán này được báo hiệu có thể phụ thuộc vào các khối mã hóa 30 và/hoặc khối dự đoán;

4) Các tham số dự đoán có thể được báo hiệu trên mỗi khối mã hóa hoặc, nếu có mặt, trên mỗi khối dự đoán với một kiểu tham số dự đoán được gửi đi phụ thuộc, ví dụ vào chế độ dự đoán. Các chế độ dự đoán có thể có, ví dụ có thể gồm có dưới dự đoán và/hoặc liên dự đoán;

5) Các phần tử cú pháp khác cũng có thể xuất hiện như thông tin lọc để lọc bức ảnh 10 ở khối mã hóa 30 để thu được tín hiệu dự đoán và/hoặc tín hiệu được khôi phục được sao chép;

6) Cuối cùng, thông tin số dư dưới dạng, không kể những cái khác, các hệ số biến đổi có thể được chứa trong dòng dữ liệu cho các khối mã hóa 30; trong các đơn vị khối

dư, dữ liệu dư có thể được báo hiệu; trên mỗi khối dư, sự phân tích phổ có thể, ví dụ, được thực hiện trong các đơn vị khối biến đổi đã nêu, nếu xuất hiện.

Bộ giải mã entropy 24 có thể đáp lại để thu được các phần tử cú pháp vừa nêu từ dòng dữ liệu. Để đạt được điều này, bộ giải mã entropy 24 sử dụng bước giải mã entropy thích ứng ngữ cảnh. Tức là, bộ giải mã entropy 24 cung cấp một vài ngữ cảnh. Để suy ra phần tử cú pháp nào đó từ dòng dữ liệu 12, bộ giải mã entropy 24 lựa chọn một ngữ cảnh nào đó trong số các ngữ cảnh có thể có. Sự lựa chọn trong số các ngữ cảnh có thể có được thực hiện phụ thuộc thuộc tính của vùng lân cận của một phần bức ảnh 10 mà phần tử cú pháp hiện tại thuộc về. Đối với mỗi ngữ cảnh có thể có, bộ giải mã entropy 24 quản lý các xác suất ký hiệu, cụ thể, ước lượng xác suất đối với mỗi ký hiệu có thể có của bảng chữ cái ký hiệu mà bộ giải mã entropy 24 hoạt động dựa trên bảng chữ cái ký hiệu này. "Quản lý" bao gồm cập nhật liên tục các xác suất ký hiệu đã nêu trên của các ngữ cảnh để thích ứng các xác suất ký hiệu được kết hợp với mỗi ngữ cảnh cho nội dung ảnh thực. Bằng cách này, các xác suất ký hiệu được thích ứng với các số liệu thống kê xác suất thực của các ký hiệu.

Trường hợp khác trong đó các thuộc tính của vùng lân cận ảnh hưởng đến sự khôi phục một phần hiện tại của ảnh 10 như khối mã hóa hiện tại 30, là giải mã dự đoán trong bộ dự đoán 28. Sự dự đoán bị giới hạn không chỉ đối với nội dung dự đoán trong khối mã hóa hiện tại 30, mà còn có thể xoay quanh phép dự đoán các tham số được chứa trong dòng dữ liệu 12 cho khối mã hóa hiện tại 30 như các tham số dự đoán, dữ liệu phân chia, hoặc thậm chí các hệ số biến đổi. Tức là, bộ dự đoán 28 có thể dự đoán nội dung ảnh hoặc các tham số này từ vùng phụ cận đã nêu trên để thu được tín hiệu thành văn bản mà tín hiệu này tiếp đó được kết hợp với phần dư dự đoán như thu được bởi môđun 26 từ dòng dữ liệu 12. Trong trường hợp dự đoán các tham số, bộ dự đoán 28 có thể sử dụng các phần tử cú pháp được chứa trong dòng dữ liệu như các phần dư dự đoán để thu được các giá trị thực tế của tham số dự đoán. Bộ dự đoán 28 sử dụng giá trị tham số dự đoán về sau để thu được tín hiệu dự đoán vừa nêu được kết hợp với phần dư dự đoán trong bộ tổ hợp 27.

"Vùng lân cận" nêu trên cơ bản bao phủ phần bên trái phía trên của chu vi phần hiện tại mà phần tử cú pháp hiện tại được giải mã entropy cho phần này hoặc phần tử

cú pháp được dự đoán hiện tại thuộc về. Trên Fig.25, vùng lân cận này được minh họa ở 34 minh họa cho một khối mã hóa 30.

Lệnh mã hóa/giải mã được định ra giữa các khối mã hóa 30: tại mức thô nhất, các khối cây mã hóa 32 của ảnh 10 được quét theo thứ tự quét 36, được minh họa là sự quét mành theo hàng từ trên cùng xuống dưới cùng. Trong mỗi khối cây mã hóa, các khối mã hóa 30 được quét theo thứ tự giao nhau thứ nhất theo chiều sâu, sao cho, trong mỗi mức thứ bậc, khối cây mã hóa 32 cơ bản được quét cũng theo sự quét mành theo hàng từ trên cùng xuống dưới cùng.

Thứ tự mã hóa được định ra giữa các khối mã hóa 30 hài hòa với sự định nghĩa vùng lân cận 34 được sử dụng để suy ra thuộc tính trong vùng lân cận để lựa chọn các ngữ cảnh và/hoặc thực hiện dự đoán không gian trong đó vùng lân cận 34 gần như bao phủ các phần của ảnh 10 mà ảnh này đã trải qua giải mã phù hợp với thứ tự mã hóa. Bất cứ khi nào một phần của vùng lân cận 34 bao phủ các phần không có sẵn của ảnh 10, dữ liệu mặc định được sử dụng. Ví dụ, khuôn mẫu vùng lân cận 34 có thể kéo dài ra phía ngoài bức ảnh 10. Tuy nhiên, một khả năng khác đó là vùng lân cận 34 kéo dài vào trong lát lân cận.

Các lát, ví dụ chia bức ảnh 10 dọc thứ tự mã hóa/giải mã được định ra dọc các khối mã hóa 30, cụ thể, mỗi lát là một chuỗi liên tục không bị gián đoạn của các khối mã hóa 30 dọc thứ tự khối mã hóa đã nêu trên. Trên Fig.25, các lát được chỉ ra bằng các đường chấm gạch 14. Thứ tự được định ra giữa các lát 14 là từ thành phần chạy của các khối mã hóa 30 kế tiếp như được chỉ ra bên trên. Nếu phần phần tử cú pháp 18 của lát 14 nào đó chỉ ra rằng lát được giải mã trong chế độ thứ nhất, thì bộ giải mã entropy 24 cho phép giải mã entropy thích ứng ngữ cảnh để suy ra các ngữ cảnh vượt qua các biên lát. Tức là, vùng lân cận trong không gian 34 được sử dụng để chọn các ngữ cảnh trong giải mã entropy dữ liệu đề cập đến lát hiện tại 14. Ví dụ, trong trường hợp của Fig.25, lát số 3 có thể là lát đã được giải mã hiện tại, và trong phần tử cú pháp giải mã entropy liên quan đến khối mã hóa 30 hoặc phần nào đó được chứa trong đó, bộ giải mã entropy 24 có thể sử dụng các thuộc tính xuất phát từ các phần giải mã nằm trong lát lân cận như lát số 1. Bộ dự đoán 28 hoạt động tương tự: cho các lát của

chế độ thứ nhất 20, bộ dự đoán 28 sử dụng sự dự đoán không gian vượt qua biên lát bao quanh lát hiện tại.

Tuy nhiên, đối với các lát có chế độ thứ hai 22 được kết hợp với các lát đó, cụ thể, phần phần tử cú pháp 18 cho các lát này chỉ ra chế độ thứ hai 22, bộ giải mã entropy 24 và bộ dự đoán 28 giới hạn việc suy ra các ngữ cảnh entropy và giải mã dự đoán phụ thuộc vào các thuộc tính liên quan đến các phần chỉ nằm trong lát hiện tại. Rõ ràng, hiệu suất mã hóa chịu sự hạn chế này. Mặt khác, các lát của chế độ thứ hai 22 cho phép phá vỡ các sự liên phụ thuộc giữa các chuỗi lát. Do đó, các lát của chế độ thứ hai 22 có thể được đặt rải rác trong bức ảnh 10 hoặc trong video mà bức ảnh 10 thuộc về để cho phép các điểm đồng bộ hóa lại. Tuy nhiên, không cần thiết là mỗi bức ảnh 10 có ít nhất một lát trong chế độ thứ hai 22.

Như đã đề cập bên trên, chế độ thứ nhất và thứ hai 20 và 22 cũng khác nhau về sự khởi tạo các xác suất ký hiệu của chúng. Các lát được mã hóa trong chế độ thứ hai 22, thu được trong bộ giải mã entropy 24 khởi tạo lại các xác suất độc lập của lát đã được giải mã trước bất kỳ, cụ thể, được giải mã trước theo thứ tự được định ra giữa các lát. Các xác suất ký hiệu, ví dụ, được thiết lập đến các giá trị mặc định đã được biết đến cho cả phía bộ giải mã và bộ mã hóa, hoặc các giá trị khởi tạo được chứa trong các lát đã được mã hóa trong chế độ thứ hai 22.

Tức là, đối với các lát được mã hóa/được giải mã trong chế độ thứ hai 22, sự thích ứng của các xác suất ký hiệu luôn luôn bắt đầu ngay từ lúc bắt đầu các lát này. Do đó, độ chính xác thích ứng là tồi đối với các lát này tại lúc bắt đầu các lát này.

Mọi điều là khác nhau trong các lát được mã hóa/giải mã trong chế độ thứ nhất 20. Đối với các lát sau, phép khởi tạo các xác suất ký hiệu được thực hiện bởi bộ giải mã entropy 24 phụ thuộc vào các trạng thái được lưu của các xác suất ký hiệu của lát đã được giải mã trước đó. Bất cứ lúc nào lát được mã hóa/giải mã trong chế độ thứ nhất 20, có phần đầu của lát, ví dụ được định vị khác với phía bên trái của bức ảnh 10, cụ thể không ở phía mà từ phía đó sự quét mành 36 bắt đầu chạy theo hàng trước khi tạo bậc tới hàng kế tiếp ở phía dưới, các xác suất ký hiệu như thu được tại lúc kết thúc giải mã entropy, lát ngay trước đó được chấp nhận. Điều này được minh họa, ví dụ, trên Fig.2 bằng mũi tên 38 cho lát số 4. Lát số 4 có phần bắt đầu của nó đâu đó nằm

giữa phía bên phải và phía bên trái của ảnh 10 và do đó, trong bước khởi tạo các xác suất ký hiệu, bộ giải mã entropy 24 chấp nhận, trong bước khởi tạo các xác suất ký hiệu, các xác suất ký hiệu mà được thu trong việc giải mã entropy lát ngay trước đó, cụ thể lát số 3, cho đến khi kết thúc lát, cụ thể, bao gồm cập nhật liên tục các xác suất ký hiệu trong lúc giải mã entropy lát số 3 cho đến đầu kết thúc của nó.

Các lát có chế độ thứ hai 22 được kết hợp với nó, tuy nhiên, các lát này có tại lúc bắt đầu tại phía bên trái của bức ảnh 10, ví dụ như, lát số 5 không thích ứng với các xác suất ký hiệu như thu được sau khi kết thúc giải mã entropy lát số 4 ngay trước đó, bởi vì nó ngăn bộ giải mã 5 không giải mã song song ảnh 10 bằng cách sử dụng xử lý mặt đầu sóng. Hơn nữa, như được chỉ ra bên trên, bộ giải mã entropy 24 thích ứng các xác suất ký hiệu như đã thu được sau khi kết thúc giải mã entropy khối cây mã hóa 32 – theo thứ tự mã hóa/giải mã 36 – thứ hai trong hàng khối cây giải mã – theo thứ tự mã hóa/giải mã 36 – ngay trước đó như được minh họa bởi mũi tên 40.

Trên Fig.25, ví dụ, ảnh 10 được phân chia thành ba hàng khối cây mã hóa và bốn cột khối rễ cây mã hóa 32 và mỗi hàng khối cây mã hóa được chia nhỏ thành hai lát 14, sao cho phần bắt đầu của từng lát thứ hai trùng với đơn vị mã hóa thứ nhất theo thứ tự đơn vị mã hóa của hàng khối rễ cây mã hóa tương ứng. Bộ giải mã entropy 24, do đó có thể sử dụng xử lý mặt đầu sóng trong việc giải mã ảnh 10, bằng cách giải mã từng hàng khối rễ cây mã hóa song song với việc bắt đầu giải mã các hàng khối rễ cây mã hóa này theo cách thức so le, bắt đầu từ hàng khối rễ cây mã hóa thứ nhất đến cao nhất, tiếp đó đến thứ bai, và rồi đến thứ ba.

Tất nhiên, sự phân chia các khối 32 theo cách thức đệ quy thành các khối mã hóa thêm nữa 30 là tùy ý và do đó, theo nghĩa chung hơn, các khối 32 cũng có thể được gọi là "các khối mã hóa". Tức là, nói chung hơn, ảnh 10 có thể được phân chia thành các khối mã hóa 32 được sắp đặt trong các hàng và các cột và có thứ tự quét mành 36 được định ra giữa các khối mã hóa với nhau, và bộ giải mã 5 có thể được xem xét để kết hợp mỗi lát 14 với tập con liên tục của các khối mã hóa 32 theo thứ tự quét mành 36 do đó các tập con đi theo nhau dọc thứ tự quét mành 36 theo thứ tự lát.

Như đã trở nên rõ ràng từ phần thảo luận bên trên, bộ giải mã 5 hoặc, cụ thể hơn là bộ giải mã entropy 24 có thể được tạo cấu hình để lưu các xác suất ký hiệu như thu

được trong việc giải mã entropy thích ứng ngữ cảnh lát bất kỳ đến khối mã hóa thứ hai trong hàng khối mã hóa theo thứ tự quét mành 36. Để khởi tạo các xác suất ký hiệu cho việc giải mã entropy thích ứng ngữ cảnh lát hiện tại có chế độ thứ nhất 20 được kết hợp với nó, bộ giải mã 5, hoặc, cụ thể hơn, bộ giải mã entropy 24, kiểm tra xem liệu khối mã hóa thứ nhất 32 của một tập con liên tục các khối mã hóa 32 được kết hợp với lát hiện tại là khối mã hóa thứ nhất 32 trong hàng khối mã hóa theo thứ tự quét mành 36. Nếu vậy, các xác suất ký hiệu cho việc giải mã entropy thích ứng ngữ cảnh của lát hiện tại được khởi tạo như được giải thích đối với hàng 40, cụ thể phụ thuộc vào các xác suất ký hiệu được lưu như thu được trong giải mã entropy ngữ cảnh lát đã giải mã trước đến khối mã hóa thứ hai trong hàng khối mã hóa theo thứ tự quét mành 36. Nếu không, phép khởi tạo các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại được thực hiện phụ thuộc vào các xác suất ký hiệu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát được giải mã trước đó đến phần kết thúc lát đã được giải mã trước đó, cụ thể, theo mũi tên 38. Một lần nữa, trong trường hợp khởi tạo theo 38, trạng thái đã lưu tại lúc kết thúc giải mã entropy, lát ngay trước đó trong thứ tự lát 36 được dự định trong khi trong trường hợp khởi tạo 40, thì lát đã được giải mã trước gồm phần kết thúc khối thứ hai của hàng ngay trước đó của khối 32 theo thứ tự khối 36.

Như được minh họa bởi đường nét đứt trên Fig.24, bộ giải mã có thể được tạo cấu hình để đáp lại phần phần tử cú pháp 18 nằm trong lát hiện tại của các lát 14 để giải mã lát hiện tại theo một trong ít nhất ba chế độ. Tức là, có thể là chế độ thứ ba 42 ngoài các chế độ 20 và 22. Chế độ thứ ba 42 có thể khác với chế độ thứ hai 22 ở chỗ dự đoán vượt qua các biên được cho phép, trong đó việc mã hóa/giải mã entropy vẫn được giới hạn để không vượt qua các biên lát.

Bên trên, hai phương án được đưa ra đối với phần phần tử cú pháp 18. Bảng bên dưới tổng kết hai phương án này.

	Phương án 1	Phương án 2
Phần phần tử cú pháp	dependent_slice_flag, no_cabac_reset_flag	dependent_slice_flag

Chế độ 1	dependent_slice_flag = 1, no_cabac_reset_flag = 1	dependent_slice_flag = 1,
Chế độ 2	dependent_slice_flag = 0	dependent_slice_flag = 0
Chế độ 3	dependent_slice_flag = 1, no_cabac_reset_flag = 0	dependent_slice_flag = 1, cabac_independent_flag = 1, tiles_or_entropy_coding_sync_idc = 3

Theo một phương án, phần phần tử cú pháp 18 được tạo ra bởi riêng dependent_slice_flag, trong khi theo phương án khác, sự kết hợp của dependent_slice_flag và no_cabac_reset_flag tạo ra phần phần tử cú pháp. Xem xét quá trình đồng bộ hóa các biến ngữ cảnh theo sự khởi tạo các xác suất ký hiệu phụ thuộc vào các trạng thái được lưu của các xác suất ký hiệu của lát đã được giải mã trước đó được đề cập. Cụ thể, bộ giải mã có thể được tạo cấu hình để, nếu last_ctb_cabac_init_flag=0 và tiles_or_entropy_coding_sync_idc=2, lưu các xác suất ký hiệu như được thu trong việc giải mã entropy thích ứng ngữ cảnh, lát đã được giải mã trước đó đến khối giải mã thứ hai trong hàng theo thứ tự quét mành, và, trong việc khởi tạo các xác suất ký hiệu để giải mã entropy thích ứng ngữ cảnh lát hiện tại theo chế độ thứ nhất, kiểm tra xem liệu khối mã hóa thứ nhất của tập con liên tục của các khối mã hóa đã gắn với lát hiện tại là khối mã hóa thứ nhất trong hàng theo thứ tự quét mành hay chưa, và, nếu có, khởi tạo các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác xuất ký hiệu đã lưu như được thu trong việc giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đến khối giải mã thứ hai trong hàng theo thứ tự quét mành, và nếu không, khởi tạo các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu như thu được trong việc giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đến lúc kết thúc lát đã được giải mã trước.

