(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) **1-0052078**

(51)[2022.01] **H04N 19/70**; H04N 19/12; H04N 19/186; H04N 19/103; H04N 19/136

(13) **B**

(21) 1-2022-06528
(22) 29/03/2021
(86) PCT/US2021/024673 29/03/2021
(87) WO2021/195628 30/09/2021
(30) 63/001,235 27/03/2020 US
(45) 25/09/2025 450
(43) 26/12/2022 417A
(73) BEIJING DAJIA INTERNET INFORMATION TECHNOLOGY CO., LTD. (CN)
Room 101D1-7, 1st Floor, Building 1, No. 6, Shangdi West Road, Haidian District, Beijing, 100085, China
(72) JHU, Hong-Jheng (CN); WANG, Xianglin (US); XIU, Xiaoyu (CN); CHEN, Yi-wen (CN); MA, Tsung-chuan (CN); CHEN, Wei (CN); YU, Bing (CN).
(74) Công ty TNHH Trường Xuân (AGELESS CO.,LTD.)

(54) PHƯƠNG PHÁP GIẢI MÃ DỮ LIỆU VIĐEO, THIẾT BỊ ĐIỆN TỬ, VÀ PHƯƠNG TIỆN LƯU TRỮ KHÔNG NHẤT THỜI CÓ THỂ ĐỌC ĐƯỢC TRÊN MÁY TÍNH

(21) 1-2022-06528

(57) Sáng chế đề xuất thiết bị điện tử, phương pháp giải mã dữ liệu viđeo và phương tiện lưu trữ không nhất thời có thể đọc được trên máy tính. Phương pháp này bao gồm các bước: nhận, từ luồng bit, nhiều phần tử cú pháp kết hợp với đơn vị lập mã, trong đó nhiều phần tử cú pháp chỉ ra loại cây lập mã của đơn vị lập mã, và liệu chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã hay không; và phù hợp với việc xác định rằng loại cây lập mã của đơn vị lập mã là cây đơn, và chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã: vô hiệu hóa chế độ bảng màu cho đơn vị lập mã khi đơn vị lập mã có kích thước bằng hoặc nhỏ hơn ngưỡng định trước. Theo vài phương án, vô hiệu hóa chế độ bảng màu cho đơn vị lập mã bao gồm: vô hiệu hóa chế độ bảng màu cho cả thành phần luma và thành phần chroma của đơn vị lập mã, hoặc vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã.

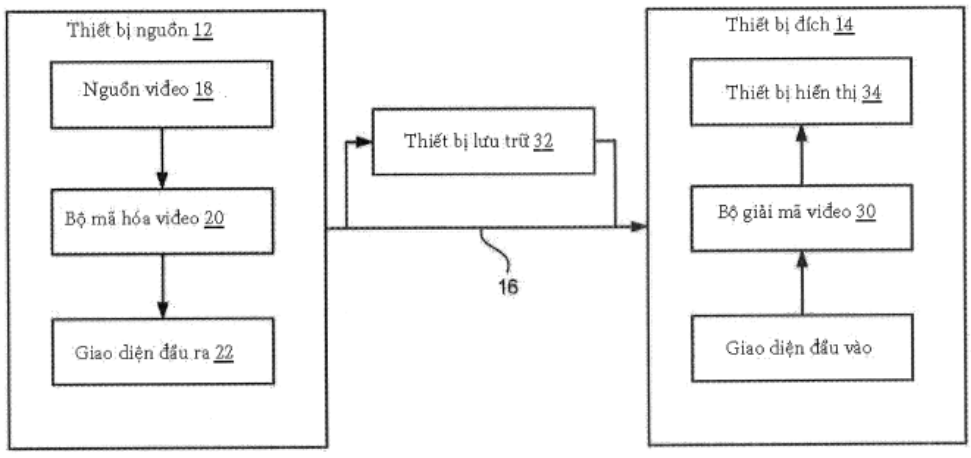


FIG. 1

Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến lập mã và nén dữ liệu viđeo, và cụ thể là đề cập đến phương pháp giải mã dữ liệu viđeo, thiết bị điện tử và phương tiện lưu trữ không nhất thời có thể đọc được trên máy tính.

Tình trạng kỹ thuật của sáng chế

Viđeo kỹ thuật số được hỗ trợ bởi nhiều thiết bị điện tử, như tivi kỹ thuật số, máy tính xách tay hoặc máy tính để bàn, máy tính bảng, máy ảnh kỹ thuật số, thiết bị ghi kỹ thuật số, đầu phát truyền thông kỹ thuật số, máy chơi game viđeo, điện thoại thông minh, thiết bị hội nghị truyền hình viđeo, thiết bị phát trực tuyến viđeo, v.v. Các thiết bị điện tử truyền, nhận, mã hóa, giải mã và/hoặc lưu trữ dữ liệu viđeo kỹ thuật số bằng cách triển khai các tiêu chuẩn nén/giải nén viđeo được xác định bởi tiêu chuẩn MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, lập mã viđeo nâng cao (Advanced Video Coding: AVC), lập mã viđeo hiệu năng cao (High Efficiency Video Coding: HEVC), và lập mã viđeo đa năng (Versatile Video Coding: VVC). Nén viđeo thường bao gồm thực hiện dự đoán không gian (nội khung) và/hoặc dự đoán thời gian (liên khung) để giảm hoặc loại bỏ sự dư thừa vốn có trong dữ liệu viđeo. Đối với lập mã viđeo dựa vào khối, khung viđeo được phân vùng thành một hoặc nhiều lát (slice), mỗi lát có nhiều khối viđeo, mà có thể cũng được đề cập đến là đơn vị cây lập mã (coding tree unit: CTU). Mỗi CTU có thể chứa một đơn vị lập mã (CU) hoặc chia đệ quy thành các CU nhỏ hơn cho đến khi đạt được kích thước CU tối thiểu xác định trước. Mỗi CU (còn được đặt tên là CU lá) chứa một hoặc nhiều đơn vị biến đổi (bộ phận biến đổi: TU) và mỗi CU cũng chứa một hoặc nhiều đơn vị dự đoán (prediction unit: PU). Mỗi CU có thể được lập mã ở chế độ liên khung, nội khung hoặc IBC. Khối viđeo trong lát được lập mã nội khung (I) của khung viđeo được mã hóa sử dụng dự đoán không gian đối với các mẫu tham chiếu trong các khối lân cận trong cùng một khung viđeo. Các khối viđeo trong lát lập mã liên khung (P hoặc B) của khung viđeo

có thể sử dụng dự đoán không gian đối với các mẫu tham chiếu trong các khối lân cận trong cùng một khung viđeo hoặc dự đoán thời gian đối với các mẫu tham chiếu trong các khung viđeo tham chiếu trước đó và/hoặc tương lai khác.

Dự đoán không gian và thời gian căn cứ vào khối tham chiếu mà được mã hóa trước đó, ví dụ, khối lân cận, dẫn đến khối dự đoán cho khối viđeo hiện thời được lập mã. Quy trình tìm kiếm khối tham chiếu có thể được hoàn thành bởi thuật toán đối sánh khối. Dữ liệu dư thể hiện sự khác biệt pixel giữa khối hiện thời được lập mã và khối dự đoán được đề cập đến là khối dư hoặc lỗi dự đoán. Khối lập mã liên (inter) được mã hóa theo vectơ chuyển động mà chỉ đến khối tham chiếu trong khung tham chiếu tạo ra khối dự đoán, và khối dư. Quy trình xác định vectơ chuyển động thường được đề cập đến là ước lượng chuyển động. Khối lập mã nội (intra) được mã hóa theo chế độ dự đoán nội và khối dư. Để nén thêm, khối dư được biến đổi từ miền pixel đến miền biến đổi, ví dụ, miền tần số, tạo các hệ số biến đổi dư, mà sau đó có thể được lượng tử hóa. Hệ số biến đổi đã lượng tử hóa, đầu tiên được sắp xếp theo dãy hai chiều, có thể được quét để tạo ra vectơ một chiều của các hệ số biến đổi, và sau đó được mã háo entropy thành viđeo luồng bit để đạt được hiệu quả nén hơn nữa.

Sau đó, viđeo luồng bit đã mã hóa được lưu lại trong phương tiện lưu trữ có thể đọc được trên máy tính (ví dụ, bộ nhớ flash) được truy cập bởi một thiết bị điện tử khác có khả năng thu viđeo kỹ thuật số hoặc được truyền trực tiếp đến thiết bị điện tử theo cách có dây hoặc không dây. Sau đó, thiết bị điện tử sẽ thực hiện giải nén viđeo (đây là một quá trình ngược lại với quá trình nén viđeo được mô tả ở trên) nhờ, ví dụ, phân tích cú pháp luồng bit viđeo mã hóa để thu được các phần tử cú pháp từ luồng bit và tái cấu trúc dữ liệu viđeo kỹ thuật số thành định dạng gốc của nó từ luồng bit viđeo được mã hóa căn cứ vào ít nhất một phần trên các phần tử cú pháp thu được từ luồng bit, và hiển thị dữ liệu viđeo kỹ thuật số được tái tạo trên màn hình của thiết bị điện tử.

Với chất lượng viđeo kỹ thuật số từ mức độ rõ nét cao, lên đến 4Kx2K hoặc thậm chí 8Kx4K, lượng dữ liệu viđeo được mã hóa/giải mã sẽ tăng lên theo cấp số nhân. Đó là một thách thức liên tục về cách dữ liệu viđeo có thể được mã hóa/giải mã hiệu quả hơn trong khi vẫn duy trì chất lượng hình ảnh của dữ liệu viđeo được giải mã.

Bản chất kỹ thuật của sáng chế

Sáng chế mô tả việc thực hiện liên quan đến mã hóa và giải mã dữ liệu viđeo và cụ thể hơn là hệ thống và phương pháp mã hóa và giải mã viđeo sử dụng chế độ bảng màu.

Theo khía cạnh thứ nhất của sáng chế, phương pháp giải mã dữ liệu viđeo bao gồm: nhận, từ luồng bit, nhiều phần tử cú pháp kết hợp với đơn vị lập mã, trong đó nhiều phần tử cú pháp chỉ ra loại cây lập mã của đơn vị lập mã, và liệu chế độ cây kép cục bộ (local dual tree) được kích hoạt cho đơn vị lập mã; và phù hợp với việc xác định rằng loại cây lập mã của đơn vị lập mã là cây đơn, và chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã: vô hiệu hóa chế độ bảng màu cho đơn vị lập mã khi đơn vị lập mã có kích thước bằng hoặc nhỏ hơn ngưỡng định trước.

Theo vài phương án, vô hiệu hóa chế độ bảng màu cho đơn vị lập mã bao gồm: vô hiệu hóa chế độ bảng màu cho cả thành phần luma và thành phần chroma của đơn vị lập mã.

Theo vài phương án, vô hiệu hóa chế độ bảng màu cho đơn vị lập mã bao gồm: vô hiệu hóa chế độ bảng màu chỉ cho thành phần chroma của đơn vị lập mã.

Theo khía cạnh thứ hai của sáng chế, thiết bị điện tử bao gồm một hoặc nhiều bộ xử lý, bộ nhớ và nhiều chương trình được lưu trong bộ nhớ. Các chương trình, khi được chạy bằng một hoặc nhiều bộ xử lý, làm cho thiết bị điện tử thực hiện phương pháp giải mã dữ liệu viđeo như được mô tả ở trên.

Theo khía cạnh thứ ba của sáng chế, phương tiện lưu trữ không nhất thời có thể đọc được trên máy tính lưu nhiều chương trình để chạy bằng thiết bị điện tử có một hoặc nhiều bộ xử lý. Các chương trình, khi được chạy bằng một hoặc nhiều bộ xử lý, làm cho thiết bị điện tử thực hiện phương pháp giải mã dữ liệu viđeo như mô tả ở trên.

Mô tả vắn tắt các hình vẽ

Các hình vẽ đi kèm, nằm trong sáng chế cung cấp sự hiểu biết thêm về việc thực hiện và được kết hợp ở đây và tạo thành một phần của sáng chế, minh họa việc thực hiện và cùng với phần mô tả để giải thích các nguyên tắc cơ bản. Các số tham chiếu giống nhau để chỉ các bộ phận tương đương nhau.

FIG. 1 là sơ đồ khối minh họa hệ thống mã hóa và giải mã viđeo điển hình phù hợp với vài phương án thực hiện của sáng chế.

FIG. 2 là sơ đồ khối minh họa bộ mã hóa viđeo điển hình phù hợp với vài phương án thực hiện của sáng chế.

FIG. 3 là sơ đồ khối minh họa bộ giải mã viđeo điển hình phù hợp với vài phương án thực hiện của sáng chế.

FIG. 4A đến 4E là sơ đồ khối minh họa cách một khung được phân vùng đệ quy thành nhiều khối viđeo có kích thước và hình dạng khác nhau phù hợp với vài phương án thực hiện của sáng chế.

FIG. 5A đến 5D là sơ đồ khối minh họa ví dụ về việc sử dụng bảng bảng màu để lập mã dữ liệu viđeo phù hợp với vài phương án thực hiện của sáng chế.

FIG. 6 là lưu đồ minh họa quy trình điển hình mà nhờ đó bộ giải mã viđeo thực hiện kỹ thuật giải mã dữ liệu viđeo phù hợp với vài phương án thực hiện của sáng chế.

FIG. 7 là sơ đồ khối minh họa một ví dụ về công cụ mã hóa số học nhị phân thích ứng với ngữ cảnh (Context-adaptive binary arithmetic coding: CABAC) phù hợp với vài phương án thực hiện của sáng chế.

Mô tả chi tiết của sáng chế

Giờ đây, việc tham khảo sẽ được thực hiện chi tiết ở các phương án cụ thể, các ví dụ được minh họa trong các hình vẽ đi kèm. Trong phần mô tả chi tiết sau đây, nhiều chi tiết cụ thể không giới hạn được trình bày để hỗ trợ việc hiểu sáng chế. Nhưng rõ ràng đối với người có hiểu biết trung bình trong lĩnh vực này là có thể sử dụng các phương án thay thế khác nhau mà không lệch khỏi phạm vi yêu cầu bảo hộ và đối tượng được thực hiện mà không cần các chi tiết cụ thể này. Ví dụ, người có hiểu biết trung bình trong lĩnh vực này sẽ hiểu rằng đối tượng được thể hiện ở đây có thể được thực hiện trên nhiều loại thiết bị điện tử có khả năng thu viđeo kỹ thuật số.

FIG. 1 là sơ đồ khối minh họa hệ thống điển hình 10 để mã hóa và giải mã khối viđeo song song phù hợp với vài phương án thực hiện của sáng chế. Như được thể hiện trong FIG. 1, hệ thống 10 bao gồm thiết bị nguồn 12 mà tạo ra và mã hóa dữ liệu viđeo được giải mã ở thời điểm muộn hơn bởi thiết bị đích 14. Thiết bị nguồn 12 và thiết bị

đích 14 có thể bao gồm bất kỳ thiết bị điện tử nào trong số nhiều loại thiết bị điện tử, bao gồm máy tính để bàn hoặc máy tính xách tay, máy tính bảng, điện thoại thông minh, hộp giải mã tín hiệu số, TV kỹ thuật số, máy ảnh, thiết bị hiển thị, đầu phát truyền thông kỹ thuật số, bảng điều khiển trò chơi viđeo, thiết bị phát viđeo hoặc tương tự. Trong vài phương án thực hiện, thiết bị nguồn 12 và thiết bị đích 14 được trang bị khả năng truyền thông không dây.

Trong vài phương án thực hiện, thiết bị đích 14 có thể nhận dữ liệu viđeo đã được mã hóa để được giải mã thông qua liên kết 16. Liên kết 16 có thể bao gồm loại phương tiện truyền thông bất kỳ hoặc thiết bị có khả năng di chuyển dữ liệu viđeo được mã hóa từ thiết bị nguồn 12 đến thiết bị đích 14. Trong một ví dụ, liên kết 16 có thể bao gồm phương tiện truyền thông để kích hoạt thiết bị nguồn 12 truyền dữ liệu viđeo đã được mã hóa trực tiếp tới thiết bị đích 14 theo thời gian thực. Dữ liệu viđeo đã được mã hóa có thể được môđun hóa theo tiêu chuẩn truyền thông, như giao thức truyền thông không dây, và được truyền tới thiết bị đích 14. Phương tiện truyền thông có thể bao gồm phương tiện truyền thông có dây hoặc không dây bất kỳ, như phổ tần số vô tuyến (radio frequency: RF) hoặc một hoặc nhiều đường truyền vật lý. Phương tiện truyền thông có thể tạo ra một phần của mạng dựa trên gói, như mạng vùng cục bộ, mạng vùng rộng, hoặc mạng toàn cầu như Internet. Phương tiện truyền thông có thể bao gồm bộ định tuyến, bộ chuyển mạch, trạm gốc hoặc bất kỳ thiết bị nào khác có thể hữu ích để tạo điều kiện truyền thông từ thiết bị nguồn 12 đến thiết bị đích 14.

Theo vài phương án thực hiện khác, dữ liệu viđeo đã được mã hóa có thể được truyền từ giao diện đầu ra 22 đến thiết bị lưu trữ 32. Sau đó, dữ liệu viđeo đã được mã hóa trong thiết bị lưu trữ 32 có thể được truy cập bằng thiết bị đích 14 thông qua giao diện đầu vào 28. Thiết bị lưu trữ 32 có thể bao gồm bất kỳ trong số nhiều phương tiện lưu trữ dữ liệu truy cập cục bộ hoặc phân phối như ổ cứng, đĩa Blu-ray, DVD, CD-ROM, bộ nhớ flash, bộ nhớ khả biến hoặc bộ nhớ không khả biến, hoặc phương tiện lưu trữ số bất kỳ để lưu trữ dữ liệu viđeo đã được mã hóa. Theo ví dụ khác, thiết bị lưu trữ 32 có thể tương ứng với máy chủ tệp hoặc thiết bị lưu trữ trung gian khác mà có thể giữ dữ liệu viđeo đã được mã hóa được tạo ra bởi thiết bị nguồn 12. Thiết bị

đích 14 có thể truy cập dữ liệu viđeo được lưu từ thiết bị lưu trữ 32 thông qua phát trực tuyến hoặc tải xuống. Máy chủ tệp có thể là loại máy tính bất kỳ có khả năng lưu dữ liệu viđeo đã được mã hóa và truyền dữ liệu viđeo đã được mã hóa này đến thiết bị đích 14. Máy chủ tệp điển hình bao gồm máy chủ web, (ví dụ, dùng cho website), máy chủ FTP, thiết bị lưu trữ gắn kèm mạng (network attached storage: NAS), hoặc ổ đĩa cục bộ. Thiết bị đích 14 có thể truy cập dữ liệu viđeo đã được mã hóa thông qua sự kết nối dữ liệu tiêu chuẩn bao gồm, kênh không dây (ví dụ, kết nối Wi-Fi), kết nối có dây (ví dụ, DSL, modem cáp, v.v.), hoặc kết hợp cả hai mà thích hợp cho việc truy cập dữ liệu viđeo đã được mã hóa được lưu trên máy chủ tệp. Việc truyền dữ liệu viđeo đã được mã hóa từ thiết bị lưu trữ 32 có thể là truyền trực tuyến, truyền tải xuống hoặc kết hợp cả hai.

Như được thể hiện trong FIG. 1, thiết bị nguồn 12 bao gồm nguồn viđeo 18, bộ mã hóa viđeo 20 và giao diện đầu ra 22. Nguồn viđeo 18 có thể bao gồm nguồn như thiết bị quay viđeo, ví dụ, camera quay viđeo, kho lưu trữ viđeo chứa viđeo đã quay trước đó, giao diện nguồn cấp dữ liệu viđeo để nhận viđeo từ nhà cung cấp nội dung viđeo và/hoặc hệ thống đồ họa máy tính để tạo ra dữ liệu đồ họa máy tính, hoặc kết hợp các nguồn này. Là một ví dụ, nếu nguồn viđeo 18 là camera quay viđeo của hệ thống giám sát an ninh, thiết bị nguồn 12 và thiết bị đích 14 có thể tạo ra cuộc gọi qua camera hoặc cuộc gọi viđeo. Tuy nhiên, các phương án thực hiện được mô tả trong sáng chế này có khả năng ứng dụng cho lập mã viđeo chung, và có thể được áp dụng cho ứng dụng không dây và/hoặc có dây.

Viđeo đã quay, chụp trước hoặc do máy tính tạo có thể được mã hóa bởi bộ mã hóa viđeo 20. Dữ liệu viđeo đã được mã hóa có thể được truyền trực tiếp đến thiết bị đích 14 thông qua giao diện đầu ra 22 của thiết bị nguồn 12. Dữ liệu viđeo đã được mã hóa có thể cũng (hoặc theo cách khác) được lưu lên thiết bị lưu trữ 32 để truy cập sau bằng thiết bị đích 14 hoặc thiết bị khác, để giải mã và/hoặc phát lại. Giao diện đầu ra 22 có thể còn bao gồm modem và/hoặc bộ phát.

Thiết bị đích 14 bao gồm giao diện đầu vào 28, bộ giải mã viđeo 30, và thiết bị hiển thị 34. Giao diện đầu vào 28 có thể bao gồm bộ phận nhận và/hoặc modem và

nhận dữ liệu viđeo đã được mã hóa qua liên kết 16. Dữ liệu viđeo đã được mã hóa được truyền thông qua liên kết 16, hoặc được cung cấp trên thiết bị lưu trữ 32, có thể bao gồm nhiều phần tử cú pháp được tạo ra bởi bộ mã hóa viđeo 20 để dùng bởi bộ giải mã viđeo 30 trong giải mã dữ liệu viđeo. Phần tử cú pháp này có thể nằm trong dữ liệu viđeo đã được mã hóa được truyền trên phương tiện truyền thông, được lưu trong phương tiện lưu trữ, hoặc được lưu ở máy chủ tệp.

Trong vài phương án thực hiện, thiết bị đích 14 có thể bao gồm thiết bị hiển thị 34, có thể là thiết bị hiển thị tích hợp và thiết bị hiển thị bên ngoài mà được tạo cấu hình để giao tiếp với thiết bị đích 14. Thiết bị hiển thị 34 hiển thị dữ liệu viđeo được giải mã cho người dùng, và có thể bao gồm bất kỳ trong số nhiều thiết bị hiển thị như màn hình tinh thể lỏng (LCD), màn hình plasma, màn hình điốt phát quang hữu cơ (OLED) hoặc loại thiết bị hiển thị khác.

Bộ mã hóa viđeo 20 và bộ giải mã viđeo 30 có thể vận hành theo tiêu chuẩn độc quyền hoặc tiêu chuẩn công nghiệp, như VVC, HEVC, MPEG-4, Part 10, tiêu chuẩn lập mã viđeo nâng cao (Advanced Video Coding: AVC), hoặc các phần mở rộng của các tiêu chuẩn đó. Cần hiểu rằng sáng chế này không giới hạn ở tiêu chuẩn lập mã/giải mã viđeo cụ thể và có thể áp dụng được cho tiêu chuẩn lập mã/giải mã viđeo khác. Thường dự tính rằng bộ mã hóa viđeo 20 của thiết bị nguồn 12 có thể được tạo cấu hình để mã hóa dữ liệu viđeo theo tiêu chuẩn bất kỳ ở hiện tại hoặc trong tương lai. Tương tự, cũng dự tính chung rằng bộ giải mã viđeo 30 của thiết bị đích 14 có thể được tạo cấu hình để giải mã dữ liệu viđeo theo tiêu chuẩn bất kỳ ở hiện tại hoặc trong tương lai.

Bộ mã hóa viđeo 20 và bộ giải mã viđeo 30, mỗi trong số chúng có thể được triển khai dưới dạng bất kỳ trong số nhiều hệ mạch của bộ mã hóa, như một hoặc nhiều bộ vi xử lý, bộ xử lý tín hiệu kỹ thuật số (DSP), mạch tích hợp dành riêng cho ứng dụng (ASIC), mảng cổng có thể lập trình trường (FPGA), logic rời rạc, phần mềm, phần cứng, phần sụn hoặc bất kỳ tổ hợp nào của chúng. Khi được triển khai một phần trong phần mềm, một thiết bị điện tử có thể lưu trữ các lệnh cho phần mềm trong một phương tiện máy tính đọc được phù hợp, không nhất thời và thực hiện các lệnh trong

phần cứng bằng cách sử dụng một hoặc nhiều bộ xử lý để thực hiện thao tác lập mã/giải mã viđeo được bộc lộ trong sáng chế này. Mỗi trong số bộ mã hóa viđeo 20 và bộ giải mã viđeo 30 có thể có trong một hoặc nhiều bộ mã hóa hoặc bộ giải mã, một trong số đó có thể được tích hợp như một phần của bộ mã hóa/giải mã kết hợp (CODEC) trong một thiết bị tương ứng.

