(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) **1-0050302**

(51)[2020.01] **H04N 19/82**; H04N 19/85

(13) **B**

(21) 1-2022-02767

(22) 23/06/2020

(86) PCT/JP2020/024648 23/06/2020

(87) WO 2021/070427 15/04/2021

(30) 62/913,065 09/10/2019 US; 62/950,000 18/12/2019 US

(45) 25/08/2025 449

(43) 25/08/2022 413A

(73) SHARP KABUSHIKI KAISHA (JP)
1, Takumi-cho, Sakai-ku, Sakai City, Osaka 590-8522, Japan

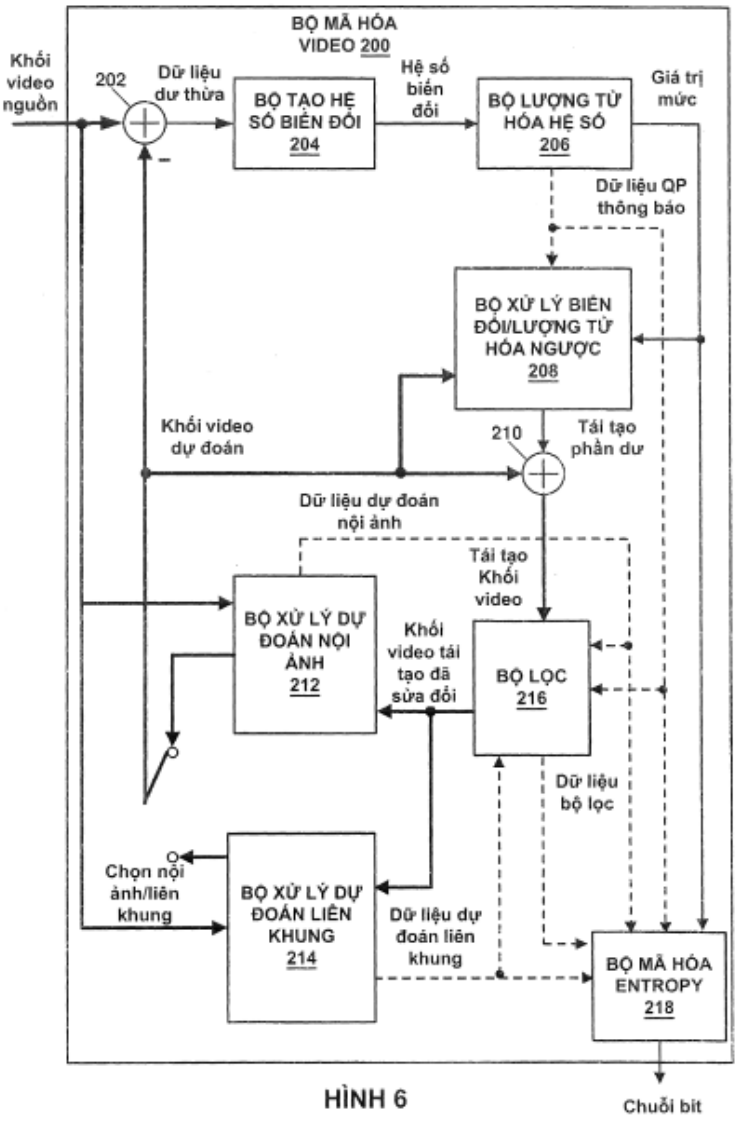
(72) MISRA, Kiran Mukesh (IN); BOSSEN, Frank (NL); SEGALL, Christopher Andrew (US); DESHPANDE, Sachin G. (US).

(74) Công ty TNHH một thành viên Sở hữu trí tuệ VCCI (VCCI-IP CO.,LTD)

(54) PHƯƠNG PHÁP LỌC DỮ LIỆU VIDEO ĐƯỢC TÁI TẠO VÀ THIẾT BỊ GIẢI MÃ DỮ LIỆU VIDEO

(21) 1-2022-02767

(57) Công bố phương pháp lọc dữ liệu video được tái tạo. Phương pháp này bao gồm: bước phân tách phần tử cú pháp thứ nhất được sử dụng để thiết lập hệ số bộ lọc thành phần chéo; bước nhập mảng mẫu của hình ảnh luma đã tái tạo; bước suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại; bước suy ra mảng hệ số bộ lọc bằng cách sử dụng hệ số bộ lọc thành phần chéo; bước suy ra biến bằng cách sử dụng mảng hệ số bộ lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma; và bước suy ra biến tỷ lệ bằng cách sử dụng biến, trong đó biến được sửa đổi bằng cách tính tổng của một mẫu khối sắc độ hiện tại, được xác định bằng vị trí đã xác định trước, và biến tỷ lệ.



HÌNH 6

Lĩnh vực kỹ thuật được đề cập

Sáng chế này liên quan đến kỹ thuật mã hóa video và cụ thể hơn là liên quan đến kỹ thuật giảm lỗi tái tạo.

Tình trạng kỹ thuật của sáng chế

Có thể tích hợp các chức năng video kỹ thuật số vào nhiều loại thiết bị, bao gồm tivi kỹ thuật số, máy tính xách tay hoặc máy tính để bàn, máy tính bảng, thiết bị ghi âm kỹ thuật số, thiết bị nghe nhìn kỹ thuật số, thiết bị chơi game, điện thoại di động, điện thoại thông minh, thiết bị chụp hình y tế và các thiết bị tương tự. Có thể mã hóa video kỹ thuật số theo chuẩn mã hóa video. Chuẩn mã hóa video xác định định dạng của chuỗi bit theo chuẩn bao gồm dữ liệu video được mã hóa. Chuỗi bit theo chuẩn là cấu trúc dữ liệu có thể được thiết bị giải mã video thu và giải mã để tạo dữ liệu video đã được khôi phục. Các chuẩn mã hóa video có thể kết hợp các kỹ thuật nén video. Ví dụ về các chuẩn mã hóa video bao gồm ISO/IEC MPEG-4 Visual và ITU-T H.264 (còn gọi là ISO/IEC MPEG-4 AVC) và Mã hóa video hiệu quả cao (HEVC). HEVC được mô tả trong phần Mã hóa video hiệu quả cao (HEVC), Tài liệu về chuẩn ITU-T H.265, tháng 12 năm 2016, được kết hợp vào tài liệu này bằng viện dẫn và được gọi là ITU-T H.265 trong tài liệu này. Các phần mở rộng và cải tiến của ITU-T H.265 hiện đang được xem xét để phát triển các chuẩn mã hóa video thế hệ tiếp theo. Ví dụ: Nhóm chuyên gia mã hóa video ITU-T (VCEG) và ISO/IEC (Nhóm chuyên gia hình ảnh chuyển động (MPEG) (gọi chung là Nhóm hợp tác chung về thiết kế chuẩn nén video (JVET)) đang làm việc để phát triển công nghệ mã hóa video được chuẩn hóa với khả năng nén vượt trội đáng kể so với khả năng nén của chuẩn HEVC hiện tại. Mô hình khai thác chung 7 (JEM 7), Mô tả thuật toán của mô hình thử nghiệm khai thác chung 7 (JEM 7), ISO/IEC JTC1/SC29/WG11 Tài liệu: JVET-G1001, tháng 7 năm 2017, Torino, IT, được kết hợp trong tài liệu này bằng viện dẫn, mô tả các tính năng mã hóa trong nghiên cứu mô hình thử nghiệm phối hợp của JVET là có khả năng nâng cao công nghệ mã hóa video vượt khả năng của chuẩn ITU-T H.265. Xin lưu ý rằng các tính năng mã hóa của JEM 7 được triển khai trong phần mềm tham chiếu JEM. Như được sử dụng trong tài liệu này, thuật ngữ JEM có thể đề cập chung đến các thuật toán có trong JEM 7 và việc triển khai phần mềm tham chiếu JEM. Hơn nữa, để đáp lại "Lời kêu gọi đề xuất ý tưởng nén video với khả năng vượt chuẩn HEVC" do VCEG và MPEG cùng đưa ra, có nhiều

nhóm đã đề xuất nhiều mô tả về công cụ mã hóa video tại Hội nghị lần thứ 10 về ISO/IEC JTC1/SC29/WG11 diễn ra từ ngày 16-20 tháng 4 năm 2018, tại San Diego, CA. Từ nhiều mô tả về các công cụ mã hóa video, văn bản nháp ban đầu hình thành về thông số kỹ thuật mã hóa video được mô tả trong "Mã hóa video đa năng (Bản nháp 1)," Hội nghị lần thứ 10 về ISO/IEC JTC1/SC29/WG11 diễn ra từ ngày ngày 16-20 tháng 4 năm 2018, tại San Diego, CA, tài liệu JVET-J1001-v2, được kết hợp vào tài liệu này bằng viện dẫn và được gọi là JVET-J1001. Sự phát triển chuẩn mã hóa video thế hệ kế tiếp của VCEG và MPEG hiện tại được gọi là dự án Mã hóa video đa năng (VVC). "Mã hóa video đa năng (Bản dự thảo 5)," Hội nghị lần thứ 14 về ISO/IEC JTC1/SC29/WG11 diễn ra từ ngày 19 đến 27 tháng 3 năm 2019 tại Geneva, CH, tài liệu JVET-N1001-v8, được kết hợp vào tài liệu này bằng viện dẫn và được gọi là JVET-N1001, biểu thị sự lặp lại văn bản dự thảo về thông số kỹ thuật mã hóa video tương ứng với dự án VVC. "Mã hóa video đa năng (Bản dự thảo 6)," Hội nghị lần thứ 15 về ISO/IEC JTC1/SC29/WG11 diễn ra từ ngày 3 đến 12 tháng 7 năm 2019 tại Gothenburg, SE, tài liệu JVET-O2001-vE, được kết hợp vào tài liệu này bằng viện dẫn, và được gọi là JVET-O2001, sự lặp lại của văn bản dự thảo về thông số kỹ thuật mã hóa video tương ứng với dự án VVC.