Do đó, nói cách khác, theo phương án thứ hai đối với cú pháp, bộ giải mã sẽ khôi phục ảnh 10 từ dòng dữ liệu 12 mà do ảnh được mã hóa trong các đơn vị lát 14 thành dòng dữ liệu mà ảnh 10 được phân chia thành các đơn vị lát này, trong đó bộ giải mã được tạo cấu hình để giải mã các lát 14 từ dòng dữ liệu 12 theo thứ tự lát 16 và bộ giải mã đáp lại phần phần tử cú pháp 18, cụ thể dependent_slice_flag nằm trong lát hiện tại của các lát, để giải mã lát hiện tại theo một trong ít nhất hai chế độ 20, 22. Theo chế độ thứ nhất 20 trong ít nhất hai chế độ, cụ thể nếu dependent_slice_flag=1, bộ giải mã giải mã lát hiện tại từ dòng dữ liệu 12 sử dụng sự giải mã entropy thích ứng ngữ cảnh 24 bao gồm suy ra ngữ cảnh vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của ngữ cảnh và khởi tạo 38, 40 các xác suất ký hiệu phụ thuộc vào các trạng thái được lưu của các xác suất ký hiệu của lát đã được mã hóa trước, và giải mã dự đoán vượt qua các biên lát, và theo chế độ thứ hai 22 trong ít nhất hai chế độ, cụ thể dependent_slice_flag=0, bộ giải mã giải mã lát hiện tại từ dòng dữ liệu 12 sử dụng sự giải mã entropy thích ứng ngữ cảnh với giới hạn suy ra ngữ cảnh để không vượt qua các biên lát, sự cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập với lát đã được mã hóa trước bất kỳ, và giải mã dự đoán với giới hạn sự giải mã dự đoán để không vượt qua các biên lát. Ảnh 10 có thể được phân chia trong các khối mã hóa 32 được đặt trong các hàng và cột và có thứ tự quét mành 36 được định ra giữa các hàng và cột với nhau, và bộ giải mã được tạo cấu hình để kết hợp mỗi lát 14 với tập con liên tục của các khối mã hóa 32 theo thứ tự quét mành 36 sao cho các tập con đi theo nhau dọc thứ tự quét mành 36 phù hợp với thứ tự lát. Bộ giải mã có thể được tạo cấu hình để, cụ thể, đáp ứng với tiles_or_entropy_coding_sync_idc=2, lưu các xác suất ký hiệu thu được trong việc giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đến khối mã hóa thứ hai 32 trong hàng theo thứ tự quét mành 36, và, trong khởi tạo các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh lát hiện tại theo chế độ thứ nhất, kiểm tra xem liệu khối mã hóa thứ nhất của tập con liên tục gồm các khối giải mã 32 đã kết hợp với lát hiện tại là khối giải mã thứ nhất 32 trong hàng phù hợp với thứ tự quét mành hay chưa, và nếu có, khởi tạo 40 các xác suất ký hiệu để giải mã entropy thích ứng ngữ cảnh lát hiện tại phụ thuộc vào các xác suất ký hiệu đã được lưu như thu được trong giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đó đến khối mã hóa thứ hai trong

hàng theo thứ tự quét mành 36, và nếu không, khởi tạo 38 các xác suất ký hiệu cho giải mã entropy thích ứng ngữ cảnh lát hiện tại phụ thuộc vào các xác suất ký hiệu như thu được trong giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đến lúc kết thúc lát đã được giải mã trước đó. Bộ giải mã có thể được tạo cấu hình để đáp lại phần phần tử cú pháp (18) nằm trong lát hiện tại của các lát 14, để giải mã lát hiện tại theo một trong ít nhất ba chế độ, cụ thể trong một trong chế độ thứ nhất 20 và chế độ thứ ba 42 hoặc chế độ thứ hai 22, trong đó bộ giải mã được tạo cấu hình để, theo chế độ thứ ba 42, cụ thể nếu dependent_slice_flag=1 và tiles_or_entropy_coding_sync_idc=3, giải mã lát hiện tại từ dòng dữ liệu sử dụng giải mã entropy thích ứng ngữ cảnh với việc giới hạn sự suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập với lát đã được giải mã hóa trước bất kỳ, và giải mã dự đoán vượt qua các biên lát, trong đó một trong các chế độ thứ nhất và thứ ba được lựa chọn phụ thuộc vào phần tử cú pháp, cụ thể cabac_independent_flag. Bộ giải mã có thể còn được tạo cấu hình để, cụ thể nếu tiles_or_entropy_coding_sync_idc=0,1, và 3 ("3" khi cabac_independent flag=0), lưu các xác suất ký hiệu như thu được trong việc giải mã entropy thích ứng cảnh lát đã được giải mã trước đến khi kết thúc lát đã được giải mã hóa trước đó, và, trong khởi tạo các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại theo chế độ thứ nhất, khởi tạo các xác suất ký hiệu để giải mã entropy thích ứng ngữ cảnh lát hiện tại phụ thuộc vào các xác suất ký hiệu đã được lưu. Bộ giải mã có thể được tạo cấu hình để, cụ thể nếu tiles_or_entropy_coding_sync_idc=1, trong chế độ thứ nhất và thứ hai, giới hạn giải mã dự đoán trong các ô mà ảnh được chia nhỏ thành các ô này.

Tất nhiên, bộ mã hóa có thể thiết lập cú pháp đã thể hiện bên trên để cho phép bộ giải mã đạt được các thuận lợi đã được chỉ ra bên trên. Bộ mã hóa có thể xử lý song song, như bộ mã hóa đa lõi nhưng không cần sau này. Để mã hóa ảnh 10 thành dòng dữ liệu 12 trong các đơn vị của các lát 14, bộ mã hóa được tạo cấu hình để mã hóa các lát 14 thành dòng dữ liệu 12 theo thứ tự lát 16. Bộ mã hóa sẽ xác định phần phần tử cú pháp 18 cho, và mã hóa phần phần tử cú pháp thành, lát hiện tại trong các lát sao cho phần phần tử cú pháp báo hiệu lát hiện tại được mã hóa theo một trong ít nhất hai chế

độ 20, 22, và nếu lát hiện tại được mã hóa theo chế độ thứ nhất 20 trong ít nhất hai chế độ, mã hóa lát hiện tại thành dòng dữ liệu 12 sử dụng sự mã hóa entropy thích ứng ngữ cảnh 24 bao gồm việc suy ra các ngữ cảnh vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo 38, 40 các xác suất kí hiệu phụ thuộc vào các trạng thái được lưu của các xác suất ngữ cảnh của lát đã được mã hóa trước, và mã hóa dự đoán vượt qua các biên lát, và nếu lát hiện tại được mã hóa theo chế độ thứ hai 22 trong ít nhất hai chế độ, mã hóa lát hiện tại thành dòng dữ liệu 12 sử dụng mã hóa entropy thích ứng ngữ cảnh với việc giới hạn suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập về lát đã được mã hóa trước bất kỳ, và mã hóa dự đoán với giới hạn mã hóa dự đoán để không vượt qua các biên lát. Trong khi ảnh 10 có thể được phân chia trong các khối mã hóa 32 được sắp đặt theo các hàng và cột và có thứ tự quét mành 36 được định ra giữa các hàng và cột, bộ mã hóa có thể được tạo cấu hình để kết hợp mỗi lát 14 với một tập con liên tục các khối mã hóa 32 theo thứ tự quét mành 36 sao cho các tập con đi theo nhau dọc theo thứ tự quét mành 36 theo thứ tự lát. Bộ mã hóa có thể được tạo cấu hình để lưu các xác suất ký hiệu như được thu trong mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đến khối mã hóa thứ hai 32 trong hàng theo thứ tự quét mành 36, và, trong khởi tạo các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh của lát hiện tại theo chế độ thứ nhất, kiểm tra xem liệu khối mã hóa thứ nhất của tập con liên tục của các khối mã hóa 32 có được liên kết với lát hiện tại là khối mã hóa thứ nhất 32 trong hàng theo thứ tự quét mành hay không, và nếu có, khởi tạo 40 các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu được lưu như được thu trong mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đến khối mã hóa thứ hai trong hàng theo thứ tự quét mành 36, và, nếu không, khởi tạo 38 các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu như được thu trong bước mã hóa entropy thích ứng ngữ cảnh lát đã được giải mã trước đến lúc kết thúc lát đã được mã hóa trước. Bộ mã hóa có thể được tạo cấu hình để mã hóa phần phần tử cú pháp 18 thành lát hiện tại trong các lát 14 sao cho lát hiện tại được báo hiệu được mã hóa thành tại đó theo một trong ít nhất ba chế độ, cụ thể trong một trong chế độ thứ nhất (20) và chế độ thứ ba (42) hoặc chế

độ thứ hai (22), trong đó bộ mã hóa được tạo cấu hình để ** theo chế độ thứ ba (42), mã hóa lát hiện tại thành dòng dữ liệu sử dụng mã hóa entropy thích ứng ngữ cảnh cùng với việc giới hạn suy ra các ngữ cảnh để không đi vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập về lát đã được mã hóa trước bất kỳ, và mã hóa dự đoán vượt qua các biên lát, trong đó bộ mã hóa phân biệt giữa một trong các chế độ thứ nhất và thứ ba sử dụng phần tử cú pháp, cụ thể ví dụ là cabac_independent_flag. Bộ mã hóa có thể được tạo cấu hình để xác định phần tử cú pháp chung như dependent_slices_present_flag và được viết giống nhau thành dòng dữ liệu với việc vận hành trong một trong ít nhất hai chế độ hoạt động chung phụ thuộc vào phần tử cú pháp chung, cụ thể, với , theo chế độ vận hành chung thứ nhất, thực hiện mã hóa phần phần tử cú pháp cho mỗi lát, và theo chế độ vận hành chung thứ hai, sử dụng một cách chắc chắn một chế độ khác trong ít nhất hai chế độ khác với chế độ thứ nhất. Bộ mã hóa có thể được tạo cấu hình theo các chế độ thứ nhất và thứ hai, tiếp tục một cách chắc chắn và liên tục cập nhật các xác suất ký hiệu từ lúc bắt đầu đến khi kết thúc lát hiện tại. Bộ mã hóa có thể được tạo cấu hình để lưu các xác suất ký hiệu như đã thu được trong mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đến lúc kết thúc lát đã được mã hóa trước, và, trong khởi tạo các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh lát hiện tại theo chế độ thứ nhất, khởi tạo các xác suất ký hiệu cho sự mã hóa entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu đã được lưu. Và bộ mã hóa có thể, ví dụ, trong chế độ thứ nhất và thứ hai, giới hạn sự mã hóa dự đoán trong các ô mà ảnh được chia nhỏ thành các ô này.

Cấu trúc có thể có của bộ mã hóa được mô tả trên Fig.26 cho mục đích trọn vẹn. Bộ dự đoán 70 hoạt động gần như giống bộ dự đoán 28, cụ thể, thực hiện dự đoán, và cũng xác định, bằng cách tối ưu hóa, ví dụ, các tham số mã hóa bao gồm các tham số và chế độ dự đoán. Các môđun 26 và 27 cũng xuất hiện trong bộ giải mã. Bộ trừ 72 xác định số dư dự đoán không tổn hao mà số dư này tiếp đó, là tổn hao – bởi việc sử dụng sự lượng tử hóa và, tùy ý, sử dụng sự biến đổi phân tích phổ - được mã hóa trong môđun lượng tử hóa và biến đổi 74. Bộ mã hóa entropy 76 thực hiện mã hóa entropy thích ứng ngữ cảnh.

Bên cạnh các ví dụ cú pháp cụ thể bên trên, ví dụ khác được chỉ ra bên dưới với việc thể hiện sự phù hợp giữa các thuật ngữ được sử dụng dưới đây và các thuật ngữ được sử dụng bên trên.

Cụ thể, mà chưa được chỉ ra cụ thể bên trên, các lát phụ thuộc không chỉ "phụ thuộc" ở chỗ cùng cho phép khai thác kiến thức đã được biết đến từ bên ngoài biên của nó, như được chỉ ra bên trên, có các ngữ cảnh entropy được thích ứng nhanh hơn, hoặc đạt được dự đoán không gian tốt hơn do cho phép vượt qua biên của nó. Hơn nữa, để tiết kiệm chi phí tốc độ đã được tiêu dùng để định ra các đoạn đầu lát bằng cách chia tách ảnh thành các lát, các lát phụ thuộc chấp nhận một phần cú pháp đoạn đầu lát từ các lát trước, cụ thể, phần đoạn đầu cú pháp lát này không được truyền lại cho các lát phụ thuộc. Điều này được thể hiện ví dụ, trên Fig.16 ở 100 và trên Fig.21 ở 102, mà theo đó loại lát, ví dụ, được chấp nhận từ lát trước. Bằng cách thức này, sự chia nhỏ bức ảnh thành các lát, như lát độc lập và các lát phụ thuộc, ít tốn kém hơn về mặt tiêu thụ bit đắt tiền.

Nó là tính phụ thuộc như vừa đề cập dẫn đến ví dụ được chỉ ra bên dưới, cho cách diễn đạt hơi khác: các lát được định ra thành các phần đơn vị của bức ảnh mà tại đó cú pháp đoạn đầu lát có thể thiết lập riêng biệt. Do đó, các lát gồm có một – sử dụng phép đặt tên gọi phía trên – lát độc lập/đều/thông thường, giờ được gọi là đoạn lát độc lập và không, một hoặc nhiều hơn – sử dụng phép đặt tên gọi phía trên – các lát phụ thuộc, giờ được gọi là các đoạn lát phụ thuộc.

Fig.27, ví dụ, thể hiện ảnh được phân chia thành hai lát, một lát được tạo ra bởi các đoạn lát 14_1 đến 14_3 và đoạn kia chỉ được tạo ra bởi đoạn lát 14_4. Các chỉ số 1 đến 4 thể hiện thứ tự lát trong thứ tự mã hóa. Fig.28a và b thể hiện ví dụ khác trong trường hợp chia nhỏ bức ảnh 10 thành hai lát, với, trong trường hợp Fig.28a, một lát được tạo ra bởi tất cả năm đoạn lát 14, bao phủ cả hai ô 50_1 và 50_2 – chỉ số lại tăng lên theo thứ tự mã hóa, và trong trường hợp của Fig.28a, hai lát lần lượt được tạo ra bởi các đoạn lát 14_1 và 14_2 và 14_3 và 14_4, chia nhỏ ô 50_1, và lát khác được tạo ra bởi các đoạn lát 14_5-14_6 bao phủ ô 50_2.

Sau đây là các định nghĩa:

Đoạn lát phụ thuộc: đoạn lát mà các giá trị của một số phần tử cú pháp của đoạn đầu đoạn lát cho đoạn lát này được suy ra từ các giá trị cho đoạn lát độc lập trước đó theo thứ tự giải mã – trước đó, trong các phương án bên trên được gọi là lát phụ thuộc.

Đoạn lát độc lập: đoạn lát mà các giá trị của các phần tử cú pháp của đoạn đầu lát cho đoạn lát này không được suy ra từ các giá trị cho đoạn lát trước – trước đây, trong các phương án bên trên được gọi là lát chuẩn.

Lát: một lượng số nguyên các đơn vị cây mã hóa được chứa trong một đoạn lát độc lập và tất cả các đoạn lát phụ thuộc tiếp theo (nếu có) đi trước đoạn lát độc lập kế tiếp (nếu có) trong cùng đơn vị truy cập/ảnh.

Đoạn đầu lát: đoạn đầu đoạn lát của đoạn lát độc lập là đoạn lát hiện tại hoặc là đoạn lát độc lập đi trước đoạn lát phụ thuộc hiện tại.

Đoạn lát: một lượng số nguyên các đơn vị cây mã hóa được sắp xếp liên tiếp trong quét ô và được chứa trong một đơn vị NAL đơn lẻ, việc chia từng bức ảnh thành các đoạn lát được gọi là sự phân chia.

Đoạn đầu đoạn lát: một phần của đoạn lát đã được mã hóa chứa các chi tiết dữ liệu đề cập đến các đơn vị cây mã hóa thứ nhất hoặc tất cả các đơn vị cây mã hóa được mô tả trong đoạn lát.

Sự báo hiệu của "các chế độ" 20 và 22, cụ thể "đoạn lát phụ thuộc" và "đoạn lát độc lập" có thể là như sau:

Trong một số đơn vị NAL thêm như PPS, phần tử cú pháp có thể được sử dụng để báo hiệu về việc liệu việc sử dụng các lát phụ thuộc được tạo ra hoặc không đối với một bức ảnh nhất định của một chuỗi cho các bức ảnh nhất định:

dependent_slice_segments_enabled_flag = 1 chỉ ra sự có mặt của phần tử cú pháp dependent_slice_segment_flag trong các đoạn đầu đoạn lát. dependent_slice_segments_enabled_flag = 0 chỉ ra sự vắng mặt của phần tử cú pháp dependent_slice_segment_flag trong các đoạn đầu đoạn lát.

dependent_slice_segments_enabled_flag là tương tự trong phạm vi với dependent_slices_present_flag đã mô tả trước đó.

Tương tự, dependent_slice_flag có thể được gọi là dependent_slice_segment_flag để tính đến danh pháp khác cho các lát.

dependent_slice_segment_flag = 1 chỉ ra rằng giá trị của từng phần tử cú pháp đoạn đầu lát là không xuất hiện trong đoạn đầu của đoạn lát hiện tại được suy ra là bằng với giá trị của phần tử cú pháp đoạn đầu lát tương ứng trong đoạn đầu lát, cụ thể, đoạn đầu đoạn lát của đoạn lát độc lập trước.

Trong cùng một mức, như mức ảnh, phần tử cú pháp sau đây có thể được kể đến:

entropy_coding_sync_enabled_flag = 1 chỉ ra rằng quá trình đồng bộ hóa riêng cho các biến ngữ cảnh được dẫn ra trước khi giải mã đơn vị cây mã hóa mà đơn vị cây mã hóa này bao gồm khối cây mã hóa thứ nhất của một hàng gồm các khối cây mã hóa trong từng ô trong từng bức ảnh đề cập đến PPS, và quá trình lưu trữ riêng cho các biến ngữ cảnh được dẫn ra sau khi giải mã đơn vị cây mã hóa mà đơn vị cây mã hóa này bao gồm khối cây mã hóa thứ hai của một hàng gồm các khối cây mã hóa trong từng ô trong từng bức ảnh đề cập đến PPS. entropy_coding_sync_enabled_flag = 0 chỉ ra rằng không có quá trình đồng bộ hóa riêng nào cho các biến ngữ cảnh được yêu cầu dẫn ra trước khi giải mã đơn vị cây mã hóa mà đơn vị cây mã hóa này bao gồm khối cây mã hóa thứ nhất của một hàng gồm các khối cây mã hóa trong từng ô trong từng bức ảnh đề cập đến PPS, và không có quá trình lưu trữ riêng nào cho các biến ngữ cảnh được yêu cầu dẫn ra sau khi giải mã đơn vị cây mã hóa mà đơn vị cây mã hóa này bao gồm khối cây mã hóa thứ hai của một hàng gồm các khối cây mã hóa trong từng ô trong từng bức ảnh đề cập đến PPS.

Yêu cầu tính tương thích dòng bit rằng giá trị của entropy_coding_sync_enabled_flag sẽ là như nhau đối với tất cả các PPS mà các PPS này được kích hoạt trong CVS.

Khi entropy_coding_sync_enabled_flag = 1 và khối cây mã hóa thứ nhất trong lát không phải là khối cây mã hóa thứ nhất của một hàng các khối cây mã hóa trong ô, yêu cầu tính tương thích dòng bit là khối cây mã hóa cuối cùng trong lát sẽ thuộc về cùng hàng các khối cây mã hóa như khối cây mã hóa thứ nhất trong lát.

Khi entropy_coding_sync_enabled_flag = 1 và khối cây mã hóa thứ nhất trong đoạn lát không phải là khối cây mã hóa thứ nhất của một hàng gồm các khối cây mã hóa trong ô, yêu cầu tính tương thích dòng bit là khối cây mã hóa cuối cùng trong đoạn lát sẽ thuộc về cùng hàng các khối cây mã hóa như khối cây mã hóa thứ nhất trong đoạn lát.

Như đã mô tả, thứ tự mã hóa/giải mã giữa các CTB 30 đưa đến theo thứ tự đường quét theo hàng từ đỉnh xuống đáy bắt đầu với việc quét ô thứ nhất sau đó tới ô kế tiếp, nếu nhiều hơn một ô xuất hiện trong bức ảnh.

Bộ giải mã 5 – và do đó là bộ mã hóa – hoạt động như sau trong giải mã (mã hóa) entropy các đoạn lát 14 của bức ảnh:

A1) Bất cứ lúc nào phần tử cú pháp được giải mã/mã hóa hiện tại synEl là phần tử cú pháp thứ nhất của ô 50, đoạn lát 14 hoặc hàng của CTB, quy trình khởi tạo của Fig.29 bắt đầu.

A2) Nếu không thì, giải mã phần tử cú pháp này diễn ra sử dụng các ngữ cảnh entropy hiện tại.

A3) Nếu phần tử cú pháp hiện tại là phần tử cú pháp cuối cùng trong CTB 30, thì quy trình lưu trữ ngữ cảnh entropy như được thể hiện trên Fig.30 được bắt đầu.

A4) Quy trình tiến hành ở A1) với phần tử cú pháp kế tiếp.

Trong quy trình khởi tạo, kiểm tra 200 xem liệu synEI có phải là phần tử cú pháp thứ nhất của đoạn lát 14 hoặc ô 50 hay không. Nếu có, các ngữ cảnh được khởi tạo độc lập với đoạn lát trước bất kỳ trong bước 202. Nếu không, kiểm tra 204 xem liệu synEI có phải là phần tử cú pháp thứ nhất của hàng các CTB 30 và entropy_coding_sync_enabled_flag =1 hay không. Nếu có, kiểm tra 206 xem liệu trong hàng CTB 30 trước của cùng ô, CTB 30 thứ hai có sẵn hay không (xem Fig.23). Nếu có, sự chấp nhận ngữ cảnh theo 40 được thực hiện trong bước 210 sử dụng các xác suất ngữ cảnh được lưu trữ hiện tại cho các sự chấp nhận loại 40. Nếu không, các ngữ cảnh được khởi tạo độc lập với đoạn lát trước bất kỳ trong bước 202. Nếu kiểm tra 204 bộc lộ không, thì được kiểm tra trong bước 212, liệu synEl có phải là phần tử cú pháp thứ nhất trong CTB thứ nhất của đoạn lát phụ thuộc 14 hay không và liệu

dependent_slice_segement_flag =1 không, và nếu có, sự chấp nhận ngữ cảnh theo 38 được thực hiện trong bước 214 sử dụng các xác suất ngữ cảnh được lưu trữ hiện tại cho các sự chấp nhận loại 38. Sau bất cứ các bước 214, 212, 210 và 202 nào, giải mã/mã hóa được bắt đầu một cách thực sự.