FIG. 2 là sơ đồ khối minh họa bộ mã hóa viđeo điển hình 20 phù hợp với vài phương án thực hiện được mô tả trong sáng chế này. Bộ mã hóa viđeo 20 có thể thực hiện lập mã dự đoán liên và nội khối viđeo trong các khung viđeo. Lập mã dự đoán nội khối dựa vào dự đoán không gian để giảm hoặc loại bỏ dư thừa không gian trong dữ liệu viđeo trong một khung viđeo hoặc hình ảnh nhất định. Mã hóa dự đoán liên dựa vào dự đoán thời gian để giảm hoặc loại bỏ dư thừa thời gian trong dữ liệu viđeo trong các khung viđeo liền kề hoặc hình ảnh của một chuỗi viđeo.

Như được thể hiện trong FIG. 2, bộ mã hóa viđeo 20 bao gồm bộ nhớ dữ liệu viđeo 40, bộ phận xử lý dự đoán 41, đệm hình ảnh được giải mã (decoded picture buffer: DPB) 64, bộ cộng 50, bộ phận xử lý biến đổi 52, đơn vị lượng tử hóa 54, và đơn vị mã hóa entropy 56. Bộ phận xử lý dự đoán 41 còn bao gồm bộ phận ước lượng chuyển động 42, bộ phận bù chuyển động 44, bộ phận phân vùng 45, bộ phận xử lý dự đoán nội 46, và đơn vị (bộ phận) sao chép khối (BC) nội 48. Theo vài phương án thực hiện, bộ mã hóa viđeo 20 còn bao gồm bộ lượng tử hóa nghịch đảo 58, bộ phận xử lý biến đổi nghịch đảo 60, và bộ cộng 62 dùng cho tái cấu trúc khối viđeo. Bộ lọc khử khối (không được thể hiện) có thể được định vị giữa bộ cộng 62 và DPB 64 để lọc ranh giới khối để loại bỏ các thành phần lạ có khối lớn khỏi viđeo được tái cấu trúc. Bộ lọc trong vòng (không được thể hiện) cũng có thể được sử dụng ngoài bộ lọc khử khối để lọc đầu ra của bộ cộng 62. Bộ mã hóa viđeo 20 có thể ở dạng một đơn vị phần cứng cố định hoặc có thể lập trình được hoặc có thể được chia cho một hoặc nhiều bộ phần cứng lập trình được hoặc cố định được minh họa.

Bộ nhớ dữ liệu viđeo 40 có thể lưu dữ liệu viđeo để mã hóa được bằng các bộ phận của bộ mã hóa viđeo 20. Dữ liệu viđeo trong bộ nhớ dữ liệu viđeo 40 có thể thu được, ví dụ, từ nguồn viđeo 18. DPB 64 là đệm mà lưu dữ liệu viđeo tham chiếu để sử

dụng trong mã hóa dữ liệu viđeo bằng bộ mã hóa viđeo 20 (ví dụ, chế độ lập mã dự đoán liên hoặc nội). Bộ nhớ dữ liệu viđeo 40 và DPB 64 có thể tạo ra bằng bất kỳ trong số nhiều thiết bị nhớ. Trong nhiều ví dụ, bộ nhớ dữ liệu viđeo 40 có thể on-chip (trên chip) với các thành phần khác của bộ mã hóa viđeo 20, hoặc off-chip (ngoài chip) liên quan đến các thành phần đó.

Như được thể hiện trong FIG. 2, sau khi nhận dữ liệu viđeo, bộ phận phân vùng 45 trong bộ phận xử lý dự đoán 41 phân vùng dữ liệu viđeo thành khối viđeo. Sự phân vùng này có thể cũng bao gồm phân vùng khung viđeo thành các lát (slice), ngói (tile) hoặc đơn vị lập mã (CU) lớn hơn khác theo cấu trúc phân tách được định trước như cấu trúc cây tứ phân kết hợp với dữ liệu viđeo. Khung viđeo có thể được chia thành nhiều khối viđeo (hoặc tập hợp của khối viđeo được đề cập đến là ngói). Bộ phận xử lý dự đoán 41 có thể lựa chọn một trong số nhiều chế độ lập mã dự đoán khả thi, như một trong số nhiều chế độ lập mã nội hoặc một trong số nhiều chế độ lập mã liên, đối với khối viđeo hiện thời căn cứ vào các kết quả lỗi (ví dụ, tốc độ lập mã và độ méo). Bộ phận xử lý dự đoán 41 có thể cung cấp khối lập mã dự đoán liên hoặc nội được tạo ra cho bộ cộng 50 để tạo ra khối dư và cho bộ cộng 62 để tái cấu trúc khối được mã hóa để sử dụng làm một phần của khung tham chiếu sau đó. Bộ phận xử lý dự đoán 41 còn cung cấp phần tử cú pháp, như các vectơ chuyển động, chỉ báo chế độ nội, thông tin phân vùng, và thông tin khác như thông tin cú pháp, cho đơn vị mã hóa entropy 56.

Để lựa chọn chế độ lập mã dự đoán nội thích hợp cho khối viđeo hiện thời, bộ phận xử lý dự đoán nội 46 trong bộ phận xử lý dự đoán 41 có thể thực hiện lập mã dự đoán nội của khối viđeo hiện thời liên quan đến một hoặc nhiều khối lân cận trong cùng một khung làm khối hiện thời được lập mã để cung cấp dự đoán không gian. Bộ phận ước lượng chuyển động 42 và bộ phận bù chuyển động 44 trong bộ phận xử lý dự đoán 41 thực hiện lập mã dự đoán liên của khối viđeo hiện thời liên quan đến một hoặc nhiều khối dự đoán trong một hoặc nhiều khung tham chiếu để cung cấp dự đoán thời gian. Bộ mã hóa viđeo 20 có thể thực hiện nhiều lần lập mã, ví dụ, để lựa chọn chế độ lập mã thích hợp cho mỗi khối dữ liệu viđeo.

Trong vài phương án thực hiện, bộ phận ước lượng chuyển động 42 xác định chế độ dự đoán liên cho khung viđeo hiện thời bằng cách tạo ra vectơ chuyển động, mà chỉ ra sự dịch chuyển của đơn vị dự đoán (prediction unit: PU) của khối viđeo trong khung viđeo hiện thời liên quan đến khối dự đoán trong khung viđeo tham chiếu, theo mẫu xác định trước trong một chuỗi các khung viđeo. Ước lượng chuyển động, được thực hiện bởi bộ phận ước lượng chuyển động 42, là quy trình tạo ra các vectơ chuyển động, mà ước lượng chuyển động cho khối viđeo. Vectơ chuyển động, ví dụ, có thể chỉ ra sự dịch chuyển của PU của khối viđeo trong khung viđeo hiện thời hoặc hình ảnh liên quan đến khối dự đoán trong khung tham chiếu (hoặc đơn vị được lập mã khác) liên quan đến khối hiện thời được lập mã trong khung hiện thời (hoặc đơn vị lập mã khác). Mẫu định trước có thể chỉ định khung viđeo trong chuỗi dưới dạng khung P hoặc khung B. Đơn vị BC nội 48 có thể xác định các vectơ, ví dụ, vectơ khối, cho lập mã BC nội theo cách tương tự với việc xác định các vectơ chuyển động bằng bộ phận ước lượng chuyển động 42 cho phép dự đoán liên, hoặc có thể sử dụng bộ phận ước lượng chuyển động 42 để xác định vectơ khối.

Khối dự đoán là khối của khung tham chiếu mà được coi là phù hợp chặt chẽ với PU của khối viđeo được lập mã về sự khác biệt pixel, có thể được xác định bằng tổng chênh lệch tuyệt đối (sum of absolute difference: SAD), tổng chênh lệch bình phương (sum of square difference: SSD) hoặc các chỉ mục chênh lệch khác. Trong vài phương án thực hiện, bộ mã hóa viđeo 20 có thể tính toán giá trị cho các vị trí pixel số nguyên phụ của khung tham chiếu được lưu trong DPB 64. Ví dụ, bộ mã hóa viđeo 20 có thể nội suy các giá trị của các vị trí pixel một phần tư, các vị trí pixel một phần tám hoặc các vị trí pixel phân số khác của khung tham chiếu. Do đó, bộ phận ước lượng chuyển động 42 có thể thực hiện việc tìm kiếm chuyển động liên quan đến các vị trí pixel đầy đủ và vị trí pixel phân số và xuất ra vectơ chuyển động với độ chính xác pixel phân số.

Bộ phận ước lượng chuyển động 42 tính toán vectơ chuyển động cho PU của khối viđeo trong khung được lập mã dự đoán liên bằng cách so sánh vị trí của PU với vị trí của khối dự đoán của khung tham chiếu được lựa chọn từ danh mục khung tham chiếu thứ nhất (List 0) hoặc danh mục khung tham chiếu thứ hai (List 1), mỗi trong số

chúng xác định một hoặc nhiều khung tham chiếu lưu trong DPB 64. Bộ phận ước lượng chuyển động 42 gửi vectơ chuyển động được tính toán cho bộ phận bù chuyển động 44 và sau đó cho đơn vị mã hóa entropy 56.

Bù chuyển động, được thực hiện bởi bộ phận bù chuyển động 44, có thể liên quan đến việc tìm nạp hoặc tạo ra khối dự đoán căn cứ vào vectơ chuyển động được xác định bởi bộ phận ước lượng chuyển động 42. Đến khi nhận vectơ chuyển động cho PU của khối viđeo hiện thời, bộ phận bù chuyển động 44 có thể đặt khối dự đoán mà vectơ chuyển động trỏ đến trong một hoặc nhiều danh mục khung tham chiếu, lấy lại khối dự đoán từ DPB 64, và chuyển tiếp khối dự đoán đến bộ cộng 50. Sau đó, bộ cộng 50 tạo ra khối viđeo dư của các giá trị khác nhau về pixel bằng cách trừ đi các giá trị pixel của khối dự đoán được cung cấp bởi bộ phận bù chuyển động 44 từ các giá trị pixel của khối viđeo hiện thời được lập mã. Các giá trị khác nhau về pixel tạo ra khối viđeo dư có thể bao gồm thành phần khác nhau về luma hoặc chroma hoặc cả hai. Bộ phận bù chuyển động 44 có thể cũng tạo ra phần tử cú pháp kết hợp với khối viđeo của khung viđeo để sử dụng bởi bộ giải mã viđeo 30 trong giải mã khối viđeo của khung viđeo. Các phần tử cú pháp có thể bao gồm, ví dụ, các phần tử cú pháp định rõ vectơ chuyển động được sử dụng để xác định khối dự đoán, cờ bất kỳ chỉ ra chế độ dự đoán, hoặc bất kỳ thông tin cú pháp khác được mô tả ở đây. Lưu ý rằng bộ phận ước lượng chuyển động 42 và bộ phận bù chuyển động 44 có thể được tích hợp cao, nhưng được minh họa riêng biệt cho các mục đích khái niệm.

Trong vài phương án thực hiện, đơn vị BC nội 48 có thể tạo ra các vectơ và tìm nạp khối dự đoán theo cùng một cách với cách đã được mô tả ở trên liên quan đến bộ phận ước lượng chuyển động 42 và bộ phận bù chuyển động 44, nhưng với các khối dự đoán ở trong cùng một khung dưới dạng khối hiện thời được lập mã và với các vectơ được đề cập đến là vectơ khối mà đối diện với vectơ chuyển động. Cụ thể là, đơn vị BC nội 48 có thể xác định chế độ dự đoán nội để sử dụng mã hóa khối hiện thời. Theo vài ví dụ, đơn vị BC nội 48 có thể mã hóa khối hiện thời sử dụng các chế độ dự đoán nội, ví dụ, trong quá trình chuyển tác mã hóa riêng biệt và kiểm tra hiệu suất của chúng thông qua phân tích tốc độ-độ méo. Tiếp theo, đơn vị BC nội 48 có thể lựa

chọn, trong số nhiều chế độ nội được kiểm tra, chế độ dự đoán nội thích hợp để sử dụng và tạo ra bộ chỉ báo chế độ nội theo đó. Ví dụ, đơn vị BC nội 48 có thể tính toán giá trị tốc độ-độ méo bằng cách sử dụng phép phân tích tốc độ-độ méo đối với nhiều chế độ dự đoán nội được kiểm tra, và lựa chọn chế độ dự đoán nội có đặc điểm tốc độ-độ méo tốt nhất trong số các chế độ được kiểm tra làm chế độ dự đoán nội thích hợp để sử dụng. Phép phân tích tốc độ-độ méo thường xác định độ méo (hoặc lỗi) giữa khối được mã hóa và khối không được mã hóa ban đầu mà đã được mã hóa để tạo ra khối được mã hóa, cũng như tốc độ bit (tức là một số bit) được sử dụng để tạo ra khối được mã hóa. Đơn vị BC nội 48 có thể tính toán tỷ lệ từ độ méo và tỷ lệ cho nhiều khối được mã hóa để xác định chế độ dự đoán nội nào thể hiện giá trị tốc độ-độ méo tốt nhất cho khối.

Trong các ví dụ khác, đơn vị BC nội 48 có thể sử dụng bộ phận ước lượng chuyển động 42 và bộ phận bù chuyển động 44, toàn bộ hoặc một phần, để thực hiện các chức năng này cho phép dự đoán BC nội theo việc thực hiện được mô tả ở đây. Trong trường hợp khác, đối với sao chép khối nội, khối dự đoán có thể là khối mà được coi là khớp chặt chẽ với khối được lập mã, về sự chênh lệch pixel, mà có thể được xác định bằng tổng chênh lệch tuyệt đối (SAD), tổng của chênh lệch bình phương (SSD) hoặc các chỉ mục chênh lệch khác và việc xác định khối dự đoán có thể bao gồm việc tính toán các giá trị cho các vị trí pixel số nguyên phụ.

Liệu khối dự đoán là từ cùng một khung theo phép dự đoán nội khung, hoặc khung khác theo phép dự đoán liên khung, thì bộ mã hóa viđeo 20 có thể tạo ra khối viđeo dư bằng cách trừ đi các giá trị pixel của khối dự đoán từ các giá trị pixel của khối viđeo hiện thời được lập mã, tạo ra các giá trị pixel khác nhau. Các giá trị pixel khác nhau này tạo ra khối viđeo dư có thể bao gồm cả sự khác biệt về thành phần luma và chroma.

Bộ phận xử lý dự đoán nội 46 có thể dự đoán nội cho khối viđeo hiện thời, như một sự thay thế cho dự đoán liên được thực hiện bởi bộ phận ước lượng chuyển động 42 và bộ phận bù chuyển động 44, hoặc dự đoán sao chép khối nội được thực hiện bởi đơn vị BC nội 48, như được mô tả ở trên. Cụ thể là, bộ phận xử lý dự đoán nội 46 có

thể xác định chế độ dự đoán nội để sử dụng mã hóa khối hiện thời. Để làm như vậy, bộ phận xử lý dự đoán nội 46 có thể mã hóa khối hiện thời sử dụng nhiều chế độ dự đoán nội, ví dụ, trong suốt quá trình chuyển tác mã hóa riêng biệt, và bộ phận xử lý dự đoán nội 46 (hoặc bộ phận lựa chọn chế độ, theo vài ví dụ) có thể lựa chọn chế độ dự đoán nội thích hợp để sử dụng từ các chế độ dự đoán nội được kiểm tra. Bộ phận xử lý dự đoán nội 46 có thể cung cấp thông tin chỉ dẫn về chế độ dự đoán nội được lựa chọn đối với khối cho đơn vị mã hóa entropy 56. Đơn vị mã hóa entropy 56 có thể mã hóa thông tin chỉ định chế độ dự đoán nội được lựa chọn trong luồng bit.

Sau khi bộ phận xử lý dự đoán 41 xác định khối dự đoán cho khối viđeo hiện thời thông qua hoặc là dự đoán liên hoặc là dự đoán nội, bộ cộng 50 tạo ra khối viđeo dư bằng cách trừ đi khối dự đoán từ khối viđeo hiện thời. Dữ liệu viđeo dư trong khối dư có thể có trong một hoặc nhiều bộ phận biến đổi (TU) và được cung cấp cho bộ phận xử lý biến đổi 52. Bộ phận xử lý biến đổi 52 biến đổi dữ liệu viđeo dư thành các hệ số biến đổi dư sử dụng phép biến đổi, như biến đổi cosin rời rạc (discrete cosine transform: DCT) hoặc một biến đổi tương tự về mặt khái niệm.

Bộ phận xử lý biến đổi 52 có thể gửi hệ số biến đổi thu được tới đơn vị lượng tử hóa 54. Đơn vị lượng tử hóa 54 lượng tử hóa hệ số biến đổi để giảm thêm tốc độ bit. Xử lý lượng tử hóa có thể cũng làm giảm độ sâu bit kết hợp với một số hoặc tất cả các hệ số biến đổi. Độ lượng tử hóa có thể được thay đổi bằng cách điều chỉnh thông số lượng tử hóa. Theo vài ví dụ, đơn vị lượng tử hóa 54 sau đó có thể thực hiện quét ma trận bao gồm hệ số biến đổi được lượng tử hóa. Theo cách khác, đơn vị mã hóa entropy 56 có thể thực hiện quét.

Sau lượng tử hóa, đơn vị mã hóa entropy 56 mã hóa entropy hệ số biến đổi được lượng tử hóa thành luồng bit viđeo sử dụng, ví dụ, phép lập mã độ dài biến tương thích theo ngữ cảnh (CAVLC), lập mã số học nhị phân tương thích theo ngữ cảnh (CABAC), lập mã số học nhị phân tương thích theo ngữ cảnh (SBAC), lập mã entropy phân vùng theo khoảng xác suất (PIPE) hoặc một phương pháp hoặc kỹ thuật mã hóa entropy khác. Luồng bit được mã hóa, sau đó có thể được truyền tới bộ giải mã viđeo 30, hoặc được lưu trữ trong thiết bị lưu trữ 32 cho phép biến đổi sau đó hoặc thu hồi

bởi bộ giải mã viđeo 30. Đơn vị mã hóa entropy 56 có thể cũng mã hóa entropy các vectơ chuyển động và phần tử cú pháp khác cho khung viđeo hiện thời được lập mã.

Bộ lượng tử hóa nghịch đảo 58 và bộ phận xử lý biến đổi nghịch đảo 60 sử dụng lượng tử hóa nghịch đảo và biến đổi nghịch đảo tương ứng, để tái tạo cấu trúc khối viđeo dư trong miền pixel để tạo ra khối tham chiếu cho phép dự đoán của khối viđeo khác. Như được lưu ý ở trên, bộ phận bù chuyển động 44 có thể tạo ra khối dự đoán bù chuyển động từ một hoặc nhiều khối tham chiếu của các khung được lưu trong DPB 64. Bộ phận bù chuyển động 44 có thể cũng áp dụng một hoặc nhiều bộ lọc nội suy cho khối dự đoán để tính toán giá trị pixel số nguyên phụ để sử dụng trong ước lượng chuyển động.

Bộ cộng 62 cộng khối dư được tái tạo cấu trúc vào khối dự đoán bù chuyển động được tạo ra bởi bộ phận bù chuyển động 44 để tạo ra khối tham chiếu lưu trữ trong DPB 64. Khối tham chiếu sau đó có thể được sử dụng bởi đơn vị BC nội 48, bộ phận ước lượng chuyển động 42 và bộ phận bù chuyển động 44 dưới dạng khối dự đoán để dự đoán liên khối viđeo khác trong khung viđeo tiếp theo.

FIG. 3 là sơ đồ khối minh họa bộ giải mã viđeo 30 điển hình phù hợp với vài phương án thực hiện của sáng chế. Bộ giải mã viđeo 30 bao gồm bộ nhớ dữ liệu viđeo 79, bộ giải mã entropy 80, bộ phận xử lý dự đoán 81, bộ lượng tử hóa nghịch đảo 86, bộ phận xử lý biến đổi nghịch đảo 88, bộ cộng 90, và DPB 92. Bộ phận xử lý dự đoán 81 còn bao gồm bộ phận bù chuyển động 82, bộ phận dự đoán nội 84, và đơn vị BC nội 85. Bộ giải mã viđeo 30 có thể thực hiện quy trình giải mã thường đối ứng với quy trình mã hóa được mô tả ở trên liên quan đến bộ mã hóa viđeo 20 kết hợp với FIG. 2. Ví dụ, bộ phận bù chuyển động 82 có thể tạo ra dữ liệu dự đoán căn cứ vào vectơ chuyển động được nhận từ bộ giải mã entropy 80, trong khi bộ phận dự đoán nội 84 có thể tạo ra dữ liệu dự đoán căn cứ vào bộ chỉ báo chế độ dự đoán nội nhận được từ bộ giải mã entropy 80.

Theo vài ví dụ, bộ phận bộ giải mã viđeo 30 có thể được phân công nhiệm vụ để thực hiện sáng chế. Ngoài ra theo vài ví dụ, việc thực hiện sáng chế có thể được chia ra trong số một hoặc nhiều bộ phận của bộ giải mã viđeo 30. Ví dụ, đơn vị BC nội 85

có thể thực hiện sáng chế một mình hoặc kết hợp với các bộ phận khác của bộ giải mã viđeo 30, như bộ phận bù chuyển động 82, bộ phận dự đoán nội 84, và bộ giải mã entropy 80. Theo vài ví dụ, bộ giải mã viđeo 30 có thể không chứa đơn vị BC nội 85 và chức năng của đơn vị BC nội 85 có thể được thực hiện bằng các thành phần khác của bộ phận xử lý dự đoán 81, như bộ phận bù chuyển động 82.

Bộ nhớ dữ liệu viđeo 79 có thể lưu trữ dữ liệu viđeo, như luồng bit viđeo được mã hóa, để được giải mã bởi thành phần khác của bộ giải mã viđeo 30. Dữ liệu viđeo được lưu trong bộ nhớ dữ liệu viđeo 79 có thể thu được, ví dụ, từ thiết bị lưu trữ 32, từ nguồn viđeo cục bộ, như camera, truyền thông mạng có dây hoặc không dây của dữ liệu viđeo, hoặc bằng cách truy cập phương tiện lưu trữ vật lý (ví dụ, ổ đĩa flash hoặc đĩa cứng). Bộ nhớ dữ liệu viđeo 79 có thể bao gồm bộ đệm hình ảnh được lập mã (coded picture buffer: CPB) mà lưu dữ liệu viđeo đã được mã hóa từ luồng bit viđeo được mã hóa. Đệm hình ảnh được giải mã (DPB) 92 của bộ giải mã viđeo 30 lưu dữ liệu viđeo tham chiếu để sử dụng trong giải mã dữ liệu viđeo bởi bộ giải mã viđeo 30 (ví dụ, trong chế độ lập mã dự đoán nội hoặc liên). Bộ nhớ dữ liệu viđeo 79 và DPB 92 có thể được tạo ra bởi bất kỳ trong số nhiều thiết bị nhớ như bộ nhớ truy cập ngẫu nhiên động (DRAM), bao gồm DRAM đồng bộ (SDRAM), RAM điện trở từ (MRAM), RAM điện trở (RRAM) hoặc các loại thiết bị nhớ khác. Với mục đích minh họa, bộ nhớ dữ liệu viđeo 79 và DPB 92 được mô tả như hai thành phần riêng biệt của bộ giải mã viđeo 30 trong FIG. 3. Nhưng người có hiểu biết trung bình trong lĩnh vực này sẽ hiểu rằng bộ nhớ dữ liệu viđeo 79 và DPB 92 có thể được cung cấp bởi cùng một thiết bị nhớ hoặc các thiết bị nhớ riêng biệt. Theo vài ví dụ, bộ nhớ dữ liệu viđeo 79 có thể là on-chip (trên chip) với các thành phần khác nhau của bộ giải mã viđeo 30, hoặc off-chip (ngoài chip) liên quan đến các thành phần khác nhau.

Trong suốt quá trình giải mã, bộ giải mã viđeo 30 nhận luồng bit viđeo được mã hóa mà thể hiện khối viđeo của khung viđeo được mã hóa và các phần tử cú pháp kết hợp. Bộ giải mã viđeo 30 có thể nhận các phần tử cú pháp ở mức độ khung viđeo và/hoặc mức độ khối viđeo. Bộ giải mã entropy 80 của bộ giải mã viđeo 30 giải mã entropy luồng bit để tạo ra hệ số lượng tử hóa, vectơ chuyển động hoặc bộ chỉ báo chế

độ dự đoán nội, và các phần tử cú pháp khác. Bộ giải mã entropy 80 sau đó chuyển tiếp các vectơ chuyển động và các phần tử cú pháp khác tới bộ phận xử lý dự đoán 81.