Các kỹ thuật nén video cho phép giảm yêu cầu lưu trữ và truyền dữ liệu video. Các kỹ thuật nén video có thể giúp giảm yêu cầu về dữ liệu bằng cách tận dụng các phần dư thừa vốn có trong một chuỗi video. Các kỹ thuật nén video có thể chia nhỏ một chuỗi video thành các phần nhỏ hơn liên tiếp (nghĩa là nhóm hình ảnh trong một chuỗi video, một hình ảnh trong nhóm các hình ảnh, các vùng trong một hình ảnh, vùng phụ trong một vùng). Các kỹ thuật mã hóa dự đoán nội ảnh (ví dụ: kỹ thuật dự đoán theo không gian trong hình ảnh) và kỹ thuật dự đoán liên khung (tức là liên ảnh (theo thời gian)) có thể được dùng để tạo ra các giá trị chênh lệch giữa đơn vị dữ liệu video được mã hóa và đơn vị dữ liệu video tham chiếu. Các giá trị chênh lệch có thể được coi là dữ liệu dư thừa. Dữ liệu dư thừa có thể được mã hóa dưới dạng hệ số biến đổi lượng tử hóa. Các phần tử cú pháp có thể liên quan đến dữ liệu dư thừa và đơn vị mã hóa tham chiếu (ví dụ: chỉ số chế độ dự đoán nội bộ và thông tin chuyển động). Dữ liệu dư thừa và các phần tử cú pháp có thể được mã hóa entropy. Dữ liệu dư thừa được mã hóa entropy và các phần tử cú pháp có thể được bao gồm trong cấu trúc dữ liệu tạo thành chuỗi bit theo chuẩn.

Bản chất kỹ thuật của sáng chế

Theo một ví dụ, phương pháp lọc dữ liệu video được tái tạo, phương pháp này bao gồm: bước phân tích cú pháp phần tử cú pháp đầu tiên được sử dụng để thiết lập hệ số

lọc thành phần chéo; bước nhập mảng mẫu của hình ảnh luma đã tái tạo; bước suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại; bước suy ra mảng hệ số lọc bằng cách sử dụng hệ số lọc thành phần chéo; bước suy ra biến bằng cách sử dụng mảng hệ số lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma; và bước suy ra biến tỷ lệ bằng cách sử dụng biến, trong đó biến được sửa đổi bằng cách tính tổng của một mẫu khối sắc độ hiện tại, được xác định bằng vị trí đã xác định trước, và biến tỷ lệ.

Theo một ví dụ, thiết bị mã hóa dữ liệu video, thiết bị này bao gồm một hoặc nhiều bộ xử lý được tạo cấu hình để: mã hóa phần tử cú pháp đầu tiên được sử dụng để thiết lập hệ số lọc thành phần chéo; nhập mảng mẫu của hình ảnh luma đã tái tạo; suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại; suy ra mảng hệ số lọc bằng cách sử dụng hệ số lọc thành phần chéo; suy ra biến bằng cách sử dụng mảng hệ số lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma; và suy ra biến tỷ lệ bằng cách sử dụng biến đó, trong đó biến được sửa đổi bằng cách tính tổng của một mẫu khối sắc độ hiện tại, được xác định bằng vị trí đã xác định trước, và biến tỷ lệ.

Theo một ví dụ, thiết bị giải mã dữ liệu video, thiết bị này bao gồm một hoặc nhiều bộ xử lý được tạo cấu hình để: giải mã phần tử cú pháp đầu tiên được sử dụng để thiết lập hệ số lọc thành phần chéo; nhập mảng mẫu của hình ảnh luma đã tái tạo; suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại; suy ra mảng hệ số lọc bằng cách sử dụng hệ số lọc thành phần chéo; suy ra biến bằng cách sử dụng mảng hệ số lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma; và suy ra biến tỷ lệ bằng cách sử dụng biến đó, trong đó biến được sửa đổi bằng cách tính tổng của một mẫu khối sắc độ hiện tại, được xác định bằng vị trí đã xác định trước, và biến tỷ lệ.

Mô tả vắn tắt các hình vẽ

HÌNH 1 là sơ đồ khái niệm minh họa ví dụ về một nhóm ảnh được mã hóa theo cách phân vùng đa cây tứ phân theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 2A là sơ đồ khái niệm minh họa ví dụ về việc mã hóa khối dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 2B là sơ đồ khái niệm minh họa ví dụ về việc mã hóa khối dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 3 là sơ đồ khái niệm minh họa ví dụ về định dạng lấy mẫu thành phần video mà có thể được sử dụng theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 4A là sơ đồ khái niệm minh họa ví dụ về các loại vị trí cho định dạng lấy mẫu thành phần video mà có thể được sử dụng theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 4B là sơ đồ khái niệm minh họa ví dụ về các loại vị trí cho định dạng lấy mẫu thành phần video mà có thể được sử dụng theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 4C là sơ đồ khái niệm minh họa ví dụ về các loại vị trí cho định dạng lấy mẫu thành phần video mà có thể được sử dụng theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 4D là sơ đồ khái niệm minh họa ví dụ về các loại vị trí cho định dạng lấy mẫu thành phần video mà có thể được sử dụng theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 4E là sơ đồ khái niệm minh họa ví dụ về các loại vị trí cho định dạng lấy mẫu thành phần video mà có thể được sử dụng theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 4F là sơ đồ khái niệm minh họa ví dụ về các loại vị trí cho định dạng lấy mẫu thành phần video mà có thể được sử dụng theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 5 là sơ đồ khối minh họa ví dụ về hệ thống mà có thể được tạo cấu hình để mã hóa và giải mã dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 6 là sơ đồ khối minh họa ví dụ về bộ mã hóa video mà có thể được tạo cấu hình để mã hóa dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 7 là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để mã hóa dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 8 là sơ đồ khái niệm minh họa ví dụ về lỗi tái tạo nhiều thành phần của dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 9A là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 9B là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 9C là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 9D là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 9E là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 9F là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 10 là sơ đồ khái niệm minh họa ví dụ về việc giảm lỗi tái tạo bằng cách lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 11A là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 11B là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 11C là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 11D là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 12A là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 12B là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 13A là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 13B là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 13C là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 14A là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 14B là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 14C là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 14D là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 14E là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 14F là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 15A là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 15B là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 15C là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 15D là sơ đồ khái niệm minh họa ví dụ về các vị trí hệ số lọc mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 16A là sơ đồ khái niệm minh họa ví dụ về bộ đệm dòng ảo mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 16B là sơ đồ khái niệm minh họa ví dụ về bộ đệm dòng ảo mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 16C là sơ đồ khái niệm minh họa ví dụ về bộ đệm dòng ảo mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 16D là sơ đồ khái niệm minh họa ví dụ về bộ đệm dòng ảo mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 17 là sơ đồ khối minh họa ví dụ về bộ giải mã video mà có thể được tạo cấu hình để giải mã dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 18 là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để mã hóa dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 19A là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 19B là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 19C là sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 20A là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 20B là sơ đồ khái niệm minh họa ví dụ về các mẫu giá đỡ mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này.

Mô tả chi tiết các phương án thực hiện sáng chế

Đơn xin cấp bằng sáng chế này liên quan đến Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/865,933, nộp ngày 24 tháng 6 năm 2019; Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/870,752, nộp ngày 4 tháng 7 năm 2019; Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/886,891, nộp ngày 14 tháng 8 năm 2019; Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/899,053, nộp ngày 11 tháng 9 năm 2019; Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/901,679, nộp ngày 17 tháng 9 năm 2019; Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/904,399, nộp ngày 23 tháng 9 năm 2019; Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/905,312, nộp ngày 24 tháng 9 năm 2019; Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/910,317, nộp ngày 3 tháng 10 năm 2019; và Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/913,065, nộp ngày 9 tháng 10 năm 2019, mỗi đơn đều được kết hợp bằng viện dẫn.