Các đoạn lát phụ thuộc với dependent_slice_segement_flag = 1, do đó giúp làm giảm thêm độ trễ mã hóa/giải mã với gần như không có bất lợi về hiệu suất mã hóa nào.

Trong quy trình lưu trữ của Fig.30, kiểm tra trong bước 300 xem liệu synEl đã được mã hóa/giải mã là phần tử cú pháp cuối cùng của CTB 30 thứ hai của hàng CTB 30 và entropy_coding_sync_enabled_flag =1 hay không. Nếu có, các ngữ cảnh entropy hiện tại được lưu trữ trong bước 302, cụ thể các xác suất mã hóa entropy của các ngữ cảnh, trong bộ lưu trữ là riêng biệt cho các sự chấp nhận 40. Tương tự, được kiểm tra trong bước 304, ngoài các bước 300 hoặc 302, liệu synEl đã được mã hóa/giải mã là phần tử cú pháp cuối của đoạn lát 14, và dependent_slice_segement_flag =1 hay không. Nếu có, các ngữ cảnh entropy hiện tại được lưu trữ trong bước 306, cụ thể, các xác suất mã hóa entropy của các ngữ cảnh, trong bộ lưu trữ là riêng biệt cho các sự chấp nhận 38.

Lưu ý rằng bất cứ sự truy vấn kiểm tra nào về việc liệu phần tử cú pháp là synEl thứ nhất của hàng CTB hay không, khai thác ví dụ, phần tử cú pháp slice_adress 400 nằm trong các đoạn đầu của đoạn lát, cụ thể, phần tử cú pháp khởi đầu bộc lộ vị trí của phần đầu của đoạn lát tương ứng dọc thứ tự giải mã.

Trong việc khôi phục ảnh 10 từ dòng dữ liệu 12 sử dụng sự xử lý WPP, bộ giải mã có thể khai thác một cách chính xác phần cú pháp khởi đầu gần đây 400 để lấy được các điểm vào dòng con WPP. Vì mỗi đoạn lát gồm có phần cú pháp khởi đầu 400 chỉ ra vị trí bắt đầu giải mã của đoạn lát tương ứng trong ảnh 10, bộ giải mã có thể nhận diện các điểm vào các dòng con WPP mà các đoạn lát được hợp thành nhóm trong các dòng con WPP này, bằng cách nhận diện, sử dụng các phần cú pháp khởi đầu 400 của đoạn lát, các đoạn lát bắt đầu ở phía bên trái của ảnh. Bộ giải mã có thể tiếp đó, giải mã song song các dòng con WPP theo cách thức so le với việc bắt đầu liên tiếp việc giải mã các dòng con WPP theo thứ tự lát. Các đoạn lát có thể thậm chí

là nhỏ hơn so với chiều rộng một bức ảnh, cụ thể, một hàng CTB, do đó việc truyền chúng có thể bị chèn vào giữa các dòng con WPP để làm giảm thêm toàn bộ độ trễ từ đầu đến cuối sự truyền. Bộ mã hóa cung cấp từng lát (14) với phần cú pháp khởi đầu 400 chỉ ra vị trí bắt đầu mã hóa của lát tương ứng trong ảnh (10) và nhóm các lát thành các dòng con WPP do đó đối với mỗi dòng con WPP, lát thứ nhất trong thứ tự lát bắt đầu ở phía bên trái của ảnh. Bộ mã hóa có thể thậm chí, chính nó sử dụng xử lý WPP trong việc mã hóa bức ảnh: bộ mã hóa mã hóa song song các dòng con WPP theo cách thức so le với việc bắt đầu liên tục việc mã hóa các dòng con WPP theo thứ tự lát.

Nhân đây, các khía cạnh sau của việc sử dụng các phần cú pháp khởi đầu của đoạn lát làm thành cách thức để định vị các điểm vào của dòng con WPP, có thể được sử dụng mà không cần khái niệm lát độc lập.

Có thể thực hiện cho tất cả để xử lý song song ảnh 10, bằng cách thiết lập các biến trên như sau:

	tiles_enabled_flag==0	tiles_enabled_flag==1	entropy_cooding_sync_enable_flag ==1	tiles_enabled_flag==1	entropy_cooding_sync_enable_flag ==1
dependent_slice_segment_flag	0/1	0/1	0/1	0/1	0/1
Sự chia nhỏ ảnh thành các ô	Không	có	Không	có	Không
Sự chia nhỏ lát với các điểm vào để chỉ ra sự bắt đầu dòng con WPP hoặc ô.	Không	có	có	Không	Không
Xử lý song song có thể có	Trong các lát	Trong các ô	WPP	Trong các ô	WPP

Thậm chí sẽ có tính khả thi để trộn WPP với phân chia ô. Trong trường hợp đó, người ta có thể xử lý các ô như những bức ảnh riêng lẻ: mỗi ô sử dụng WPP sẽ gồm một lát có một hoặc nhiều đoạn lát phụ thuộc, và việc kiểm tra trong bước 300 và 208 sẽ dựa vào CTB thứ hai trong hàng CTB nêu trên trong cùng ô, chỉ khi các bước 204 và A1 sẽ dựa vào CTB 30 thứ nhất trong hàng CTB 30 của ô hiện tại. Ttrường hợp đó, bảng trên có thể được mở rộng:

	tiles_enabled_flag	tiles_enabled_flag==1	entropy_cooding_sync_enable_flag ==1	tiles_enabled_flag==1	entropy_cooding_sync_enable_flag ==1	entropy_cooding_sync_enable_flag ==1	entropy_cooding_sync_enable_flag ==1
dependent_slice_segment_flag	0/1	0/1	0/1	0/1	0/1	0/1	0/1
Chia nhỏ ảnh thành các ô	Không	Có	Không	Có	Không	Có	Có
Chia nhỏ lát với các điểm vào để chỉ ra sự bắt đầu dòng con WPP	Không	Có	Có	Không	Không	Không (mỗi dòng con của ô trong lát phụ thuộc riêng của nó)	Có (mỗi ô trong lát riêng biệt và các điểm vào cho các dòng con WPP

hoặc ô.							trong lát của ô)
Xử lý song song có thể có	trong các lát	Trong các ô	WPP	Trong các ô	WPP	Các dòng con WPP trong các ô	Các dòng con WPP trong các ô

Lưu ý ngắn gọn, sự mở rộng về sau cũng là có thể có với phương án 2. Phương án 2 cho phép xử lý sau đây:

	Mức cú pháp				
tiles_or_entropy_coding_sync_idc	Trên mỗi bức ảnh	0	1	2	3
cabac_independent_flag	Mỗi lát	Không xuất hiện	Không xuất hiện	Không xuất hiện	0/1
dependent_slice_flag	Trên mỗi lát	0/1	0/1	0/1	0/1
Chia nhỏ ảnh thành các ô		không	có	không	không
Chia nhỏ lát với các điểm vào để chỉ ra sự bắt đầu dòng con WPP hoặc ô		không	có	có	không
Xử lý song song có thể có		Trong các lát	Trong các ô	WPP	Mã hóa entropy các lát

Nhưng với các sự mở rộng sau đây, bảng dưới đây sẽ đưa đến:

Bổ sung vào các ngữ nghĩa của tập hợp tham số ảnh:

Nếu tiles_or_entropy_coding_sync_idc = 4, mỗi hàng, nhưng hàng CTB thứ nhất sẽ được chứa trong lát khác với dependent_slice_flag = 1. Các CTB của các hàng khác nhau phải không được xuất hiện trong cùng một lát. Có thể có nhiều hơn một lát trên mỗi hàng CTB.

Nếu tiles_or_entropy_coding_sync_idc = 5, từng CTB , nhưng ô thứ nhất phải được chứa trong lát khác. Các CTB của các ô khác nhau phải không được xuất hiện trong cùng lát. Có thể có hơn 1 lát trên mỗi ô.

Xem Fig.31 để giải thích thêm.

Tức là, bảng bên trên có thể được mở rộng

	Mức cú pháp								
tiles_or_entropy_coding_sync_idc	Mỗi bức ảnh	0	1	2	3	5	4	6	7
cabac_independent_flag	Mỗi lát	Không xuất hiện	Không xuất hiện	Không xuất hiện	0/1	Không xuất hiện	Không xuất hiện	Không xuất hiện	Không xuất hiện
dependent_slice_flag	Mỗi lát	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
Chia nhỏ bức ảnh thành các ô		Không	Có	Không	Không	Có	Không	Có	Có
Chia nhỏ lát với các điểm vào để chỉ ra sự		Không	Có	Có	Khô	Khô	Khôn	Không (mỗi	Có

bắt đầu dòng con WPP hoặc ô.					ng	ng	g	dòng con của ô trong lát độc lập riêng của nó)	(mỗi ô trong lát riêng biệt và các điểm vào cho các dòng con WPP trong lát của ô)
Xử lý song song có thể sử dụng		Trong các lát	Trong các ô	WPP	Mã hóa entropy các lát	Trong các ô	WPP	Các dòng con WPP trong các ô	Các dòng con WPP trong các ô

Với các phương án nêu trên, cần lưu ý rằng bộ giải mã có thể được tạo cấu hình để, ví dụ, đáp lại tiles_or_entropy_coding_sync_idc=1,2, trong chế độ thứ nhất và thứ hai, đọc thông tin từ lát hiện tại bộc lộ sự chia nhỏ lát hiện tại thành các phần con song song, trong đó các phần con song song có thể là các dòng con WPP hoặc các ô, làm dừng sự giải mã entropy thích ứng ngữ cảnh tại lúc kết thúc phần con song song thứ nhất và bắt đầu lại giải mã entropy thích ứng ngữ cảnh một lần nữa tại lúc bắt đầu phần con song song kế tiếp bất kỳ bao gồm, trong chế độ thứ nhất, khởi tạo các xác

suất ký hiệu phụ thuộc vào các trạng thái đã lưu của các xác suất ký hiệu của phần con song song trước đó và, trong chế độ thứ hai, khởi tạo các xác suất ký hiệu độc lập với lát đã mã hóa trước bất kỳ và phần con song song đã giải mã trước bất kỳ.

Do đó, sự mô tả bên trên bộc lộ các phương pháp để mã hóa độ trễ thấp, giải mã, đóng gói và truyền dữ liệu video có cấu trúc như được cung cấp bởi tiêu chuẩn mã hóa HEVC, như được cấu trúc trong các ô, các dòng con xử lý song song mặt đầu sóng (WPP), các lát hoặc các lát entropy.

Đặc biệt, được định ra làm thế nào để vận chuyển dữ liệu đã được mã hóa song song trong kịch bản hội thoại để đạt tới độ trễ nhỏ nhất trong mã hóa, giải mã và quá trình truyền. Do đó, phương pháp mã hóa song song dạng đường liên hợp, truyền và giải mã để cho phép các ứng dụng độ trễ nhỏ nhất như trò chơi, phẫu thuật từ xa, v.v.

Hơn nữa, các phương án nêu trên đóng khe hở của xử lý song song mặt đầu sóng làm cho nó có thể sử dụng trong các kịch bản truyền độ trễ thấp. Do đó, định dạng đóng gói mới cho các dòng con WPP 0 được thể hiện, lát phụ thuộc. Lát phụ thuộc này có thể chứa dữ liệu lát entropy, dòng con WPP, hàng đầy đủ của các LCU, chỉ một đoạn của lát, trong đó đoạn đầu lát đã truyền trước đó cũng áp dụng cho dữ liệu đoạn được chứa. Dữ liệu được chứa được báo hiệu trong đoạn đầu lát con.

Cuối cùng lưu ý rằng, đặt tên các lát mới cũng có thể là "tập con / các lát trọng số nhẹ", nhưng tên "lát phụ thuộc" được thấy là tốt hơn.

Sự báo hiệu được thể hiện mô tả mức song song hóa trong mã hóa và vận chuyển.

Mặc dù một số khía cạnh được mô tả trong ngữ cảnh của thiết bị, rõ ràng rằng các khía cạnh này cũng thể hiện sự mô tả phương pháp tương ứng, trong đó khối hoặc thiết bị tương ứng với bước phương pháp hoặc đặc điểm của bước phương pháp. Tương tự, các khía cạnh được mô tả trong ngữ cảnh của bước phương pháp cũng thể hiện sự mô tả khối tương ứng hoặc mục hoặc đặc điểm của thiết bị tương ứng. Một số hoặc tất cả các bước phương pháp có thể được thực hiện bằng cách (hoặc sử dụng) thiết bị phần cứng, tương tự ví dụ, bộ vi xử lý, máy tính lập trình được hoặc mạch điện

tử. Theo một số phương án, một hoặc nhiều bước phương pháp quan trọng nhất nào đó có thể được thực hiện bởi thiết bị này.

Phụ thuộc vào các yêu cầu thực hiện nhất định, các phương án của sáng chế có thể được thực hiện trong phần cứng hoặc trong phần mềm. Việc thực hiện có thể được thực hiện sử dụng môi trường lưu trữ dạng số, ví dụ như đĩa mềm, DVD, Blu-Ray, CD, ROM, PROM, EPROM, EEPROM hoặc bộ nhớ FLASH, có các tín hiệu đọc được bằng điện được lưu trữ trên đó, chúng kết hợp (hoặc có khả năng kết hợp) với hệ thống máy tính lập trình được do đó phương pháp tương ứng được thực hiện. Do đó, môi trường lưu trữ dạng số có thể là máy tính đọc được.

Một số phương án theo sáng chế gồm có vật mang dữ liệu có các tín hiệu điều khiển đọc được bằng điện, các tín hiệu này có khả năng kết hợp với hệ thống máy tính lập trình được, do đó một trong các phương pháp đã mô tả ở đây được thực hiện.

Thông thường, các phương án của sáng chế có thể được thực hiện như một sản phẩm chương trình máy tính với mã chương trình, mã chương trình hoạt động để thực hiện một trong các phương pháp khi sản phẩm chương trình máy tính chạy trên máy tính. Mã chương trình có thể ví dụ được lưu trữ trên vật mang đọc được bằng máy.

Các phương án khác gồm có chương trình máy tính để thực hiện một trong các phương pháp được mô tả ở đây, đã được lưu trữ trên vật mang đọc được bằng máy.

Nói cách khác, phương án của phương pháp theo sáng chế là chương trình máy tính có mã chương trình để thực hiện một trong các phương pháp đã được mô tả trong sáng chế, khi chương trình máy tính chạy trên máy tính.

Phương án khác của các phương pháp là vật mang dữ liệu (hoặc môi trường lưu trữ dạng số, hoặc môi trường đọc được bằng máy tính) gồm có, chương trình máy tính đã được ghi trên vật mang dữ liệu để thực hiện một trong các phương pháp đã được mô tả ở đây. Vật mang dữ liệu, môi trường lưu trữ dạng số hoặc môi trường đã được ghi thường là hữu hình và/hoặc không tạm thời.

Do đó phương án khác của phương pháp theo sáng chế là dòng dữ liệu hoặc chuỗi các tín hiệu mô tả chương trình máy tính để thực hiện một trong các phương pháp đã được mô tả ở đây. Dòng dữ liệu hoặc chuỗi tín hiệu có thể ví dụ được tạo cấu

hình được truyền thông qua kết nối truyền thông dữ liệu, ví dụ thông qua Internet (liên mạng).

Phương án khác gồm có phương tiện xử lý, ví dụ máy tính, hoặc thiết bị logic lập trình được, được tạo cấu hình để hoặc được thích ứng để thực hiện một trong các phương pháp đã được mô tả ở đây.

Phương án khác gồm có máy tính đã được cài đặt chương trình máy tính trên đó mà chương trình máy tính này để thực hiện một trong các phương pháp đã được mô tả ở đây.

Phương án khác theo sáng chế gồm có thiết bị hoặc hệ thống được tạo cấu hình để truyền (ví dụ, bằng điện hoặc tùy ý) chương trình máy tính để thực hiện một trong các phương pháp đã được mô tả ở đây đến máy thu. Máy thu có thể, ví dụ là máy tính, thiết bị di động, thiết bị nhớ hoặc tương tự. Thiết bị hoặc hệ thống có thể, ví dụ, gồm có máy chủ tập tin để truyền chương trình máy tính đến máy thu.

Theo một số phương án, thiết bị logic lập trình được (ví dụ mảng cổng lập trình được dạng trường) có thể được sử dụng để thực hiện một số hoặc tất cả các chức năng của các phương pháp đã được mô tả ở đây. Theo một số phương án, mảng cổng lập trình được dạng trường có thể kết hợp với bộ vi xử lý để thực hiện một trong các phương pháp đã được mô tả ở đây. Nói chung, các phương pháp ưu tiên được thực hiện bởi thiết bị phần cứng bất kỳ.

Phương pháp mô tả bên trên chỉ mang mính minh họa các nguyên lý của sáng chế. Cần hiểu rằng các thay đổi và cải biến các phương án và các chi tiết đã mô tả ở đây sẽ là rõ ràng đối với người có chuyên môn trong lĩnh vực kỹ thuật tương ứng. Do đó, chỉ bởi phạm vi của các điểm yêu cầu bảo hộ sẽ làm giới hạn mục đích của sáng chế chứ không phải bởi sự mô tả cụ thể bởi bản mô tả và sự giải thích các phương án ở đây.

YÊU CẦU BẢO HỘ

1. Bộ giải mã để khôi phục ảnh (10) từ dòng dữ liệu (12) mà do ảnh được mã hóa trong các đơn vị lát (14) thành dòng dữ liệu này mà ảnh (10) được phân chia thành các đơn vị lát này, trong đó bộ giải mã được tạo cấu hình để giải mã các lát (14) từ dòng dữ liệu (12) theo thứ tự lát (16) và bộ giải mã đáp lại phần phần tử cú pháp (18) nằm trong lát hiện tại của các lát, để giải mã lát hiện tại theo một chế độ trong số ít nhất hai chế độ (20, 22), và

theo chế độ thứ nhất (20) trong số ít nhất hai chế độ, giải mã lát hiện tại từ dòng dữ liệu (12) sử dụng bước giải mã entropy thích ứng ngữ cảnh (24) bao gồm việc suy ra các ngữ cảnh vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo (38, 40) các xác suất ký hiệu phụ thuộc vào các trạng thái đã lưu của các xác suất ký hiệu của lát được giải mã trước đó, và giải mã dự đoán vượt qua các biên lát, và

theo chế độ thứ hai (22) trong số ít nhất hai chế độ, giải mã lát hiện tại từ dòng dữ liệu (12) sử dụng bước giải mã entropy thích ứng ngữ cảnh với giới hạn việc suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập với lát được giải mã trước bất kỳ và giải mã dự đoán với giới hạn giải mã dự đoán để không vượt qua các biên lát,

trong đó ảnh (10) được phân chia trong các khối mã hóa (32) được sắp đặt trong các hàng và các cột và có thứ tự quét mành (36) được định ra giữa các khối mã hóa với nhau, và bộ giải mã được tạo cấu hình để kết hợp từng lát (14) với tập con liên tục của các khối mã hóa (32) trong thứ tự quét mành (36) sao cho các tập con theo sau nhau dọc thứ tự quét mành (36) theo thứ tự lát, và

trong đó bộ giải mã được tạo cấu hình để lưu các xác suất ký hiệu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đó đến khối mã hóa (32) thứ hai trong hàng theo thứ tự quét mành (36), và, trong việc khởi tạo các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh lát hiện tại theo chế độ thứ nhất, kiểm tra xem liệu khối mã hóa thứ nhất của tập con liên tục gồm các khối mã hóa

(32) đã kết hợp với lát hiện tại có phải là khối mã hóa (32) thứ nhất trong hàng theo thứ tự quét mành hay không, và, nếu có, khởi tạo (40) các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu đã lưu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đó đến khối mã hóa thứ hai trong hàng theo thứ tự quét mành (36), và, nếu không, khởi tạo (38) các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đó đến lúc kết thúc lát đã được giải mã trước đó.