Khi khung viđeo được lập mã dưới dạng khung lập mã dự đoán nội (I) hoặc cho khối dự đoán lập mã nội trong các loại khung khác nhau, bộ phận dự đoán nội 84 của bộ phận xử lý dự đoán 81 có thể tạo ra dữ liệu dự đoán cho khối viđeo của khung viđeo hiện thời căn cứ vào chế độ dự đoán nội được báo hiệu và dữ liệu tham chiếu từ các khối được giải mã trước đó của khung hiện thời.

Khi khung viđeo được lập mã dưới dạng khung lập mã dự đoán liên (tức là, B hoặc P), bộ phận bù chuyển động 82 của bộ phận xử lý dự đoán 81 tạo ra một hoặc nhiều khối dự đoán cho khối viđeo của khung viđeo hiện thời căn cứ vào vectơ chuyển động và các phần tử cú pháp khác nhận được từ bộ giải mã entropy 80. Mỗi trong số các khối dự đoán có thể được sản xuất từ khung tham chiếu trong một trong số danh mục khung tham chiếu. Bộ giải mã viđeo 30 có thể tạo cấu trúc danh mục khung tham chiếu, List 0 và List 1, sử dụng kỹ thuật tạo cấu trúc mặc định căn cứ vào khung tham chiếu được lưu trong DPB 92.

Theo vài ví dụ, khi khối viđeo được lập mã theo chế độ BC nội được mô tả ở đây, đơn vị BC nội 85 của bộ phận xử lý dự đoán 81 tạo ra khối dự đoán cho khối viđeo hiện thời căn cứ vào vectơ khối và các phần tử cú pháp khác nhận được từ bộ giải mã entropy 80. Các khối dự đoán có thể trong vùng tái cấu trúc của cùng một hình ảnh dưới dạng khối viđeo hiện thời được định rõ bởi bộ mã hóa viđeo 20.

Bộ phận bù chuyển động 82 và/hoặc đơn vị BC nội 85 xác định thông tin dự đoán cho khối viđeo của khung viđeo hiện thời bằng cách phân tích cú pháp vectơ chuyển động và các phần tử cú pháp khác, và sau đó sử dụng thông tin dự đoán để tạo ra các khối dự đoán cho khối viđeo hiện thời được lập mã. Ví dụ, bộ phận bù chuyển động 82 sử dụng vài trong số các phần tử cú pháp nhận được để xác định chế độ dự đoán (ví dụ, dự đoán liên hoặc nội) được sử dụng để lập mã khối viđeo của khung viđeo, loại khung dự đoán liên khung (ví dụ, B hoặc P), thông tin tạo cấu trúc cho một hoặc nhiều danh mục khung tham chiếu cho khung này, vectơ chuyển động cho mỗi khối viđeo mã hóa dự đoán liên khung của khung này, trạng thái dự đoán liên khung

cho mỗi khối viđeo được lập mã dự đoán liên của khung, và thông tin khác để giải mã khối viđeo trong khung viđeo hiện thời.

Tương tự, đơn vị BC nội 85 có thể sử dụng vài trong số các phần tử cú pháp đã nhận, ví dụ, cờ, để xác định rằng khối viđeo hiện thời đã được dự đoán sử dụng chế độ BC nội, thông tin tạo cấu trúc mà khối viđeo của khung này của nó là trong vùng tái cấu trúc và nên được lưu trong DPB 92, các vectơ khối cho mỗi khối viđeo dự đoán BC nội của khung này, tình trạng dự đoán BC nội cho mỗi khối viđeo dự doán BC nội của khung này và thông tin khác để giải mã khối viđeo trong khung viđeo hiện thời.

Bộ phận bù chuyển động 82 có thể cũng thực hiện nội suy bằng cách sử dụng các bộ lọc nội suy như được sử dụng bởi bộ mã hóa viđeo 20 trong suốt quá trình mã hóa của khối viđeo để tính toán các giá trị nội suy cho pixel số nguyên phụ của khối tham chiếu. Trong trường hợp này, bộ phận bù chuyển động 82 có thể xác định bộ lọc nội suy được sử dụng bởi bộ mã hóa viđeo 20 từ phần tử cú pháp đã nhận và sử dụng bộ lọc nội suy để tạo ra các khối dự đoán.

Bộ lượng tử hóa nghịch đảo 86 lượng tử hóa nghịch đảo hệ số biến đổi được lượng tử hóa được tạo ra trong luồng bit và được giải mã entropy bằng bộ giải mã entropy 80 sử dụng cùng một thông số lượng tử hóa được tính toán bởi bộ mã hóa viđeo 20 cho mỗi khối viđeo trong khung viđeo để xác định độ lượng tử hóa. Bộ phận xử lý biến đổi nghịch đảo 88 áp dụng phép biến đổi nghịch đảo, ví dụ, DCT nghịch đảo, biến đổi số nguyên nghịch đảo, hoặc một phép biến đổi nghịch đảo tương tự về mặt khái niệm, cho hệ số biến đổi để tái tạo cấu trúc khối dư trong miền pixel.

Sau khi bộ phận bù chuyển động 82 hoặc đơn vị BC nội 85 tạo ra khối dự đoán cho khối viđeo hiện thời căn cứ vào các vectơ và các phần tử cú pháp khác, bộ cộng 90 tái cấu trúc khối viđeo đã giải mã cho khối viđeo hiện thời bằng cách tính tổng khối dư từ bộ phận xử lý biến đổi nghịch đảo 88 và khối dự đoán tương ứng được tạo ra bởi bộ phận bù chuyển động 82 và đơn vị BC nội 85. Bộ lọc trong vòng (không được thể hiện) có thể được định vị trí giữa bộ cộng 90 và DPB 92 để xử lý thêm khối viđeo được giải mã. Khối viđeo được giải mã trong khung thu được sau đó được lưu trong DPB 92, mà lưu khung tham chiếu được sử dụng cho bù chuyển động tiếp theo của

của khối viđeo tiếp theo. DPB 92, hoặc thiết bị nhớ tách rời khỏi DPB 92, có thể cũng lưu viđeo được giải mã cho việc thể hiện sau đó trên thiết bị hiển thị, như thiết bị hiển thị 34 của FIG. 1.

Trong xử lý lập mã viđeo thông thường, chuỗi viđeo thường bao gồm tập hợp sắp thứ tự khung hoặc hình ảnh. Mỗi khung có thể bao gồm ba dãy mẫu, được ký hiệu là SL, SCb và SCr. SL là dãy hai chiều của mẫu luma. SCb là dãy hai chiều của mẫu Cb chroma. SCr là dãy hai chiều của mẫu Cr chroma. Trong các ví dụ khác, khung có thể là đơn sắc và do đó bao gồm chỉ một dãy hai chiều của mẫu luma.

Như được thể hiện trong FIG. 4A, bộ mã hóa viđeo 20 (hoặc cụ thể hơn là bộ phận phân vùng 45) tạo ra sự biểu diễn được mã hóa của một khung bằng cách trước tiên là phân chia khung thành một tập hợp các đơn vị cây lập mã (CTU). Khung viđeo có thể bao gồm một số nguyên của CTU được xếp thứ tự liên tiếp theo thứ tự quét raster từ trái sang phải và từ đỉnh xuống đáy. Mỗi CTU là đơn vị lập mã logic lớn nhất và chiều rộng và chiều cao của CTU được báo hiệu bằng bộ mã hóa viđeo 20 trong tập hợp thông số chuỗi, sao cho tất cả CTU trong chuỗi viđeo có cùng một kích thước là một trong số 128×128, 64×64, 32×32, và 16×16. Nhưng nên lưu ý rằng sáng chế không nhất thiết giới hạn ở kích thước cụ thể. Như được thể hiện trong FIG. 4B, mỗi CTU có thể bao gồm một khối cây lập mã (coding tree block: CTB) của mẫu luma, hai khối cây lập mã tương ứng của khối chroma, và phần tử cú pháp được sử dụng để lập mã mẫu khối cây lập mã. Phần tử cú pháp mô tả thuộc tính của các loại khác nhau của bộ phận khối được lập mã của pixel và chuỗi viđeo có thể được tái cấu trúc như thế nào ở bộ giải mã viđeo 30, bao gồm dự đoán liên hoặc nội, chế độ dự đoán nội, vectơ chuyển động, và các thông số khác. Trong hình ảnh đơn sắc hoặc hình ảnh có ba mặt phẳng màu riêng, CTU có thể bao gồm khối cây lập mã đơn và các phần tử cú pháp được sử dụng để lập mã các mẫu của khối cây lập mã. Khối cây lập mã có thể là khối NxN của các mẫu.

Để đạt được hiệu suất tốt hơn, bộ mã hóa viđeo 20 có thể thực hiện một cách đệ quy phân vùng cây như phân vùng cây nhị phân, phân vùng cây tam phân, phân vùng cây tứ phân hoặc kết hợp cả hai trên khối cây lập mã của CTU và chia CTU thành các

đơn vị lập mã (CU) nhỏ hơn. Như được mô tả trong FIG. 4C, 64x64 CTU 400 lần đầu tiên được chia thành bốn CU nhỏ hơn, mỗi CU có kích thước khối là 32x32. Trong số bốn CU nhỏ hơn, CU 410 và CU 420 mỗi cái được chia thành bốn CU có kích thước 16x16 theo kích thước khối. Hai CU 16x16 430 và 440 được chia thành bốn CU có kích thước 8x8 theo kích thước khối. FIG. 4D mô tả cấu trúc cây dữ liệu tứ phân minh họa kết quả cuối cùng của quy trình phân vùng của CTU 400 như được mô tả trong FIG. 4C, mỗi nút lá của cây tứ phân tương ứng với một CU có kích thước tương ứng nằm trong khoảng từ 32x32 đến 8x8. Tương tự CTU được mô tả trong FIG. 4B, mỗi CU có thể bao gồm khối lập mã (coding block: CB) của mẫu luma và hai khối lập mã tương ứng của mẫu chroma của khung có cùng một kích thước, và phần tử cú pháp được sử dụng để lập mã mẫu của khối lập mã. Trong hình ảnh đơn sắc hoặc hình ảnh có ba mặt phẳng màu riêng, CU có thể bao gồm khối lập mã đơn và cấu trúc cú pháp được sử dụng để lập mã các mẫu của khối lập mã. Cần lưu ý rằng phân vùng cây tứ phân được mô tả trong các FIG. 4C và 4D chỉ dành cho mục đích minh họa và một CTU có thể được chia thành các CU để thích ứng với các đặc điểm cục bộ khác nhau căn cứ vào phân vùng cây nhị phân/tam phân/tứ phân. Trong cấu trúc cây đa loại, một CTU được phân vùng bởi cấu trúc cây tứ phân và mỗi CU lá cây tứ phân có thể còn được phân vùng bởi cấu trúc cây nhị phân và tam phân. Như được thể hiện trong FIG. 4E, có năm kiểu phân vùng, tức là, phân vùng bậc bốn, phân vùng nhị phân ngang, phân vùng nhị phân dọc, phân vùng bậc ba ngang và phân vùng bậc ba dọc.

Trong vài phương án thực hiện, bộ mã hóa viđeo 20 có thể còn phần vùng khối lập mã của CU thành một hoặc nhiều khối dự đoán (PB) MxN. Khối dự đoán là khối hình chữ nhật (hình vuông hoặc không vuông) của khối mà cùng một phép dự đoán trên đó, nội hoặc liên được áp dụng. Bộ phận dự đoán (prediction unit: (PU) của CU có thể bao gồm khối dự đoán của mẫu luma, hai khối dự đoán tương ứng của mẫu chroma, và phần tử cú pháp được sử dụng để dự đoán các khối dự đoán. Trong hình ảnh đơn sắc hoặc hình ảnh có ba mặt phẳng màu riêng, PU có thể bao gồm khối dự đoán đơn và cấu trúc cú pháp được sử dụng để dự đoán khối dự đoán. Bộ mã hóa

viđeo 20 có thể tạo ra khối luma dự đoán, Cb và Cr cho luma, khối dự đoán Cb và Cr cho mỗi PU của CU.

Bộ mã hóa viđeo 20 có thể sử dụng dự đoán nội hoặc dự đoán liên để tạo ra khối dự đoán cho PU. Nếu bộ mã hóa viđeo 20 sử dụng dự đoán nội để tạo ra khối dự đoán của PU, bộ mã hóa viđeo 20 có thể tạo ra khối dự đoán của PU căn cứ vào các mẫu giải mã của khung kết hợp với PU. Nếu bộ mã hóa viđeo 20 sử dụng dự đoán liên để tạo ra khối dự đoán của PU, bộ mã hóa viđeo 20 có thể tạo ra khối dự đoán của PU căn cứ vào các mẫu giải mã của một hoặc nhiều khung khác với khung kết hợp với PU.

Sau khi bộ mã hóa viđeo 20 tạo ra khối luma dự đoán luma, Cb, và Cr cho một hoặc nhiều PU của CU, bộ mã hóa viđeo 20 có thể tạo ra khối dư luma cho CU bằng cách trừ đi khối luma dự đoán của CU khỏi khối lập mã luma gốc sao cho mỗi mẫu trong khối dư luma của CU chỉ ra sự khác nhau giữa mẫu luma trong một trong số khối luma dự đoán của CU và mẫu tương ứng trong khối lập mã luma gốc CU. Tương tự, bộ mã hóa viđeo 20 có thể tạo ra khối dư Cb và khối dư Cr cho CU tương ứng, sao cho mỗi mẫu trong khối dư Cb của CU chỉ ra sự khác nhau giữa mẫu Cb trong một trong số khối Cb dự đoán của CU và mẫu tương ứng trong khối lập mã Cb gốc của CU và mỗi mẫu trong khối dư Cr của Cu có thể chỉ ra sự khác nhau giữa mẫu Cr trong một trong số khối Cr dự đoán của CU và mẫu tương ứng trong khối lập mã Cr gốc của CU.

Ngoài ra, như được minh họa trong FIG. 4C, bộ mã hóa viđeo 20 có thể sử dụng phân vùng cây tứ phân để phân tích các khối dư luma, Cb, và Cr của CU thành một hoặc nhiều khối biến đổi luma, Cb, và Cr. Khối biến đổi là khối hình chữ nhật (vuông hoặc không vuông) của mẫu mà áp dụng cùng một phép biến đổi trên đó. Đơn vị biến đổi (TU) của CU có thể bao gồm khối biến đổi của mẫu luma, hai khối biến đổi tương ứng của các mẫu chroma, và phần tử cú pháp được sử dụng để biến đổi các mẫu khối biến đổi. Do đó, mỗi TU của CU có thể kết hợp với khối biến đổi luma, khối biến đổi Cb, và khối biến đổi Cr. Theo vài ví dụ, khối biến đổi luma kết hợp với TU có thể là khối con của khối dư luma của CU. Khối biến đổi Cb có thể là khối con của khối dư Cb của CU. Khối biến đổi Cr có thể là khối con của khối dư Cr của CU. Trong hình

ảnh đơn sắc hoặc hình ảnh có ba mặt phẳng màu riêng, TU có thể bao gồm khối biến đổi đơn và cấu trúc cú pháp được sử dụng để biến đổi mẫu của khối biến đổi.

Bộ mã hóa viđeo 20 có thể áp dụng một hoặc nhiều phép biến đổi cho khối biến đổi luma của TU để tạo ra luma khối hệ số cho TU. Khối hệ số có thể là dãy hai chiều của hệ số biến đổi. Hệ số biến đổi có thể là một đại lượng vô hướng. Bộ mã hóa viđeo 20 có thể áp dụng một hoặc nhiều phép biến đổi cho khối biến đổi Cb của TU để tạo ra khối hệ số Cb cho TU. Bộ mã hóa viđeo 20 có thể áp dụng một hoặc nhiều phép biến đổi cho khối biến đổi Cr của TU để tạo ra khối hệ số Cr cho TU.

Sau khi tạo ra khối hệ số (ví dụ, khối hệ số luma, khối hệ số Cb hoặc khối hệ số Cr), bộ mã hóa viđeo 20 có thể lượng tử hóa khối hệ số. Lượng tử hóa thường để chỉ quy trình, trong đó hệ số biến đổi được lượng tử hóa để giảm một cách khả thi một lượng dữ liệu được sử dụng để thể hiện các hệ số biến đổi, tạo ra hiệu quả nén hơn nữa. Sau khi bộ mã hóa viđeo 20 lượng tử hóa khối hệ số, bộ mã hóa viđeo 20 có thể mã hóa entropy phần tử cú pháp chỉ ra hệ số biến đổi được lượng tử hóa. Ví dụ, bộ mã hóa viđeo 20 có thể thực hiện lập mã số học nhị phân tương thích ngữ cảnh (Context-Adaptive Binary Arithmetic Coding: CABAC) trên các phần tử cú pháp chỉ ra hệ số biến đổi được lượng tử hóa. Cuối cùng, bộ mã hóa viđeo 20 có thể xuất ra luồng bit mà bao gồm trình tự bit mà tạo ra sự thể hiện của khung được lập mã và dữ liệu kết hợp, mà hoặc là được lưu trong thiết bị lưu trữ 32 hoặc được truyền tới thiết bị đích 14.

Sau khi nhận luồng bit được tạo ra bởi bộ mã hóa viđeo 20, bộ giải mã viđeo 30 có thể phân tích cú pháp luồng bit để thu được các phần tử cú pháp từ luồng bit đó. Bộ giải mã viđeo 30 có thể tái cấu trúc khung của dữ liệu viđeo dựa vào ít nhất một phần trên phần tử cú pháp thu được từ luồng bit. Quy trình tái cấu trúc dữ liệu viđeo thường đối ứng với quy trình mã hóa được thực hiện bởi bộ mã hóa viđeo 20. Ví dụ, bộ giải mã viđeo 30 có thể thực hiện phép biến đổi nghịch đảo trên các khối hệ số kết hợp với TU của CU hiện thời để tái cấu trúc các khối dư kết hợp với TU của CU hiện thời. Bộ giải mã viđeo 30 cũng tái cấu trúc khối lập mã của CU hiện thời bằng cách cộng các mẫu của khối dự đoán cho PU của CU hiện thời với các mẫu tương ứng của khối biến

đổi của TU của CU hiện thời. Sau khi tái cấu trúc khối lập mã cho mỗi CU của khung, bộ giải mã viđeo 30 có thể tái cấu trúc khung.

Như được lưu ý ở trên, lập mã viđeo đạt được hiệu quả nén viđeo sử dụng chủ yếu hai chế độ, tức là, dự đoán nội khung (hoặc dự đoán nội) và dự đoán liên khung (hoặc dự đoán liên). Lập mã dựa vào bảng màu là bảng lập mã khác mà được chấp nhận bởi các tiêu chuẩn lập mã viđeo. Trong lập mã dựa vào bảng màu, mà có thể thích hợp đặc biệt với lập mã nội dung tạo trên màn hình, bộ lập mã viđeo (ví dụ, bộ mã hóa viđeo 20 hoặc bộ giải mã viđeo 30) tạo ra bảng bảng màu của các màu thể hiện dữ liệu viđeo của khối thu được. Bảng bảng màu bao gồm các giá trị pixel chi phối tốt nhất (ví dụ, được sử dụng thường xuyên) trong khối nhất định. Các giá trị pixel không thường xuyên được thể hiện trong dữ liệu viđeo của khối thu được hoặc không có trong bảng bảng màu hoặc có trong bảng bảng màu dưới dạng màu thoát ra.

Mỗi mục trong bảng bảng màu bao gồm chỉ mục cho các giá trị pixel tương ứng trong bảng bảng màu. Các chỉ mục bảng màu cho các mẫu trong khối có thể được lập mã để chỉ ra mục mà từ bảng bảng màu được sử dụng để dự đoán hoặc tái cấu trúc mẫu này. Chế độ bảng màu bắt đầu với quy trình tạo ra biến độc lập dự đoán bảng màu cho khối thứ nhất của hình ảnh, lát, ngói hoặc tạo nhóm khác như vậy của khối viđeo. Như sẽ được giải thích dưới đây, biến độc lập dự đoán bảng màu cho các khối viđeo tiếp theo thường được tạo ra bằng cách cập nhật biến độc lập dự đoán bảng màu được sử dụng trước đó. Nhằm mục đích minh họa, giả sử rằng biến độc lập dự đoán bảng màu được xác định ở mức độ hình ảnh (khung hình). Theo cách khác, hình ảnh có thể bao gồm nhiều khối lập mã, mỗi khối có bảng bảng màu của nó, nhưng có một biến độc lập dự đoán bảng màu cho toàn bộ hình ảnh.

Để giảm bit cần cho báo hiệu các mục bảng màu trong luồng bit viđeo, bộ giải mã viđeo có thể sử dụng biến độc lập dự đoán bảng màu để xác định các mục bảng màu mới trong bảng bảng màu được sử dụng để tái cấu trúc khối viđeo. Ví dụ, biến độc lập dự đoán bảng màu có thể bao gồm các mục bảng màu từ bảng bảng màu được sử dụng trước đó hoặc thậm chí được khởi đầu với bảng bàng màu được sử dụng gần đây nhất bằng cách chứa toàn bộ mục của bảng bảng màu được sử dụng gần đây nhất.

Trong vài phương án thực hiện, biến độc lập dự đoán bảng màu có thể bao gồm ít hơn toàn bộ mục từ bảng bảng màu sử dụng gần nhất và sau đó kết hợp với vài mục từ bảng bảng màu được sử dụng trước đây khác. Biến độc lập dự đoán bảng màu có thể có cùng một kích thước với bảng bảng màu được sử dụng để lập mã các khối khác nhau hoặc có thể lớn hơn hoặc nhỏ hơn bảng bảng màu được sử dụng để lập mã các khối khác. Trong một ví dụ, biến độc lập dự đoán bảng màu được thực hiện dưới dạng bảng vào trước ra trước (first-in-first-out: FIFO) bao gồm 64 mục bảng màu.

Để tạo ra bảng bảng màu cho một khối của dữ liệu viđeo từ biến độc lập dự đoán bảng màu, bộ giải mã viđeo có thể nhận, từ luồng bit viđeo được mã hóa, cờ một bit cho mỗi mục của biến độc lập dự đoán bảng màu. Cờ một bit có thể có giá trị thứ nhất (ví dụ, một giá trị nhị phân) chỉ rằng mục kết hợp của biến độc lập dự đoán bảng màu có trong bảng bảng màu hoặc giá trị thứ hai (ví dụ, giá trị không nhị phân) chỉ ra rằng mục kết hợp của biến độc lập dự đoán bảng màu không có trong bảng bảng màu. Nếu kích thước của biến độc lập dự đoán bảng màu lớn hơn bảng bảng màu được sử dụng cho khối của dữ liệu viđeo, sau đó bộ giải mã viđeo có thể dừng nhận nhiều cờ khi kích thước lớn nhất cho bảng bảng màu đạt được.

Trong vài phương án thực hiện, vài mục trong bảng bảng màu có thể được trực tiếp báo hiệu trong luồng bit viđeo được mã hóa thay vì được xác định bằng cách sử dụng biến độc lập dự đoán bảng màu. Đối với các mục này, bộ giải mã viđeo có thể nhận, từ luồng bit viđeo được mã hóa, ba giá trị bit m riêng chỉ ra các giá trị pixel cho thành phần luma và hai thành phần chroma kết hợp với mục này, trong đó m thể hiện cho độ sâu bit của dữ liệu viđeo. So với nhiều giá trị bit m cần cho các mục bảng màu được báo hiệu trực tiếp, các mục bảng màu này được suy ra từ biến độc lập dự đoán bảng màu chỉ yêu cầu cờ một bit. Do đó, báo hiệu vài hoặc toàn bộ mục bảng màu sử dụng biến độc lập dự đoán bảng màu có thể giảm đáng kể một số lượng bit cần để báo hiệu mục của bảng bảng màu mới, nhờ đó cải thiện được hiệu quả lập mã tổng thể của lập mã chế độ bảng màu.

Trong nhiều ví dụ, biến độc lập dự đoán bảng màu cho một khối được xác định căn cứ vào bảng bảng màu được sử dụng để lập mã một hoặc nhiều khối được lập mã

trước đó. Nhưng khi lập mã đơn vị cây lập mã thứ nhất trong hình ảnh, lát hoặc ngói, bảng bảng màu của khối lập mã trước đó có thể không có sẵn. Do đó, biến độc lập dự đoán bảng màu không thể được tạo ra sử dụng toàn bộ bảng bảng màu được sử dụng trước đó. Trong trường hợp này, chuỗi của bộ khởi tạo biến độc lập dự đoán bảng màu có thể được báo hiệu trong tập hợp thông số chuỗi (sequence parameter set: SPS) và/hoặc tập hợp thông số hình ảnh (picture parameter set: PPS), mà là các giá trị được sử dụng để tạo ra biến độc lập dự đoán bảng màu khi bảng bảng màu được sử dụng trước đó không có sẵn. SPS thường chỉ cấu trúc cú pháp của phần tử cú pháp mà áp dụng cho dãy hình ảnh viđeo được lập mã liên tiếp gọi là chuỗi viđeo lập mã (coded video sequence: CVS) khi được xác định bởi nội dung của phần tử cú pháp được phát hiện trong PPS được đề cập tới bởi phần tử cú pháp được phát hiện trong mỗi tiêu đề phân đoạn lát. PPS thường chỉ cấu trúc cú pháp của phần tử cú pháp mà áp dụng cho một hoặc nhiều hình ảnh riêng trong CVS khi được xác định bởi phần tử cú pháp được phát hiện trong mỗi tiêu đề phân đoạn lát. Do đó, SPS thường được coi là cấu trúc cú pháp ở mức độ cao hơn so với PPS, có nghĩa là phần tử cú pháp có trong SPS thường thay đổi ít thường xuyên hơn và áp dụng cho một phần lớn hơn dữ liệu viđeo so với các phần tử cú pháp có trong PPS.