Nói chung, sáng chế này mô tả nhiều kỹ thuật được dùng để mã hóa dữ liệu video. Đặc biệt, sáng chế này mô tả các kỹ thuật để giảm lỗi tái tạo. Lưu ý rằng mặc dù các kỹ thuật của sáng chế này được mô tả liên quan đến ITU-T H.264, ITU-T H.265, JEM, JVET-N1001, và JVET-O2001, nhưng các kỹ thuật của sáng chế này cũng thường được áp dụng để mã hóa video. Ví dụ: các kỹ thuật mã hóa được mô tả trong tài liệu này có thể được kết hợp vào hệ thống mã hóa video, (bao gồm hệ thống mã hóa video dựa trên các chuẩn mã hóa video trong

tương lai) bao gồm cấu trúc khối video, kỹ thuật dự đoán nội ảnh, kỹ thuật dự đoán liên khung, kỹ thuật biến đổi, kỹ thuật lọc và/hoặc kỹ thuật mã hóa entropy ngoài các kỹ thuật khác có trong ITU-T H.265, JEM, JVET-N1001 và JVET-O2001. Do đó, việc tham chiếu đến ITU-T H.264, ITU-T H.265, JEM, JVET-N1001 và/hoặc JVET-O2001 là nhằm mục đích mô tả và không được hiểu là giới hạn phạm vi của các kỹ thuật được mô tả trong tài liệu này. Hơn nữa, xin lưu ý rằng việc kết hợp bằng cách tham chiếu các tài liệu ở đây nhằm mục đích mô tả và không được hiểu là hạn chế hoặc tạo sự mơ hồ đối với các thuật ngữ được dùng trong tài liệu này. Ví dụ: trong trường hợp một tham chiếu kết hợp cung cấp một định nghĩa khác về thuật ngữ với một tham chiếu kết hợp khác và/hoặc khi thuật ngữ được dùng trong tài liệu này, thì thuật ngữ này phải được giải thích theo cách bao hàm một cách rộng rãi từng định nghĩa tương ứng và/hoặc theo cách bao hàm từng định nghĩa cụ thể trong phương án thay thế.

Theo một ví dụ, phương pháp bao gồm bước dữ liệu mẫu được tái tạo cho thành phần dữ liệu video hiện tại, bước thu dữ liệu mẫu được tái tạo cho một hoặc nhiều thành phần dữ liệu video khác, bước dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác, và bước áp dụng bộ lọc cho dữ liệu mẫu đã tái tạo cho thành phần dữ liệu video hiện tại dựa trên bộ lọc thành phần chéo đã dẫn xuất và dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác.

Theo một ví dụ, thiết bị bao gồm một hoặc nhiều bộ xử lý được tạo cấu hình để thu dữ liệu mẫu đã tái tạo cho thành phần dữ liệu video hiện tại, thu dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác, dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác, và áp dụng bộ lọc cho dữ liệu mẫu đã tái tạo đối với thành phần dữ liệu video hiện tại dựa trên bộ lọc thành phần chéo đã dẫn xuất và dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác.

Theo một ví dụ, môi trường lưu trữ máy tính có thể đọc được không chuyển tiếp bao gồm các lệnh được lưu trữ trên đó, khi được thực thi sẽ khiến một hoặc nhiều bộ xử lý của thiết bị thu dữ liệu mẫu đã tái tạo cho thành phần dữ liệu video hiện tại, thu dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác, dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác, và áp dụng bộ lọc cho dữ liệu mẫu đã tái tạo đối với thành phần dữ liệu video hiện tại dựa trên bộ lọc thành phần chéo đã dẫn xuất và dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác.

Theo một ví dụ, thiết bị bao gồm phương tiện để thu dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác, phương tiện để dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác, và phương tiện để áp dụng bộ lọc cho dữ liệu mẫu đã tái tạo đối với thành phần dữ liệu video hiện tại dựa trên bộ lọc thành phần chéo đã dẫn xuất và dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác.

Chi tiết của một hoặc nhiều ví dụ được trình bày trong các hình vẽ kèm theo và mô tả bên dưới. Các tính năng, đối tượng và lợi thế khác sẽ trở nên rõ ràng từ phần mô tả và các hình vẽ, cũng như từ các yêu cầu bảo hộ.

Nội dung video bao gồm các chuỗi video gồm một loạt khung hình (hoặc hình ảnh). Một loạt khung hình cũng có thể được gọi là một nhóm ảnh (GOP). Mỗi khung hình video hoặc hình ảnh có thể được chia thành một hoặc nhiều vùng. Các vùng có thể được xác định theo đơn vị cơ sở (ví dụ: khối video) và các bộ quy tắc xác định vùng. Ví dụ: quy tắc xác định vùng có thể là vùng phải là số nguyên của khối video được bố trí trong hình chữ nhật. Hơn nữa, các khối video trong vùng có thể được sắp xếp theo mẫu quét (ví dụ: quét raster). Thuật ngữ khối video nói chung, như được sử dụng trong tài liệu này, có thể đề cập đến một vùng của hình ảnh hoặc có thể đề cập cụ thể hơn đến mảng giá trị mẫu lớn nhất có thể được mã hóa theo dự đoán, các phân vùng của chúng và/hoặc cấu trúc tương ứng. Hơn nữa, thuật ngữ khối video hiện tại có thể đề cập đến một vùng của hình ảnh đang được mã hóa hoặc giải mã. Khối video có thể được định nghĩa là một mảng các giá trị mẫu. Xin lưu ý rằng trong một số trường hợp, giá trị pixel có thể được mô tả bao gồm các giá trị mẫu cho các thành phần tương ứng của dữ liệu video, cũng có thể được gọi là thành phần màu, (ví dụ: thành phần luma (Y) và chroma (Cb và Cr) hoặc các thành phần màu đỏ, xanh lục và xanh lam). Xin lưu ý rằng trong một số trường hợp, các thuật ngữ giá trị pixel và giá trị mẫu sẽ được dùng để thay thế cho nhau. Hơn nữa, trong một số trường hợp, pixel hoặc mẫu có thể được gọi là pel. Định dạng lấy mẫu video, cũng có thể được gọi là định dạng sắc độ, có thể xác định số lượng mẫu sắc độ có trong khối video so với số lượng mẫu luma có trong khối video. Ví dụ: đối với định dạng lấy mẫu 4:2:0, tốc độ lấy mẫu đối với thành phần luma gấp đôi tốc độ lấy mẫu của các thành phần sắc độ theo cả hướng ngang và hướng dọc.

Bộ mã hóa video có thể thực hiện mã hóa theo dự đoán trên các khối video và các phân vùng của chúng. Khối video và các phân vùng của chúng có thể được gọi là các nút. ITU-T H.264 chỉ định khối macro bao gồm các mẫu luma có kích thước 16x16. Có nghĩa là, trong

ITU-T H.264, hình ảnh được phân đoạn thành các khối macro. ITU-T H.265 chỉ định cấu trúc Đơn vị cây mã hóa (CTU) tương tự (còn được gọi là đơn vị mã hóa lớn nhất (LCU)). Trong ITU-T H.265, hình ảnh được phân đoạn thành các CTU. Trong ITU-T H.265, đối với hình ảnh, có thể thiết lập kích thước CTU bao gồm các mẫu luma 16x16, 32x32 hoặc 64x64. Trong ITU-T H.265, CTU bao gồm các Khối cây mã hóa (CTB) tương ứng cho từng thành phần của dữ liệu video (ví dụ: luma (Y) và chroma (Cb và Cr). Cần lưu ý rằng video có một thành phần luma và hai thành phần sắc độ tương ứng có thể được mô tả là có hai kênh, nghĩa là kênh luma và kênh sắc độ. Hơn nữa, trong ITU-T H.265, CTU có thể được phân vùng theo cấu trúc phân vùng cây tứ phân (QT), do đó, các CTB của CTU được phân vùng thành các Khối mã hóa (CB). Nghĩa là, trong ITU-T H.265, CTU có thể được phân chia thành các nút lá cây tứ phân. Theo ITU-T H.265, một CB luma cùng với hai CB sắc độ tương ứng và các phần tử cú pháp liên quan được gọi là đơn vị mã hóa (CU). Trong ITU-T H.265, kích thước tối thiểu cho phép của CB có thể được báo hiệu. Trong ITU-T H.265, kích thước nhỏ nhất cho phép của CB luma là các mẫu luma 8x8. Trong ITU-T H.265, quyết định mã hóa vùng ảnh bằng dự đoán nội bộ hoặc dự đoán liên khung được thực hiện ở cấp CU.