2. Bộ giải mã theo điểm 1, trong đó bộ giải mã được tạo cấu hình để đáp lại phần phần tử cú pháp (18) nằm trong lát hiện tại của các lát (14), để giải mã lát hiện tại theo một chế độ trong số ít nhất ba chế độ, cụ thể, trong một chế độ trong chế độ thứ nhất (20) và chế độ thứ ba (42) hoặc chế độ thứ hai (22), trong đó bộ giải mã được tạo cấu hình để

theo chế độ thứ ba (42), giải mã lát hiện tại từ dòng dữ liệu sử dụng giải mã entropy thích ứng ngữ cảnh với giới hạn việc suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập với lát đã được giải mã trước đó bất kỳ, và giải mã dự đoán vượt qua các biên lát,

trong đó một trong các chế độ thứ nhất và thứ ba được chọn phụ thuộc vào phần tử cú pháp.

3. Bộ giải mã theo điểm 1 hoặc điểm 2, trong đó bộ giải mã được tạo cấu hình để đáp lại phần tử cú pháp chung trong dòng dữ liệu để vận hành một chế độ trong số ít nhất hai chế độ vận hành chung, với, theo chế độ vận hành chung thứ nhất, thực hiện sự đáp ứng với phần phần tử cú pháp cho mỗi lát, và, theo chế độ vận hành chung thứ hai, sử dụng chắc chắn một chế độ khác trong số ít nhất hai chế độ khác với chế độ thứ nhất.

4. Bộ giải mã theo điểm bất kỳ trong số các điểm từ 1 đến 3, trong đó bộ giải mã được tạo cấu hình để theo các chế độ thứ nhất và thứ hai, tiếp tục một cách chắc chắn và liên tiếp cập nhật liên tục các xác suất ký hiệu từ lúc bắt đầu đến lúc kết thúc lát hiện tại.

5. Bộ giải mã theo điểm bất kỳ trong số các điểm từ 1 đến 4, trong đó bộ giải mã được tạo cấu hình để lưu các xác suất ký hiệu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đó đến lúc kết thúc lát đã được giải mã trước đó.

6. Bộ giải mã theo điểm bất kỳ trong số các điểm từ 1 đến 5, trong đó bộ giải mã được tạo cấu hình để, trong chế độ thứ nhất và thứ hai, giới hạn việc giải mã dự đoán trong phạm vi các ô mà ảnh được chia nhỏ thành các ô này.

7. Bộ giải mã theo điểm bất kỳ trong số các điểm từ 1 đến 6, trong đó bộ giải mã được tạo cấu hình để, trong chế độ thứ nhất và thứ hai, đọc thông tin từ lát hiện tại bộc lộ sự chia nhỏ lát hiện tại thành các phần con song song, ngừng giải mã entropy thích ứng ngữ cảnh tại lúc kết thúc phần con song song thứ nhất và tiếp tục lại việc giải mã entropy thích ứng ngữ cảnh một lần nữa tại lúc bắt đầu phần con song song kế tiếp bất kỳ bao gồm, trong chế độ thứ nhất, khởi tạo các xác suất ký hiệu phụ thuộc vào các trạng thái đã lưu của các xác suất ký hiệu của phần con song song trước và, trong chế độ thứ hai, khởi tạo các xác suất ký hiệu độc lập với lát đã được giải mã trước bất kỳ và phần con song song đã được giải mã trước bất kỳ.

8. Bộ giải mã theo điểm bất kỳ trong số các điểm từ 1 đến 7, trong đó bộ giải mã được tạo cấu hình để, theo chế độ thứ nhất (20) trong số ít nhất hai chế độ, sao chép cho lát hiện tại một phần của cú pháp đoạn đầu lát từ lát trước đã được giải mã trong chế độ thứ hai.

9. Bộ giải mã theo điểm bất kỳ trong số các điểm nêu trên, trong đó bộ giải mã được tạo cấu hình để khôi phục ảnh (10) từ dòng dữ liệu (12) sử dụng xử lý song song mặt đầu sóng (WPP), trong đó mỗi lát (14) gồm có phần cú pháp khởi đầu (400) chỉ ra vị trí bắt đầu giải mã của lát tương ứng nằm trong ảnh (10) và trong đó bộ giải mã được tạo cấu hình để

nhận diện các điểm vào của các dòng con WPP mà do các lát được nhóm lại thành các dòng con WPP này, bằng cách nhận diện, sử dụng các phần cú pháp khởi đầu của lát, các lát bắt đầu tại phía bên trái của bức ảnh, và

giải mã song song các dòng con WPP theo cách thức so le với việc bắt đầu liên tục việc giải mã các dòng con WPP theo thứ tự lát.

10. Bộ giải mã theo điểm bất kỳ trong số các điểm từ 1 đến 9, trong đó bộ giải mã được tạo cấu hình để chia các khối biến đổi sắc độ khác hơn các khối biến đổi độ sáng đáp ứng với thông tin trong dòng dữ liệu.

11. Bộ mã hóa để mã hóa ảnh (10) thành dòng dữ liệu (12) trong các đơn vị lát (14) mà ảnh (10) được phân chia thành các đơn vị lát này, trong đó bộ mã hóa được tạo cấu hình để mã hóa các lát (14) thành dòng dữ liệu (12) theo thứ tự lát (16) và bộ mã hóa được tạo cấu hình để

xác định phần phần tử cú pháp (18) cho, và mã hóa phần phần tử cú pháp thành, lát hiện tại trong số các lát sao cho phần phần tử cú pháp báo hiệu lát hiện tại sẽ được mã hóa theo một chế độ trong số ít nhất hai chế độ (20, 22), và

nếu lát hiện tại sẽ được mã hóa theo chế độ thứ nhất (20) trong số ít nhất hai chế độ, mã hóa lát hiện tại thành dòng dữ liệu (12) sử dụng bước mã hóa entropy thích ứng ngữ cảnh (24) bao gồm việc suy ra các ngữ cảnh vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo (38, 40) các xác suất ký hiệu phụ thuộc vào các trạng thái đã lưu của các xác suất ký hiệu của lát đã được mã hóa trước đó, và việc mã hóa dự đoán vượt qua các biên lát, và

nếu lát hiện tại sẽ được mã hóa theo chế độ thứ hai (22) trong số ít nhất hai chế độ, mã hóa lát hiện tại thành dòng dữ liệu (12) sử dụng mã hóa entropy thích ứng ngữ cảnh với giới hạn sự suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập với lát đã được mã hóa trước bất kỳ, và mã hóa dự đoán với việc giới hạn mã hóa dự đoán để không vượt qua các biên lát,

trong đó ảnh (10) được phân chia trong các khối mã hóa (32) được sắp xếp trong các hàng và các cột và có thứ tự quét mành (36) được định ra giữa các khối mã hóa với nhau, và bộ mã hóa được tạo cấu hình để kết hợp từng lát (14) với tập con liên tục của các khối mã hóa (32) theo thứ tự quét mành (36) sao cho các tập con theo sau nhau dọc theo thứ tự quét mành (36) theo thứ tự lát, và

trong đó bộ mã hóa được tạo cấu hình để lưu các xác suất ký hiệu như thu được trong bước mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đó đến khối mã hóa (32) thứ hai trong hàng theo thứ tự quét mành (36), và, trong bước khởi tạo các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh lát hiện tại theo chế độ thứ nhất, kiểm tra xem liệu khối mã hóa thứ nhất của tập con liên tục của các khối mã hóa (32) được kết hợp với lát hiện tại có phải là khối mã hóa (32) thứ nhất trong hàng theo thứ tự quét mành hay không, và, nếu có, khởi tạo (40) các xác suất ký hiệu cho việc mã hóa entropy thích ứng ngữ cảnh lát hiện tại phụ thuộc vào các xác suất ký hiệu đã thu được trong bước mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đến khối mã hóa thứ hai trong hàng theo thứ tự quét mành (36), và, nếu không, khởi tạo (38) các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh lát hiện tại phụ thuộc vào các xác suất ký hiệu như thu được trong bước mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đến lúc kết thúc lát đã được mã hóa trước đó.

12. Bộ mã hóa theo điểm 11, trong đó bộ mã hóa được tạo cấu hình để chia các khối biến đổi sắc độ khác với các khối biến đổi độ sáng.

13. Bộ mã hóa theo điểm 11, trong đó bộ mã hóa được tạo cấu hình để mã hóa phần phần tử cú pháp (18) thành lát hiện tại trong số các lát (14) sao cho lát hiện tại được báo hiệu sẽ được mã hóa tại đó theo một trong ít nhất ba chế độ, cụ thể, theo một trong chế độ thứ nhất (20) và thứ ba (42) hoặc chế độ thứ hai (22), trong đó bộ mã hóa được tạo cấu hình để

theo chế độ thứ ba (42), mã hóa lát hiện tại thành dòng dữ liệu sử dụng mã hóa entropy thích ứng ngữ cảnh với giới hạn việc suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập với lát được mã hóa trước bất kỳ, và mã hóa dự đoán vượt qua các biên lát,

trong đó bộ mã hóa phân biệt giữa một chế độ trong số các chế độ thứ nhất và thứ ba sử dụng phần tử cú pháp.

14. Bộ mã hóa theo điểm 11, trong đó bộ mã hóa được tạo cấu hình để xác định phần tử cú pháp chung và viết phần tử cú pháp thành dòng dữ liệu cùng với vận hành theo

một chế độ trong số ít nhất hai chế độ vận hành chung phụ thuộc vào phần tử cú pháp chung, cụ thể, với, theo chế độ vận hành chung thứ nhất, thực hiện mã hóa phần phần tử cú pháp cho mỗi lát, và, theo chế độ vận hành chung thứ hai, sử dụng chắc chắn một chế độ khác trong số ít nhất hai chế độ khác với chế độ thứ nhất.

15. Phương pháp khôi phục ảnh (10) từ dòng dữ liệu (12) mà do ảnh được mã hóa trong các đơn vị lát (14) thành dòng dữ liệu này mà ảnh (10) được phân chia thành các đơn vị lát này, trong đó phương pháp gồm có giải mã các lát (14) từ dòng dữ liệu (12) theo thứ tự lát (16) và phương pháp đáp lại phần phần tử cú pháp (18) nằm trong lát hiện tại trong số các lát, để giải mã lát hiện tại theo một chế độ trong số ít nhất hai chế độ (20, 22), trong đó

theo chế độ thứ nhất (20) trong số ít nhất hai chế độ, lát hiện tại được giải mã từ dòng dữ liệu (12) sử dụng giải mã entropy thích ứng ngữ cảnh (24) bao gồm việc suy ra các ngữ cảnh vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo (38, 40) các xác suất ký hiệu phụ thuộc vào các trạng thái đã lưu của các xác suất ký hiệu của lát đã được giải mã trước đó, và giải mã dự đoán vượt qua các biên lát, và

theo chế độ thứ hai (22) trong số ít nhất hai chế độ, lát hiện tại được giải mã từ dòng dữ liệu (12) sử dụng giải mã entropy thích ứng ngữ cảnh với giới hạn việc suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập với lát đã được giải mã trước đó bất kỳ, và giải mã dự đoán với việc giới hạn giải mã dự đoán để không vượt qua các biên lát,

trong đó ảnh (10) được phân chia trong các khối mã hóa (32) được sắp xếp trong các hàng và các cột và có thứ tự quét mành (36) được định ra giữa các khối mã hóa với nhau, và phương pháp bao gồm bước kết hợp từng lát (14) với tập con liên tục các khối mã hóa (32) theo thứ tự quét mành (36) sao cho các tập con theo sau nhau dọc thứ tự quét mành (36) theo thứ tự lát, và

trong đó phương pháp bao gồm lưu các xác suất ký hiệu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đó đến khối mã

hóa (32) thứ hai trong hàng theo thứ tự quét mành (36), và, trong việc khởi tạo xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại theo chế độ thứ nhất, kiểm tra xem liệu khối mã hóa thứ nhất của tập con liên tục của các khối mã hóa (32) đã kết hợp với lát hiện tại có phải là khối mã hóa (32) thứ nhất trong hàng theo thứ tự quét mành hay không, và, nếu có, khởi tạo (40) các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu đã lưu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đó đến khối mã hóa thứ hai trong hàng theo thứ tự quét mành (36), và, nếu không, khởi tạo (38) các xác suất ký hiệu cho bước giải mã entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu như thu được trong bước giải mã entropy thích ứng ngữ cảnh lát đã được giải mã trước đến lúc kết thúc lát đã được giải mã trước.

16. Phương pháp mã hóa ảnh (10) thành dòng dữ liệu (12) trong các đơn vị lát (14) do mà ảnh (10) được phân chia thành các đơn vị lát này, trong đó phương pháp gồm có mã hóa các lát (14) thành dòng dữ liệu (12) theo thứ tự lát (16) và phương pháp gồm có

xác định phần phần tử cú pháp (18) cho, mã hóa phần phần tử cú pháp thành, lát hiện tại trong số các lát sao cho phần phần tử cú pháp báo hiệu lát hiện tại sẽ được mã hóa theo một chế độ trong số ít nhất hai chế độ (20, 22), và

nếu lát hiện tại sẽ được mã hóa theo chế độ thứ nhất (20) trong số ít nhất hai chế độ, mã hóa lát hiện tại thành dòng dữ liệu (12) sử dụng bước mã hóa entropy thích ứng ngữ cảnh (24) bao gồm việc suy ra các ngữ cảnh vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo (38, 40) các xác suất ký hiệu phụ thuộc các trạng thái đã lưu của các xác suất ký hiệu của lát đã được mã hóa trước, và mã hóa dự đoán vượt qua các biên lát, và

nếu lát hiện tại sẽ được mã hóa theo chế độ thứ hai (22) trong số ít nhất hai chế độ, mã hóa lát hiện tại thành dòng dữ liệu (12) sử dụng mã hóa entropy thích ứng ngữ cảnh với giới hạn việc suy ra các ngữ cảnh để không vượt qua các biên lát, cập nhật liên tục các xác suất ký hiệu của các ngữ cảnh và khởi tạo các xác suất ký hiệu độc lập

với lát đã mã hóa trước bất kỳ, và mã hóa dự đoán với giới hạn mã hóa dự đoán để không vượt qua các biên lát,

trong đó ảnh (10) được phân chia trong các khối mã hóa (32) được sắp xếp trong các hàng và các cột và có thứ tự quét mành (36) được định ra giữa các khối mã hóa với nhau, và phương pháp bao gồm bước kết hợp từng lát (14) với tập con liên tục các khối mã hóa (32) theo thứ tự quét mành (36) sao cho các tập con theo sau nhau dọc thứ tự quét mành (36) theo thứ tự lát, và

trong đó phương pháp còn bao gồm lưu các xác suất ký hiệu như thu được trong bước mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đó đến khối mã hóa (32) thứ hai trong hàng theo thứ tự quét mành (36), và, trong việc khởi tạo xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh lát hiện tại theo chế độ thứ nhất, kiểm tra xem liệu khối mã hóa thứ nhất của tập con liên tục các khối mã hóa (32) đã kết hợp với lát hiện tại có phải là khối mã hóa (32) thứ nhất trong hàng theo thứ tự quét mành hay không, và, nếu có, khởi tạo (40) các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu đã lưu như thu được trong bước mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đó đến khối mã hóa thứ hai trong hàng theo thứ tự quét mành (36), và, nếu không, khởi tạo (38) các xác suất ký hiệu cho bước mã hóa entropy thích ứng ngữ cảnh của lát hiện tại phụ thuộc vào các xác suất ký hiệu như thu được trong bước mã hóa entropy thích ứng ngữ cảnh lát đã được mã hóa trước đến lúc kết thúc lát đã được mã hóa trước.

17. Vật ghi lưu trữ dạng số đã lưu trữ trên đó dòng dữ liệu, trong đó dòng dữ liệu được mã hóa sử dụng phương pháp theo điểm 16.

18. Vật ghi lưu trữ dạng số đã lưu trữ trên đó dòng dữ liệu theo điểm 17, trong đó các khối biến đổi sắc độ được chia khác với các khối biến đổi độ sáng.

19. Vật ghi có thể đọc đường bằng máy tính chứa chương trình máy tính có mã chương trình để thực hiện phương pháp theo điểm 15, khi chương trình máy tính chạy trên máy tính.

20. Vật ghi có thể đọc đường bằng máy tính chứa chương trình máy tính có mã chương trình để thực hiện phương pháp theo điểm 16, khi chương trình máy tính chạy trên máy tính.