FIG. 5A đến 5B là các sơ đồ khối minh họa các ví dụ về việc sử dụng bảng bảng màu để lập mã dữ liệu viđeo phù hợp với vài phương án thực hiện của sáng chế.

Đối với báo hiệu chế độ bảng màu (PLT), chế độ bảng màu được lập mã như chế độ dự đoán cho đơn vị lập mã, tức là, chế độ dự đoán cho đơn vị lập mã có thể là MODE_INTRA, MODE_INTER, MODE_IBC và MODE_PLT. Nếu chế độ bảng màu được sử dụng, các giá trị pixel trong CU được thể hiện bằng tập hợp nhỏ nhất các màu đại diện. Tập hợp được đề cập đến là bảng màu. Đối với các pixel có các giá trị gần với màu bảng màu, các chỉ mục bảng màu được báo hiệu. Đối với các pixeo có các giá trị ngoài bảng màu, các pixel được biểu thị bằng ký hiệu thoát và các giá trị pixel đã lượng tử hóa được báo hiệu trực tiếp. Cú pháp và ngữ nghĩa liên quan của chế độ bảng màu trong thông số kỹ thuật dự thảo VVC hiện tại được minh họa trong bảng 1 và bảng 2 tương ứng dưới đây.

Để giải mã khối được mã hóa chế độ bảng màu, bộ giải mã cần giải mã các màu bảng màu và các chỉ mục từ luồng bit. Màu bảng màu được định rõ bởi bảng bảng màu và được mã hóa bởi cú pháp lập mã bảng bảng màu (ví dụ, palette_predictor_run, num_signaled_palette_entries, new_palette_entries). Cờ thoát, palette_escape_val_present_flag, được báo hiệu cho mỗi CU để chỉ ra nếu các ký hiệu thoát được thể hiện trong CU hiện thời. Nếu các ký hiệu thoát có mặt, bảng bảng màu được tăng thêm một mục nhập nữa và chỉ mục cuối cùng được gán cho chế độ thoát. Các chỉ mục bảng màu của tất cả các pixel trong CU tạo ra ánh xạ chỉ mục bảng màu và được mã hóa bằng cú pháp lập mã ánh xạ chỉ mục bảng màu (ví dụ, num_palette_indices_minus1, palette_idx_idc, copy_above_indices_for_final_run_flag, palette_transpose_flag, copy_above_palette_indices_flag, palette_run_prefix, palette_run_suffix). Ví dụ về CU được lập mã chế độ bảng màu được minh họa trong Fig. 5A, trong đó kích thước bảng màu là 4. Ba mẫu thứ nhất trong CU sử dụng toàn bộ bảng màu 2, 0, và 3 tương ứng cho việc tái cấu trúc. Mẫu "x" trong CU thể hiện ký hiệu thoát. Cờ mức độ CU, palette_escape_val_present_flag, chỉ ra liệu ký hiệu thoát bất kỳ nào có mặt trong CU hay không. Nếu ký hiệu thoát có mặt, kích thước bảng màu được tăng thêm một và chỉ mục cuối cùng được sử dụng để chỉ ra biểu tượng thoát. Do đó, trong Fig. 5A, chỉ mục 4 được gán cho biểu tượng thoát.

Nếu chỉ mục bảng màu (ví dụ, chỉ mục 4 trong FIG. 5A) tương ứng với ký hiệu thoát, phí tổn bổ sung được báo hiệu để chỉ ra các màu tương ứng của mẫu.

Theo vài phương án, ở phía bộ mã hóa, cần thiết suy ra bảng màu thích hợp được sử dụng với CU. Để tạo ra bảng màu cho lập mã mất dữ liệu, một thuật toán phân cụm k-mean biến đổi được sử dụng. Mẫu đầu tiên của khối được thêm vào bảng màu. Sau đó, đối với mỗi khối mẫu tiếp theo, tổng của sự chênh lệch tuyệt đối (SAD) giữa mẫu và mỗi màu bảng màu hiện tại được tính. Nếu độ méo cho mỗi thành phần là nhỏ hơn giá trị ngưỡng cho mục bảng màu tương ứng với SAD nhỏ nhất, mẫu được bổ sung vào cụm này thuộc về mục bảng màu. Trường hợp khác, mẫu được bổ sung là mục

bảng màu mới. Khi số lượng mẫu được ánh xạ tới một cụm vượt quá ngưỡng, trọng tâm cho cụm đó được cập nhật và trở thành mục bảng màu của cụm đó.

Trong bước tiếp theo, các cụm được sắp xếp theo thứ tự sử dụng giảm dần. Sau đó, mục bảng màu tương ứng với mỗi mục được cập nhật. Thông thường, trọng tâm cụm được sử dụng làm mục bảng màu. Nhưng phép phân tích độ méo được thực hiện để phân tích xem liệu bất kỳ mục nào từ biến độc lập dự đoán bảng màu có thể thích hợp hơn để được sử dụng làm mục bảng màu cập nhật thay cho trọng tâm khi chi phí lập mã các mục bảng màu được tính toán. Quy trình này được tiếp tục cho đến khi các cụm được xử lý hoặc kích thước bảng màu tối đa đạt được. Cuối cùng, nếu cụm có chỉ mẫu đơn và mục bảng màu tương ứng không trong biến độc lập dự đoán bảng màu, mẫu được chuyển đổi thành ký hiệu thoát. Ngoài ra, các mục bảng màu kép được loại bỏ và các cụm của nó được trộn vào.

Sau khi suy ra bảng màu, mỗi mẫu trong khối được gán cho chỉ mục của mục bảng màu gần nhất (trong SAD). Sau đó, các mẫu được gán cho chế độ 'INDEX' hoặc 'COPY_ABOVE'. Đối với mỗi mẫu có thể sử dụng chế độ 'INDEX' hoặc 'COPY_ABOVE', việc chạy cho từng chế độ được xác định. Sau đó, chi phí lập mã chế độ được tính toán. Chế độ mà chi phí thấp hơn được chọn.

Đối với lập mã bảng bảng màu, biến độc lập dự đoán bảng màu được duy trì. Kích thước tối đa của bảng màu và kích thước tối đa của biến độc lập dự đoán bảng màu có thể đều được báo hiệu trong SPS (hoặc mức độ lập mã khác như PPS, tiêu đề lát, v.v.). Biến độc lập dự đoán bảng màu được khởi đầu ở thời điểm bắt đầu của mỗi lát, tại đó biến độc lập dự đoán bảng màu được thiết lập mại về 0. Đối với mỗi mục trong biến độc lập dự đoán bảng màu, cờ sử dụng lại được báo hiệu để chỉ ra liệu nó là một phần của bảng màu hiện tại hay không. Như được thể hiện trong Fig. 5B, cờ sử dụng lại, palette_predictor_run, được gửi. Sau đó, số lượng các mục bảng màu mới được báo hiệu sử dụng mã Golomb theo cấp số nhân của thứ tự 0 thông qua cú pháp num_signaled_palette_entries. Cuối cùng, giá trị thành phần cho các mục bảng màu mới, new_palette_entries[], được báo hiệu. Sau khi lập mã CU hiện thời, biến độc lập dự đoán bảng màu được cập nhật sử dụng bảng màu hiện thời và các mục từ biến độc

lập dự đoán bảng màu trước đó mà không được sử dụng lại trong bảng màu hiện thời sẽ được bổ sung vào điểm cuối cùng của biến độc lập dự đoán bảng màu mới đến khi kích thước cho phép đạt được.

Đối với lập mã ánh xạ chỉ mục bảng màu, các chỉ mục được lập mã sử dụng các phép quét ngang qua theo chiều ngang hoặc dọc như được thể hiện trong Fig. 5C. Thứ tự quét được báo hiệu rõ ràng trong luồng bit sử dụng palette_transpose_flag.

Các chỉ mục bảng màu được lập mã sử dụng hai chế độ mẫu bảng màu: 'INDEX' và 'COPY_ABOVE'. Trong chế độ 'INDEX', chỉ mục bảng màu được báo hiệu rõ ràng. Trong chế độ 'COPY_ABOVE', chỉ mục bảng màu của mẫu trong hàng trên được sao chép. Đối với cả chế độ 'INDEX' và 'COPY_ABOVE', giá trị chạy được báo hiệu để chỉ định số pixel được mã hóa bằng cách sử dụng cùng một chế độ. Chế độ được báo hiệu bằng cờ ngoại trừ hàng trên cùng khi quét ngang được sử dụng hoặc cột đầu tiên khi quét dọc được sử dụng hoặc khi chế độ trước đó là 'COPY_ABOVE'.

Theo vài phương án, thứ tự lập mã đối với ánh xạ chỉ mục là như sau: Thứ nhất, số lượng giá trị chỉ mục cho CU được báo hiệu sử dụng cú pháp num_palette_indices_minus1, mà sau đó là báo hiệu giá trị chỉ mục thực tế cho toàn bộ CU sử dụng cú pháp palette_idx_idc. Cả số lượng chỉ mục cũng như giá trị chỉ mục đều được mã hóa ở chế độ bỏ qua (bypass). Điều này nhóm các bin được mã hóa bỏ qua liên quan đến chỉ mục lại với nhau. Sau đó chế độ bảng màu (INDEX hoặc COPY_ABOVE) và thao tác chạy được báo hiệu theo cách xen kẽ sử dụng cú pháp copy_above_palette_indices_flag, palette_run_prefix và palette_run_suffix. copy_above_palette_indices_flag là cờ lập mã ngữ cảnh (chỉ một bin), các từ mã của palette_run_prefix được xác định thông qua quy trình được mô tả trong bảng 3 dưới đây và 5 bin đầu tiên được lập mã ngữ cảnh. palette_run_suffix được lập mã dưới dạng bỏ qua bin (bypass bin). Cuối cùng, các giá trị thoát thành phần tương ứng với các mẫu thoát cho toàn bộ CU được nhóm lại với nhau và được mã hóa ở chế độ bỏ qua. Phần tử cú pháp bổ sung, copy_above_indices_for_final_run_flag, được báo hiệu sau khi báo hiệu các giá trị chỉ mục. Phần tử cú pháp này, kết hợp với số lượng các chỉ mục,

loại bỏ sự cần thiết phải báo hiệu giá trị chạy tương ứng với lần chạy cuối cùng trong khối.

Trong VVC (VTM), cây kép được kích hoạt cho I-slice mà phân tách sự phân vùng đơn vị lập mã đối với các thành phần luma và chroma. Kết quả là, bảng màu được áp dụng lên luma (thành phần Y) và chroma (các thành phần Cb và Cr) riêng rẽ. Nếu cây kép được vô hiệu hóa, bảng màu sẽ được áp dụng lên cùng thành phần Y, Cb, Cr.

Bảng 1: Cú pháp của lập mã bảng màu

palette_coding(x0, y0, cbWidth, cbHeight, startComp, numComps) {	Bộ mô tả
palettePredictionFinished = 0	
NumPredictedPaletteEntries = 0	
for(predictorEntryIdx = 0; predictorEntryIdx < PredictorPaletteSize[startComp] && !palettePredictionFinished && NumPredictedPaletteEntries[startComp] < palette_max_size; predictorEntryIdx++) {	
palette_predictor_run	ae(v)
if(palette_predictor_run != 1) {	
if(palette_predictor_run > 1)	
predictorEntryIdx += palette_predictor_run − 1	
PalettePredictorEntryReuseFlags[predictorEntryIdx] = 1	
NumPredictedPaletteEntries++	
} else	
palettePredictionFinished = 1	
}	
if(NumPredictedPaletteEntries < palette_max_size)	
num_signaled_palette_entries	ae(v)
for(cIdx = startComp; cIdx < (startComp + numComps); cIdx++)	

for(i = 0; i < num_signaled_palette_entries; i++)	
new_palette_entries[cIdx][i]	ae(v)
if(CurrentPaletteSize[startComp] > 0)	
palette_escape_val_present_flag	ae(v)
if(MaxpaletteIndex > 0) {	
num_palette_indices_minus1	ae(v)
adjust = 0	
for(i = 0; i <= num_palette_indices_minus1; i++) {	
if(MaxpaletteIndex − adjust > 0) {	
palette_idx_idc	ae(v)
PaletteIndexIdc[i] = palette_idx_idc	
}	
adjust = 1	
}	
copy_above_indices_for_final_run_flag	ae(v)
palette_transpose_flag	ae(v)
}	
if(treeType != DUAL_TREE_CHROMA && palette_escape_val_present_flag) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	
}	
if(treeType != DUAL_TREE_LUMA && palette_escape_val_present_flag) {	
if(cu_chroma_qp_offset_enabled_flag && !IsCuChromaQpOffsetCoded) {	

cu_chroma_qp_offset_flag	ae(v)
if(cu_chroma_qp_offset_flag)	
cu_chroma_qp_offset_idx	ae(v)
}	
}	
remainingNumIndices = num_palette_indices_minus1 + 1	
PaletteScanPos = 0	
log2CbWidth = Log2(cbWidth)	
log2CbHeight = Log2(cbHeight)	
while(PaletteScanPos < cbWidth*cbHeightt) {	
xC = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][0]	
yC = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][1]	
if(PaletteScanPos > 0) {	
xcPrev = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos − 1][0]	
ycPrev = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos − 1][1]	
}	
PaletteRunMinus1 = cbWidth * cbHeight − PaletteScanPos − 1	
RunToEnd = 1	
CopyAboveIndicesFlag[xC][yC] = 0	
if(MaxpaletteIndex > 0)	

if(((!palette_transpose_flag && yC > 0) || (palette_transpose_flag && xC > 0)) && CopyAboveIndicesFlag[xcPrev][ycPrev] = = 0)	
if(remainingNumIndices > 0 && PaletteScanPos < cbWidth* cbHeight − 1) {	
copy_above_palette_indices_flag	ae(v)
CopyAboveIndicesFlag[xC][yC] = copy_above_palette_indices_flag	
} else {	
if(PaletteScanPos = = cbWidth * cbHeight − 1 && remainingNumIndices > 0)	
CopyAboveIndicesFlag[xC][yC] = 0	
else	
CopyAboveIndicesFlag[xC][yC] = 1	
}	
if (CopyAboveIndicesFlag[xC][yC] = = 0) {	
currNumIndices = num_palette_indices_minus1 + 1 − remainingNumIndices	
PaletteIndexMap[xC][yC] = PaletteIndexIdc[currNumIndices]	
}	
if(MaxpaletteIndex > 0) {	
if(CopyAboveIndicesFlag[xC][yC] = = 0)	
remainingNumIndices − = 1	
if(remainingNumIndices > 0 || CopyAboveIndicesFlag[xC][yC] != copy_above_indices_for_final_run_flag) {	

PaletteMaxRunMinus1 = cbWidth * cbHeight − PaletteScanPos − 1 − remainingNumIndices − copy_above_indices_for_final_run_flag	
RunToEnd = 0	
if(PaletteMaxRunMinus1 > 0) {	
palette_run_prefix	ae(v)
if((palette_run_prefix > 1) && (PaletteMaxRunMinus1 != (1 << (palette_run_prefix − 1))))	
palette_run_suffix	ae(v)
}	
}	
}	
runPos = 0	
while (runPos <= PaletteRunMinus1) {	
xR = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][0]	
yR = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][1]	
if(CopyAboveIndicesFlag[xC][yC] = = 0) {	
CopyAboveIndicesFlag[xR][yR] = 0	
PaletteIndexMap[xR][yR] = PaletteIndexMap[xC][yC]	
} else {	
CopyAboveIndicesFlag[xR][yR] = 1	
if (!palette_transpose_flag)	

PaletteIndexMap[xR][yR] = PaletteIndexMap[xR][yR − 1]	
else	
PaletteIndexMap[xR][yR] = PaletteIndexMap[xR − 1][yR]	
}	
runPos++	
PaletteScanPos ++	
}	
}	
if(palette_escape_val_present_flag) {	
for(cIdx = startComp; cIdx < (startComp + numComps); cIdx++)	
for(sPos = 0; sPos < cbWidth* cbHeight; sPos++) {	
xC = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][sPos][0]	
yC = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][sPos][1]	
if(PaletteIndexMap[cIdx][xC][yC] = = MaxpaletteIndex) {	
palette_escape_val	ae(v)
PaletteEscapeVal[cIdx][xC][yC] = palette_escape_val	
}	
}	
}	
}	

Bảng 2: Ngữ nghĩa của lập mã bảng màu

Trong ngữ nghĩa sau, các chỉ mục dãy x0, y0 chỉ định vị trí (x0, y0) của mẫu luma trên cùng bên trái của khối mã hóa được xem xét liên quan đến mẫu luma trên cùng

bên trái của hình ảnh. Các chỉ mục dãy xC, yC xác định vị trí (xC, yC) của mẫu so với mẫu luma trên cùng bên trái của hình ảnh. Chỉ mục mảng startComp chỉ định thành phần màu đầu tiên của bảng bảng màu hiện thời. startComp bằng 0 chỉ ra thành phần Y; startComp bằng 1 chỉ ra thành phần Cb; startComp bằng 2 chỉ ra thành phần Cr. numComps chỉ rõ số lượng thành phần màu trong bảng bảng màu hiện thời.

Biến độc lập dự đoán bảng màu bao gồm các mục bảng màu từ đơn vị lập mã trước đó mà được sử dụng để dự đoán các mục trong bảng màu hiện thời. Biến PredictorPaletteSize[startComp] chỉ rõ kích thước của biến độc lập dự đoán bảng màu cho thành phần màu thứ nhất của bảng bảng màu hiện thời startComp.

Biến PalettePredictorEntryReuseFlags[i] bằng 1 chỉ rõ rằng mục thứ i trong biến độc lập dự đoán bảng màu được sử dụng lại trong bảng màu hiện thời. PalettePredictorEntryReuseFlags[i] bằng 0 chỉ rõ rằng mục thứ i trong biến độc lập dự đoán bảng màu không phải là mục trong bảng màu hiện thời. Tất cả phần tử của dãy PalettePredictorEntryReuseFlags[i] được khởi tạo là 0.

palette_predictor_run được sử dụng để xác định số lượng số không đứng trước mục nhập khác 0 trong dãy PalettePredictorEntryReuseFlags.

Là một yêu cầu tương thích luồng bit là giá trị của palette_predictor_run sẽ trong phạm vi 0 đến (PredictorPaletteSize − predictorEntryIdx), bao gồm nơi mà predictorEntryIdx tương ứng với vị thí hiện thời trong dãy PalettePredictorEntryReuseFlags. Biến NumPredictedPaletteEntries chỉ rõ số lượng các mục trong bảng màu hiện thời mà được sử dụng lại từ bảng màu của biến độc lập dự đoán. Giá trị của NumPredictedPaletteEntries sẽ ở trong phạm vi bao gồm 0 đến palette_max_size.

num_signaled_palette_entries chỉ rõ số lượng các mục trong bảng màu hiện thời mà được báo hiệu rõ ràng cho thành phần màu thứ nhất của bảng bảng màu hiện thời startComp.

Khi num_signaled_palette_entries không được thể hiện, nó được suy ra là bằng 0.

Biến CurrentPaletteSize[startComp] chỉ rõ kích thước của bảng màu hiện thời cho thành phần màu thứ nhất của bảng bảng màu hiện thời startComp và được suy ra như

sau:

CurrentPaletteSize[startComp] = NumPredictedPaletteEntries + num_signaled_palette_entries

Giá trị của CurrentPaletteSize[startComp] sẽ ở trong phạm vi bao gồm từ 0 đến palette_max_size.

new_palette_entries[cIdx][i] chỉ rõ giá trị cho mục bảng màu được báo hiệu thứ i cho thành phần màu cIdx.

Biến PredictorPaletteEntries[cIdx][i] chỉ rõ phần tử thứ i trong bảng màu của bộ báo hiệu cho các thành phần màu cIdx.

Biến CurrentPaletteEntries[cIdx][i] chỉ rõ phần tử thứ i trong bảng màu hiện thời cho thành phần màu cIdx và được suy ra như sau:

```
NumPredictedPaletteEntries = 0
for( i = 0; i < PredictorPaletteSize[ startComp ]; i++ )
    if( PalettePredictorEntryReuseFlags[ i ] ) {
        for( cIdx =startComp; cIdx < ( startComp + numComps ); cIdx++ )
            CurrentPaletteEntries[ cIdx ][ NumPredictedPaletteEntries ] =
PredictorPaletteEntries[ cIdx ][ i ]
        NumPredictedPaletteEntries++
        }
for( cIdx = startComp; cIdx < (startComp + numComps); cIdx++)
    for( i = 0; i < num_signaled_palette_entries[startComp]; i++ )
        CurrentPaletteEntries[ cIdx ][ NumPredictedPaletteEntries + i ] =
new_palette_entries[ cIdx ][ i ]
```

palette_escape_val_present_flag bằng 1 chỉ rõ rằng đơn vị lập mã hiện thời chứa ít nhất một mẫu lập mã thoát. escape_val_present_flag bằng 0 chỉ rõ rằng không có mẫu lập mã thoát nào trong đơn vị lập mã hiện thời. Khi không có mặt, giá trị của palette_escape_val_present_flag được suy ra là bằng 1.

Biến MaxpaletteIndex chỉ rõ giá trị khả thi tối đa cho chỉ mục bảng màu đối với đơn vị lập mã hiện thời. Giá trị của MaxpaletteIndex được thiết lập bằng

CurrentPaletteSize[startComp] − 1 + palette_escape_val_present_flag.

num_palette_indices_minus1 cộng 1 là số lượng của các chỉ mục được báo hiệu hoặc suy ra rõ ràng cho khối hiện thời.

Khi num_palette_indices_minus1 không có mặt, nó được suy ra là bằng 0.

palette_idx_idc là một chỉ báo của chỉ mục cho bảng bảng màu, CurrentPaletteEntries. Giá trị của palette_idx_idc sẽ ở trong phạm vi bao gồm từ 0 đến MaxpaletteIndex, cho chỉ mục thứ nhất trong khối và trong phạm vi bao gồm từ 0 đến (MaxpaletteIndex − 1) cho các chỉ mục còn lại trong khối.

Khi palette_idx_idc không có mặt, nó được suy ra là bằng 0.

Biến PaletteIndexIdc[i] lưu palette_idx_idc thứ n được báo hiệu và suy ra chính xác.

Tất cả các phần tử của dãy PaletteIndexIdc[i] được khởi tạo là 0.

copy_above_indices_for_final_run_flag bằng 1 chỉ rõ rằng các chỉ mục bảng màu của các vị trí cuối cùng trong đơn vị lập mã được sao chép từ các chỉ mục bảng màu trong hàng trên nếu quét ngang theo chiều ngang được sử dụng hoặc các chỉ mục bảng màu ở cột trái nếu quét ngang theo chiều dọc được sử dụng. copy_above_indices_for_final_run_flag bằng 0 chỉ rõ rằng các chỉ mục bảng màu của vị trí cuối cùng trong đơn vị lập mã được sao chép từ PaletteIndexIdc[num_palette_indices_minus1].

Khi copy_above_indices_for_final_run_flag không có mặt, nó được suy ra là bằng 0.

palette_transpose_flag bằng 1 chỉ rõ rằng quét ngang dọc được áp dụng để quét các chỉ mục cho các mẫu trong đơn vị lập mã hiện thời. palette_transpose_flag bằng 0 chỉ rõ rằng quét ngang theo chiều ngang được áp dụng để quét các chỉ mục cho các mẫu trong đơn vị lập mã hiện thời. Khi không có mặt, giá trị của palette_transpose_flag được suy ra là bằng 0.