Trong ITU-T H.265, CU được liên kết với cấu trúc đơn vị dự đoán (PU) có gốc tại CU. Trong ITU-T H.265, các cấu trúc PU cho phép tách các CB luma và sắc độ nhằm mục đích tạo ra các mẫu tham chiếu tương ứng. Nghĩa là, trong ITU-T H.265, CB luma và sắc độ có thể được chia thành các khối dự đoán (PB) luma và sắc độ tương ứng, trong đó PB bao gồm một khối các giá trị mẫu được áp dụng cùng một dự đoán. Trong ITU-T H.265, CB có thể được phân chia thành 1, 2 hoặc 4 PB. ITU-T H.265 hỗ trợ kích thước PB từ các mẫu 64x64 đến 4x4. Trong ITU-T H.265, PB vuông được hỗ trợ cho dự đoán nội bộ, trong đó CB có thể tạo thành PB hoặc CB có thể được tách thành bốn PB vuông. Trong ITU-T H.265, ngoài các PB vuông, các PB hình chữ nhật được hỗ trợ cho dự đoán liên khung, trong đó CB có thể giảm một nửa theo chiều dọc hoặc chiều ngang để tạo thành PB. Ngoài ra, xin lưu ý rằng trong ITU-T H.265, đối với dự đoán liên khung, bốn phân vùng PB không đối xứng được hỗ trợ, trong đó CB được phân vùng thành hai PB ở một phần tư chiều cao (ở trên cùng hoặc dưới cùng) hoặc chiều rộng (ở bên trái hoặc bên phải) của CB. Dữ liệu dự đoán nội ảnh (ví dụ: các phần tử cú pháp trong chế độ dự đoán nội ảnh) hoặc dữ liệu dự đoán liên khung (ví dụ: các phần tử cú pháp dữ liệu về chuyển động) tương ứng với PB được dùng để tạo ra các giá trị mẫu tham chiếu và/hoặc dự đoán cho PB.

JEM chỉ định CTU có kích thước tối đa là các mẫu luma 256x256. JEM chỉ định cấu trúc khối cây tứ phân cộng với cây nhị phân (QTBT). Trong JEM, cấu trúc QTBT cho phép các nút lá cây tứ phân được phân chia thêm bằng cấu trúc cây nhị phân (BT). Nghĩa là, trong JEM, cấu trúc cây nhị phân cho phép các nút lá cây tứ phân được chia đệ quy theo chiều dọc hoặc chiều ngang. Trong JVET-N1001 và JVET-O2001, CTU được phân vùng theo cấu trúc cây đa kiểu hình cộng với cây tứ phân (QTMT hoặc QT+MTT). QTMT trong JVET-N1001 và JVET-O2001 tương tự như QTBT trong JEM. Tuy nhiên, trong JVET-N1001 và JVET-O2001, ngoài việc biểu thị các phép tách nhị phân, cây đa kiểu hình có thể biểu thị cái gọi là phép tách tam phân (hoặc cây tam phân (TT)). Phép tách tam phân chia một khối theo chiều dọc hoặc chiều ngang thành ba khối. Trong trường hợp tách TT theo chiều dọc, khối được chia theo một phần tư chiều rộng của khối từ cạnh trái và một phần tư chiều rộng từ cạnh phải và trong trường hợp tách TT theo chiều ngang, khối được chia bằng một phần tư chiều cao từ cạnh trên và bằng một phần tư chiều cao từ cạnh dưới. Tham chiếu lần nữa đến HÌNH 1, HÌNH 1 minh họa ví dụ về CTU được phân chia thành các nút lá cây tứ phân, và các nút lá cây tứ phân được phân chia thêm theo phép phân tách BT hoặc phân tách TT. Nghĩa là, trong HÌNH 1, đường nét đứt biểu thị các phép tách nhị phân và tam phân bổ sung trong một cây tứ phân.

Như mô tả ở trên, mỗi khung hình video hoặc hình ảnh có thể được chia thành một hoặc nhiều vùng. Ví dụ: theo ITU-T H.265, mỗi khung hình video hoặc hình ảnh có thể được phân vùng để có một hoặc nhiều mảnh và được phân vùng thêm để có một hoặc nhiều ô, trong đó mỗi mảnh gồm một chuỗi CTU (ví dụ: trong thứ tự quét mành) và một ô là một dãy các CTU tương ứng với diện tích hình chữ nhật của một ảnh. Xin lưu ý rằng một mảnh, trong ITU-T H.265, là một chuỗi gồm một hoặc nhiều phân đoạn mảnh bắt đầu bằng một phân đoạn mảnh độc lập và chứa tất cả các phân đoạn mảnh phụ thuộc tiếp theo (nếu có) trước phân đoạn mảnh độc lập tiếp theo (nếu có). Một phân đoạn mảnh, tương tự như một mảnh, là một chuỗi các CTU. Do đó, trong một số trường hợp, các thuật ngữ mảnh và phân đoạn mảnh có thể được dùng thay thế cho nhau để chỉ một chuỗi CTU được bố trí theo thứ tự quét raster. Ngoài ra, xin lưu ý rằng trong ITU-T H.265, một ô có thể bao gồm các CTU có trong nhiều hơn một mảnh và một mảnh có thể bao gồm các CTU có trong nhiều hơn một ô. Tuy nhiên, ITU-T H.265 quy định rằng cần phải thỏa mãn một hoặc cả hai điều kiện sau: (1) Tất cả các CTU trong một mảnh đều phải thuộc cùng một ô; và (2) Tất cả các CTU trong một ô đều thuộc cùng một mảnh.

Đối với JVET-N1001 và JVET-O2001, các mảnh cần phải bao gồm một số nguyên các khối (brick) thay vì chỉ cần bao gồm một số nguyên các CTU. Trong JVET-N1001 và JVET-O2001, khối (brick) là vùng hình chữ nhật của các hàng CTU bên trong một ô cụ thể trong hình ảnh. Ngoài ra, trong JVET-N1001 và JVET-O2001, một ô có thể được phân chia thành nhiều khối (brick), mỗi khối này bao gồm một hoặc nhiều hàng CTU bên trong ô. Một viên gạch không được chia thành nhiều viên gạch cũng được gọi là một viên gạch. Tuy nhiên, một khối là tập con thực sự của ô không được gọi là ô. Do đó, một mảnh bao gồm một tập hợp CTU không tạo thành vùng hình chữ nhật của hình ảnh có thể hoặc không được hỗ trợ trong một số kỹ thuật mã hóa video. Hơn nữa, cần lưu ý là trong một số trường hợp một mảnh cần phải gồm một số nguyên các ô hoàn chỉnh và trong trường hợp này được gọi là một nhóm ô. Các kỹ thuật được mô tả trong tài liệu này có thể áp dụng cho các khối, mảnh, ô và/hoặc các nhóm ô. HÌNH 1 là sơ đồ khái niệm minh họa ví dụ về một nhóm hình ảnh bao gồm các mảnh. Theo ví dụ minh họa trong HÌNH 1, Pic_3 được minh họa là bao gồm hai mảnh (nghĩa là $Slice_0$ và $Slice_1$). Theo ví dụ minh họa trong HÌNH 1, $Slice_0$ bao gồm một brick (khối), nghĩa là $Brick_0$ và $Slice_1$ bao gồm hai khối, nghĩa là $Brick_1$ và $Brick_2$. Cần lưu ý rằng trong một số trường hợp, $Slice_0$ và $Slice_1$ có thể đáp ứng yêu cầu và được phân loại dưới dạng nhóm ô và/hoặc nhóm các ô.

Đối với mã hóa dự đoán nội bộ, chế độ dự đoán nội bộ có thể chỉ định vị trí của các mẫu tham chiếu trong một hình ảnh. Trong ITU-T H.265, chế độ dự đoán nội bộ có thể được xác định bao gồm chế độ dự đoán phẳng (nghĩa là điều chỉnh bề mặt), chế độ dự đoán DC (nghĩa là trung bình tổng thể phẳng) và chế độ dự đoán góc 33 (predMode: 2-34). Trong JEM, các chế độ dự đoán nội bộ khả thi có thể bao gồm chế độ dự đoán phẳng, chế độ dự đoán DC và chế độ dự đoán góc 65. Xin lưu ý rằng có thể gọi các chế độ dự đoán phẳng và DC là các chế độ dự đoán vô hướng và có thể gọi các chế độ dự đoán góc là các chế độ dự đoán có hướng. Xin lưu ý rằng có thể áp dụng chung các kỹ thuật được mô tả trong tài liệu này bất kể số lượng các chế độ dự đoán có thể xác định là bao nhiêu.