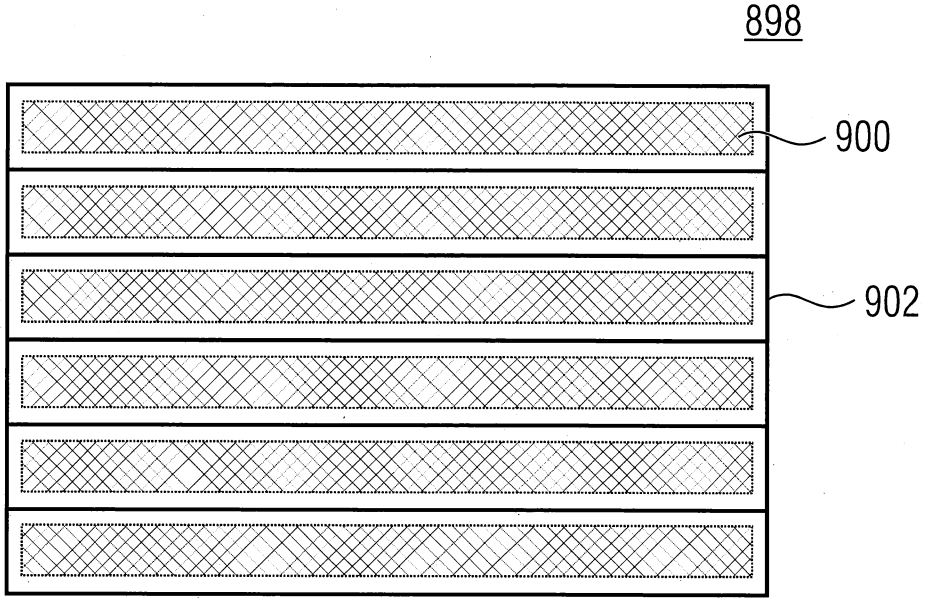


FIG 1

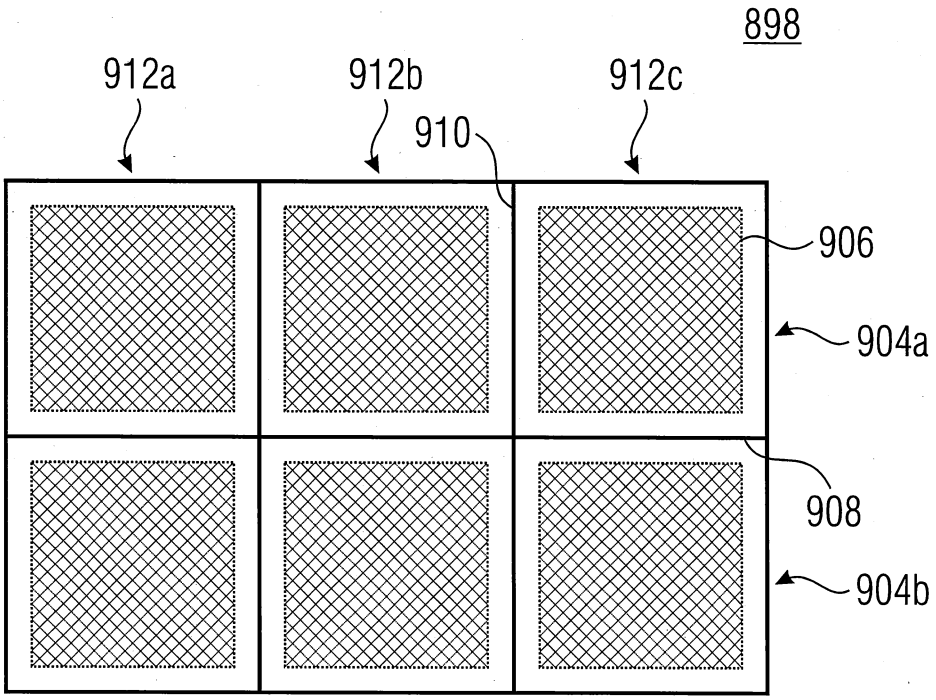


FIG 2

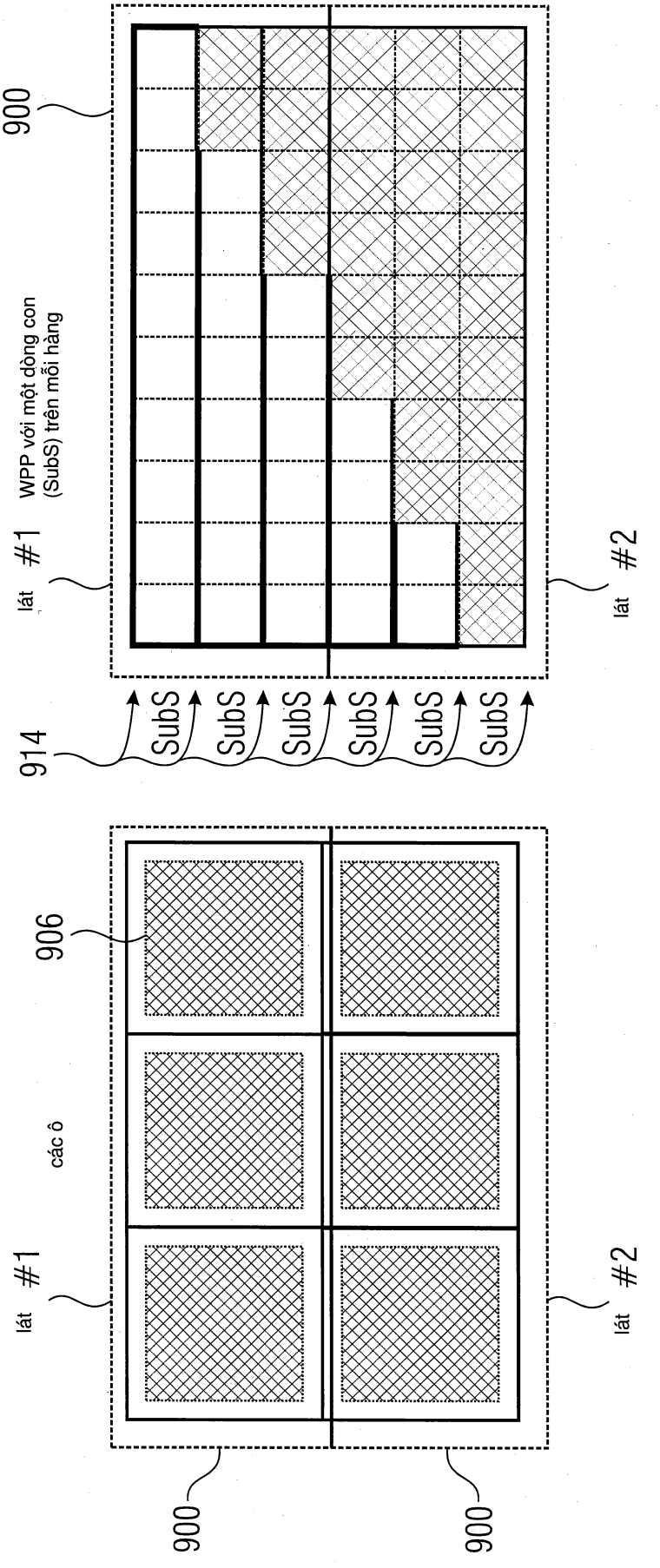

900
WPP với một dòng con (SubS) trên mỗi hàng
lát #1
lát #2
914
SubS
SubS
SubS
SubS
SubS
SubS
906
các ô
lát #1
lát #2
900
900

FIG 3

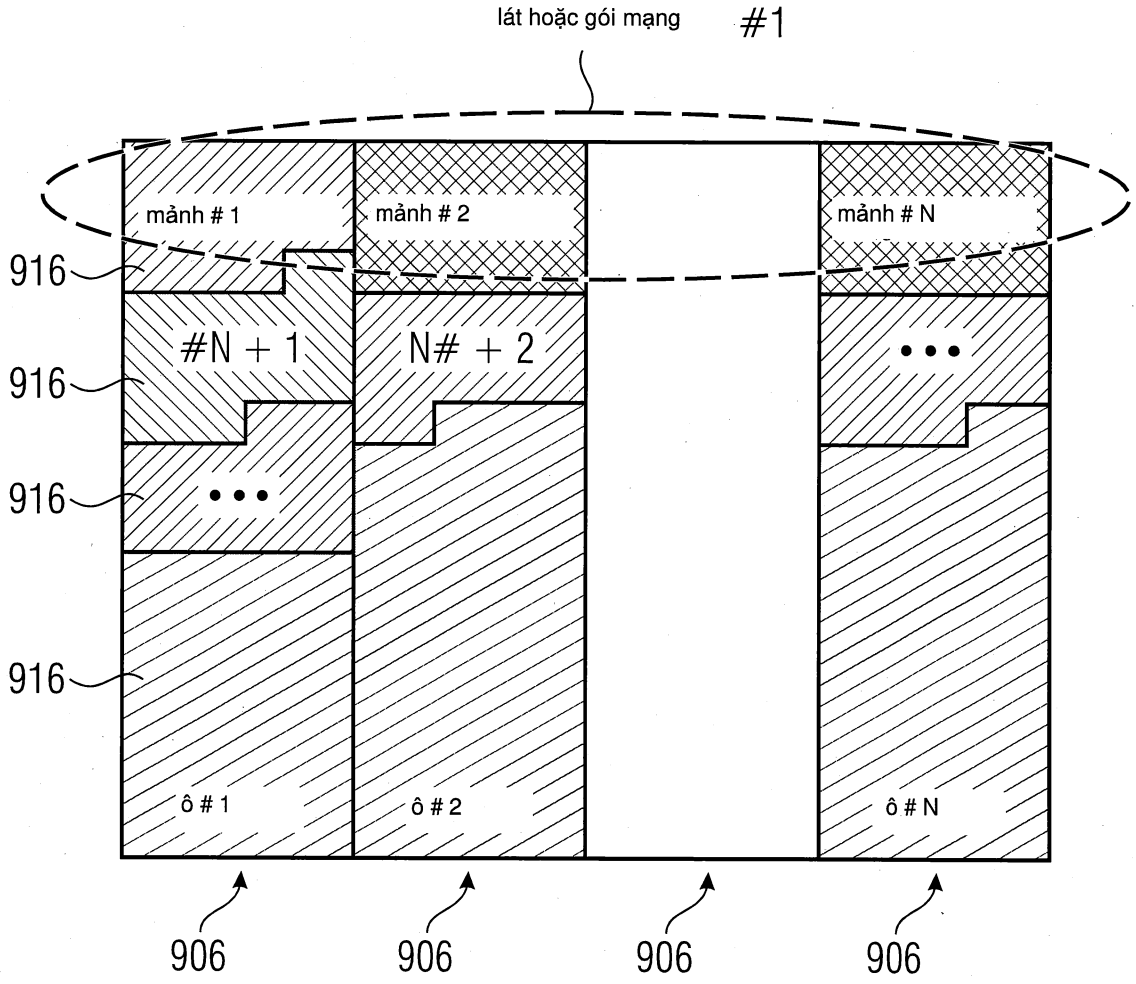

lát hoặc gói mạng
#1
mảnh # 1
mảnh # 2
mảnh # N
916
916
916
916
#N + 1
N# + 2
• • •
• • •
ô # 1
ô # 2
ô # N
906
906
906
906

FIG 4

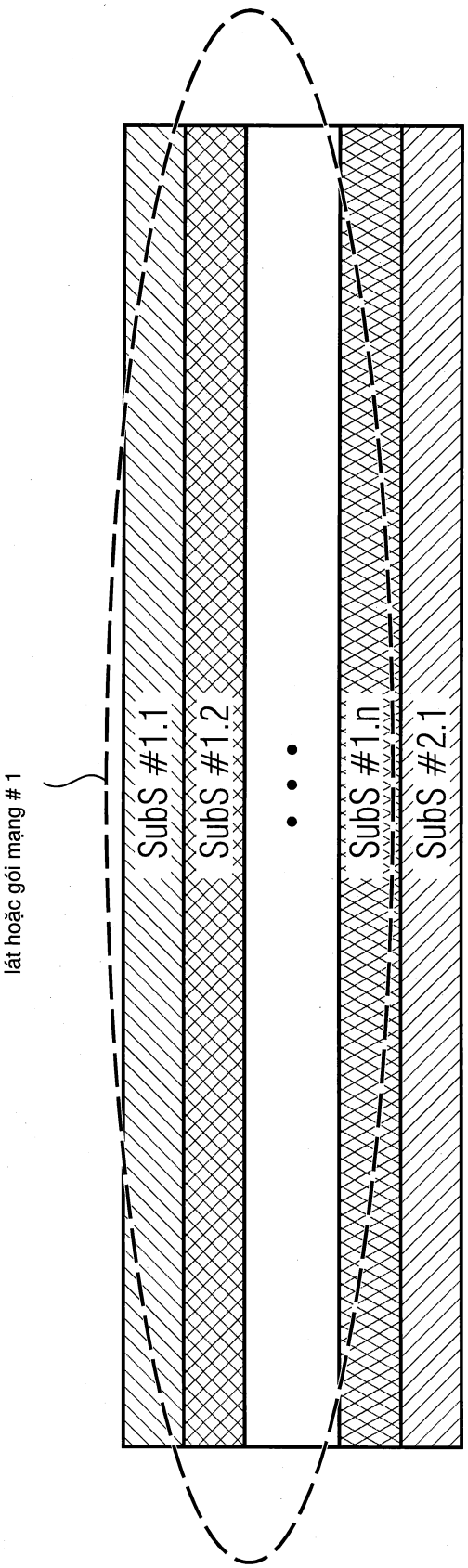


FIG 5

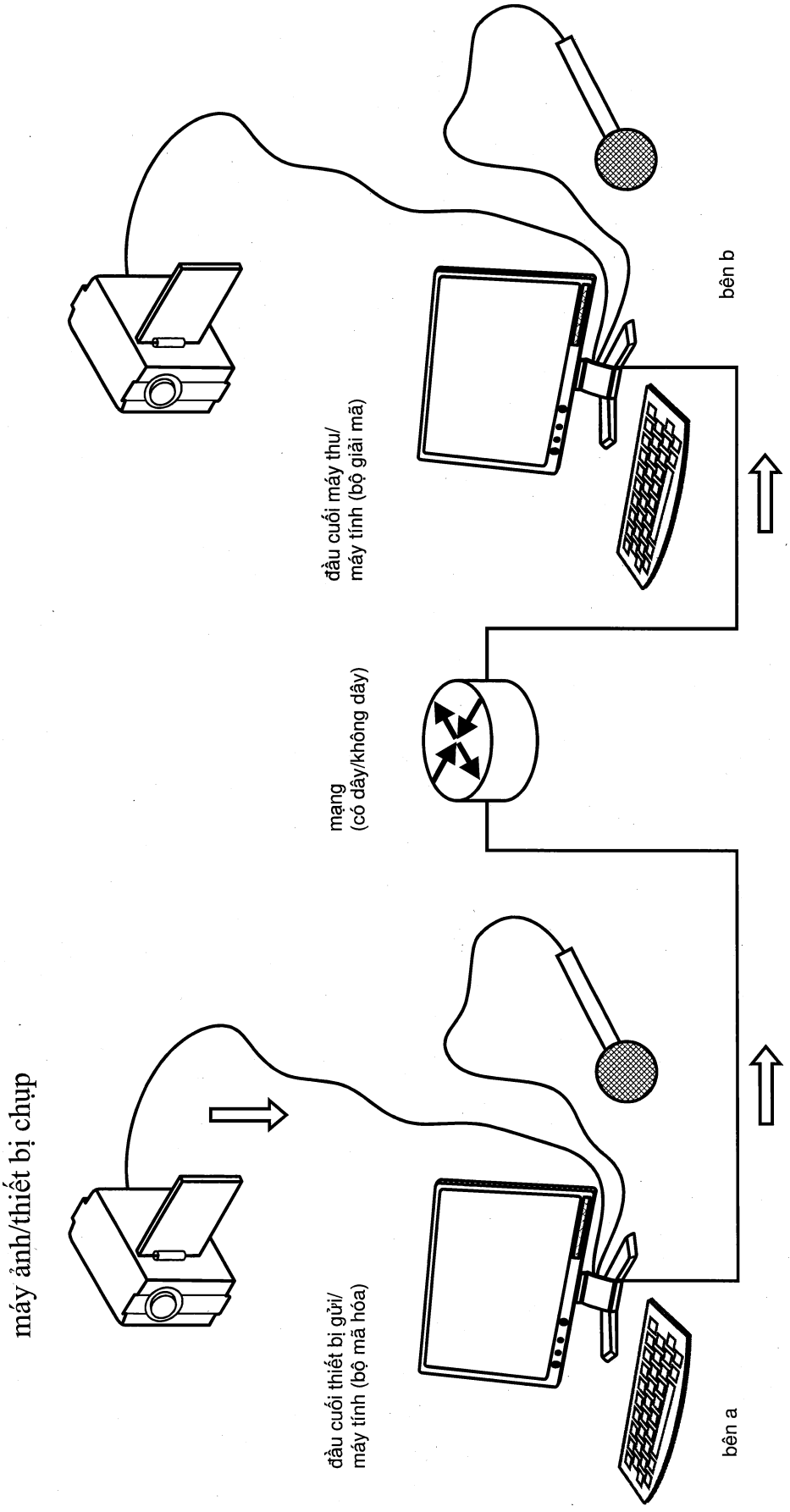

máy ảnh/thiết bị chụp
đầu cuối thiết bị gửi/
máy tính (bộ mã hóa)
mạng
(có dây/không dây)
đầu cuối máy thu/
máy tính (bộ giải mã)
bên a
bên b