Dãy TraverseScanOrder chỉ rõ thứ tự quét cho lập mã bảng màu. TraverseScanOrder được gán thứ tự quét ngang HorTravScanOrder nếu palette_transpose_flag bằng 0 và TraverseScanOrder được gán thứ tự quét dọc VerTravScanOrder nếu palette_transpose_flag bằng 1.

copy_above_palette_indices_flag bằng 1 chỉ rõ rằng chỉ mục bảng màu bằng chỉ mục

bảng màu ở cùng một vị trí trong hàng trên nếu quét ngang theo chiều ngang được sử dụng hoặc cùng một vị trí trong cột trái nếu quét ngang theo chiều dọc được sử dụng. copy_above_palette_indices_flag bằng 0 chỉ rõ rằng sự chỉ báo của chỉ mục bảng màu của mẫu được lập mã trong luồng bit hoặc được suy ra.

Biến CopyAboveIndicesFlag[xC][yC] bằng 1 chỉ rõ rằng chỉ mục bảng màu được sao chép từ chỉ mục bảng màu trong hàng trên (quét ngang) hoặc cột trái (quét dọc). CopyAboveIndicesFlag[xC][yC] bằng 0 chỉ rõ rằng chỉ mục bảng màu được lập mã hoặc được suy ra chính xác trong luồng bit. Các chỉ mục dãy xC, yC chỉ rõ vị trí (xC, yC) của mẫu liên quan đến mẫu luma trái trên đỉnh của hình ảnh. Giá trị của PaletteIndexMap[xC][yC] sẽ ở trong phạm vi bao gồm từ 0 đến (MaxpaletteIndex − 1).

Biến PaletteIndexMap[xC][yC] chỉ rõ chỉ mục bảng màu, mà là chỉ mục cho dãy được thể hiện bởi CurrentPaletteEntries. Các chỉ mục dãy xC, yC chỉ rõ vị trí (xC, yC) của mẫu liên quan đến mẫu luma bên trái trên đỉnh của hình ảnh. Giá trị của PaletteIndexMap[xC][yC] sẽ ở trong phạm vi bao gồm từ 0 đến MaxpaletteIndex.

Biến adjustedRefPaletteIndex được suy ra như sau:

```
adjustedRefPaletteIndex = MaxpaletteIndex + 1
if( PaletteScanPos > 0 ) {
    xcPrev =
x0 + TraverseScanOrder[ log2CbWidth ][ log2bHeight ][ PaletteScanPos − 1 ][ 0 ]
    ycPrev =
y0 + TraverseScanOrder[ log2CbWidth ][ log2bHeight ][ PaletteScanPos − 1 ][ 1 ]
    if( CopyAboveIndicesFlag[ xcPrev ][ ycPrev ] = = 0 ) {
        adjustedRefPaletteIndex = PaletteIndexMap[ xcPrev ][ ycPrev ] {
    }
    else {
        if( !palette_transpose_flag )
            adjustedRefPaletteIndex = PaletteIndexMap[ xC ][ yC − 1 ]
        else
```

adjustedRefPaletteIndex = PaletteIndexMap[xC − 1][yC]

}

}

Khi CopyAboveIndicesFlag[xC][yC] bằng 0, biến CurrPaletteIndex được suy ra như sau:

if(CurrPaletteIndex >= adjustedRefPaletteIndex)

CurrPaletteIndex++

palette_run_prefix, khi có mặt, chỉ rõ phần tiền tố trong mã nhị phân của PaletteRunMinus1.

palette_run_suffix được sử dụng trong phép suy ra biến PaletteRunMinus1. Khi không có mặt, giá trị của palette_run_suffix được suy ra là bằng 0.

Khi RunToEnd bằng 0, biến PaletteRunMinus1 được suy ra như sau:

- Nếu PaletteMaxRunMinus1 bằng 0, PaletteRunMinus1 được thiết lập bằng 0.
- Trường hợp khác (PaletteMaxRunMinus1 lớn hơn 0) áp dụng điều sau:
  - Nếu palette_run_prefix nhỏ hơn 2, áp dụng điều sau:

    PaletteRunMinus1 = palette_run_prefix
  - Trường hợp khác (palette_run_prefix lớn hơn hoặc bằng 2), áp dụng điều sau:

    PrefixOffset = 1 << (palette_run_prefix − 1)

    PaletteRunMinus1 = PrefixOffset + palette_run_suffix

Biến PaletteRunMinus1 được sử dụng như sau:

- Nếu CopyAboveIndicesFlag[xC][yC] bằng 0, PaletteRunMinus1 chỉ rõ số vị trí liên tiếp trừ đi 1 với cùng một chỉ mục bảng màu.
- Trường hợp khác, nếu palette_transpose_flag bằng 0, PaletteRunMinus1 chỉ rõ số vị trí liên tiếp trừ đi 1 với cùng một chỉ mục bảng màu như được sử dụng ở vị trí tương ứng trong hàng trên.
- Trường hợp khác, PaletteRunMinus1 chỉ rõ số vị trí liên tiếp trừ đi 1 với cùng một chỉ mục bảng màu như được sử dụng ở vị trí tương ứng trong cột trái.

Khi RunToEnd bằng 0, biến PaletteMaxRunMinus1 đại diện cho giá trị lớn nhất có thể cho PaletteRunMinus1 và nó là một yêu cầu về tương thích của luồng bit là giá trị

của PaletteMaxRunMinus1 sẽ lớn hơn hoặc bằng 0.

palette_escape_val chỉ rõ giá trị mẫu được lập mã thoát lượng tử hóa cho một thành phần.

Biến PaletteEscapeVal[cIdx][xC][yC] chỉ rõ giá trị thoát của mẫu mà PaletteIndexMap[xC][yC] cho nó bằng MaxpaletteIndex và palette_escape_val_present_flag bằng 1. Chỉ mục dãy cIdx chỉ rõ thành phần màu. Các chỉ mục dãy xC, yC chỉ rõ vị trí (xC, yC) của mẫu liên quan đến mẫu luma bên trái trên đỉnh của hình ảnh.

Là một yêu cầu về tương thích của luồng bit, PaletteEscapeVal[cIdx][xC][yC] sẽ ở trong phạm vi bao gồm từ 0 đến (1 << ($BitDepth_Y$ + 1)) − 1, đối với cIdx bằng 0, và trong phạm vi bao gồm từ 0 đến (1 << ($BitDepth_C$ + 1)) − 1, đối với cIdx khác 0.

Bảng 3: Từ mã nhị phân và lựa chọn ngữ cảnh CABAC cho cú pápx palette_run_prefix

Từ mã nhị phân của palette_run_prefix được suy ra thông qua xử lý nhị phân hóa truncated rice như được mô tả dưới đây với thông số nhập vào cMax = Floor(Log2(PaletteMaxRunMinus1)) + 1, cRiceParam = 0.

Xử lý nhị phân hóa Truncated Rice

Đầu vào cho thao tác xử lý này là yêu cầu đối với nhị phân hóa truncated Rice (TR), cMax và cRiceParam.

Đầu ra cho thao tác xử lý này là nhị phân hóa TR liên kết với mỗi giá trị symbolVal với chuỗi bin tương ứng.

Chuỗi bin TR khi có mặt là sự ghép nối của chuỗi bin tiền bố và chuỗi bin hậu tố.

Đối với phép suy ra chuỗi bin tiền tố, áp dụng điều sau:

- Giá trị tiền tố của symbolVal, prefixVal, được suy ra như sau:

  prefixVal = symbolVal >> cRiceParam

- Giá trị tiền tố của chuỗi bin TR được chỉ rõ như sau:
  - Nếu prefixVal nhỏ hơn cMax >> cRiceParam, chuỗi bin tiền tố là chuỗi bin có chiều dài prefixVal + 1 được lập chỉ mục bởi binIdx. Các bin cho binIdx nhỏ hơn prefixVal bằng 1. Bin có binIdx bằng prefixVal bằng 0. Bảng 3-1

dưới đây minh họa chuỗi bin của phép nhị phân hóa một toán hạng này cho prefixVal.

– Trường hợp khác, chuỗi bin là chuỗi bit có chiều dài cMax >> cRiceParam với tất cả bin bằng 1.

Bảng: 3-1 – Chuỗi bin của phép nhị phân hóa một toán hạng (mang thông tin)

Tiền tố Val	Chuỗi bin					
0	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
...						
binIdx	0	1	2	3	4	5

Khi cMax lớn hơn symbolVal và cRiceParam lớn hơn 0, hậu tố của chuỗi bin TR có mặt và được suy ra như sau:

– Giá trị hậu tố suffixVal được suy ra như sau:

suffixVal = symbolVal − ((prefixVal) << cRiceParam)

– Hậu tố của chuỗi bin TR được chỉ rõ bằng gọi phép xử lý nhị phân hóa có chiều dài cố định (fixed-length: FL) được chỉ rõ trong đặc điểm kỹ thuật VVC cho suffixVal với giá trị cMax bằng (1 << cRiceParam) − 1.

Lưu ý – Đối với thông số đầu vào cRiceParam = 0, phép nhị phân hóa TR chính xác là phép nhị phân hóa truncated unary và nó luôn được gọi với giá trị cMax bằng giá trị có thể lớn nhất của phần tử cú pháp được giải mã.

Phép suy ra của ctxInc cho phần tử cú pháp palette_run_prefix

Đầu vào cho phép xử lý này là chỉ mục bin binIdx và các phần tử cú pháp copy_above_palette_indices_flag và palette_idx_idc.

Đầu ra của phép xử lý này là biến ctxInc.

Biến ctxInc được suy ra như sau:

– Nếu copy_above_palette_indices_flag bằng 0 và binIdx bằng 0, ctxInc được suy ra như sau:

ctxInc = (palette_idx_idc < 1) ? 0 : ((palette_idx_idc < 3) ? 1 : 2)

– Trường hợp khác, ctxInc được đề xuất trong bảng 3-2:

Bảng: 3-2 – Đặc điểm kỹ thuật của

ctxIdxMap[copy_above_palette_indices_flag][binIdx]

binIdx	0	1	2	3	4	> 4
copy_above_palette_indices_flag == 1	5	6	6	7	7	Bỏ qua
copy_above_palette_indices_flag == 0	0, 1, 2	3	3	4	4	Bỏ qua

Là nhóm hệ số (coefficient group: CG) được sử dụng trong lập mã hệ số biến đổi, CU được chia thành nhiều nhóm hệ số dựa vào nhiều đường, mỗi nhóm bao gồm m mẫu, khi chỉ mục chạy, giá trị chỉ mục bảng màu và màu được lượng tử hóa cho chế độ thoát được mã hóa/phân tích cú pháp tuần tự cho mỗi CG. Kết quả là, pixel trong CG dựa trên dòng có thể được tái cấu trúc sau khi phân tích cú pháp phần tử cú pháp, ví dụ, chỉ mục chạy, giá trị chỉ mục bảng màu và màu lượng tử hóa thoát cho CG, mà làm giảm mạnh yêu cầu đệm trong chế độ bảng màu trong VTM6.0, trong đó các phần tử cú pháp cho toàn bộ CU phải được phân tích cú pháp (và được lưu) trước khi tái cấu trúc.

Trong ứng dụng này, mỗi CU của chế độ bảng màu được chia thành nhiều đoạn có m mẫu ($m = 8$ trong phép kiểm tra này) căn cứ vào chế độ quét ngang, như được thể hiện trong Fig. 5D.

Thứ tự mã hóa cho việc lập mã chạy bảng màu trong mỗi đoạn là như sau: đối với mỗi pixel, một ngữ cảnh được lập hóa bin run_copy_flag = 0 được báo hiệu chỉ ra rằng pixel có cùng chế độ với pixel trước đó, tức là pixel được quét trước đó và pixel hiện tại đều thuộc loại chạy COPY_ABOVE hoặc pixel được quét trước đó và pixel

hiện tại đều là của loại chạy INDEX và cùng một giá trị chỉ mục. Nếu không, run_copy_flag = 1 được báo hiệu.

Nếu pixel hiện tại và pixel trước đó ở chế độ khác, một bin copy_above_palette_indices_flag được lập mã ngữ cảnh sẽ được báo hiệu chỉ ra kiểu chạy, tức là INDEX hoặc COPY_ABOVE, của pixel. Trong trường hợp này, bộ giải mã không phải phân tích cú pháp kiểu chạy nếu mẫu ở hàng đầu tiên (quét ngang theo chiều ngang) hoặc trong cột đầu tiên (quét ngang dọc) kể từ khi chế độ INDEX được sử dụng mặc định. Bộ giải mã cũng không phải phân tích cú pháp loại chạy nếu loại chạy đã phân tích cú pháp trước đó là COPY_ABOVE.

Sau khi lập mã chạy bảng màu của pixel trong một đoạn, các giá trị chỉ mục (đối với chế độ INDEX) và màu thoát được lượng tử hóa được lập mã khi bỏ qua (bypass) bin và được nhóm lại ngoài mã hóa/phân tích cú pháp của các bin được lập mã theo ngữ cảnh để cải thiện thông lượng trong mỗi CG dựa trên dòng. Vì giá trị chỉ mục hiện được lập mã/phân tích cú pháp sau khi chạy lập mã, nên bộ mã hóa không phải báo hiệu số lượng giá trị chỉ mục num_palette_indices_minus1 và loại chạy cuối cùng copy_above_indices_for_final_run_flag. Cú pháp của CG chế độ bảng màu được minh họa trong bảng 4.

Bảng 4: Cú pháp của lập mã bảng màu

palette_coding(x0, y0, cbWidth, cbHeight, startComp, numComps) {	**Bộ mô tả**
palettePredictionFinished = 0	
NumPredictedPaletteEntries = 0	
for(predictorEntryIdx = 0; predictorEntryIdx < PredictorPaletteSize[startComp] && !palettePredictionFinished && NumPredictedPaletteEntries[startComp] < palette_max_size; predictorEntryIdx++) {	
palette_predictor_run	ae(v)
if(palette_predictor_run != 1) {	
if(palette_predictor_run > 1)	

predictorEntryIdx += palette_predictor_run − 1	
PalettePredictorEntryReuseFlags[predictorEntryIdx] = 1	
NumPredictedPaletteEntries++	
} else	
palettePredictionFinished = 1	
}	
if(NumPredictedPaletteEntries < palette_max_size)	
num_signaled_palette_entries	ae(v)
for(cIdx = startComp; cIdx < (startComp + numComps); cIdx++)	
for(i = 0; i < num_signaled_palette_entries; i++)	
new_palette_entries[cIdx][i]	ae(v)
if(CurrentPaletteSize[startComp] > 0)	
palette_escape_val_present_flag	ae(v)
if(MaxpaletteIndex > 0) {	
adjust = 0	
palette_transpose_flag	ae(v)
}	
if(treeType != DUAL_TREE_CHROMA && palette_escape_val_present_flag) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	
}	
if(treeType != DUAL_TREE_LUMA && palette_escape_val_present_flag) {	
if(cu_chroma_qp_offset_enabled_flag && !IsCuChromaQpOffsetCoded) {	

cu_chroma_qp_offset_flag	ae(v)
if(cu_chroma_qp_offset_flag)	
cu_chroma_qp_offset_idx	ae(v)
}	
}	
PreviousRunTypePosition = 0	
PreviousRunType = 0	
for (subSetId = 0; subSetId <= (cbWidth* cbHeight - 1) >> 4; subSetId++) {	
minSubPos = subSetId << 4	
if(minSubPos + 16 > cbWidth * cbHeight)	
maxSubPos = cbWidth * cbHeight	
else	
maxSubPos = minSubPos + 16	
RunCopyMap[0][0] = 0	
log2CbWidth = Log2(cbWidth)	
log2CbHeight = Log2(cbHeight)	
PaletteScanPos = minSubPos	
while(PaletteScanPos < maxSubPos) {	
xC = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][0]	
yC = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][1]	
if(PaletteScanPos > 0) {	
xcPrev = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos − 1][0]	

ycPrev = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos − 1][1]	
}	
if (MaxpaletteIndex > 0 && PaletteScanPos > 0) {	
run_copy_flag	ae(v)
RunCopyMap[xC][yC] = run_copy_flag	
}	
CopyAboveIndicesFlag[xC][yC] = 0	
if(MaxpaletteIndex > 0 && ! RunCopyMap[startComp][xC][yC]) {	
if(((!palette_transpose_flag && yC > 0) || (palette_transpose_flag && xC > 0)) && CopyAboveIndicesFlag[xcPrev][ycPrev] = = 0) {	
copy_above_palette_indices_flag	ae(v)
CopyAboveIndicesFlag[xC][yC] = copy_above_palette_indices_flag	
}	
PreviousRunType = CopyAboveIndicesFlag[xC][yC]	
PreviousRunTypePosition = curPos	
} else {	
CopyAboveIndicesFlag[xC][yC] = CopyAboveIndicesFlag[xcPrev][ycPrev]	
}	
}	
PaletteScanPos ++	
}	
PaletteScanPos = minSubPos	
while(PaletteScanPos < maxSubPos) {	

xC = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][0]	
yC = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos][1]	
if(PaletteScanPos > 0) {	
xcPrev = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos − 1][0]	
ycPrev = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][PaletteScanPos − 1][1]	
}	
if (MaxpaletteIndex > 0) {	
if (! RunCopyMap [xC][yC] && CopyAboveIndicesFlag[xC][yC] = = 0) {	
if(MaxpaletteIndex − adjust > 0) {	
palette_idx_idc	ae(v)
}	
adjust = 1	
}	
}	
if (! RunCopyMap [xC][yC] && CopyAboveIndicesFlag[xC][yC] = = 0) {	
CurrPaletteIndex = palette_idx_idc	
if(CopyAboveIndicesFlag[xC][yC] = = 0) {	
PaletteIndexMap[xC][yC] = CurrPaletteIndex	
} else {	

if (!palette_transpose_flag)	
PaletteIndexMap[xC][yC] = PaletteIndexMap[xC][yC − 1]	
else	
PaletteIndexMap[xC][yC] = PaletteIndexMap[xC − 1][yC]	
}	
}	
if(palette_escape_val_present_flag) {	
for(cIdx = startComp; cIdx < (startComp + numComps); cIdx++)	
for(sPos = minSubPos ; sPos < maxSubPos; sPos++) {	
xC = x0 + TraverseScanOrder[log2CbWidth][log2CbHeight][sPos][0]	
yC = y0 + TraverseScanOrder[log2CbWidth][log2CbHeight][sPos][1]	
if(PaletteIndexMap[cIdx][xC][yC] == MaxpaletteIndex) {	
palette_escape_val	ae(v)
PaletteEscapeVal[cIdx][xC][yC] = palette_escape_val	
}	
}	
}	
}	
}	

FIG. 6 là lưu đồ 600 minh họa quy trình điển hình mà nhờ đó bộ giải mã viđeo (ví dụ, bộ giải mã viđeo 30) thực hiện kỹ thuật giải mã dữ liệu viđeo phù hợp với vài phương án thực hiện của sáng chế.

Đối với chế độ bảng màu trong VVC, chế độ bảng màu có thể áp dụng cho các CU mà bằng hoặc nhỏ hơn 64x64 pixel. Theo vài phương án, kích thước khối của chế độ bảng màu tối thiểu được sử dụng để giảm độ phức tạp sao cho chế độ bảng màu được vô hiệu hóa cho các đơn vị lập mã mà có nhỏ hơn hoặc bằng kích thước khối của chế độ bảng màu tối thiểu. Ví dụ, chế độ bảng màu được vô hiệu hóa cho tất cả các khối có kích thước nhỏ hơn so với ngưỡng nào đó, ví dụ, 16 mẫu. Bởi vì có các định dạng chroma khác nhau (ví dụ, 4:4:4, 4:2:2, 4:2:0) và các loại cây lập mã khác nhau (ví dụ, SINGLE_TREE, DUAL_TREE_LUMA và DUAL_TREE_CHROMA), ngưỡng này có thể khác nhau. "SINGLE_TREE" chỉ ra rằng thành phần luma và chroma của một hình ảnh được phân vùng theo cùng một cách sao cho hai thành phần này chia sẻ cùng một bảng bảng màu và biến độc lập dự đoán bảng màu dưới chế độ bảng màu. Ngược lại, "DUAL_TREE" chỉ ra rằng thành phần luma và chroma của hình ảnh được phân vùng riêng rẽ sao cho hai thành phần này có các bảng bảng màu và biến độc lập dự đoán bảng màu khác nhau dưới một chế độ bảng màu. Ví dụ, đối với trường hợp loại "DUAL_TREE", tức là thành phần chroma được xem xét riêng biệt, chế độ bảng màu cho thành phần chroma của các CU mà nhỏ hơn hoặc bằng 16 mẫu nên được vô hiệu hóa để giảm độ phức tạp. Bảng 5 sau nêu một ví dụ về cú pháp được chỉ định.

Bảng 5: Cờ kích hoạt chế độ bảng màu dưới các loại cây lập mã và định dạng chroma khác nhau

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	Bộ mô tả
...	

if(CuPredMode[chType][x0][y0] == MODE_INTRA && sps_palette_enabled_flag && cbWidth <= 64 && cbHeight <= 64 && cu_skip_flag[x0][y0] = = 0 && modeType != MODE_TYPE_INTER && (cbWidth* cbHeight > (treeType != DUAL_TREE_CHROMA? 16:16* SubWidthC* SubHeightC)))	
pred_mode_plt_flag	u(1)
}	
...	
}	

Trong bảng 5, *pred_mode_plt_flag* chỉ rõ liệu chế độ bảng màu được kích hoạt (ví dụ, giá trị là 1) hoặc vô hiệu hóa (ví dụ, giá trị là 0) cho đơn vị lập mã. Các thông số như SubWidthC và SubHeightC được kết hợp với định dạng chroma của đơn vị lập mã như sau:

Định dạng chroma	**SubWidthC**	**SubHeightC**
Đơn sắc	1	1
4:4:4	1	1
4:2:2	2	1
4:2:0	2	2

Trong mẫu đơn sắc, chỉ có một dãy mẫu, mà mà trên danh nghĩa được coi là dãy luma. Trong lấy mẫu 4: 2: 0, mỗi dãy trong số hai mảng sắc độ có một nửa chiều cao và một nửa chiều rộng của dãy luma. Trong lấy mẫu 4: 2: 2, mỗi dãy trong số hai dãy

màu có cùng chiều cao và một nửa chiều rộng của dãy luma. Trong lấy mẫu 4: 4: 4, mỗi dãy trong số hai dãy màu có cùng chiều cao và chiều rộng với dãy luma.

Theo phương án khác, đối với trường hợp cây đơn, chế độ bảng màu được vô hiệu hóa đối với các CU có khối luma kích thước nhỏ. Trong một ví dụ, chế độ bảng màu cho các CU có khối luma mà nhỏ hơn hoặc bằng 16 pixel được vô hiệu hóa trong trường hợp cây đơn. Trong một ví dụ cụ thể, chế độ bảng màu có thể được kích hoạt cho CU 8x4 mà chứa 8x4 mẫu luma và hai mẫu 4x2 chroma vì sự kích hoạt bảng màu được đặt điều kiện trên kích thước của mẫu luma bỏ qua kích thước chroma.

Theo vài phương án, trong trường hợp cây đơn, thành phần luma (ví dụ, Y) và thành phần chroma (ví dụ, Cb và Cr) của CU được phân vùng theo nhiều cách. Trong trường hợp cây kép, thành phần luma và thành phần chroma có các cây phân vùng khác nhau. Trong trường hợp cây kép cục bộ (local dual tree), luma và thành phần chroma có bảng bảng màu khác nhau dưới trường hợp cây đơn. Dưới trường hợp cây kép cục bộ, lập mã bảng màu được áp dụng riêng cho thành phần luma và thành phần chroma trong CU.

Theo phương án khác, đối trường hợp cây kép cục bộ, chế độ bảng màu được vô hiệu hóa đối với các khối có kích thước nhỏ. Trong một ví dụ, đối với trường hợp cây kép cục bộ, chế độ bảng màu đối với CU mà nhỏ hơn so với hoặc bằng 32 pixel được vô hiệu hóa.

Theo vài phương án, việc áp dụng chế độ bảng màu không bao gồm trường hợp cây kép cục bộ (local dual tree). Trong VVC, trong trường hợp cây đơn, chế độ bảng màu có thể được áp dụng cho CU với khối luma mà bằng hoặc nhỏ hơn 64x64 pixel và lớn hơn 4x4 pixel. Trong trường hợp cây kép, chế độ bảng màu có thể áp dụng cho CU mà bằng hoặc nhỏ hơn 64x64 pixel và lớn hơn 4x4 pixel, cho cả thành phần luma và chroma. Theo phương án khác, để giảm độ phức tạp, chế độ bảng màu được vô hiệu hóa cho trường hợp các cây kép cục bộ. Bảng 6 sau nêu một ví dụ về cú pháp trong thiết kế VVC. modeType của CU bằng MODE_TYPE_INTRA trong VVC, có nghĩa là CU là ở trong trường hợp cây kép cục bộ. Sự thay đổi cho VVC được minh họa dưới đây.