Đối với mã hóa dự đoán liên khung, hình ảnh tham chiếu được xác định và vectơ chuyển động (MV) xác định các mẫu trong hình ảnh tham chiếu được dùng để tạo dự đoán cho khối video hiện tại. Ví dụ: có thể dự đoán một khối video hiện tại bằng cách sử dụng các giá trị mẫu tham chiếu nằm trong một hoặc nhiều hình ảnh được mã hóa trước đó và một vectơ chuyển động được dùng để chỉ ra vị trí của khối tham chiếu so với khối video hiện tại. Ví dụ: một vectơ chuyển động có thể mô tả thành phần dịch chuyển theo chiều ngang của vectơ chuyển

động (ví dụ: MV_x), thành phần dịch chuyển thẳng đứng của vectơ chuyển động (ví dụ: MV_y) và độ phân giải cho vectơ chuyển động (ví dụ: độ chính xác 1/4 pixel, độ chính xác 1/2 pixel, độ chính xác một pixel, độ chính xác hai pixel, độ chính xác bốn pixel). Có thể sắp xếp các hình ảnh được giải mã trước đây, có thể bao gồm ảnh xuất ra trước hoặc sau ảnh hiện tại, thành một hoặc nhiều ảnh để tham chiếu danh sách ảnh và xác định những hình ảnh này bằng giá trị chỉ số hình ảnh tham chiếu. Hơn nữa, trong kỹ thuật mã hóa dự đoán liên khung, dự đoán đơn hướng đề cập đến việc tạo một dự đoán bằng các giá trị mẫu từ một hình ảnh tham chiếu duy nhất và dự đoán kép là tạo một dự đoán bằng các giá trị mẫu tương ứng từ hai hình ảnh tham chiếu. Nghĩa là, trong dự đoán đơn hướng, một hình ảnh tham chiếu duy nhất và vectơ chuyển động tương ứng được dùng để tạo một dự đoán cho một khối video hiện tại và trong dự đoán kép. một hình ảnh tham chiếu đầu tiên và vectơ chuyển động đầu tiên tương ứng và một hình ảnh tham chiếu thứ hai và vectơ chuyển động thứ hai tương ứng được dùng để tạo một dự đoán cho một khối video hiện tại. Trong dự đoán hai hướng, các giá trị mẫu tương ứng được kết hợp (ví dụ: thêm vào, làm tròn và cắt bớt, hoặc tính trung bình theo trọng số) để tạo một dự đoán. Hình ảnh và vùng của chúng có thể được phân loại dựa trên loại chế độ dự đoán nào có thể dùng để mã hóa các khối video của chúng. Nghĩa là, đối với các vùng có loại B (ví dụ: mảnh B), có thể sử dụng các chế độ dự đoán hai hướng, dự đoán đơn hướng và dự đoán nội bộ, và có thể sử dụng dự đoán đơn hướng và các chế độ dự đoán nội bộ đối với các vùng có loại P (ví dụ: mảnh P), và đối với các vùng có loại I (ví dụ: mảnh I) thì chỉ sử dụng các chế độ dự đoán nội bộ. Như mô tả ở trên, hình ảnh tham chiếu được xác định thông qua các chỉ số tham chiếu. Theo ví dụ, đối với mảnh P, có một danh sách ảnh tham chiếu duy nhất, RefPicList0 và đối với mảnh B, có thể có một danh sách ảnh tham chiếu độc lập thứ hai, RefPicList1 ngoài RefPicList0. Xin lưu ý rằng, đối với dự đoán đơn hướng trong mảnh B, có thể dùng một trong các RefPicList0 hoặc RefPicList1 để tạo dự đoán. Ngoài ra, xin lưu ý rằng trong quá trình giải mã, khi bắt đầu giải mã một hình ảnh, danh sách ảnh tham chiếu được tạo từ hình ảnh đã giải mã trước đó sẽ được lưu trữ trong bộ đệm hình ảnh đã giải mã (DPB).

Hơn nữa, một chuẩn mã hóa có thể hỗ trợ nhiều chế độ dự đoán vectơ chuyển động khác nhau. Dự đoán vectơ chuyển động cho phép lấy giá trị của một vectơ chuyển động cho khối video hiện tại dựa trên một vectơ chuyển động khác. Ví dụ: có thể lấy tập hợp các khối ứng viên có thông tin chuyển động liên quan từ các khối lân cận theo không gian và các khối lân cận theo thời gian đến khối video hiện tại. Hơn nữa, có thể sử dụng thông tin chuyển động được tạo (hoặc mặc định) cho quá trình dự đoán vectơ chuyển động. Ví dụ về dự đoán vectơ

chuyển động bao gồm dự đoán vectơ chuyển động nâng cao (AMVP), dự đoán vectơ chuyển động theo thời gian (TMVP), còn gọi là chế độ "hợp nhất" và suy luận chuyển động "bỏ qua" và "trực tiếp". Ngoài ra, các ví dụ khác về dự đoán vectơ chuyển động bao gồm dự đoán vectơ chuyển động nâng cao theo thời gian (ATMVP) và dự đoán vectơ chuyển động theo không gian-thời gian (STMVP). Đối với dự đoán vectơ chuyển động, cả bộ mã hóa video và bộ giải mã video đều thực hiện cùng một quy trình để lấy ra một tập hợp ứng viên. Do đó, đối với một khối video hiện tại, quá trình mã hóa và giải mã sẽ tạo ra cùng một tập hợp ứng viên.

Như được mô tả ở trên, để mã hóa dự đoán liên khung, các mẫu tham chiếu trong hình ảnh đã được mã hóa trước đó được sử dụng để mã hóa các khối video trong hình ảnh hiện tại. Các hình ảnh được mã hóa trước đây có sẵn để sử dụng làm tham chiếu khi bước mã hóa hình ảnh hiện tại được gọi là hình ảnh tham chiếu. Cần lưu ý rằng thứ tự giải mã không cần thiết phải tương ứng với thứ tự đầu ra hình ảnh, nghĩa là thứ tự hình ảnh trong trình tự video theo thời gian. Trong ITU-T H.265, khi ảnh được giải mã, nó sẽ được lưu trữ vào bộ đệm hình ảnh đã giải mã (DPB) (có thể được gọi là bộ đệm khung, bộ đệm tham chiếu, bộ đệm hình ảnh tham chiếu hoặc tương tự). Trong ITU-T H.265, hình ảnh được lưu vào DPB xóa khỏi DPB khi chúng được xuất ra và không còn cần thiết để mã hóa các hình ảnh tiếp theo. Trong ITU-T H.265, việc xác định xem liệu có nên xóa hình ảnh khỏi DPB hay không được sử dụng một lần cho mỗi hình ảnh, sau khi giải mã phần đầu mảnh, nghĩa là khi bắt đầu giải mã hình ảnh. Ví dụ, xem HÌNH 1, Pic_3 được minh họa dưới dạng tham chiếu Pic_2. Tương tự, Pic_4 được minh họa dưới dạng tham chiếu Pic_1. Đối với HÌNH 1, giả sử số hình ảnh tương ứng với thứ tự giải mã, DPB sẽ được tự động thêm thông tin như sau: sau khi giải mã Pic_1, DPB này sẽ bao gồm {Pic_1}; khi bắt đầu giải mã Pic_2, DPB sẽ bao gồm {Pic_1}; sau khi giải mã Pic_2, DPB sẽ bao gồm {Pic_1, Pic_2}; khi bắt đầu giải mã Pic3, DPB sẽ bao gồm {Pic_1, Pic_2}. Sau đó Pic_3 sẽ được giải mã theo Pic_2 và sau khi giải mã Pic_3, DPB sẽ bao gồm {Pic_1, Pic_2, Pic_3}. Khi bắt đầu giải mã Pic_4, hình ảnh Pic_2 và Pic_3 sẽ được đánh dấu để xóa khỏi DPB, vì chúng không cần thiết để giải mã Pic_4 (hoặc hình ảnh bất kỳ tiếp theo không được minh họa) và giả sử Pic_2 và Pic_3 đã được xuất ra, DPB sẽ được cập nhật để bao gồm {Pic_1}. Sau đó Pic_4 sẽ được giải mã bằng Pic_1 tham chiếu. Quy trình đánh dấu hình ảnh để xóa khỏi DPB có thể được gọi là quản lý bộ hình ảnh tham chiếu (RPS).

Như đã mô tả ở trên, dữ liệu dự đoán nội ảnh hoặc dữ liệu dự đoán liên khung được dùng để tạo ra các giá trị mẫu tham chiếu cho khối các giá trị mẫu. Sự chênh lệch giữa các giá trị mẫu được bao gồm trong PB hiện tại hoặc một loại cấu trúc vùng hình ảnh khác và

các mẫu tham chiếu liên quan (ví dụ, những mẫu tham chiếu được tạo ra bằng cách sử dụng dự đoán) có thể được gọi là dữ liệu dư. Dữ liệu dư thừa có thể bao gồm các mảng giá trị chênh lệch tương ứng, tương ứng với từng thành phần của dữ liệu video. Dữ liệu dư thừa có thể nằm trong miền pixel. Có thể áp dụng một phép biến đổi, chẳng hạn như phép biến đổi cosin rời rạc (DCT), phép biến đổi sin rời rạc (DST), phép biến đổi số nguyên, phép biến đổi wavelet hoặc phép biến đổi tương tự về mặt khái niệm, cho một mảng các giá trị chênh lệch để tạo ra các hệ số biến đổi. Lưu ý rằng trong ITU-T H.265 và JVET-N1001 và JVET-O2001, CU được liên kết với cấu trúc đơn vị biến đổi (TU) có gốc ở cấp CU. Nghĩa là, có thể phân vùng một mảng các giá trị chênh lệch nhằm tạo các hệ số biến đổi (ví dụ: có thể áp dụng bốn phép biến đổi 8x8 cho mảng giá trị dư thừa 16x16). Đối với mỗi thành phần của dữ liệu video, có thể gọi các phân vùng của những giá trị chênh lệch như vậy là Khối chuyển đổi (TB). Xin lưu ý rằng, trong một số trường hợp, có thể áp dụng phép biến đổi lõi và phép biến đổi thứ cấp tiếp theo (trong bộ mã hóa video) để tạo ra các hệ số biến đổi. Đối với một bộ giải mã video, thứ tự chuyển đổi sẽ được đảo ngược.