FIG 6

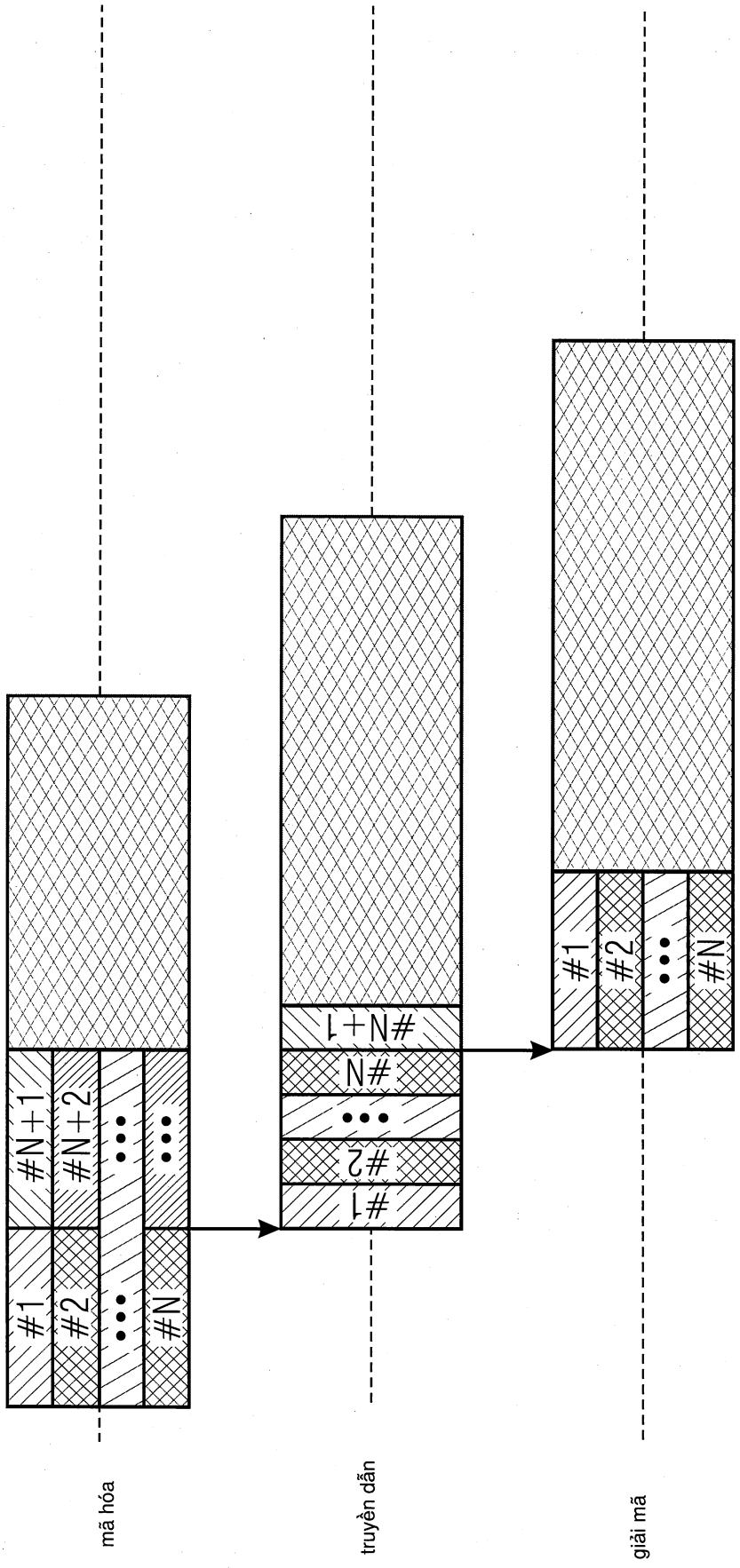


FIG 7

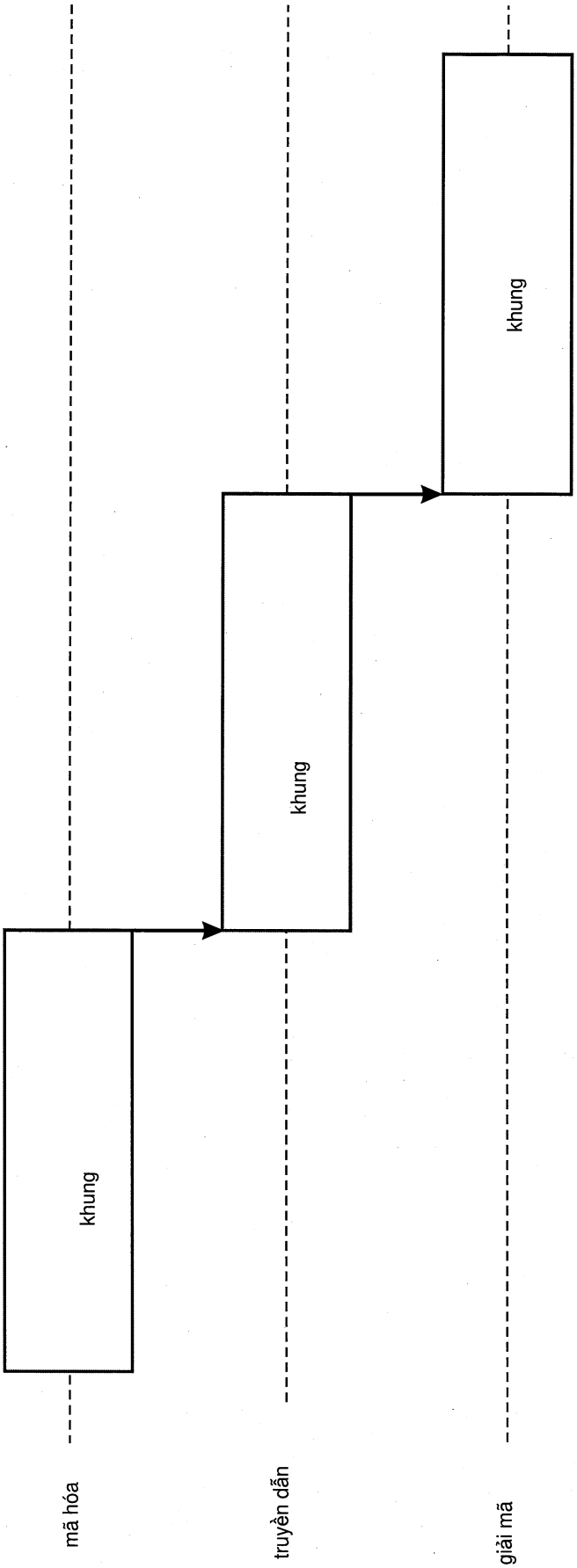


FIG 8

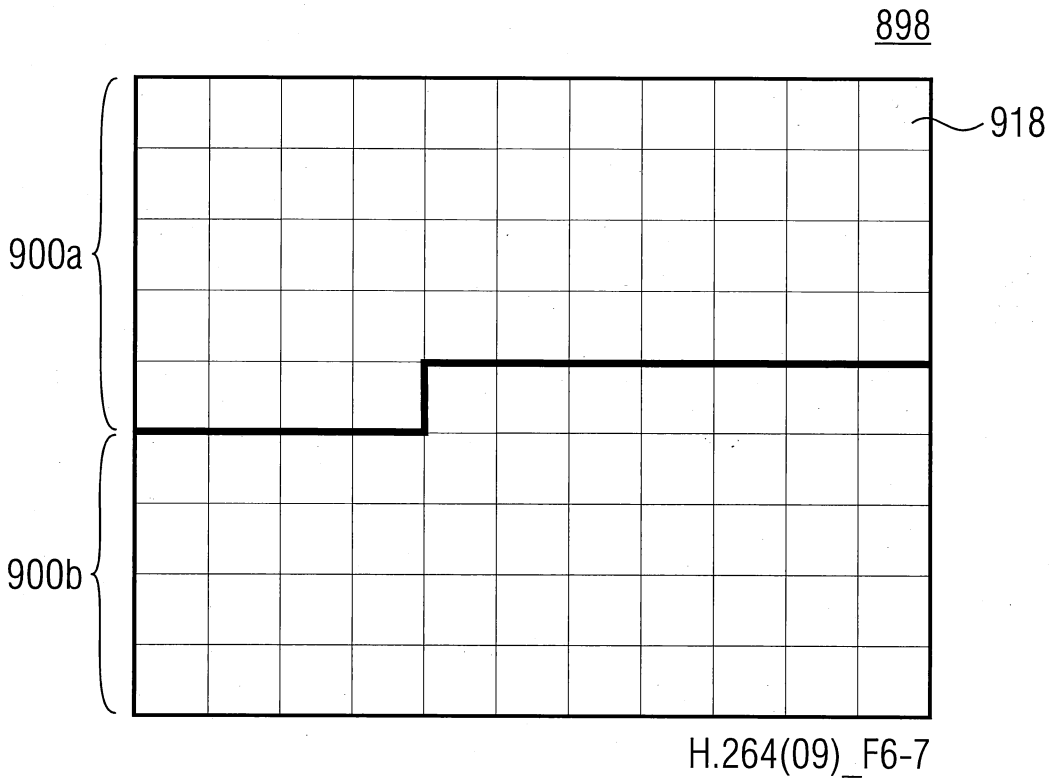

898
918
900a
900b
H.264(09)_F6-7

FIG 9

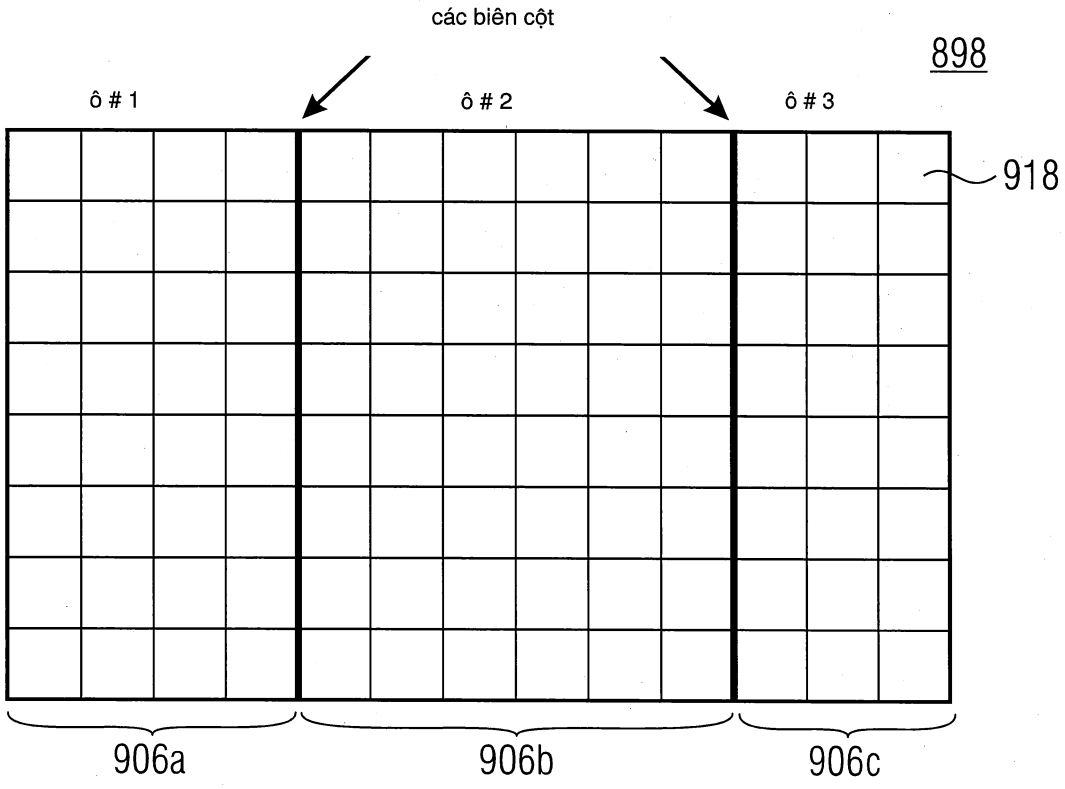

các biên cột
898
ô # 1
ô # 2
ô # 3
918
906a
906b
906c

FIG 10

seq_parameter_set_rbsp( ) {	Descriptor
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc = = 3)	
separate_colour_plane_flag	u(1)
max_temporal_layers_minus1	u(3)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
pic_cropping_flag	u(1)
if(pic_cropping_flag) {	
pic_crop_left_offset	ue(v)
pic_crop_right_offset	ue(v)
pic_crop_top_offset	ue(v)
pic_crop_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_bit_depth_luma_minus1	u(4)
pcm_bit_depth_chroma_minus1	u(4)
}	
qpprime_y_zero_transquant_bypass_flag	u(1)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
for(i = 0; i <= max_temporal_layers_minus1; i++) {	
max_dec_pic_buffering[i]	ue(v)

FIG 11A

FIG 11	FIG 11A
	FIG 11B
	FIG 11C

num_reorder_pics[i]	ue(v)
max_latency_increase[i]	ue(v)
}	
restricted_ref_pic_lists_flag	u(1)
if(restricted_ref_pic_lists_flag)	
lists_modification_present_flag	u(1)
log2_min_coding_block_size_minus3	ue(v)
log2_diff_max_min_coding_block_size	ue(v)
log2_min_transform_block_size_minus2	ue(v)
log2_diff_max_min_transform_block_size	ue(v)
if(pcm_enabled_flag) {	
log2_min_pcm_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_coding_block_size	ue(v)
}	
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enable_flag	
chroma_pred_from_luma_enabled_flag	u(1)
deblocking_filter_in_aps_enabled_flag	u(1)
seq_loop_filter_across_slices_enabled_flag	u(1)
asymmetric_motion_partitions_enabled_flag	u(1)
non_square_quadtree_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
adaptive_loop_filter_enabled_flag	u(1)
if(adaptive_loop_filter_enabled_flag)	
alf_coef_in_slice_flag	u(1)
if(pcm_enabled_flag)	
pcm_loop_filter_disable_flag	u(1)
temporal_id_nesting_flag	u(1)

FIG 11B

FIG 11	FIG 11A
	FIG 11B
	FIG 11C

if(log2_min_coding_block_size_minus3 = = 0)	
inter_4x4_enabled_flag	u(1)
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
short_term_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
tiles_or_entropy_coding_sync_idc	u(2)
if(tiles_or_entropy_coding_sync_idc = = 1) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
sps_extension_flag	u(1)
if(sps_extension_flag)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

FIG 11C

FIG 11	FIG 11A
	FIG 11B
	FIG 11C

pic_parameter_set_rbsp( ) {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
sign_data_hiding_flag	u(1)
if(sign_data_hiding _flag)	
sign_hiding_threshold	u(4)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
enable_temporal_mvp_flag	u(1)
slice_granularity	u(2)
max_cu_qp_delta_depth	ue(v)
cb_qp_offset	se(v)
cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
output_flag_present_flag	u(1)
if(tiles_or_entropy_coding_sync_idc == 1) {	
tile_info_present_flag	u(1)
tile_control_present_flag	u(1)
if(tile_info_present_flag) {	
num_tile_columns_minus1	ue(v)

FIG 12A

FIG 12 | FIG 12A / FIG 12B

num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
}	
if(tile_control_present_flag)	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles_or_entropy_coding_sync_idc == 2)	
num_substreams_minus1	ue(v)
deblocking_filter_control_present_flag	u(1)
if(slice_type == P || slice_type == B)	
log2_parallel_merge_level_minus2	ue(v)
pps_extension_flag	u(1)
if(pps_extension_flag)	
while(more_rbsp_data( ))	
pps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

FIG 12B

FIG 12	FIG 12A
	FIG 12B

slice_header( ) {	Descriptor
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if(!entropy_slice_flag) {	
pic_parameter_set_id	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if(IdrPicFlag) {	
idr_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
} else {	
pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
delta_poc_lsb_lt[i]	ue(v)
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt_minus1[i]	ue(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_interleaving_flag	u(1)
slice_sample_adaptive_offset_flag	u(1)

FIG 13A

FIG 13

FIG 13A
FIG 13B
FIG 13C

if(slice_sao_interleaving_flag & & slice_sample_adaptive_offset_flag) {	
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
}	
}	
if(scaling_list_enable_flag | | deblocking_filter_in_aps_enabled_flag | | (sample_adaptive_offset_enabled_flag) & & !slice_sao_interleaving_flag) | | adaptive_loop_filter_enabled_flag)	
aps_id	ue(v)
if(slice_type = = P | | slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
if(lists_modification_present_flag) {	
ref_pic_list_modification( )	
ref_pic_list_combination( )	
}	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
}	
if(cabac_init_present_flag && slice_type != I)	
cabac_init_flag	u(1)
if(!entropy_slice_flag) {	
slice_qp_delta	se(v)
if(deblocking_filter_control_present_flag) {	
if(deblocking_filter_in_aps_enabled_flag)	
inherit_dbl_params_from_aps_flag	u(1)
if(!inherit_dbl_params_from_aps_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)

FIG 13B

FIG 13 | FIG 13A | FIG 13B | FIG 13C

tc_offset_div2	se(v)
}	
}	
}	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if(slice_type != I && ((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_idc = = 1 && slice_type = = B))	
pred_weight_table( )	
}	
if(slice_type = = P | | slice_type = = B)	
five_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param( )	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_slice_flag)	
alf_cu_control_param( )	
}	
if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag || slice_sample_adaptive_offset_flag || !disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if(tiles_or_entropy_coding_sync_idc > 0) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
}	

FIG 13C

FIG 13	FIG 13A
	FIG 13B
	FIG 13C

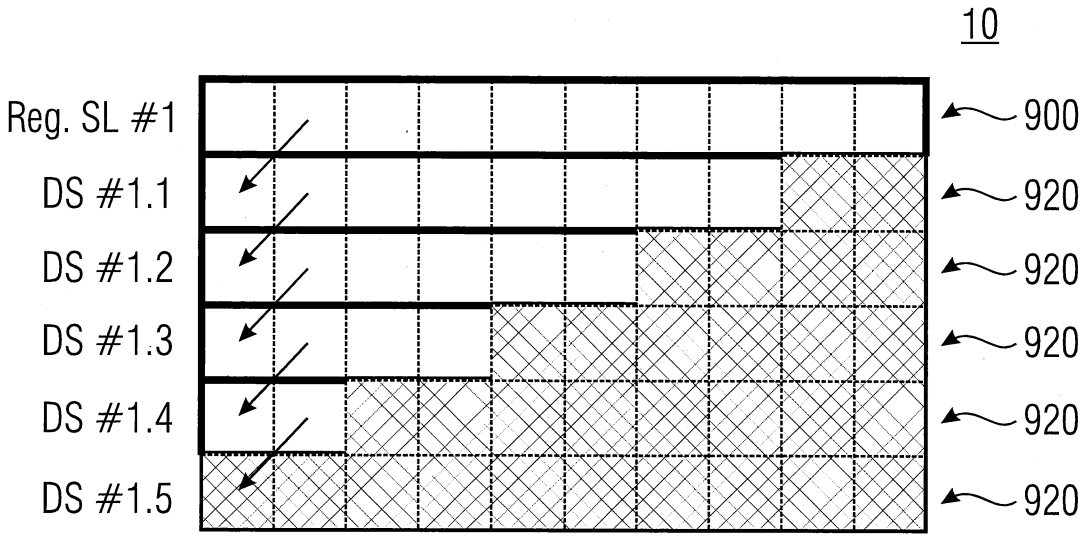


FIG 14

pic_parameter_set_rbsp( ) {	Descriptor
• • •	
dependent_slices_present_flag	u(1)
last_ctb_cabac_init_flg	u(1)
}	

FIG 15

400

slice_header( ) {	Descriptor
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
~~entropy_slice_flag~~	~~u(1)~~
~~dependent_slice_flag~~	~~u(1)~~
~~if(!entropy_slice_flag) {~~	
~~if(!dependent_slice_flag) {~~	
• • •	
~~} else {~~	
~~no_cabac_reset_flag~~	~~u(1)~~
}	
• • •	
if(cabac_init_present_flag & & slice_type != I & & ~~!no_cabac_reset_flag~~)	
cabac_init_flag	u(1)
• • •	
if(slice_type = = P | | slice_type = = B)	
five_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param( )	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_slice_flag)	
alf_cu_control_param( )	
}	
if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag | | slice_sample_adaptive_offset_flag | | !disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if (tiles_or_entropy_coding_sync_idc > = 0) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
}	

100

FIG 16

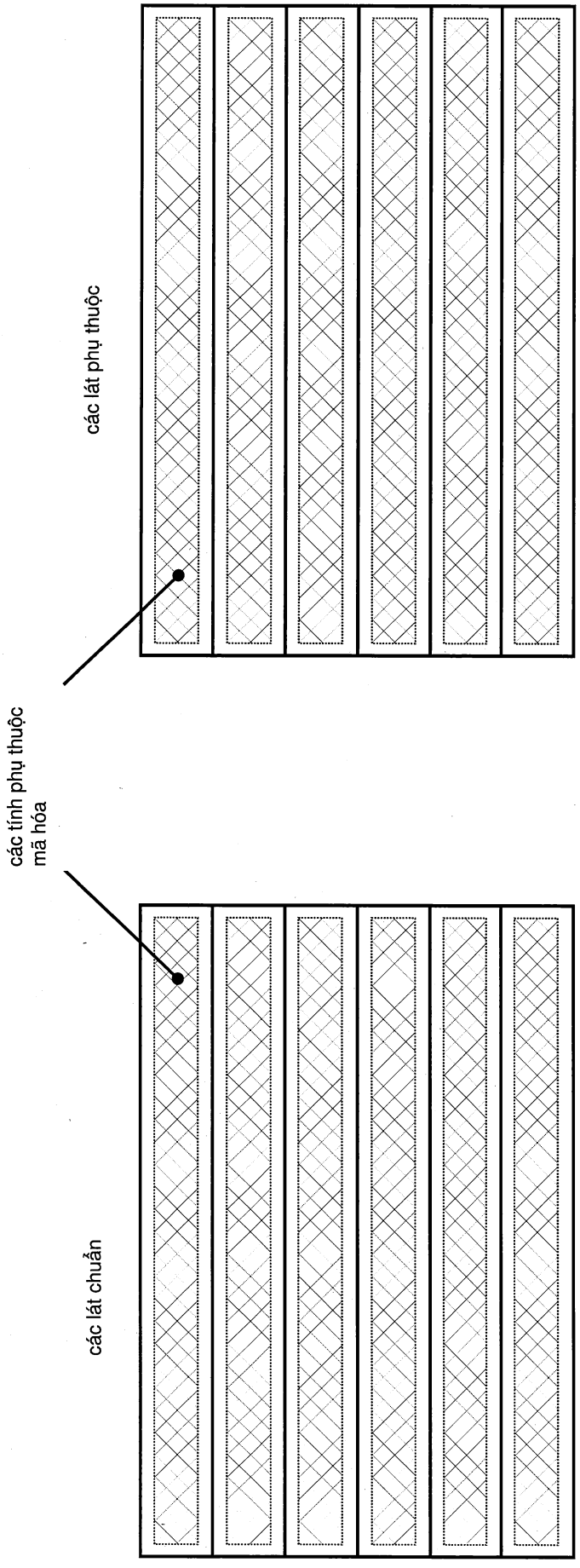
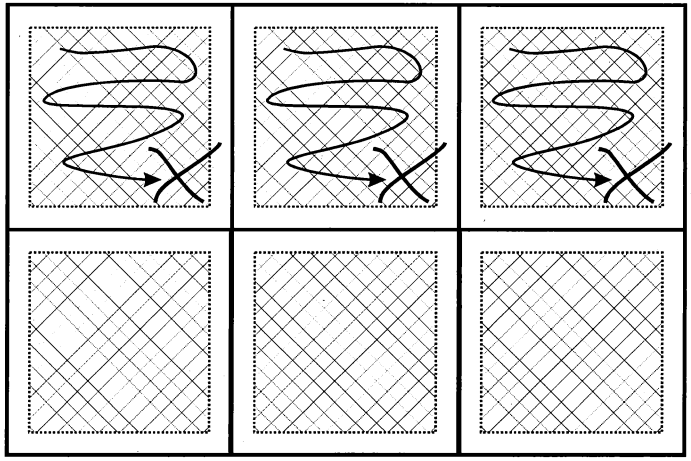