Bảng 6: Cú pháp điển hình không bao gồm chế độ bảng màu trong trường hợp cây kép cục bộ

– coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	– Bộ mô tả
– …	–
– if(CuPredMode[chType][x0][y0] == MODE_INTRA && sps_palette_enabled_flag && cbWidth <= 64 && cbHeight <= 64 && cu_skip_flag[x0][y0] = = 0 && modeType != MODE_TYPE_INTER && (cbWidth* cbHeight > (treeType != DUAL_TREE_CHROMA? 16:16* SubWidthC* SubHeightC)) && (modeType ! = MODE_TYPE_INTRA))	–
– pred_mode_plt_flag	– ae(v)
– }	–
– …	–
– }	–

Theo phương án khác, đối với trường hợp cây kép cục bộ, chế độ bảng màu chỉ được vô hiệu hóa cho thành phần chroma. Theo cách khác, dưới trường hợp cây kép cục bộ, chế độ bảng màu có thể được áp dụng cho CU luma, mà không thể áp dụng cho CU chroma. Bảng 7 sau nêu một ví dụ về cú pháp trong thiết kế VVC. Khi modeType của CU bằng MODE_TYPE_INTRA và treeType của CU bằng DUAL_TREE_CHROMA trong thiết kế VVC, có nghĩa là CU là thành phần chroma và trong trường hợp cây kép cục bộ. Sự thay đổi cho VVC được minh họa dưới đây trong bảng 7.

Bảng 7: Cú pháp điển hình của vô hiệu hóa chế độ bảng màu chỉ cho thành phần chroma trong trường hợp cây kép cục bộ

– coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	– Bộ mô tả
– …	–
– if(CuPredMode[chType][x0][y0] == MODE_INTRA && sps_palette_enabled_flag && cbWidth <= 64 && cbHeight <= 64 && cu_skip_flag[x0][y0] == 0 && modeType != MODE_TYPE_INTER && (cbWidth* cbHeight > (treeType != DUAL_TREE_CHROMA? 16:16* SubWidthC* SubHeightC)) && ((modeType != MODE_TYPE_INTRA) || (treeType != DUAL_TREE_CHROMA)))	–
– pred_mode_plt_flag	– ae(v)
– }	–
– …	–
– }	–

Theo vài phương án, đối với trường hợp cây kép cục bộ, dự đoán bảng màu được cập nhật cho cả thành phần luma và chroma. Theo đặc điểm kỹ thuật VVC hiện thời, dưới trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu được thực hiện chỉ cho thành phần chroma. Cụ thể hơn, dự đoán bảng màu có thể không được cập nhật trong khi lập mã từng CU luma trong chế độ bảng màu dưới cây kép cục bộ. Dự đoán bảng màu có thể được cập nhật sau khi lập mã thành phần chroma cuối cùng của mỗi CU chroma trong chế độ bảng màu trong cây kép cục bộ.

Quy trình cập nhật của dự đoán bảng màu được định rõ trong VVC như mô tả ở trên không hiệu quả cho hiệu suất lập mã. Theo vài phương án được bộc lộ ở đây, để cải thiện hiệu quả lập mã, dưới trường hợp cây kép cục bộ, quy trình cập nhật của phép dự đoán bảng màu được thực hiện cho cả CU luma và chroma. Cụ thể hơn, phép đoán bảng màu có thể được cập nhật trước tiên trong khi lập mã mỗi CU luma dưới cây kép cục bộ, sau đó là lập mã mỗi CU chroma dưới cùng một cây kép cục bộ. Bảng 8 sau

nêu một ví dụ về cú pháp trong thiết kế VVC. Trong thiết kế VVC, cIdx biến chỉ rõ thành phần màu/viđeo của CU hiện thời, 0 cho thành phần luma, 1 cho thành phần Cb, và 2 cho thành phần Cr. Sự thay đổi cho VVC được minh họa dưới đây.

Bảng 8: Cú pháp điển hình của quy trình cập nhật của phép dự đoán bảng màu được thực hiện cho cả thành phần luma và chroma trong trường hợp cây kép cục bộ

Quy trình giải mã cho chế độ bảng màu

– …

Khi localDualTree bằng 1, điều sau được áp dụng:

– Nếu treeType bằng DUAL_TREE_LUMA, điều sau được áp dụng cho i = 0..num_signalled_palette_entries[startComp] − 1:

CurrentPaletteEntries[1][NumPredictedPaletteEntries + i] = 1 << (BitDepth − 1) (450)

CurrentPaletteEntries[2][NumPredictedPaletteEntries + i] = 1 << (BitDepth − 1) (451)

– Trường hợp khác (nếu treeType bằng DUAL_TREE_CHROMA), điều sau được áp dụng cho i = 0..num_signalled_palette_entries[startComp] − 1:

CurrentPaletteEntries[0][NumPredictedPaletteEntries + i] = 1 << (BitDepth − 1) (452)

– Các biến CurrentPaletteSize[0], startComp, numComps và maxNumPalettePredictorSize được suy ra như sau:

CurrentPaletteSize[0] = CurrentPaletteSize[startComp]

startComp = 0

numComps = 3

maxNumPalettePredictorSize = 63

Khi một trong các điều kiện sau đúng:

– cIdx bằng 0 và numComps bằng 1;

– cIdx bằng 0 và localDualTree bằng 1;

– cIdx bằng 2;

– Giá trị PredictorPaletteSize[startComp] và dãy PredictorPaletteEntries được suy ra và được sửa đổi như sau:

– …

Theo vài phương án, phép dự đoán bảng màu được cập nhật một phần trong các trường hợp cây kép cục bộ. Như được đề cập ở trên, dưới trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu được thực hiện cho cả CU luma và chroma. Cụ thể hơn, phép dự đoán bảng màu có thể được cập nhật trước tiên trong khi lập mã từng CU luma dưới cây kép cục bộ, sau đó là lập mã từng CU chroma dưới cùng một cây kép cục bộ.

Cho rằng các CU dưới cây kép cục bộc đều là Cu kích thước nhỏ, thực hiện quy trình cập nhật của dự đoán bảng màu trên các CU này theo cách tuần tự đòi hỏi nhiều chu kỳ tính toán. Theo các phương án khác, để giảm độ phức tạp, dưới trường hợp cây kép cục bộ, một bảng bảng màu chia sẻ được sử dụng cho vài hoặc tất cả các Cu mà không cập nhật bảng.

Trong một ví dụ, đối với trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu được vô hiệu hóa dưới chế độ bảng màu. Bảng 9 sau nêu một ví dụ về cú pháp trong thiết kế VVC, trong đó các đường mã từ 450 đến 456 trong bảng 8 cũng được loại bỏ. Trong thiết kế VVC, biến cIdx chỉ rõ thành phần màu của CU hiện thời, 0 cho luma, 1 cho Cb, và 2 cho thành phần Cr. Sự thay đổi cho VVC được minh họa dưới đây.

Bảng 9: Cú pháp điển hình của quy trình cập nhật của dự đoán bảng màu được vô hiệu hóa cho cả thành phần luma và chroma trong trường hợp cây kép cục bộ

Quy trình giải mã cho chế độ bảng màu … Khi một trong các điều kiện sau đúng: – cIdx bằng 0 và numComps bằng 1 và localDualTree khác 1; – cIdx bằng 2 và localDualTree khác 1; – giá trị PredictorPaletteSize[startComp] và dãy PredictorPaletteEntries được suy ra và được sửa đổi như sau: – …

Theo vài phương án, dưới trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu trong chế độ bảng màu được vô hiệu hóa cho các CU với kích thước khối luma nhỏ hơn hoặc bằng 32 pixel. Trong trường hợp này, quy trình cập nhật của dự đoán bảng màu trong chế độ bảng màu có thể được kích hoạt cho CU 8x8 hoặc CU lớn hơn mà chứa ít nhất 8x8 mẫu luma.

Theo phương án khác, đối với trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu chỉ được vô hiệu hóa cho CU chroma. Bảng 10 sau nêu một ví dụ về cú pháp trong thiết kế VVC. Trong thiết kế VVC, biến cIdx chỉ rõ thành phần màu của CU hiện thời, 0 cho thành phần luma, 1 cho thành phần Cb, và 2 cho thành phần Cr. Sự thay đổi cho VVC được minh họa dưới đây.

Bảng 10: Cú pháp điển hình của quy trình cập nhật của dự đoán bảng màu được vô hiệu hóa chỉ cho thành phần chroma trong trường hợp cây kép cục bộ

Quy trình giải mã cho chế độ bảng màu

– …

Khi localDualTree bằng 1, áp dụng điều sau:

– Nếu treeType bằng DUAL_TREE_LUMA, điều sau được áp dụng cho i = 0..num_signalled_palette_entries[startComp] − 1:

CurrentPaletteEntries[1][NumPredictedPaletteEntries + i] =

1 << (BitDepth − 1)

CurrentPaletteEntries[2][NumPredictedPaletteEntries + i] =

1 << (BitDepth − 1)

– Các biến CurrentPaletteSize[0], startComp, numComps và maxNumPalettePredictorSize được suy ra như sau:

CurrentPaletteSize[0] = CurrentPaletteSize[startComp]

startComp = 0

numComps = 3

maxNumPalettePredictorSize = 63

Khi một trong các điều kiện sau đúng:

– cIdx bằng 0 và numComps bằng 1;

– cIdx bằng 0 và localDualTree bằng 1;

– cIdx bằng 2 và localDualTree khác 1;

– Giá trị PredictorPaletteSize[startComp] và dãy PredictorPaletteEntries được suy ra và được sửa đổi như sau:

– …

Theo vài phương án, dưới trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu trong chế độ bảng màu được cập nhật độc lập cho các thành phần viđeo khác nhau.

Như được đề cập ở trên, dưới trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu được thực hiện tuần tự. Điều này cũng có nghĩa là giải mã của CU chroma trong chế độ bảng màu dưới cây kép cục bộ không thể được bắt đầu cho đến

khi tất cả CU luma dưới cùng một cây kép cục bộ được giải mã. Điều này gây ra sự trễ không mong muốn trong thực hiện giải mã/mã hóa (codec) phần cứng.

Để giải quyết vấn đề trên, theo vài phương án, phép dự đoán bảng màu cập nhật quy trình được thực hiện độc lập cho các thành phần viđeo khác nhau (ví dụ luma và chroma) trong cây kép cục bộ, sao cho chế độ bảng màu lập mã cho thành phần chroma có thể được thực hiện song song với thành phần luma. Trong một ví dụ về phương án này, dưới cây kép cục bộ, bảng màu ở thời điểm bắt đầu cây kép cục bộ được sử dụng làm bảng màu khởi đầu cho cả CU luma và chroma.

Theo vài phương án, việc cải thiện quy trình cập nhật của dự đoán bảng màu trong trường hợp cây kép cục bộ được thực hiện. Dưới trường hợp cây kép cục bộ, quy trình cập nhật của dự đoán bảng màu được thực hiện riêng cho thành phần luma và chroma. Cụ thể hơn, dự đoán bảng màu có thể được cập nhật trước tiên trong khi lập mã mỗi CU luma dưới cây kép cục bộ, sau đó là lập mã mỗi CU chroma dưới cùng một cây kép cục bộ. Kết quả là, trong cập nhật dự đoán bảng màu trong khi lập mã CU luma dưới cây kép cục bộ, thông tin chroma của pixel được đặt cùng vị trí có thể không có sẵn và ngược lại.

Theo vài phương án, để cải thiện hiệu quả lập mã, trong cập nhật dự đoán bảng màu trong khi lập mã CU của một thành phần viđeo (ví dụ luma và/hoặc chroma) dưới cây kép cục bộ, các giá trị thành phần viđeo khác (ví dụ chroma và/hoặc luma) của ứng viên có sẵn trước đó trong bảng màu có thể được sử dụng. Trong một ví dụ cho trường hợp cây kép cục bộ, trong suốt quy trình cập nhật của dự đoán bảng màu cho thành phần luma, thành phần chroma của ứng viên có sẵn thứ nhất có thể được sử dụng làm thành phần chroma của mục bảng màu được bổ sung mới. Bảng 12 sau nêu một ví dụ về cú pháp trong thiết kế VVC. Sự thay đổi thành VVC được minh họa dưới đây.

Bảng 11: Cú pháp điển hình của quy trình cập nhật của dự đoán bảng màu sử dụng thành phần chroma của ứng viên có sẵn thứ nhất

Quy trình giải mã cho chế độ bảng màu

– …

Khi localDualTree bằng 1, áp dụng điều sau:

– Nếu treeType bằng DUAL_TREE_LUMA, điều sau được áp dụng cho i = 0..num_signalled_palette_entries[startComp] − 1:

  – Nếu NumPredictedPaletteEntries khác 0:

    CurrentPaletteEntries[1][NumPredictedPaletteEntries + i] =

    CurrentPaletteEntries[1][0]

    CurrentPaletteEntries[2][NumPredictedPaletteEntries + i] =

    CurrentPaletteEntries[2][0]

  – Trường hợp khác:

    CurrentPaletteEntries[1][NumPredictedPaletteEntries + i] =

    1 << (BitDepth − 1)

    CurrentPaletteEntries[2][NumPredictedPaletteEntries + i] =

    1 << (BitDepth − 1)

– Trường hợp khác (nếu treeType bằng DUAL_TREE_CHROMA), điều sau được áp dụng cho i = 0..num_signalled_palette_entries[startComp] − 1:

  – Nếu NumPredictedPaletteEntries khác 0:

    CurrentPaletteEntries[0][NumPredictedPaletteEntries + i] =

    CurrentPaletteEntries[0][0]

  – Trường hợp khác:

    CurrentPaletteEntries[0][NumPredictedPaletteEntries + i] =

    1 << (BitDepth − 1)

– …

Như được thể hiện trong FIG. 6, theo vài phương án, bộ giải mã viđeo 30 nhận, từ luồng bit, nhiều phần tử cú pháp kết hợp với đơn vị lập mã, và nhiều phần tử cú pháp chỉ ra loại cây lập mã của đơn vị lập mã, và liệu chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã (610).

Bộ giải mã viđeo 30, phù hợp với việc xác định rằng loại cây lập mã của đơn vị lập mã là cây đơn, và chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã (620), vô hiệu hóa chế độ bảng màu cho đơn vị lập mã khi đơn vị lập mã có kích thước bằng hoặc nhỏ hơn ngưỡng định trước (630).

Theo vài phương án, vô hiệu hóa chế độ bảng màu cho đơn vị lập mã (630) bao gồm vô hiệu hóa chế độ bảng màu cho cả thành phần luma và thành phần chroma của đơn vị lập mã (640).

Theo vài phương án, vô hiệu hóa chế độ bảng màu cho đơn vị lập mã (630) bao gồm vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã (650).

Theo vài phương án, vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã bao gồm: xác định từ nhiều phần tử cú pháp rằng chỉ ra thêm liệu thành phần viđeo của đơn vị lập mã là thành phần chroma hay thành phần luma; và phù hợp với xác định rằng thành phần viđeo của đơn vị lập mã là thành phần chroma: vô hiệu hóa chế độ bảng màu cho thành phần chroma của đơn vị lập mã.

Theo vài phương án, vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã bao gồm: xác định từ nhiều phần tử cú pháp rằng chỉ ra thêm liệu thành phần viđeo của đơn vị lập mã là thành phần chroma hoặc thành phần luma; phù hợp với xác định rằng thành phần viđeo của đơn vị lập mã là thành phần luma: nhận, từ luồng bit, cờ kích hoạt chế độ bảng màu kết hợp với thành phần luma của đơn vị lập mã; và giải mã đơn vị lập mã phù hợp với cờ kích hoạt chế độ bảng màu.

Theo vài phương án, bộ giải mã viđeo 30, nhận, từ luồng bit, cú pháp kích hoạt dự đoán bảng màu; và cập nhật dự đoán bảng màu cho thành phần luma của đơn vị lập mã theo cú pháp kích hoạt dự đoán bảng màu.

Theo vài phương án, bộ giải mã viđeo 30, vô hiệu hóa sự cập nhật của dự đoán bảng màu cho thành phần luma của đơn vị lập mã.

Theo vài phương án, bộ giải mã viđeo 30, xác định kích thước khối của chế độ bảng màu luma tối thiểu; và phù hợp với việc xác định rằng kích thước của thành phần luma của đơn vị lập mã nhỏ hơn hoặc bằng kích thước khối của chế độ bảng màu luma tối thiểu: vô hiệu hóa sự cập nhật của dự đoán bảng màu cho thành phần luma của đơn vị lập mã.

Theo vài phương án, kích thước khối của chế độ bảng màu luma tối thiểu là 32 x 32 mẫu luma.

Theo vài phương án, kích thước khối của chế độ bảng màu luma tối thiểu là 8 x 8 mẫu luma.

Theo vài phương án, phần tử cú pháp có trong nhiều phần tử cú pháp chỉ ra liệu chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã là MODE_TYPE_INTRA.

Theo vài phương án, phần tử cú pháp có trong nhiều phần tử cú pháp chỉ ra liệu thành phần viđeo của đơn vị lập mã là thành phần chroma hoặc thành phần luma là DUAL_TREE_CHROMA.

Theo vài phương án, ngưỡng định trước trong bước 630 là 32 x 32 mẫu.

Theo vài phương án, ngưỡng định trước trong bước 630 là 16 x 16 mẫu.

FIG. 7 là sơ đồ khối minh họa công cụ lập mã số học nhị phân tương thích theo ngữ cảnh (context-adaptive binary arithmetic coding: CABAC) mẫu mực phù hợp với vài phương án thực hiện của sáng chế.

Lập mã số học nhị phân tương thích theo ngữ cảnh (CABAC) là một dạng của lập mã entropy được sử dụng trong nhiều tiêu chuẩn lập mã viđeo, ví dụ H.264/MPEG-4 AVC, lập mã viđeo hiệu năng cao (High Efficiency Video Coding: HEVC) và VVC. CABAC căn cứ vào lập mã số học, với một số cải tiến và thay đổi để điều chỉnh nó cho phù hợp với nhu cầu của tiêu chuẩn lập mã viđeo. Ví dụ, CABAC lập mã ký hiệu nhị phân, mà giữ cho độ phức tạp thấp và cho phép mô hình hóa xác suất cho các bit được sử dụng thường xuyên hơn của bất kỳ ký hiệu nào. Các mô hình xác suất được lựa chọn một cách tương thích căn cứ vào bối cảnh cục bộ, cho phép mô hình hóa xác suất tốt hơn, bởi vì các chế độ lập mã thường có tương quan cục bộ tốt.

Cuối cùng, CABAC sử dụng phép chia phạm vi không có phép nhân bằng cách sử dụng các phạm vi xác suất được lượng tử hóa và các trạng thái xác suất.

CABAC có nhiều chế độ xác suất cho các ngữ cảnh khác nhau. Trước tiên nó chuyển đổi tất cả các ký hiệu không phải nhị phân sang nhị phân. Sau đó, đối với mỗi bin (hoặc bit có thời hạn), bộ lập mã chọn mô hình xác suất nào sẽ sử dụng, sau đó sử dụng thông tin từ các phần tử lân cận để tối ưu hóa ước tính xác suất. Lập mã số học cuối cùng cũng được áp dụng để nén dữ liệu.

Mô hình ngữ cảnh cung cấp các ước tính về xác suất có điều kiện của các ký hiệu lập mã. Sử dụng các mô hình ngữ cảnh phù hợp, có thể khai thác dự phòng giữa các ký hiệu nhất định bằng cách chuyển đổi giữa các mô hình xác suất khác nhau theo các ký hiệu đã được lập mã trong vùng lân cận của ký hiệu hiện tại để mã hóa. Lập mã một ký hiệu dữ liệu bao gồm các giai đoạn sau.

Nhị phân hóa: CABAC sử dụng lập mã số học nhị phân có nghĩa là chỉ mã hóa các quyết định nhị phân (1 hoặc 0). Một ký hiệu không có giá trị nhị phân (ví dụ hệ số biến đổi hoặc vectơ chuyển động) được "nhị phân hóa" hoặc chuyển đổi thành lập mã nhị phân trước khi lập mã số học. Quy trình này tương tự với quy trình chuyển đổi ký hiệu dữ liệu thành mã chiều dài khác nhau nhưng lập mã nhị phân còn được mã hóa (bởi bộ lập mã số học) trước khi truyền. Các giai đoạn được lặp lại cho mỗi bin (hoặc "bit") của ký hiệu được nhị phân hóa.

Lựa chọn mô hình ngữ cảnh: "mô hình ngữ cảnh" là mô hình xác suất cho một hoặc nhiều bin cảu ký hiệu được nhị phân hóa. Mô hình này có thể được chọn từ sự lựa chọn các mô hình có sẵn phụ thuộc vào thống kê các ký hiệu dữ liệu được lập mã gần đây. Mô hình ngữ cảnh lưu trữ xác suất của mỗi bin là "1" hoặc "0".

Mã hóa số học: Bộ lập mã số học mã hóa mỗi bin theo mô hình xác suất đã lựa chọn. Lưu ý rằng, chỉ có hai phạm vi phụ đối với mỗi bin bin (tương ứng với "0" và "1").

Cập nhật xác suất: Mô hình ngữ cảnh được lựa chọn được cập nhật căn cứ vào giá trị lập mã thực tế (ví dụ, nếu giá trị bin là "1", số lượng tần số của "1" được tăng lên).

Bằng cách phân tích mỗi giá trị phần tử cú pháp không nhị phân thành chuỗi bin, xử lý thêm mỗi giá trị bin trong CABAC phụ thuộc vào quyết định chế độ lập mã kết hợp, mà có thể được chọn làm chế độ thông thường hoặc chế độ bỏ qua. Cái sau được chọn cho các bin, được giả định là phân phối đồng nhất và do đó, toàn bộ quá trình mã hóa số học nhị phân thông thường (và giải mã) chỉ đơn giản bị bỏ qua. Trong chế độ lập mã thông thường, mỗi giá trị bin được mã hóa bằng cách sử dụng công cụ lập mã số học nhị phân thông thường, trong đó mô hình xác suất liên quan được xác định bởi một lựa chọn cố định, căn cứ vào loại phần tử cú pháp và vị trí bin hoặc chỉ mục bin (binIdx) trong sự thể hiện nhị phân hóa của phần tử cú pháp, hoặc được chọn một cách thích ứng từ hai hoặc nhiều mô hình xác suất tùy thuộc vào thông tin bên liên quan (ví dụ, lân cận không gian, thành phần, độ sâu hoặc kích thước của CU/PU/TU, hoặc vị trí bên trong TU). Lựa chọn mô hình xác suất được đề cập đến là mô hình bối cảnh. Là một quyết định thiết kế quan trọng, trường hợp thứ hai thường chỉ được áp dụng cho các bin được quan sát thường xuyên nhất, trong khi bin còn lại, thường ít được quan sát hơn, sẽ được xử lý bằng cách sử dụng khớp, điển hình là mô hình xác suất bậc không. Bằng cách này, CABAC cho phép mô hình hóa xác suất thích ứng có chọn lọc ở cấp độ ký hiệu phụ và do đó, cung cấp một công cụ hiệu quả để khai thác sự dư thừa giữa các ký hiệu với chi phí học hoặc mô hình tổng thể giảm đáng kể. Lưu ý rằng đối với cả trường hợp cố định và trường hợp thích nghi, về nguyên tắc, sự chuyển đổi từ mô hình xác suất này sang mô hình xác suất khác có thể xảy ra giữa hai bin được lập mã thông thường liên tiếp bất kỳ. Nói chung, việc thiết kế các mô hình ngữ cảnh trong CABAC phản ánh mục đích là tìm ra sự thỏa hiệp tốt giữa các mục tiêu xung đột là tránh chi phí mô hình hóa không cần thiết và khai thác các phụ thuộc thống kê mức độ lớn.

Các thông số số của mô hình xác suất trong CABAC là thích ứng, có nghĩa là sự thích ứng của xác suất mô hình với các biến thể thống kê của nguồn bin được thực hiện trên cơ sở từng bin theo kiểu thích ứng ngược và được đồng bộ hóa cả trong bộ mã hóa và bộ giải mã; quá trình này được gọi là ước lượng xác suất. Với mục đích đó, mỗi mô hình xác suất trong CABAC có thể lấy một trong số 126 trạng thái khác nhau

với các giá trị xác suất của mô hình liên quan p khác nhau trong khoảng [0: 01875; 0: 98125]. Hai tham số của mỗi mô hình xác suất được lưu trữ dưới dạng mục nhập 7 bit trong bộ nhớ ngữ cảnh: 6 bit cho mỗi trạng thái xác suất trong số 63 trạng thái xác suất đại diện cho xác suất mô hình pLPS của ký hiệu ít có khả năng xảy ra nhất (LPS) và 1 bit cho nMPS, giá trị của biểu tượng có thể xảy ra nhất (most probable symbol: MPS).