Quá trình lượng tử hóa có thể được thực hiện trực tiếp trên hệ số biến đổi hoặc giá trị mẫu dư (ví dụ: trong trường hợp lượng tử hóa mã hóa palét). Lượng tử hóa xấp xỉ các hệ số biến đổi theo biên độ được giới hạn đến tập hợp các giá trị được định rõ. Về cơ bản, quá trình lượng tử hóa sẽ tỷ lệ hóa các hệ số biến đổi nhằm thay đổi lượng dữ liệu cần thiết để biểu diễn một nhóm hệ số biến đổi. Lượng tử hóa có thể bao gồm việc chia các hệ số biến đổi (hoặc các giá trị thu được từ bước cộng giá trị bù với các hệ số biến đổi) bằng hệ số tỷ lệ lượng tử hóa và bất kỳ hàm làm tròn nào liên quan (ví dụ: làm tròn đến số nguyên gần nhất). Có thể gọi hệ số biến đổi lượng tử hóa là giá trị cấp hệ số. Lượng tử hóa ngược (hay "khử lượng tử hóa") có thể bao gồm phép nhân các giá trị cấp hệ số với hệ số tỷ lệ lượng tử hóa và mọi phép toán cộng bù hoặc làm tròn đối ứng bất kỳ. Xin lưu ý rằng, như được sử dụng trong tài liệu này, thuật ngữ quá trình lượng tử hóa trong một số trường hợp có thể đề cập đến phép chia cho hệ số tỷ lệ để tạo ra các giá trị cấp và nhân với hệ số tỷ lệ để khôi phục hệ số biến đổi trong một số trường hợp. Nghĩa là, quá trình lượng tử hóa, trong một số trường hợp, có thể đề cập đến lượng tử hóa và một số trường hợp là lượng tử hóa ngược. Hơn nữa, cần lưu ý rằng mặc dù trong một số ví dụ dưới đây, các quy trình lượng tử hóa được mô tả liên quan đến các phép toán số học liên quan đến ký hiệu thập phân, nhưng các mô tả này chỉ dành cho mục đích minh họa và không nên hiểu là mang tính chất giới hạn. Ví dụ, các kỹ thuật được mô tả trong tài liệu này có thể được thực hiện trong thiết bị sử dụng các phép

toán nhị phân và tương tự. Ví dụ, các phép toán nhân và chia được mô tả trong tài liệu này có thể được thực hiện bằng cách sử dụng các phép toán dịch chuyển bit và tương tự.

Các hệ số biến đổi lượng tử hóa và các phần tử cú pháp (ví dụ: các phần tử cú pháp chỉ ra cấu trúc mã hóa cho một khối video) có thể được mã hóa entropy theo kỹ thuật mã hóa entropy. Quá trình mã hóa entropy bao gồm bước mã hóa các giá trị phần tử cú pháp bằng các thuật toán nén dữ liệu không làm mất dữ liệu. Các ví dụ về kỹ thuật mã hóa entropy bao gồm mã hóa độ dài biến đổi thích ứng với nội dung (CAVLC), mã hóa số học nhị phân thích ứng theo ngữ cảnh (CABAC), mã hóa entropy phân vùng theo khoảng xác suất (PIPE) và các phương pháp mã hóa tương tự. Hệ số biến đổi lượng tử được mã hóa entropy và các phần tử cú pháp được mã hóa entropy tương ứng có thể tạo thành một chuỗi bit theo chuẩn có thể dùng để tái tạo dữ liệu video tại một bộ giải mã video. Quá trình mã hóa entropy, ví dụ CABAC, có thể bao gồm việc thực hiện nhị phân hóa trên các phần tử cú pháp. Nhị phân hóa đề cập đến quá trình chuyển đổi một giá trị của phần tử cú pháp thành một chuỗi một hoặc nhiều bit. Các bit này có thể được gọi là "bin" (ngăn). Nhị phân hóa có thể bao gồm một hoặc nhiều kỹ thuật mã hóa kết hợp sau: mã hóa độ dài cố định, mã hóa đơn phân, mã hóa đơn phân cắt ngắn, mã hóa Rice cắt ngắn, mã hóa Golomb, mã hóa Golomb theo cấp số nhân bậc k và mã hóa Golomb-Rice. Ví dụ: mã hóa nhị phân có thể bao gồm bước biểu diễn giá trị số nguyên 5 cho một phần tử cú pháp là 00000101 bằng kỹ thuật mã hóa nhị phân có độ dài cố định 8 bit hoặc biểu diễn giá trị nguyên 5 là 11110 bằng kỹ thuật mã hóa nhị phân mã hóa đơn phân. Như được sử dụng trong tài liệu này, mỗi thuật ngữ trong số các thuật ngữ mã hóa độ dài cố định, mã hóa đơn phân, mã hóa đơn phân cắt ngắn, mã hóa Rice cắt ngắn, mã hóa Golomb, mã hóa Golomb theo cấp số nhân bậc k và mã hóa Golomb-Rice có thể chỉ việc triển khai chung các kỹ thuật này và/hoặc triển khai các kỹ thuật mã hóa này theo cách cụ thể hơn. Ví dụ: có thể xác định cụ thể việc triển khai mã hóa Golomb-Rice theo một chuẩn mã hóa video. Trong ví dụ về CABAC, đối với một bin (ngăn) cụ thể, ngữ cảnh cung cấp giá trị trạng thái có thể xảy ra nhất (MPS) cho bin (nghĩa là MPS cho bin là một trong 0 hoặc 1) và giá trị xác suất của bin là MPS hoặc trạng thái ít có thể nhất (LPS). Ví dụ, ngữ cảnh có thể cho thấy MPS của bin là 0 và xác suất của bin là 1 là 0,3. Lưu ý rằng có thể xác định ngữ cảnh dựa trên giá trị của các bin được mã hóa trước đó, bao gồm các bin trong phần tử cú pháp hiện tại và các phần tử cú pháp đã được mã hóa trước đó. Ví dụ: giá trị của các phần tử cú pháp được liên kết với các khối video lân cận có thể được sử dụng để xác định ngữ cảnh cho bin hiện tại.

HÌNH 2A-2B là sơ đồ khái niệm minh họa các ví dụ về việc mã hóa một khối dữ liệu video. Như được minh họa trong HÌNH 2A, khối dữ liệu video hiện tại (ví dụ: CB tương ứng với thành phần video) được mã hóa bằng cách tạo ra phần dư bằng cách trừ một tập hợp giá trị dự đoán khỏi khối dữ liệu video hiện tại, thực hiện phép biến đổi trên phần dư và lượng tử hóa hệ số biến đổi để tạo giá trị mức. Như được minh họa trong HÌNH 2B, khối dữ liệu video hiện tại được giải mã bằng cách thực hiện lượng tử hóa ngược trên các giá trị mức, thực hiện biến đổi ngược và thêm tập hợp giá trị dự đoán vào phần dư thu được. Lưu ý rằng theo các ví dụ trong HÌNH 2A-2B, các giá trị mẫu của khối được tái tạo khác với các giá trị mẫu của khối video hiện tại được mã hóa. Đặc biệt, HÌNH 2B minh họa Lỗi tái tạo là chênh lệch giữa Khối hiện tại và Khối đã tái tạo. Theo cách này, kỹ thuật mã hóa có thể được coi là mất dữ liệu. Tuy nhiên, người xem video được tái tạo có thể coi sự khác biệt về giá trị mẫu là có thể chấp nhận hoặc không thể nhận thấy được.

Ngoài ra, như được minh họa trong HÌNH 2A-2B, các giá trị cấp hệ số được tạo ra bằng cách sử dụng mảng hệ số tỷ lệ. Trong ITU-T H.265, mảng hệ số tỷ lệ được tạo ra bằng cách chọn ma trận tỷ lệ và nhân từng mục nhập trong ma trận tỷ lệ với hệ số tỷ lệ lượng tử hóa. Trong ITU-T H.265, ma trận tỷ lệ được chọn một phần dựa trên chế độ dự đoán và thành phần màu, trong đó ma trận tỷ lệ có các kích thước sau được xác định: 4x4, 8x8, 16x16 và 32x32. Lưu ý rằng theo một số ví dụ, ma trận tỷ lệ có thể cung cấp cùng một giá trị cho mỗi mục nhập (nghĩa là tất cả hệ số được chia tỷ lệ theo một giá trị). Trong ITU-T H.265, giá trị của hệ số tỷ lệ lượng tử hóa có thể được xác định bằng thông số lượng tử hóa - QP. Trong ITU-T H.265, đối với độ sâu bit là 8 bit, QP có thể lấy 52 giá trị từ 0 đến 51 và sự thay đổi 1 đối với QP thường tương ứng với sự thay đổi giá trị của hệ số tỷ lệ lượng tử hóa xấp xỉ 12 %. Ngoài ra, trong ITU-T H.265, giá trị QP đối với tập hợp hệ số biến đổi có thể được dẫn xuất bằng cách sử dụng giá trị thông số lượng tử hóa dự đoán (có thể được gọi là giá trị QP dự đoán hoặc giá trị dự đoán QP) và giá trị delta thông số lượng tử hóa được tùy chọn phát tín hiệu (có thể được gọi là giá trị delta QP hoặc giá trị QP delta). Trong ITU-T H.265, thông số lượng tử hóa có thể được cập nhật cho mỗi CU và thông số lượng tử hóa tương ứng có thể được dẫn xuất cho từng kênh luma (ánh sáng) và chroma (sắc độ).