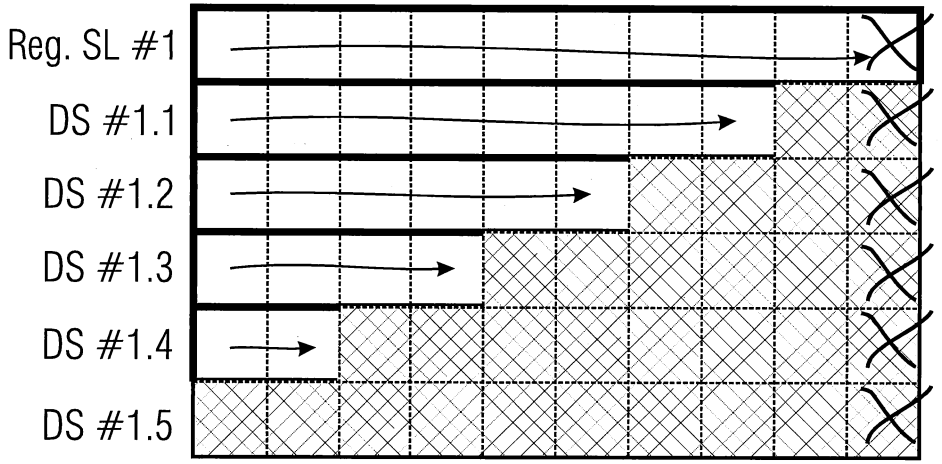
FIG 17

DS viết tắt của dependent slice
CU viết tắt của coding unit

các ô



một lát phụ thuộc (DS) trên hàng đơn vị mã hóa (CU)



không có độ trễ trong trường hợp:

$$t_{enc}(2*CU) <= t_{trans}(1*CU\_ROW)$$

FIG 18

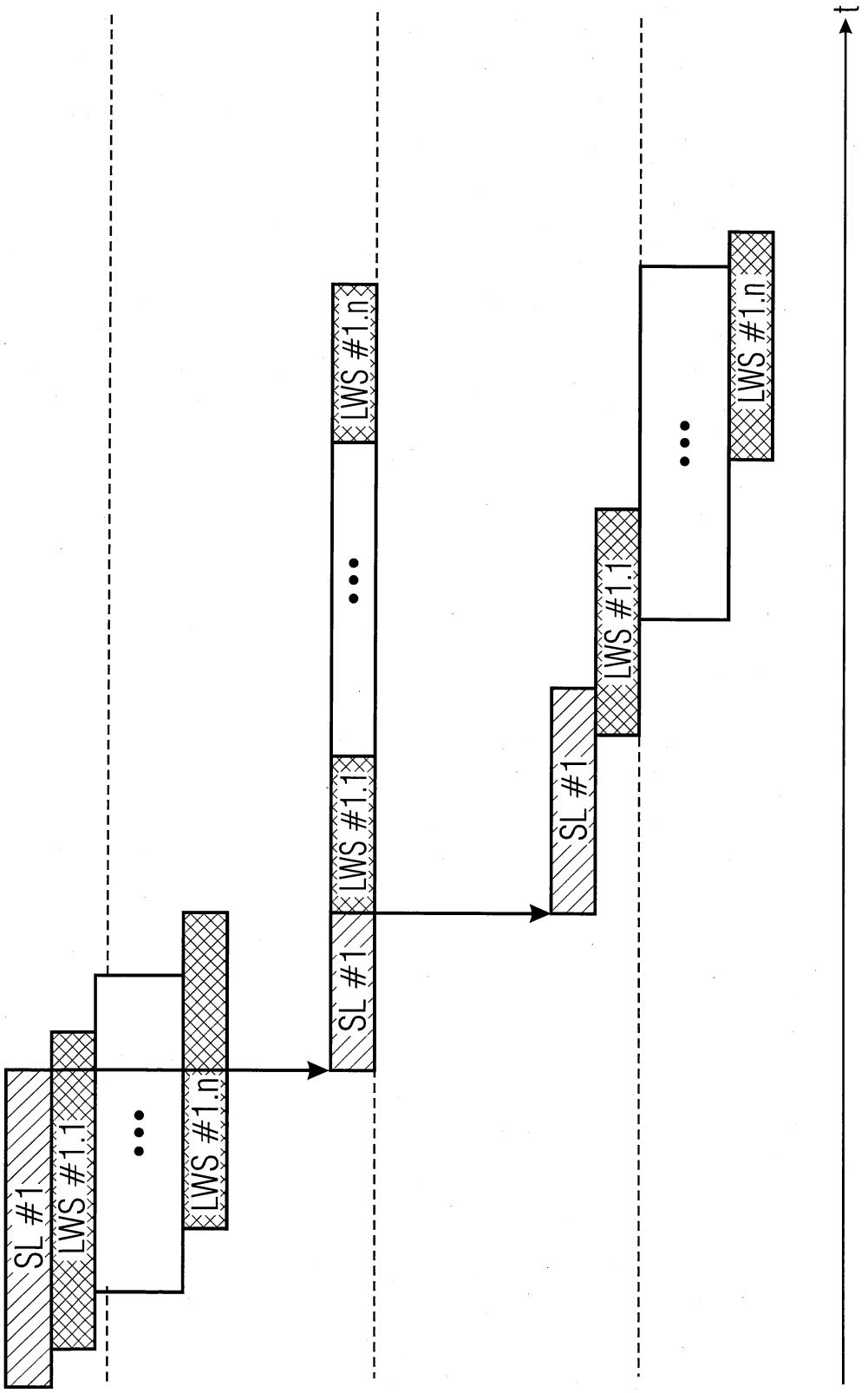

SL #1
LWS #1.1
LWS #1.n
SL #1
LWS #1.1
LWS #1.n
SL #1
LWS #1.1
LWS #1.n
t
mã hóa WPP
truyền dẫn
giải mã WPP

FIG 19

RS viết tắt của regular slice
DS viết tắt của dependent slice

các lát đều (RS) với các lát phụ thuộc (DS)

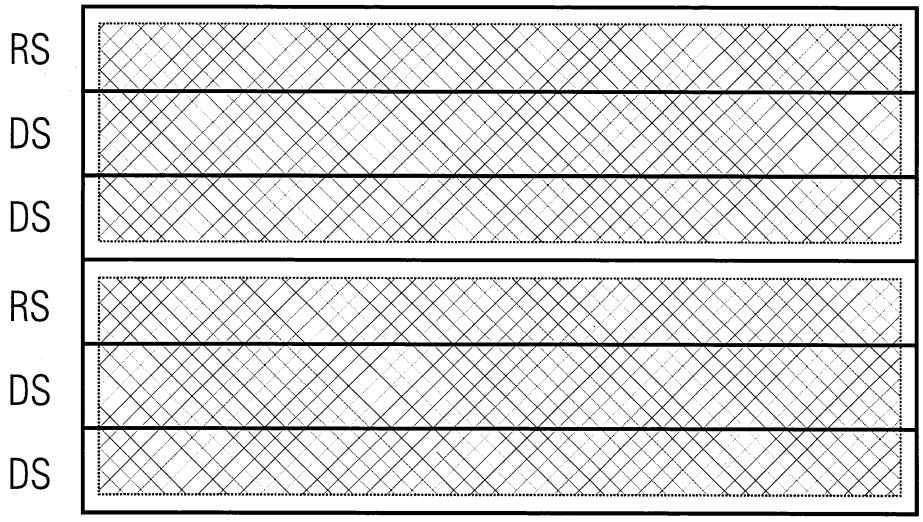


FIG 20

slice_header( ) {	Descriptor
first_slice_in_pic_flag	u(1)
pic_parameter_set_id	ue(v)
if(!first_slice_in_pic_flag)	
slice_address	u(v)
if(dependent_slice_enabled_flag && !first_slice_in_pic_flag)	
dependent_slice_flag	u(1)
if(!dependent_slice_flag) {	
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if(RapPicFlag) {	
rap_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
}	
if(!IdrPicFlag) {	
• • •	
}	
if(tiles_or_entropy_coding_sync_idc = = 1 || tiles_or_entropy_coding_sync_idc = = 2) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
if(slice_header_extension_present_flag) {	
slice_header_extension_length	ue(v)
for(i = 0; i < slice_header_extension_length; i++)	
slice_header_extension_data_byte	u(8)
}	
byte_alignment( )	
}	

400

102

FIG 21

pic_parameter_set_rbsp( ) {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
sign_data_hiding_flag	u(1)
if(sign_data_hiding _flag)	
sign_hiding_threshold	u(4)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
[Ed. (BB): not present in HM software]	
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
slice_granularity	u(2)
diff_cu_qp_delta_depth	ue(v)
cb_qp_offset	se(v)
cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
output_flag_present_flag	u(1)
transquant_bypass_enable_flag	u(1)
dependent_slice_enabled_flag	u(1)
tiles_or_entropy_coding_sync_idc	u(2)
if(tiles_or_entropy_coding_sync_idc == 1) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles_or_entropy_coding_sync_idc == 3)	
cabac_independent_flag	u(1)
• • •	
}	

FIG 22

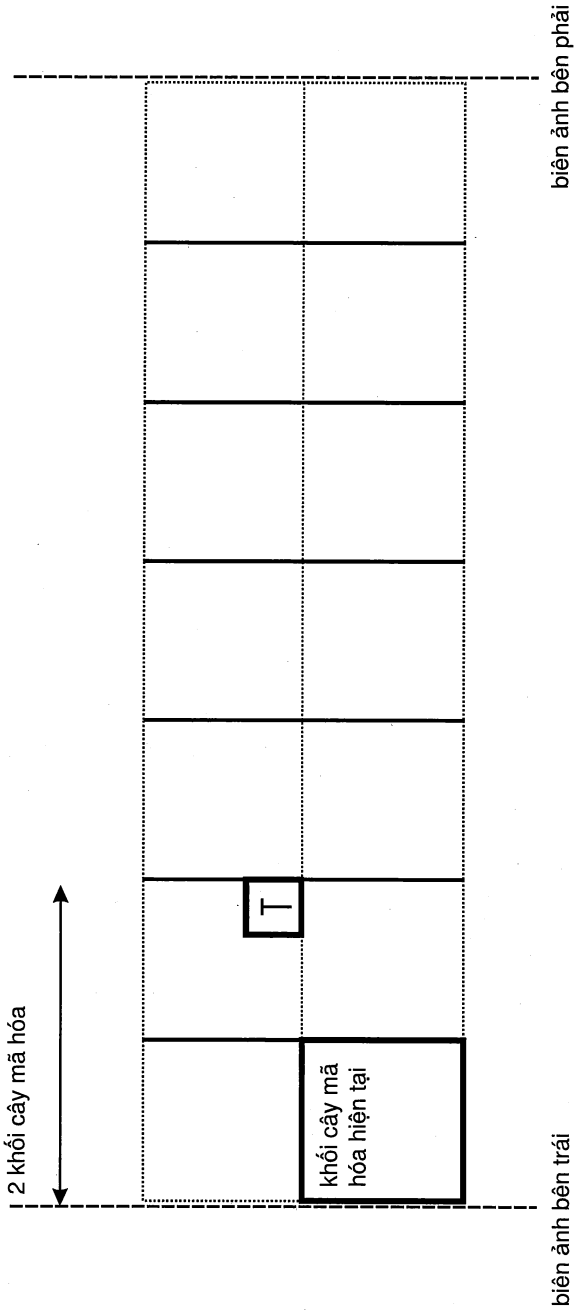

biên ảnh bên phải
T
2 khối cây mã hóa
khối cây mã hóa hiện tại
biên ảnh bên trái

FIG 23

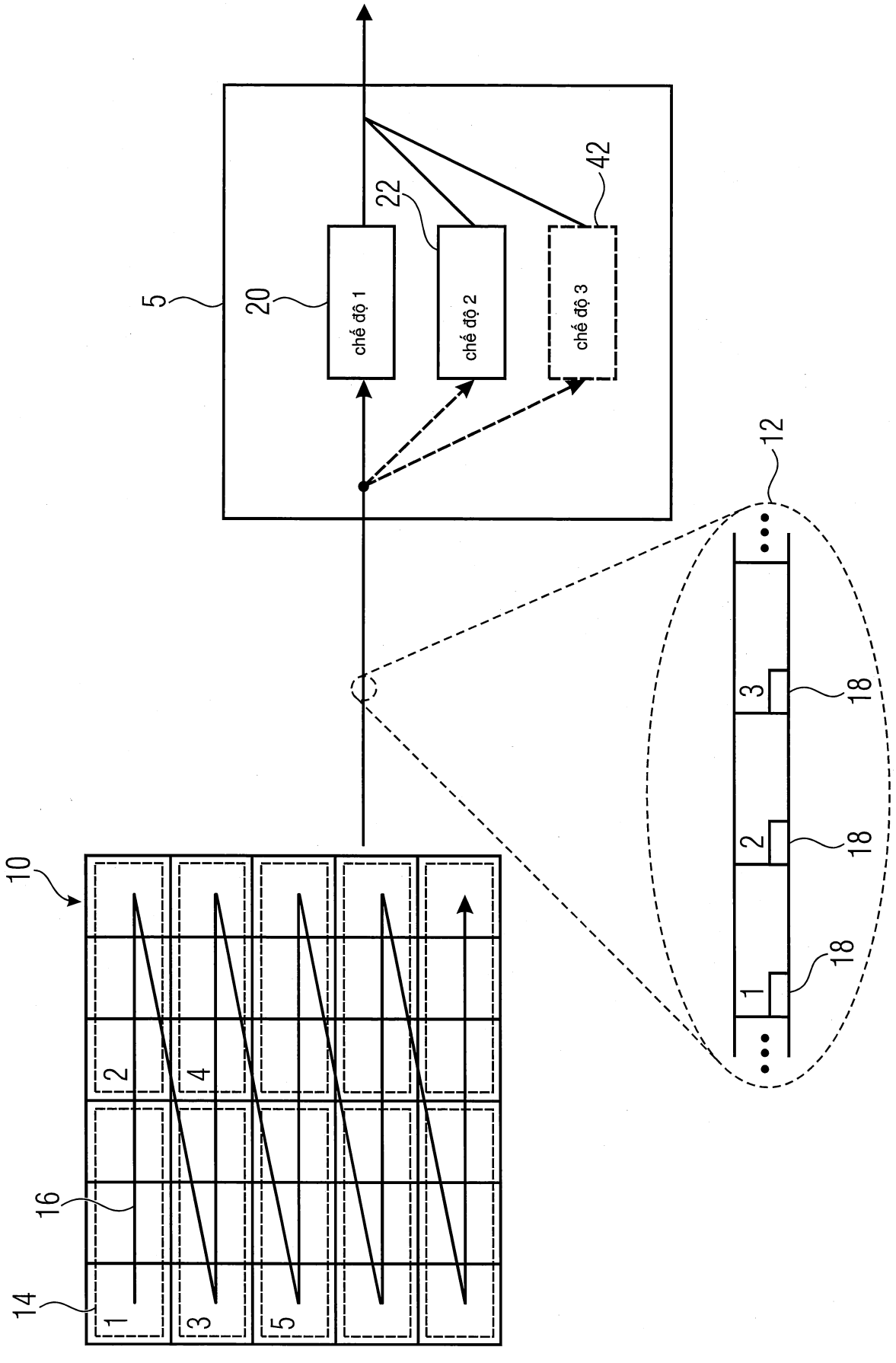


FIG 24

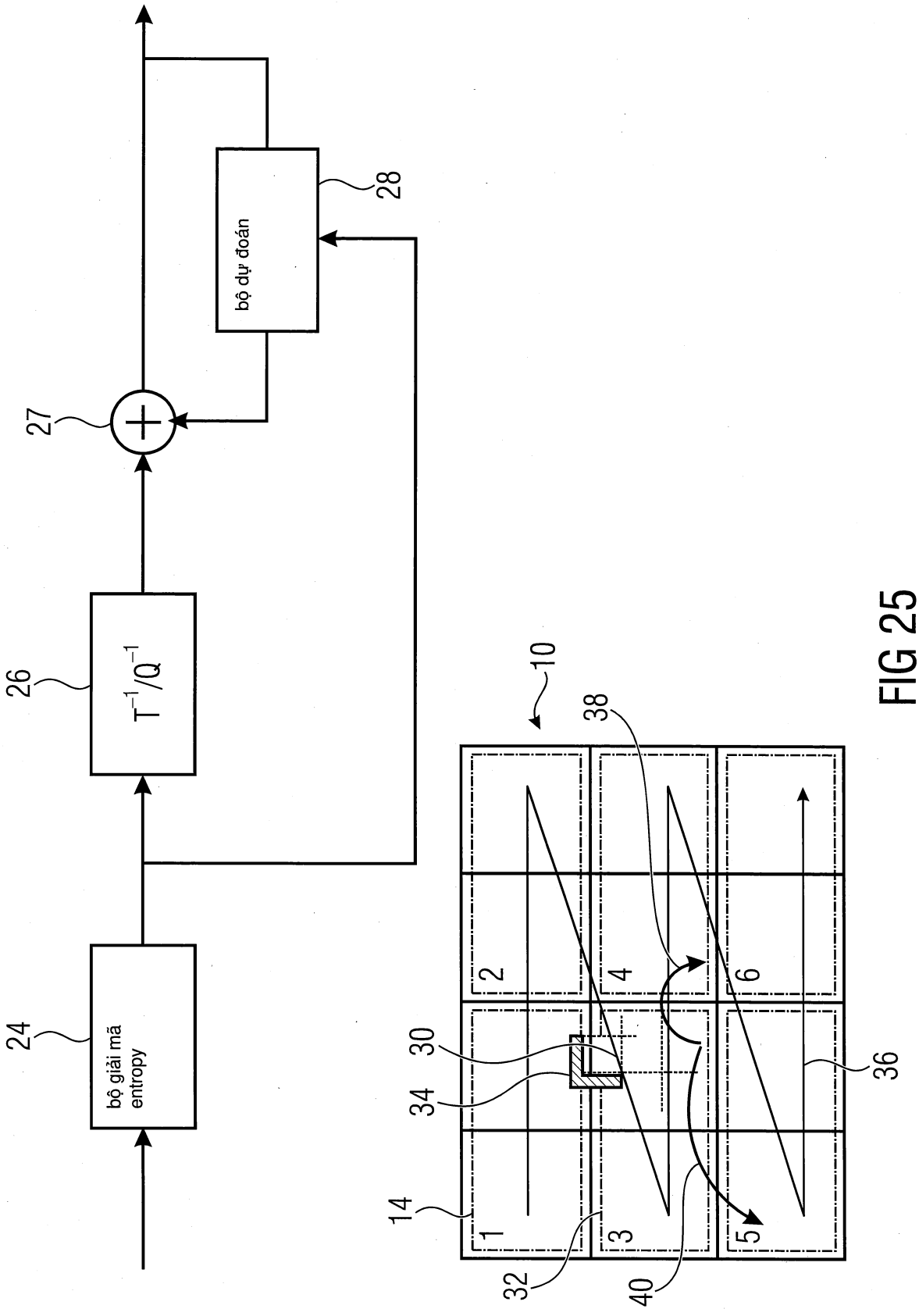

bộ giải mã entropy
24
T^{-1}/Q^{-1}
26
27
bộ dự đoán
28
10
38
30
34
14
32
40
36
1
2
3
4
5
6