Trong một hoặc nhiều ví dụ, các chức năng được mô tả có thể được triển khai trong phần cứng, phần mềm, chương trình cơ sở hoặc bất kỳ sự kết hợp nào của chúng. Nếu được triển khai trong phần mềm, các chức năng có thể được lưu trữ hoặc truyền qua, dưới dạng một hoặc nhiều lệnh hoặc mã, phương tiện lưu trữ có thể đọc được trên máy tính và được chạy bằng bộ xử lý dựa trên phần cứng. Phương tiện có thể đọc được trên máy tính có thể bao gồm phương tiện lưu trữ có thể đọc được trên máy tính mà tương ứng với phương tiện hữu hình như phương tiện lưu trữ dữ liệu hoặc phương tiện truyền thông bao gồm bất kỳ phương tiện nào tạo điều kiện thuận lợi cho việc chuyển chương trình máy tính từ nơi này sang nơi khác, ví dụ: theo một giao thức truyền thông. Theo cách này, phương tiện máy tính có thể đọc được thường có thể tương ứng với (1) phương tiện lưu trữ hữu hình mà máy tính có thể đọc được, không phải là phương tiện truyền thông nhất thời hoặc (2) phương tiện truyền thông như tín hiệu hoặc sóng mang. Phương tiện lưu trữ dữ liệu có thể là bất kỳ phương tiện sẵn có nào có thể được truy cập bởi một hoặc nhiều máy tính hoặc một hoặc nhiều bộ xử lý để truy xuất lệnh, mã và/hoặc cấu trúc dữ liệu để thực hiện các triển khai được mô tả trong sáng chế hiện tại. Một sản phẩm chương trình máy tính có thể bao gồm một phương tiện máy tính có thể đọc được.

Thuật ngữ được sử dụng trong mô tả các triển khai ở đây chỉ nhằm mục đích mô tả các triển khai cụ thể và không nhằm mục đích giới hạn phạm vi của các yêu cầu bảo hộ. Như được sử dụng trong phần mô tả của việc thực hiện sáng chế và các yêu cầu bảo hộ đi kèm, dạng số ít được dự định bao gồm cả dạng số nhiều, trừ khi ngữ cảnh chỉ rõ trường hợp khác. Cũng sẽ hiểu rằng thuật ngữ "và/hoặc" khi được sử dụng ở đây để chỉ bao gồm bất kỳ và tất cả các kết hợp có thể có của một hoặc nhiều trong số các mục được liệt kê liên quan. Cũng sẽ hiểu rằng các thuật ngữ "bao gồm" và/hoặc

"chứa" khi được sử dụng trong bản mô tả này, chỉ rõ sự có mặt của đặc điểm, chi tiết và/hoặc thành phần được đề cập, nhưng không loại trừ sự hiện diện hoặc bổ sung của một hoặc nhiều đặc điểm, chi tiết, thành phần khác, và/hoặc nhóm của chúng.

Cũng sẽ được hiểu rằng, mặc dù các thuật ngữ thứ nhất, thứ hai, v.v. có thể được sử dụng ở đây để mô tả các yếu tố khác nhau, các yếu tố này không nên bị giới hạn bởi các thuật ngữ này. Các thuật ngữ này chỉ được sử dụng để phân biệt một phần tử này với một phần tử khác. Ví dụ, điện cực thứ nhất có thể được gọi là điện cực thứ hai, và tương tự, điện cực thứ hai có thể được gọi là điện cực thứ nhất, mà không lệch khỏi phạm vi triển khai. Điện cực thứ nhất và điện cực thứ hai đều là điện cực, nhưng chúng không phải là điện cực giống nhau.

Phần mô tả của sáng chế được thể hiện với mục đích minh họa và mô tả, và không nhằm mục đích trình bày đầy đủ hoặc giới hạn đối với sáng chế ở dạng được bộc lộ. Nhiều sửa đổi, biến thể và triển khai thay thế sẽ rõ ràng đối với có hiểu biết trung bình trong lĩnh vực này có ích của những hướng dẫn được thể hiện trong phần mô tả trên và các hình vẽ kết hợp. Phương án được chọn và được mô tả để giải thích tốt nhất quy tắc của sáng chế, ứng dụng thực tiễn và để cho phép người có hiểu biết trung bình trong lĩnh vực này hiểu sáng chế cho những thực hiện khác nhau và sử dụng tốt nhất nguyên tắc cơ bản và các cách triển khai khác nhau với các sửa đổi khác nhau phù hợp với mục đích sử dụng cụ thể đã dự tính. Do đó, hiểu rằng phạm vi yêu cầu bảo hộ không bị giới hạn ở các ví dụ cụ thể của việc thực hiện sáng chế được bộc lộ và rằng những cải tiến và việc thực hiện khác dự định nằm trong phạm vi của yêu cầu bảo hộ đi kèm.

Sáng chế yêu cầu hưởng quyền ưu tiên của đơn sáng chế tạm thời nộp tại Mỹ số 63/001,235, tiêu đề "Lập mã viđeo sử dụng chế độ bảng màu" ngày 27/03/2020, mà được tham khảo kết hợp ở đây.

YÊU CẦU BẢO HỘ

1. Phương pháp giải mã dữ liệu viđeo, bao gồm các bước:

xác định, từ luồng bit, nhiều biến kết hợp với đơn vị lập mã, trong đó nhiều biến chỉ ra loại cây lập mã của đơn vị lập mã, và liệu chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã hay không; và

phù hợp với việc xác định rằng loại cây lập mã của đơn vị lập mã là cây đơn, và chế độ cây kép cục bộ (local dual tree mode) được kích hoạt cho đơn vị lập mã:

vô hiệu hóa chế độ bảng màu (palette mode) cho đơn vị lập mã khi đơn vị lập mã có kích thước bằng hoặc nhỏ hơn ngưỡng định trước.

2. Phương pháp theo điểm 1, trong đó việc vô hiệu hóa chế độ bảng màu cho đơn vị lập mã bao gồm:

vô hiệu hóa chế độ bảng màu cho cả thành phần luma và thành phần chroma của đơn vị lập mã.

3. Phương pháp theo điểm 1, còn bao gồm:

phù hợp với việc xác định rằng loại cây lập mã của đơn vị lập mã là cây đơn, và chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã:

vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã khi đơn vị lập mã có kích thước lớn hơn ngưỡng định trước.

4. Phương pháp theo điểm 3, trong đó việc vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã bao gồm:

xác định từ ít nhất một trong số nhiều biến rằng liệu thành phần viđeo của đơn vị lập mã là thành phần chroma hay thành phần luma; và

phù hợp với việc xác định rằng thành phần viđeo của đơn vị lập mã là thành phần chroma:

vô hiệu hóa chế độ bảng màu cho thành phần chroma của đơn vị lập mã.

5. Phương pháp theo điểm 3, trong đó việc vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã bao gồm:

xác định từ ít nhất một trong số nhiều biến rằng liệu thành phần viđeo của đơn vị lập mã là thành phần chroma hay thành phần luma; và

ít nhất phù hợp với việc xác định rằng thành phần viđeo của đơn vị lập mã là thành phần luma:

nhận, từ luồng bit, cờ kích hoạt chế độ bảng màu kết hợp với thành phần luma của đơn vị lập mã; và

giải mã đơn vị lập mã phù hợp với cờ kích hoạt chế độ bảng màu.

6. Phương pháp theo điểm 5, còn bao gồm:

suy ra, từ luồng bit, cờ cây kép cục bộ; và

cập nhật dự đoán bảng màu cho thành phần luma của đơn vị lập mã đáp ứng cờ cây kép cục bộ là 1.

7. Phương pháp theo điểm 6, còn bao gồm:

vô hiệu hóa việc cập nhật của phép dự đoán bảng màu cho thành phần chroma của đơn vị lập mã đáp ứng cờ cây kép cục bộ là 1.

8. Phương pháp theo điểm 5, còn bao gồm:

xác định ngưỡng kích thước khối luma của chế độ bảng màu; và

phù hợp với xác định rằng kích thước của thành phần luma của đơn vị lập mã nhỏ hơn hoặc bằng ngưỡng kích thước khối luma của chế độ bảng màu:

vô hiệu hóa việc cập nhật của phép dự đoán bảng màu cho thành phần luma của đơn vị lập mã.

9. Phương pháp theo điểm 8, trong đó ngưỡng kích thước khối luma của chế độ bảng màu là 32 x 32 mẫu luma.

10. Phương pháp theo điểm 8, trong đó ngưỡng kích thước khối luma của chế độ bảng màu là 8 x 8 mẫu luma.

11. Phương pháp theo điểm 1, trong đó xác định được liệu chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã căn cứ vào liệu có một biến trong số nhiều biến là MODE_TYPE_INTRA.

12. Phương pháp theo điểm 4, trong đó xác định được liệu thành phần viđeo của đơn vị lập mã là thành phần chroma hoặc thành phần luma căn cứ vào liệu có một biến trong số nhiều biến là DUAL_TREE_CHROMA.

13. Phương pháp theo điểm 1, trong đó ngưỡng định trước là 16.

14. Thiết bị điện tử bao gồm:

một hoặc nhiều bộ xử lý;

bộ nhớ được ghép cặp với một hoặc nhiều bộ xử lý; và

nhiều chương trình được lưu trong bộ nhớ đó, khi được chạy bằng một hoặc nhiều bộ xử lý, làm cho thiết bị điện tử thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 13.

15. Phương tiện lưu trữ không nhất thời có thể đọc được trên máy tính lưu nhiều chương trình để chạy bằng thiết bị điện tử có một hoặc nhiều bộ xử lý, trong đó nhiều chương trình, khi được chạy bằng một hoặc nhiều bộ xử lý, làm cho thiết bị điện tử này thực hiện phương pháp theo điểm bất kỳ trong số các điểm từ 1 đến 13.

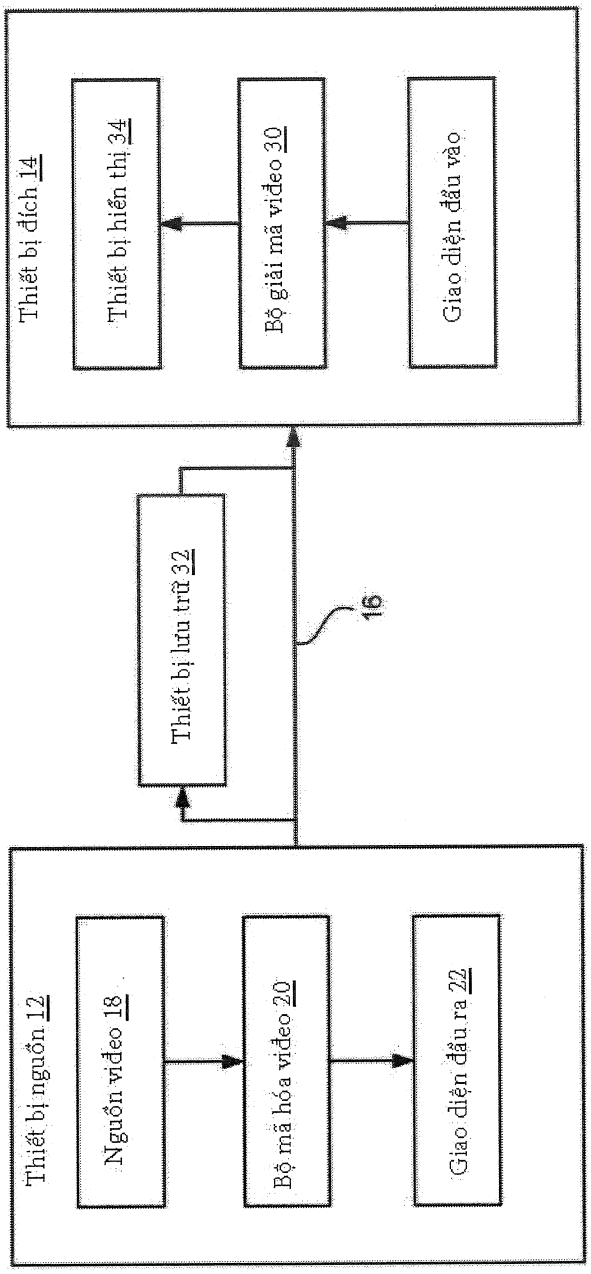

Thiết bị đích 14
Thiết bị hiển thị 34
Bộ giải mã video 30
Giao diện đầu vào
Thiết bị lưu trữ 32
16
Thiết bị nguồn 12
Nguồn video 18
Bộ mã hóa video 20
Giao diện đầu ra 22

FIG. 1

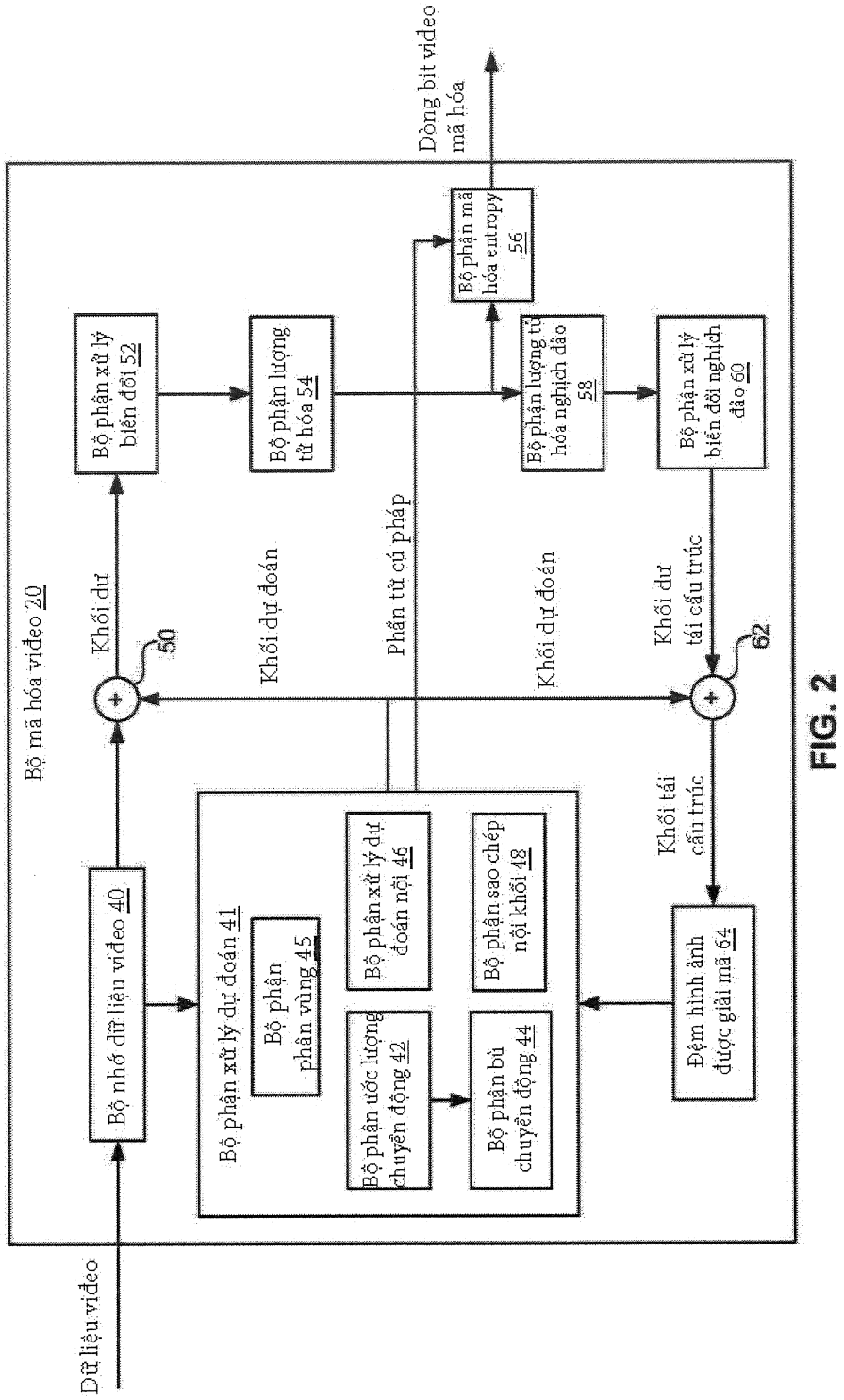

Dữ liệu video
Bộ mã hóa video 20
Bộ nhớ dữ liệu video 40
Bộ phận xử lý dự đoán 41
Bộ phận phân vùng 45
Bộ phận ước lượng chuyển động 42
Bộ phận xử lý dự đoán nội 46
Bộ phận bù chuyển động 44
Bộ phận sao chép nội khối 48
Đệm hình ảnh được giải mã 64
Khối dư
50
Bộ phận xử lý biến đổi 52
Bộ phận lượng tử hóa 54
Bộ phận mã hóa entropy 56
Dòng bit video mã hóa
Khối dự đoán
Phần tử cú pháp
Bộ phận lượng tử hóa nghịch đảo 58
Bộ phận xử lý biến đổi nghịch đảo 60
Khối dư tái cấu trúc
62
Khối tái cấu trúc

FIG. 2

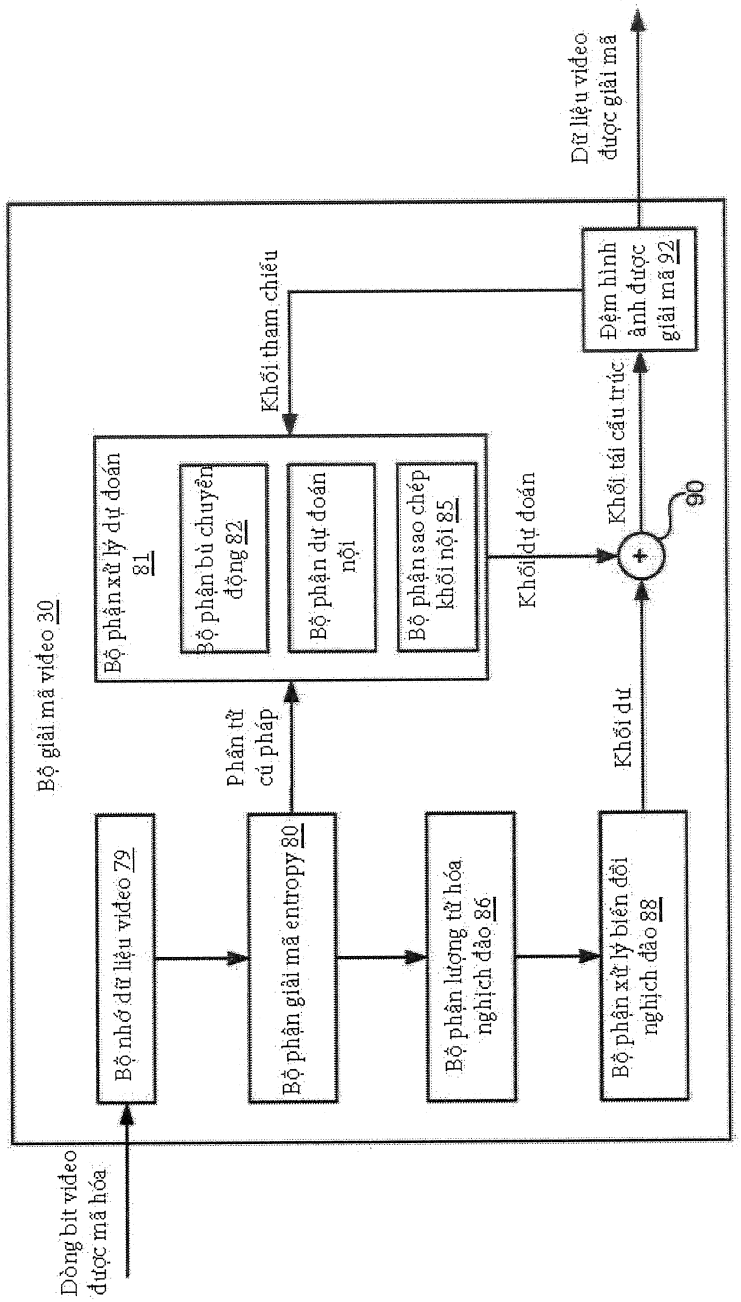

Dòng bit video được mã hóa
Bộ giải mã video 30
Bộ nhớ dữ liệu video 79
Bộ phận giải mã entropy 80
Bộ phận lượng tử hóa nghịch đảo 86
Bộ phận xử lý biến đổi nghịch đảo 88
Phần tử cú pháp
Khối dư
Bộ phận xử lý dự đoán 81
Bộ phận bù chuyển động 82
Bộ phận dự đoán nội
Bộ phận sao chép khối nội 85
Khối dự đoán
Khối tham chiếu
90
Khối tái cấu trúc
Đệm hình ảnh được giải mã 92
Dữ liệu video được giải mã