Như được mô tả ở trên, đối với ví dụ được minh họa trong HÌNH 2A-2B, giá trị mẫu của khối được tái tạo có thể khác với giá trị mẫu của khối video hiện tại được mã hóa. Ngoài ra, lưu ý rằng trong một số trường hợp, việc mã hóa dữ liệu video trên cơ sở từng khối có thể dẫn đến hiện tượng giả (ví dụ: cái gọi là giả chặn, giả băng tần, v.v.) Ví dụ: việc giả chặn có

thể khiến người dùng có thể cảm nhận được bằng mắt thường đường biên khối mã hóa của dữ liệu video tái tạo. Theo cách này, các giá trị mẫu đã tái tạo có thể được cải biến để giảm thiểu sự chênh lệch giữa các giá trị mẫu của khối video hiện tại được mã hóa và khối được tái tạo và/hoặc giảm thiểu thành phần giả do quy trình mã hóa video đưa vào. Những cải biến này có thể được gọi chung là quy trình lọc. Lưu ý rằng quy trình lọc có thể xảy ra trong quy trình lọc trong vòng hoặc quy trình lọc sau vòng. Đối với quy trình lọc trong vòng, giá trị mẫu tạo thành của quy trình lọc có thể được dùng cho khối video dự đoán (ví dụ: được lưu trữ vào bộ đệm khung tham chiếu để mã hóa tiếp theo tại bộ mã hóa video và giải mã tiếp theo tại bộ giải mã video). Đối với quy trình lọc sau vòng, giá trị mẫu thu được của quy trình lọc chỉ là đầu ra trong quy trình giải mã (ví dụ: không được sử dụng cho quy trình mã hóa tiếp theo). Ví dụ: trong trường hợp bộ giải mã video, đối với quy trình lọc trong vòng, giá trị mẫu thu được từ bước lọc khối đã tái tạo sẽ được sử dụng cho quy trình giải mã tiếp theo (ví dụ: được lưu trữ vào bộ đệm tham chiếu) và sẽ được xuất ra (ví dụ: đến màn hình). Đối với quy trình lọc sau vòng, khối đã tái tạo sẽ được sử dụng cho quy trình giải mã tiếp theo và các giá trị mẫu thu được từ bước lọc khối đã tái tạo sẽ được xuất ra.

Tách khối (hoặc tách khối), lọc tách khối hoặc áp dụng bộ lọc tách khối đề cập đến quy trình làm phẳng đường biên của các khối video được tái tạo lân cận (nghĩa là khiến người xem ít cảm nhận được đường biên hơn). Việc làm phẳng đường biên của khối video lân cận đã tái tạo có thể bao gồm bước cải biến các giá trị mẫu có trong các hàng hoặc cột liền kề với đường biên. ITU-T H.265 cung cấp vị trí áp dụng bộ lọc tách khối cho các giá trị mẫu được tái tạo trong quy trình lọc trong vòng. ITU-T H.265 bao gồm hai loại bộ lọc tách khối mà có thể được sử dụng để sửa đổi mẫu luma: Bộ lọc mạnh sửa đổi giá trị mẫu trong ba hàng hoặc cột liền kề đến đường biên và Bộ lọc yếu sửa đổi giá trị mẫu trong hàng hoặc cột ngay bên cạnh đến đường biên và sửa đổi có điều kiện các giá trị mẫu trong hàng hoặc cột thứ hai từ đường biên đó. Ngoài ra, ITU-T H.265 bao gồm một loại bộ lọc có thể được sử dụng để cải biến các mẫu sắc độ (chroma): Bộ lọc bình thường.

Ngoài việc áp dụng bộ lọc tách khối trong quy trình lọc trong vòng, ITU-T H.265 cung cấp vị trí có thể áp dụng bước lọc Độ lệch tương thích mẫu (SAO) trong quy trình lọc trong vòng. Trong ITU-T H.265, SAO là quy trình cải biến các giá trị mẫu được tách khối trong một vùng bằng cách thêm giá trị bù có điều kiện. ITU-T H.265 cung cấp hai loại bộ lọc SAO mà có thể được áp dụng cho CTB: bù dải hoặc bù cạnh. Đối với mỗi bù dải và bù cạnh, bốn giá trị bù được đưa vào chuỗi bit. Đối với bù dải, độ lệch được áp

dụng phụ thuộc vào biên độ của giá trị mẫu (ví dụ: biên độ được ánh xạ đến các dải băng mà được ánh xạ đến bốn độ lệch được báo hiệu). Đối với bù cạnh, độ lệch được áp dụng phụ thuộc vào CTB có một phân loại cạnh ngang, dọc, đường chéo thứ nhất hoặc đường chéo thứ hai (ví dụ: các phân loại được ánh xạ đến bốn độ lệch được báo hiệu).

Loại quy trình lọc khác bao gồm cái gọi là bộ lọc vòng thích ứng (ALF). ALF với khả năng thích ứng dựa trên khối được định rõ trong JEM. Trong JEM, ALF được áp dụng sau bộ lọc SAO. Lưu ý rằng ALF có thể được áp dụng cho các mẫu tái tạo độc lập với các kỹ thuật lọc khác. Quy trình áp dụng ALF được xác định trong JEM tại bộ mã hóa video, có thể được tóm tắt như sau: (1) mỗi khối 2x2 của thành phần luma cho hình ảnh tái tạo được phân loại theo chỉ số phân loại; (2) tập hợp hệ số bộ lọc được dẫn xuất cho mỗi chỉ số phân loại; (3) quyết định lọc được xác định cho thành phần luma (ánh sáng); (4) quyết định lọc được xác định cho thành phần sắc độ; và (5) thông số bộ lọc (ví dụ: hệ số và quyết định) được báo hiệu.

Theo ALF được xác định trong JEM, mỗi khối 2x2 được phân loại theo chỉ số phân loại C, trong đó C là số nguyên trong phạm vi bao gồm từ 0 đến 24. C được dẫn xuất dựa trên hướng D của nó và giá trị được lượng tử hóa của hoạt động $\hat{A}$, theo phương trình sau:

$$C = 5D + \hat{A}$$

trong đó D và $\hat{A}$, gradien của hướng ngang, dọc và hai đường chéo được tính bằng cách sử dụng Laplacian 1-D như sau:

$$g_v = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} V_{k,l},\ V_{k,l} = |2R(k,l) - R(k,l-1) - R(k,l+1)|,$$

$$g_h = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} H_{k,l},\ H_{k,l} = |2R(k,l) - R(k-1,l) - R(k+1,l)|,$$

$$g_{d1} = \sum_{k=i-2}^{i+3} \sum_{l=j-3}^{j+3} D1_{k,l},\ D1_{k,l} = |2R(k,l) - R(k-1,l-1) - R(k+1,l+1)|$$

$$g_{d2} = \sum_{k=i-2}^{i+3} \sum_{j=j-2}^{j+3} D2_{k,l},\ D2_{k,l} = |2R(k,l) - R(k-1,l+1) - R(k+1,l-1)|$$

trong đó, chỉ số i và j đề cập đến tọa độ của mẫu phía trên bên trái trong khối 2 × 2 và $R(i,j)$ biểu thị mẫu được tái tạo tại tọa độ (i,j).

Giá trị građien tối đa và tối thiểu của hướng ngang và hướng dọc có thể được đặt là:

$$g^{\text{tối đa}}{}_{h,v} = max(g_h, g_v);$$

$$g^{\text{tối thiểu}}{}_{h,v} = min(g_h, g_v).$$

và giá trị građien tối đa và tối thiểu của hai hướng chéo có thể được đặt là:

$$g^{\text{tối đa}}{}_{d0,d1} = max(g_{d0}, g_{d1});$$

$$g^{\text{tối thiểu}}{}_{d0,d1} = min(g_{d0}, g_{d1}).$$

Trong JEM, để dẫn xuất giá trị của hướng D, các giá trị tối đa và tối thiểu được so sánh với nhau và có hai ngưỡng t_1 và t_2:

Bước 1. Nếu cả $g^{\text{tối đa}}{}_{h,v} \le t_1 \cdot g^{\text{tối thiểu}}{}_{h,v}$ và $g^{\text{tối đa}}{}_{d0,d1} \le t_1 \cdot g^{\text{tối thiểu}}{}_{d0,d1}$ là true, D được đặt là 0.