FIG 25

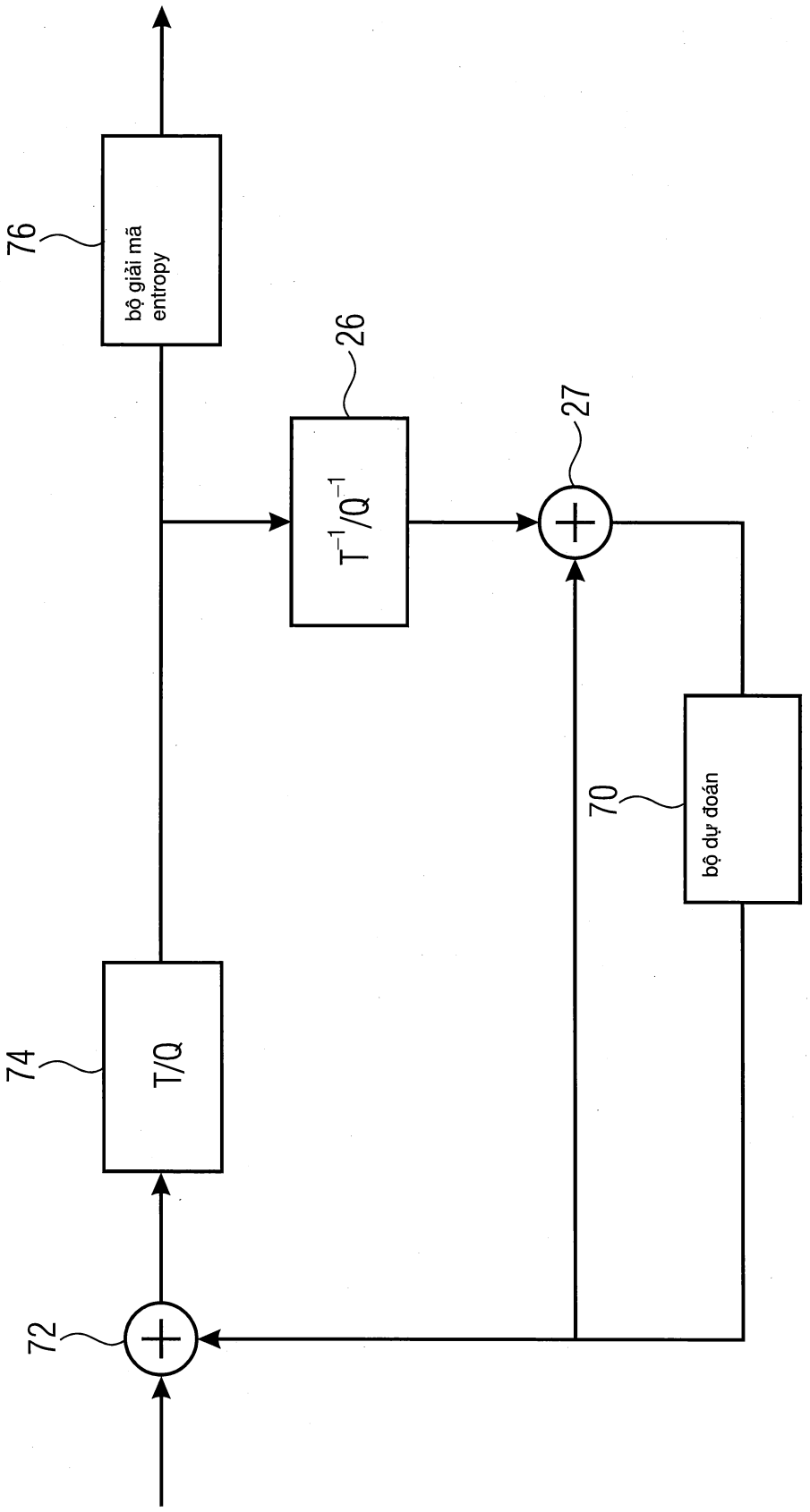


FIG 26

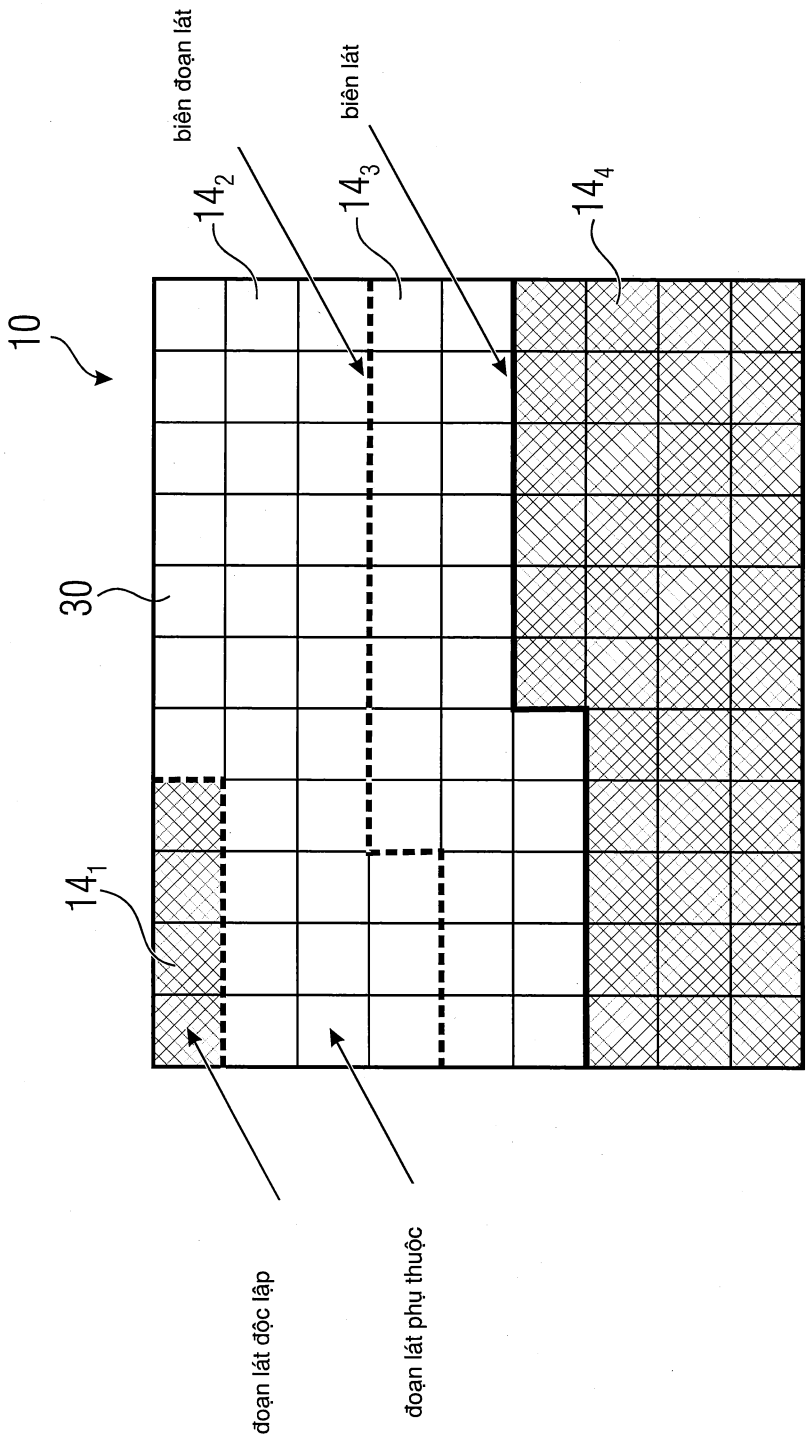

10
30
14_1
14_2
14_3
14_4
biên đoạn lát
biên lát
đoạn lát độc lập
đoạn lát phụ thuộc

FIG 27

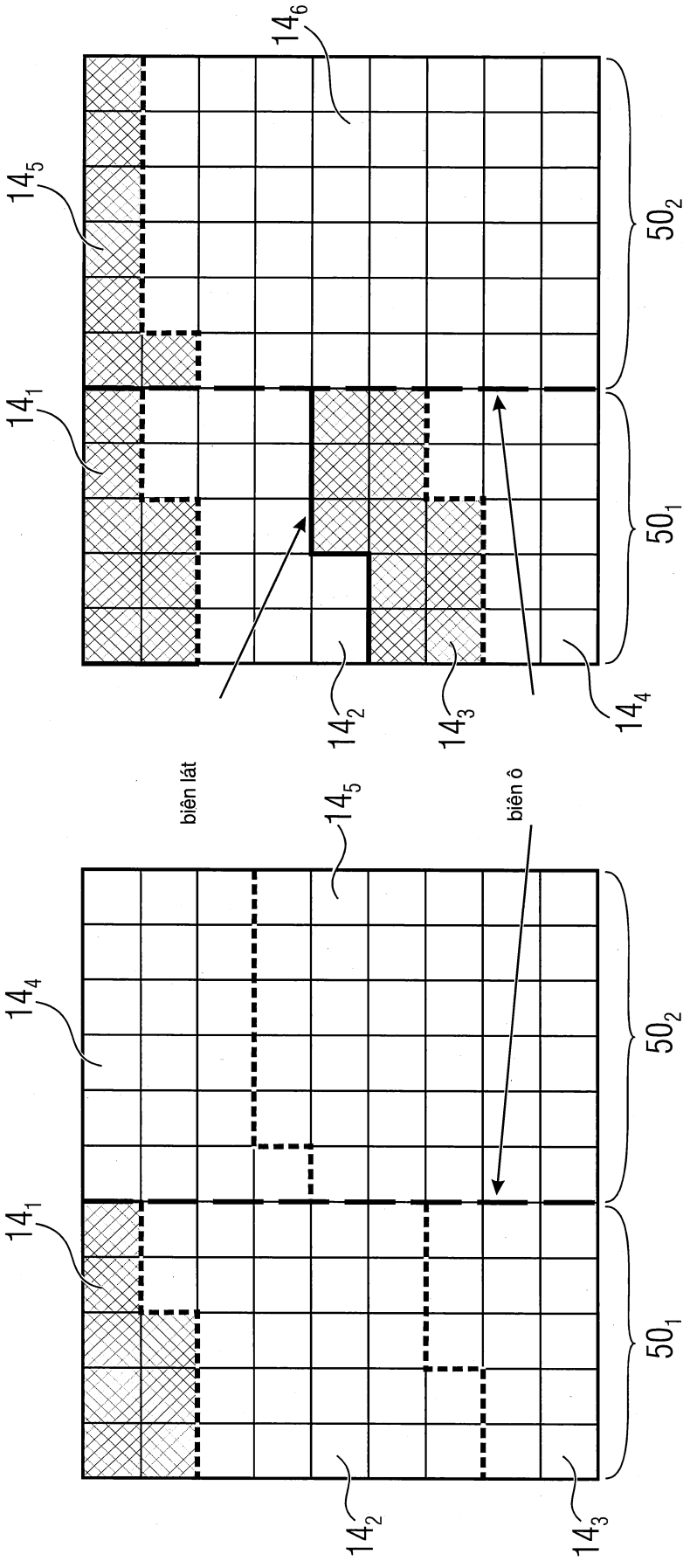


FIG 28B

FIG 28A

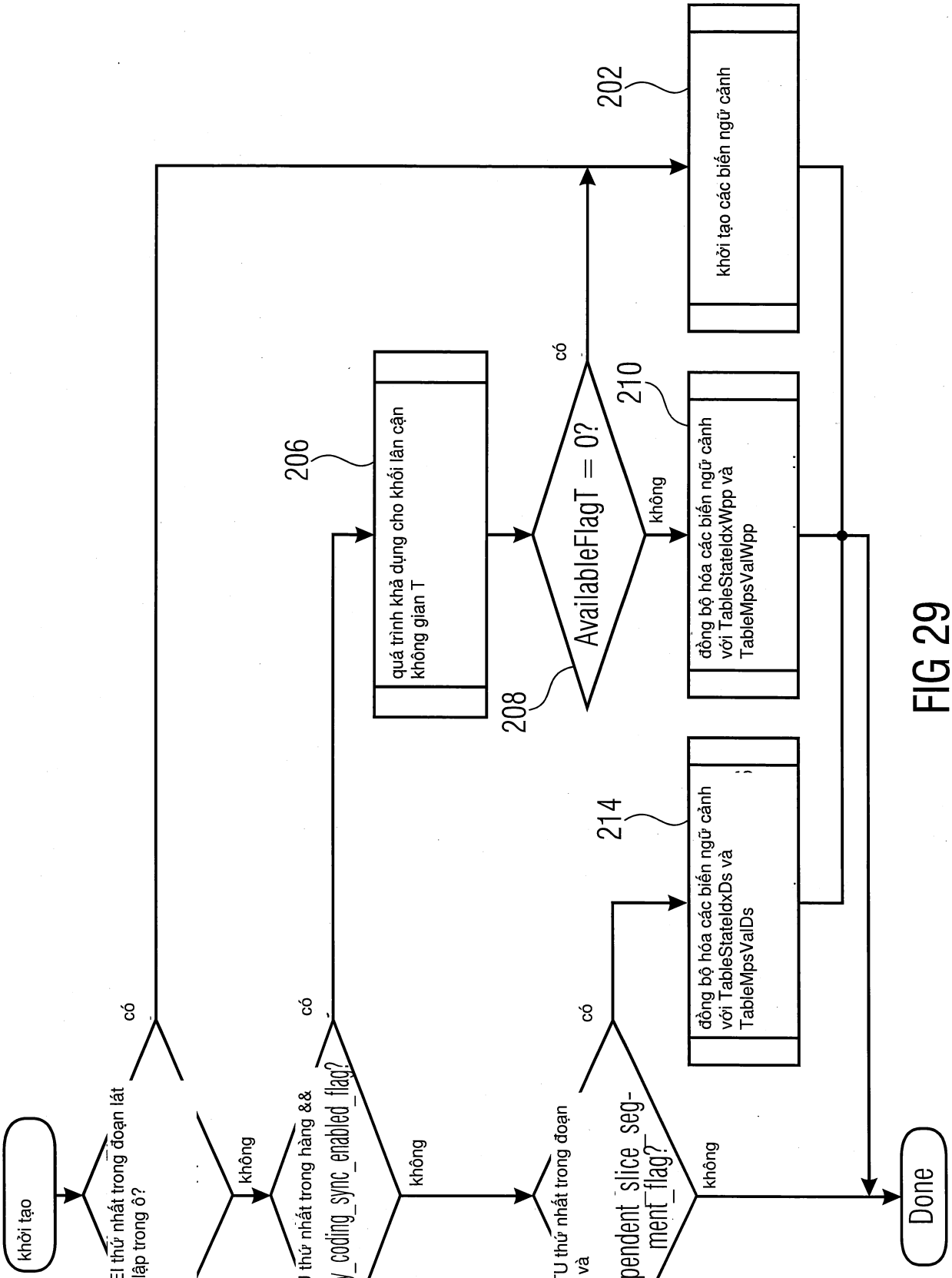

khởi tạo
200
synEI thứ nhất trong đoạn lát độc lập trong ô?
có
không
204
CTU thứ nhất trong hàng && entropy_coding_sync_enabled_flag?
có
không
212
CTU thứ nhất trong đoạn lát và dependent_slice_segment_flag?
có
không
202
khởi tạo các biến ngữ cảnh
206
quá trình khả dụng cho khối lân cận không gian T
208
AvailableFlagT = 0?
có
không
210
đồng bộ hóa các biến ngữ cảnh với TableStateIdxWpp và TableMpsValWpp
214
đồng bộ hóa các biến ngữ cảnh với TableStateIdxDs và TableMpsValDs
Done

FIG 29

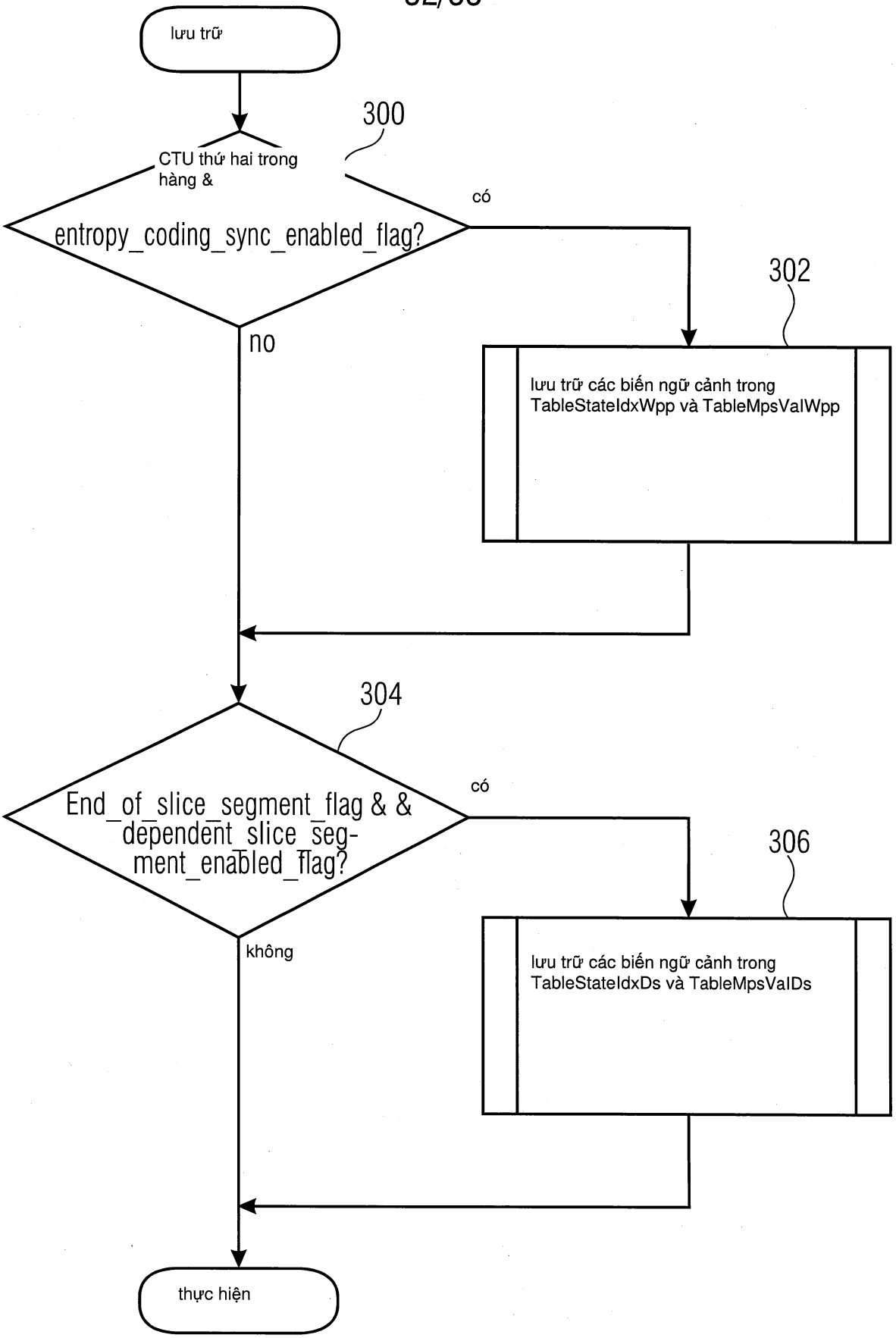

lưu trữ
300
CTU thứ hai trong hàng &
entropy_coding_sync_enabled_flag?
có
no
302
lưu trữ các biến ngữ cảnh trong TableStateIdxWpp và TableMpsValWpp
304
End_of_slice_segment_flag & & dependent_slice_segment_enabled_flag?
có
không
306
lưu trữ các biến ngữ cảnh trong TableStateIdxDs và TableMpsValDs
thực hiện

FIG 30

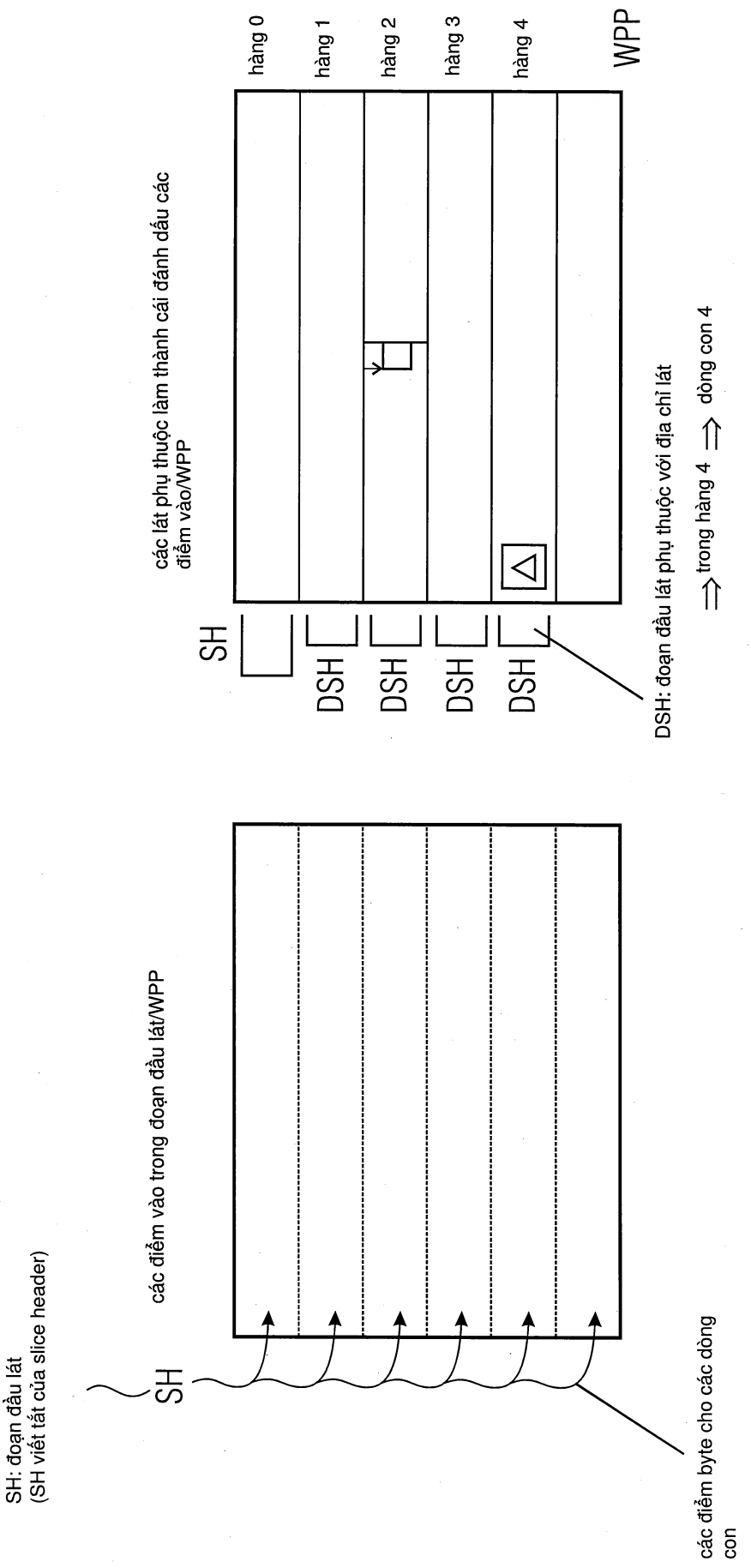

hàng 0
hàng 1
hàng 2
hàng 3
hàng 4
WPP
SH
DSH
DSH
DSH
DSH
các lát phụ thuộc làm thành cái đánh dấu các điểm vào/WPP
DSH: đoạn đầu lát phụ thuộc với địa chỉ lát
⟹ trong hàng 4 ⟹ dòng con 4
SH: đoạn đầu lát
(SH viết tắt của slice header)
các điểm vào trong đoạn đầu lát/WPP
SH
các điểm byte cho các dòng con

FIG 31