FIG. 3

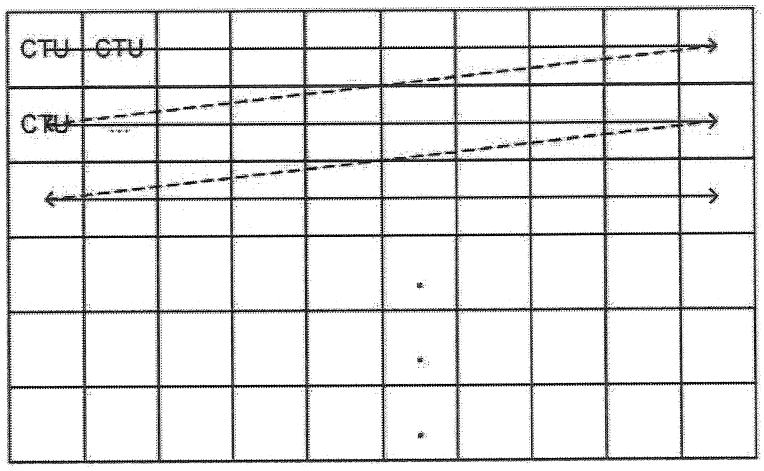

CTU
CTU
CTU

FIG. 4A

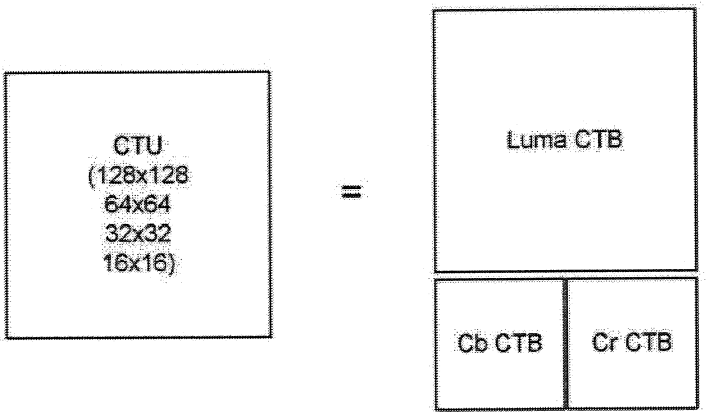

CTU
(128x128
64x64
32x32
16x16)
=
Luma CTB
Cb CTB
Cr CTB

FIG. 4B

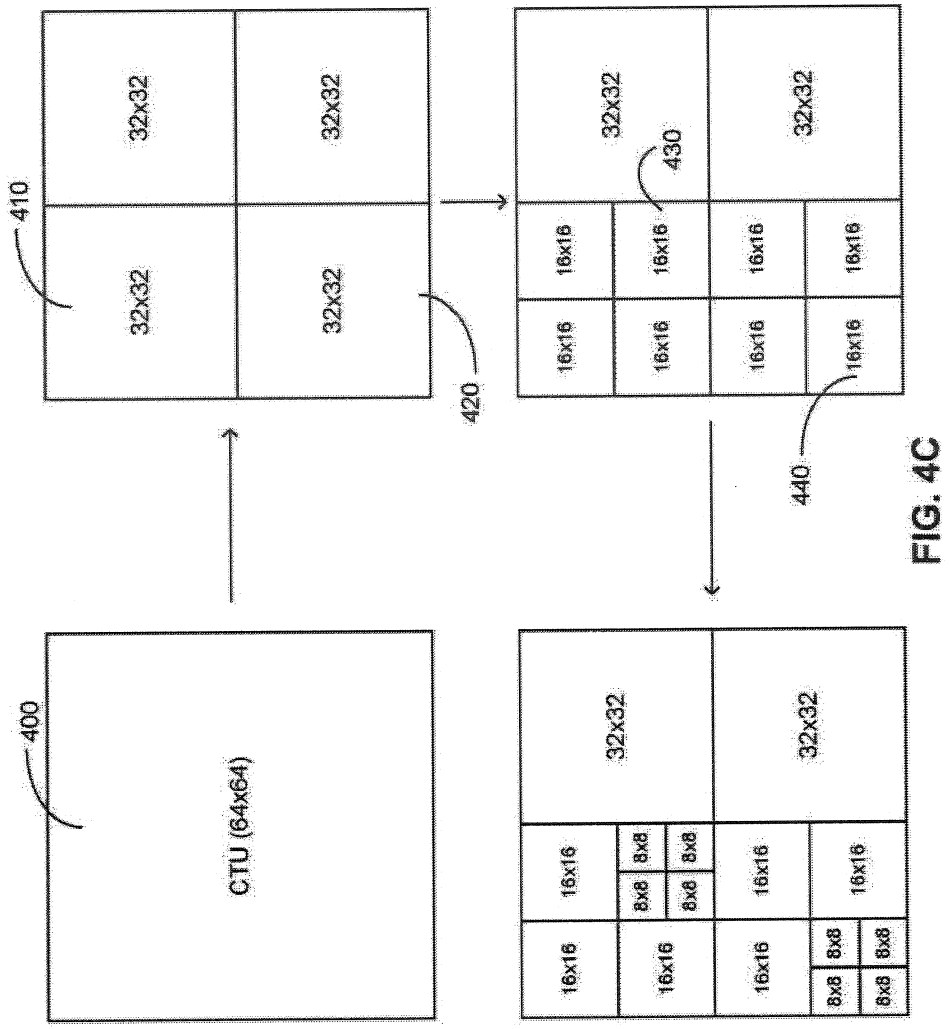

400
CTU (64x64)
410
32x32
32x32
32x32
32x32
420
430
440
16x16
8x8

FIG. 4C

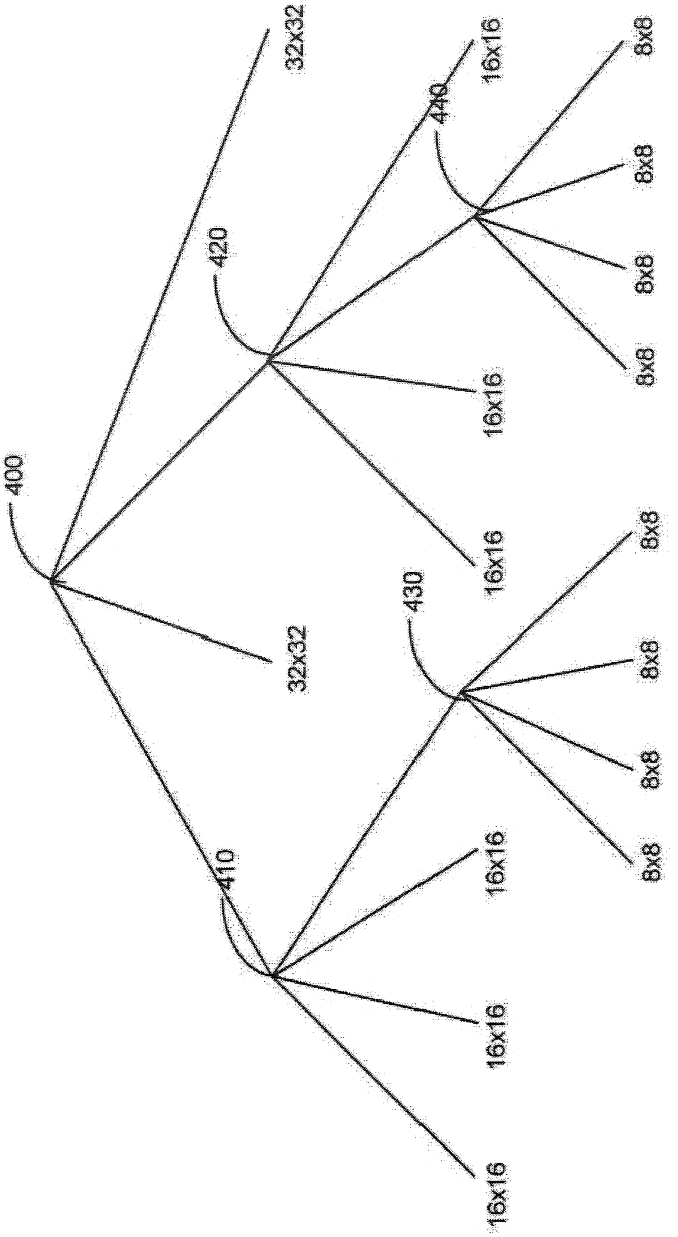

400
32x32
410
16x16
16x16
16x16
420
16x16
8x8
8x8
8x8
8x8
430
8x8
8x8
8x8
8x8
32x32

FIG. 4D

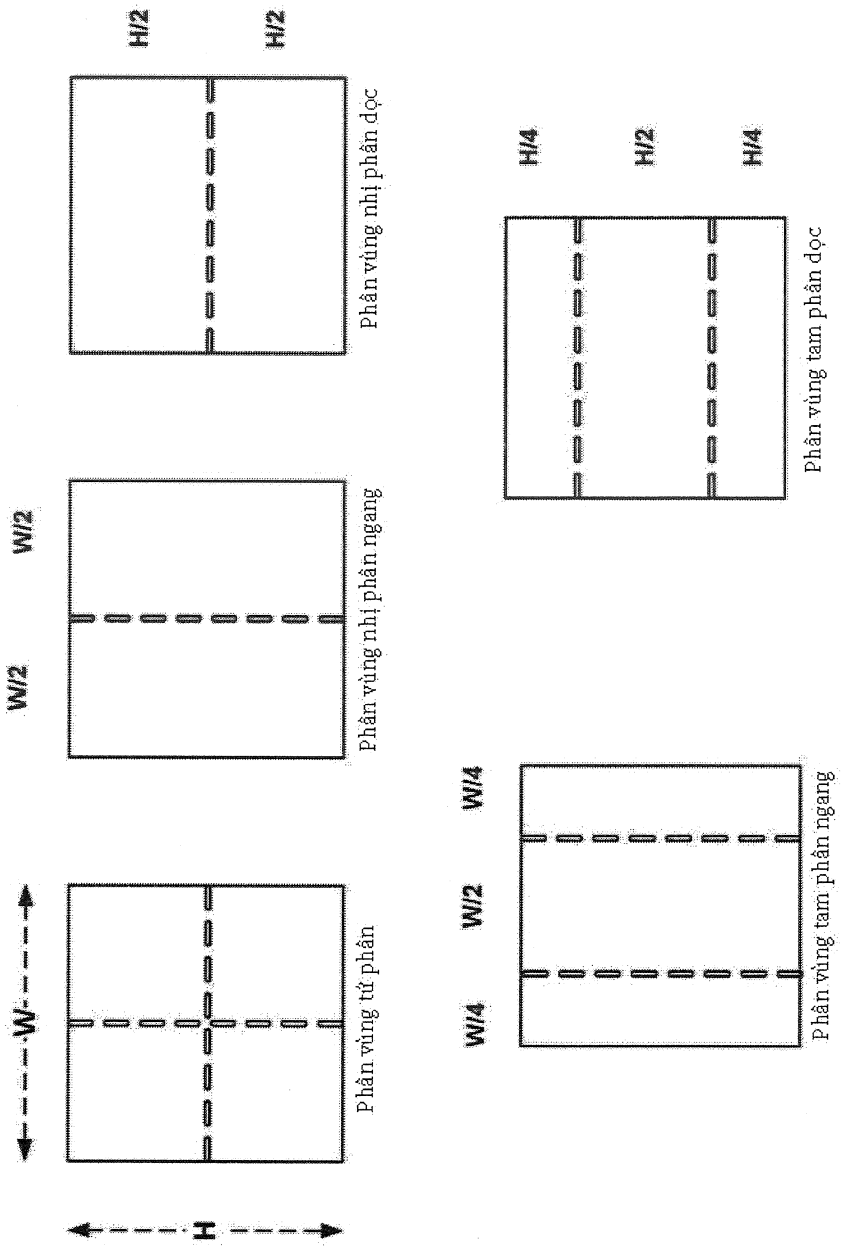

H/2
H/2
Phân vùng nhị phân dọc
W/2
W/2
Phân vùng nhị phân ngang
W
H
Phân vùng tứ phân
H/4
H/2
H/4
Phân vùng tam phân dọc
W/4
W/2
W/4
Phân vùng tam phân ngang

FIG. 4E

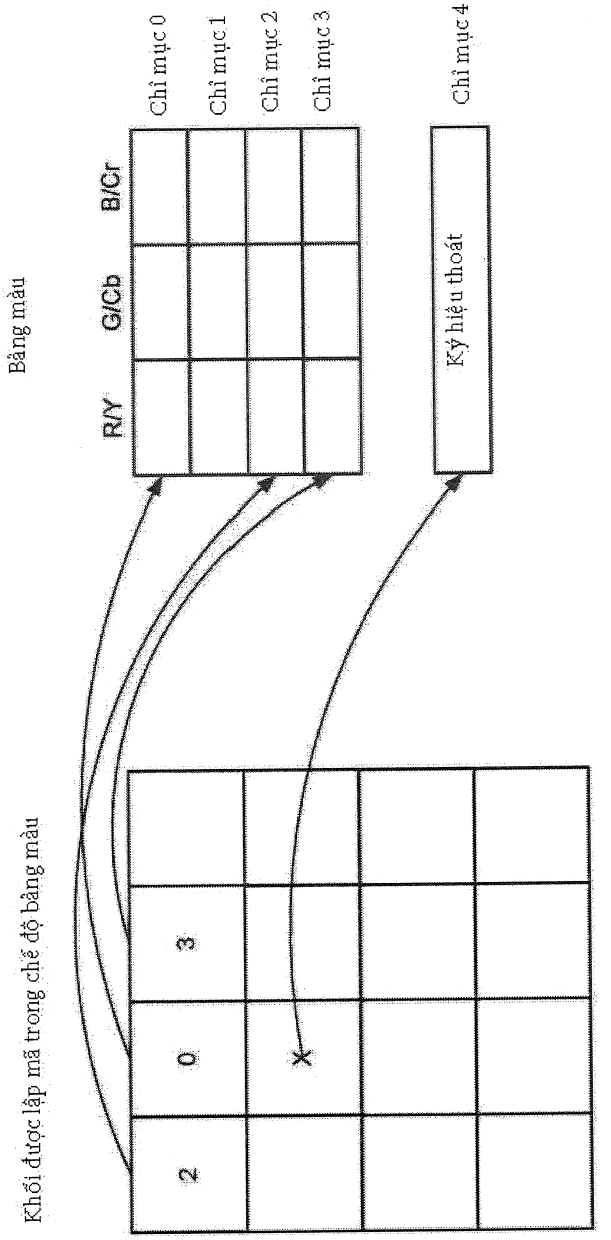
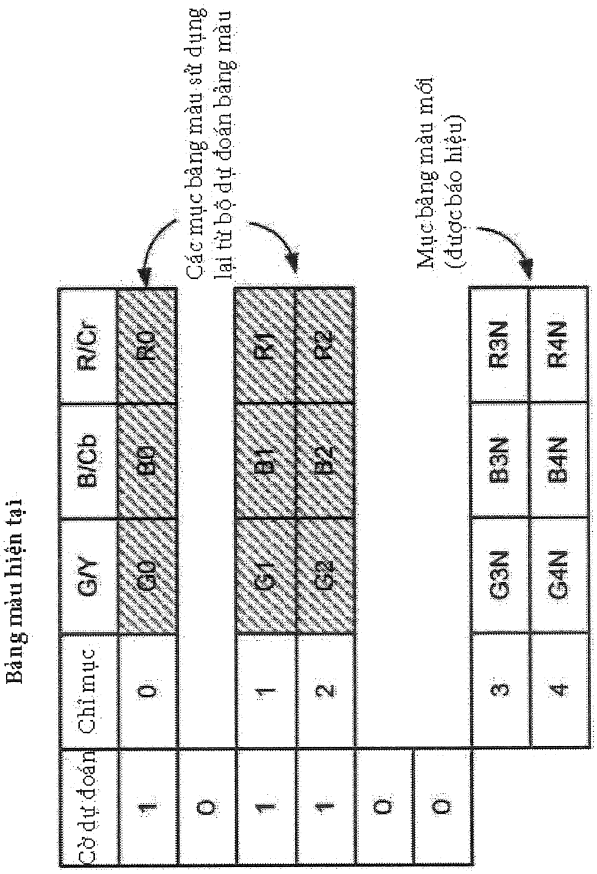

Khối được lập mã trong chế độ bảng màu
2
0
3
Bảng màu
R/Y
G/Cb
B/Cr
Chỉ mục 0
Chỉ mục 1
Chỉ mục 2
Chỉ mục 3
Ký hiệu thoát
Chỉ mục 4

FIG. 5A


Bảng màu hiện tại
Cờ dự đoán
Chỉ mục
G/Y
B/Cb
R/Cr
1
0
G0
B0
R0
0
1
1
G1
B1
R1
1
2
G2
B2
R2
0
0
3
G3N
B3N
R3N
4
G4N
B4N
R4N
Các mục bảng màu sử dụng lại từ bộ dự đoán bảng màu
Mục bảng màu mới (được báo hiệu)

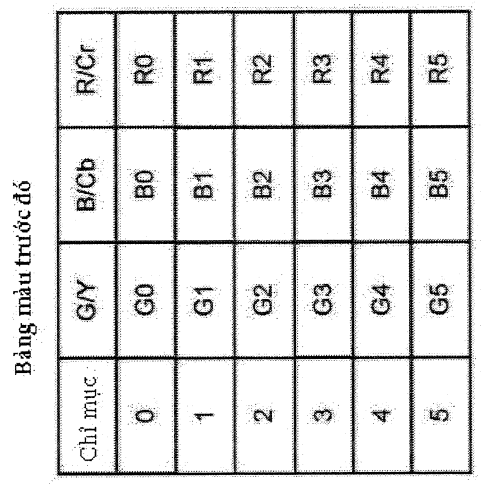
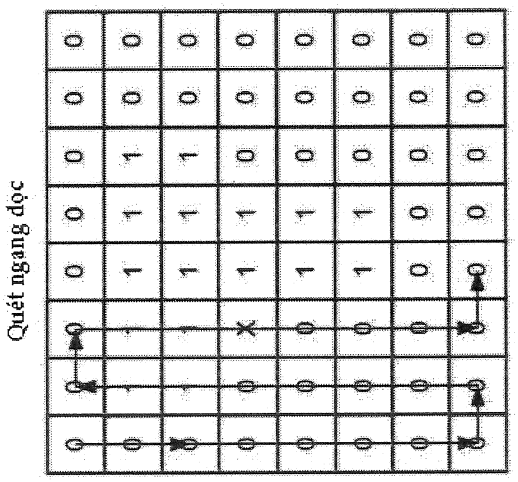

Bảng màu trước đó
Chỉ mục
G/Y
B/Cb
R/Cr
0
G0
B0
R0
1
G1
B1
R1
2
G2
B2
R2
3
G3
B3
R3
4
G4
B4
R4
5
G5
B5
R5

FIG. 5B



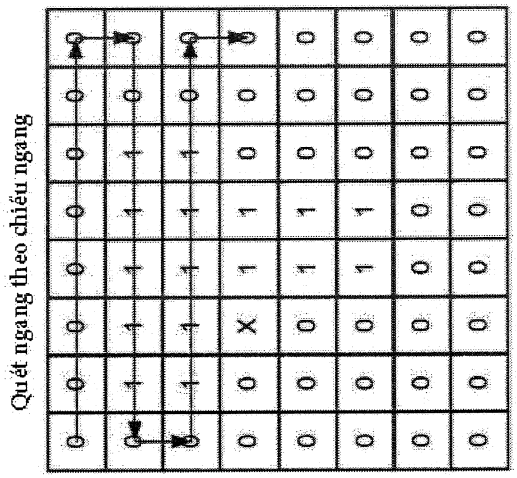


FIG. 5C

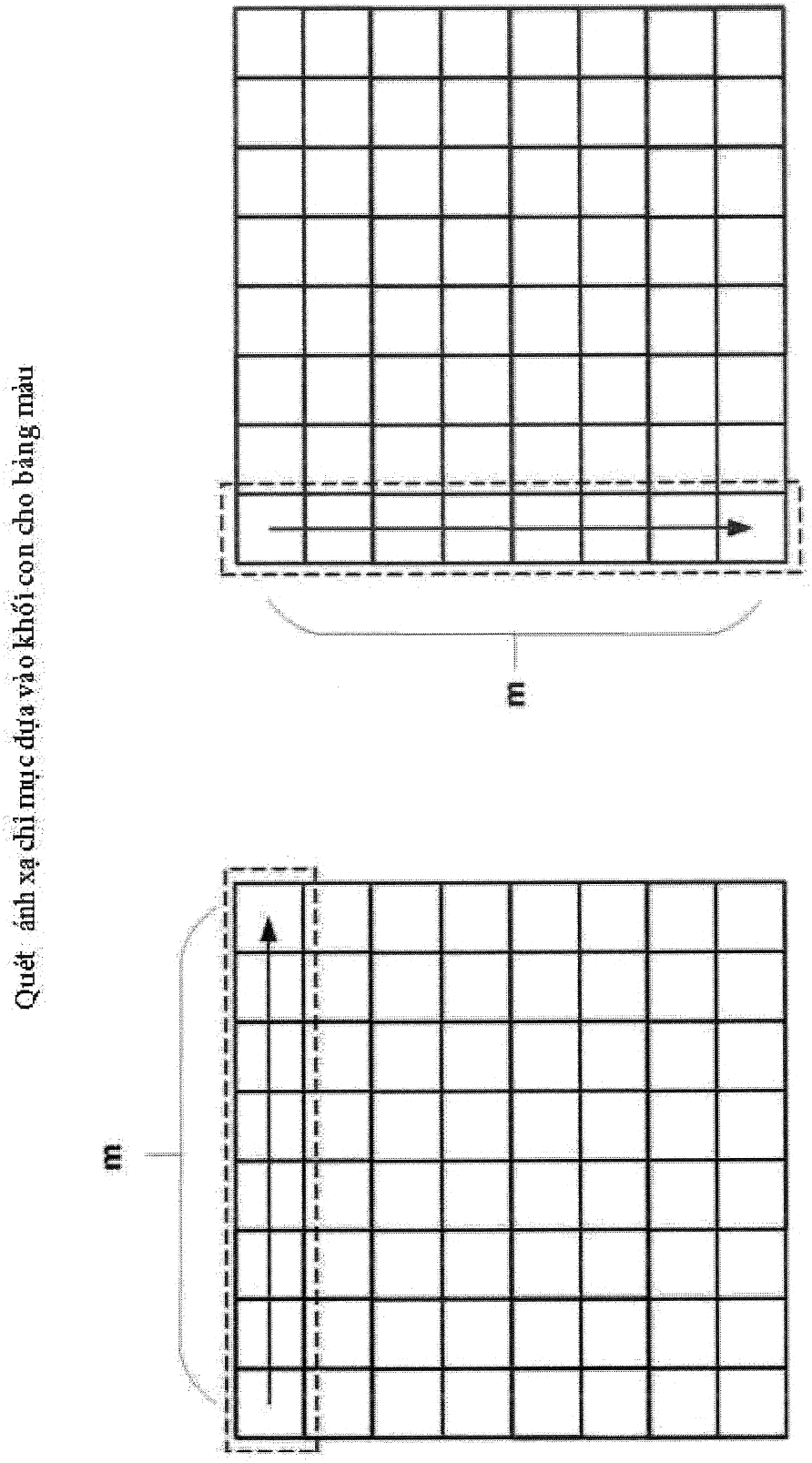


FIG. 5D

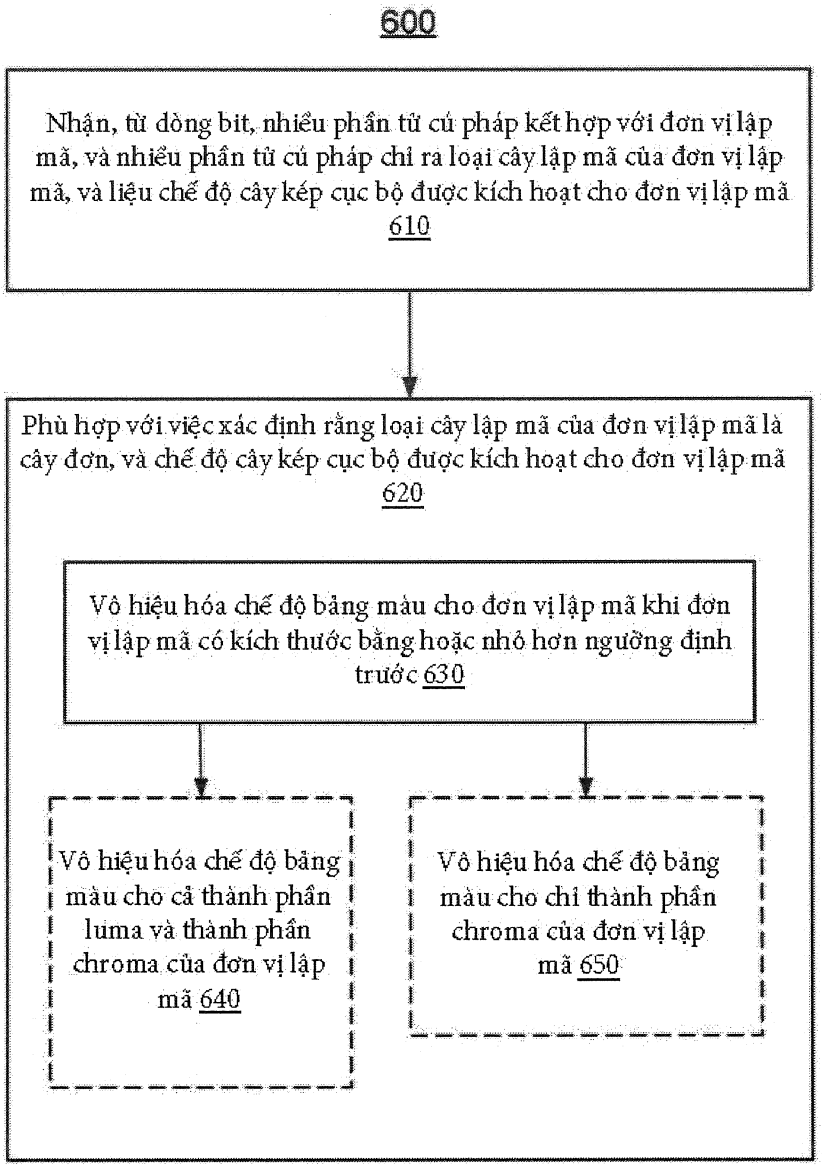

600
Nhận, từ dòng bit, nhiều phần tử cú pháp kết hợp với đơn vị lập mã, và nhiều phần tử cú pháp chỉ ra loại cây lập mã của đơn vị lập mã, và liệu chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã 610
Phù hợp với việc xác định rằng loại cây lập mã của đơn vị lập mã là cây đơn, và chế độ cây kép cục bộ được kích hoạt cho đơn vị lập mã 620
Vô hiệu hóa chế độ bảng màu cho đơn vị lập mã khi đơn vị lập mã có kích thước bằng hoặc nhỏ hơn ngưỡng định trước 630
Vô hiệu hóa chế độ bảng màu cho cả thành phần luma và thành phần chroma của đơn vị lập mã 640
Vô hiệu hóa chế độ bảng màu cho chỉ thành phần chroma của đơn vị lập mã 650

FIG. 6

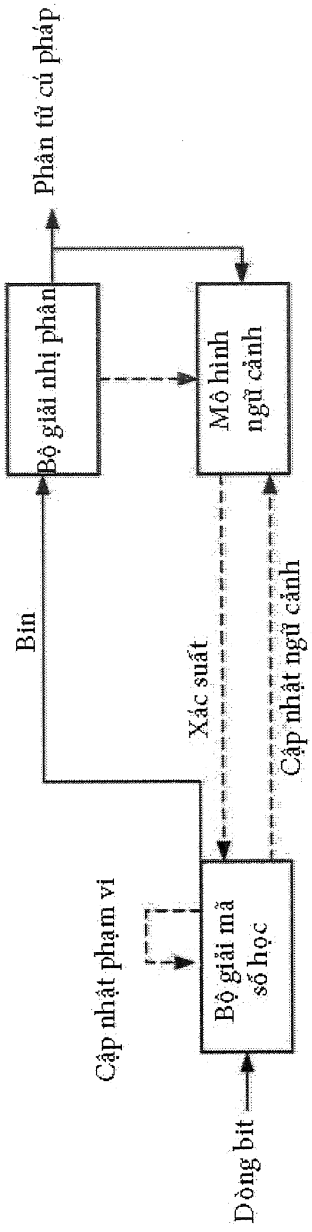

Phân tử cú pháp
Bộ giải nhị phân
Mô hình ngữ cảnh
Bin
Xác suất
Cập nhật ngữ cảnh
Cập nhật phạm vi
Bộ giải mã số học
Dòng bit

FIG. 7