Bước 2. Nếu $g^{\text{tối đa}}{}_{h,v} / g^{\text{tối thiểu}}{}_{h,v} > g^{\text{tối đa}}{}_{d0,d1} / g^{\text{tối thiểu}}{}_{d0,d1}$, tiếp tục từ Bước 3; nếu không thì tiếp tục từ Bước 4.

Bước 3. Nếu $g^{\text{tối đa}}{}_{h,v} > t_2 \cdot g^{\text{tối thiểu}}{}_{h,v}$, D được đặt là 2; nếu không thì D được đặt là 1.

Bước 4. Nếu $g^{\text{tối đa}}{}_{d0,d1} > t_2 \cdot g^{\text{tối thiểu}}{}_{d0,d1}$, D được đặt là 4; nếu không thì D được đặt là 3.

Trong JEM, giá trị hoạt động A được tính như sau:

$$A = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} (V_{k,l} + H_{k,l}).$$

trong đó, A còn được lượng tử hóa trong phạm vi từ 0 đến 4 và giá trị lượng tử hóa này được ký hiệu là $\hat{A}$.

Như được mô tả ở trên, bước áp dụng ALF được xác định trong JEM tại bộ mã hóa video bao gồm bước dẫn xuất tập hợp hệ số bộ lọc đối với mỗi chỉ số phân loại và bước xác định quyết định lọc. Lưu ý rằng việc dẫn xuất tập hợp hệ số bộ lọc và xác định các quyết định lọc có thể là quy trình lặp đi lặp lại. Nghĩa là, tập hợp hệ số bộ lọc có thể được cập nhật dựa trên quyết định lọc và quyết định lọc có thể được cập nhật dựa trên tập hợp hệ số bộ lọc đã cập nhật, và điều này có thể được lặp lại nhiều lần. Ngoài ra, bộ mã hóa video có thể triển khai các thuật toán riêng khác nhau để xác định tập hợp hệ số bộ lọc và/hoặc để xác định quyết định lọc. Các kỹ thuật được mô tả trong tài liệu này thường có thể áp dụng được bất kể cách dẫn xuất tập hợp hệ số bộ lọc cho từng chỉ số phân loại và cách xác định quyết định lọc.

Theo một ví dụ, tập hợp hệ số bộ lọc được dẫn xuất bằng cách ban đầu dẫn xuất tập hợp hệ số bộ lọc tối ưu cho từng chỉ số phân loại. Hệ số bộ lọc tối ưu được dẫn xuất bằng cách so sánh giá trị mẫu mong muốn (nghĩa là giá trị mẫu trong video nguồn) với giá trị mẫu đã tái tạo sau khi áp dụng lọc và bằng cách giảm thiểu tổng sai số bình phương (SSE) giữa giá trị mẫu mong muốn và giá trị mẫu đã tái tạo sau khi thực hiện lọc. Sau đó, hệ số tối ưu đã dẫn xuất cho từng nhóm có thể được sử dụng để thực hiện lọc cơ bản trên các mẫu được tái tạo nhằm phân tích tính hiệu quả của ALF. Nghĩa là, giá trị mẫu mong muốn, giá trị mẫu được tái tạo trước khi áp dụng ALF và giá trị mẫu được tái tạo sau khi thực hiện ALF có thể được so sánh để xác định tính hiệu quả của việc áp dụng ALF bằng cách sử dụng hệ số tối ưu.

Theo ALF đã xác định trong JEM, mỗi mẫu tái tạo R(i,j) được lọc bằng cách xác định kết quả trong giá trị mẫu R'(i,j) theo phương trình sau, trong đó theo phương trình dưới đây, L biểu thị độ dài bộ lọc và f(k,l) biểu thị hệ số bộ lọc đã giải mã.

$$R'(i,j) = \sum_{k=-L/2}^{L/2} \sum_{l=-L/2}^{L/2} f(k,l) \times R(i+k, j+l)$$

Cần lưu ý rằng JEM xác định ba hình dạng bộ lọc (hình thoi 5x5, hình thoi 7x7 và hình thoi 9x9). Cần lưu ý rằng trong JEM, phép biến đổi hình học được áp dụng cho các hệ số bộ lọc f(k,l) tùy thuộc vào giá trị gradien: g_v, g_h, g_{d1}, g_{d2}, như được cung cấp trong Bảng 1.

Giá trị gradien	Biến đổi
$g_{d2} < g_{d1}$ và $g_h < g_v$	Không biến đổi
$g_{d2} < g_{d1}$ và $g_v < g_h$	Đường chéo
$g_{d1} < g_{d2}$ và $g_h < g_v$	Lật dọc
$g_{d1} < g_{d2}$ và $g_v < g_h$	Xoay

Bảng 1

trong đó Đường chéo, Lật dọc và Xoay được xác định như sau:

$$\text{Đường chéo:} \quad f_D(k, l) = f(l, k),$$

$$\text{Lật dọc:} \quad f_v(k, l) = f(k, K-l-1)$$

$$\text{Xoay:} \quad f_R(k, l) = f(K-l-1, k)$$

trong đó, K là kích thước của bộ lọc và $0 \leq k$, $1 \leq K\text{-}1$ là tọa độ hệ số, sao cho vị trí (0,0) ở góc trên bên trái và vị trí (K-1,K-1) ở góc dưới bên phải.

JEM cung cấp vị trí có thể được báo hiệu của tối đa 25 tập hợp hệ số bộ lọc luma (nghĩa là một tập hợp cho từng chỉ số phân loại khả dĩ). Do đó, hệ số tối ưu có thể được báo hiệu cho từng chỉ số phân loại xảy ra trong vùng hình ảnh tương ứng. Tuy nhiên, nhằm tối ưu hóa lượng dữ liệu cần thiết để báo hiệu tập hợp hệ số bộ lọc luma so với tính hiệu quả của bộ lọc, tối ưu hóa biến dạng tỷ lệ (RD) có thể được thực hiện. Ví dụ: JEM cung cấp vị trí mà tập hợp hệ số bộ lọc của các nhóm phân loại lân cận có thể được hợp nhất và báo hiệu bằng cách sử dụng một mảng ánh xạ tập hợp hệ số bộ lọc đến từng chỉ số phân loại. Ngoài ra, JEM cung cấp vị trí mà dự đoán hệ số thời gian có thể được sử dụng để báo hiệu hệ số. Nghĩa là, JEM cung cấp vị trí mà tập hợp hệ số bộ lọc cho hình ảnh hiện tại có thể được dự đoán dựa trên tập hợp hệ số bộ lọc của hình ảnh tham chiếu bằng cách kế thừa tập hợp hệ số bộ lọc được sử dụng cho hình ảnh tham chiếu. JEM còn cung cấp vị trí cho hình ảnh dự đoán nội ảnh, tập hợp 16 bộ lọc cố định có thể có sẵn để dự đoán tập hợp hệ số bộ lọc. Như được mô tả ở trên, việc dẫn xuất tập hợp hệ số bộ lọc và xác định quyết định lọc có thể là quy trình lặp đi lặp lại. Nghĩa là, ví dụ như hình dạng của ALF có thể được xác định dựa trên số lượng tập hợp hệ số bộ lọc được báo hiệu và tương tự, liệu ALF có được áp dụng cho một vùng hình ảnh có thể dựa trên tập hợp hệ

số bộ lọc được báo hiệu hay không và/hoặc hình dạng của bộ lọc. Lưu ý rằng đối với bộ lọc ALF, mỗi thành phần sử dụng tập hợp các giá trị mẫu từ thành phần tương ứng làm đầu vào và dẫn xuất các giá trị mẫu đầu ra. Nghĩa là, bộ lọc ALF được áp dụng cho từng thành phần độc lập với dữ liệu trong thành phần khác. Ngoài ra, lưu ý rằng JVET-N1001 và JVET-O2001 xác định bộ lọc tách khối, SAO và ALF mà có thể được mô tả là thường dựa trên bộ lọc tách khối, SAO và ALF được cung cấp trong ITU-T H.265 và JEM.

Định dạng lấy mẫu video, cũng có thể được gọi là định dạng sắc độ, có thể xác định số lượng mẫu sắc độ có trong CU so với số lượng mẫu luma có trong CU. Ví dụ: đối với định dạng lấy mẫu 4:2:0, tốc độ lấy mẫu đối với thành phần luma gấp đôi tốc độ lấy mẫu của các thành phần sắc độ theo cả hướng ngang và hướng dọc. Do đó, đối với CU được định dạng theo định dạng 4:2:0, chiều rộng và chiều cao của mảng mẫu cho thành phần luma sẽ gấp đôi chiều rộng và chiều cao của mỗi mảng mẫu cho thành phần sắc độ. HÌNH 3 là sơ đồ khái niệm minh họa ví dụ về bộ mã hóa được định dạng theo định dạng mẫu 4:2:0. HÌNH 3 minh họa vị trí tương đối của mẫu sắc độ (chroma) so với mẫu luma trong CU. Như được mô tả ở trên, CU thường được xác định theo số lượng mẫu luma ngang và dọc. Do đó, như được minh họa trong HÌNH 3, CU 16x16 được định dạng theo định dạng mẫu 4:2:0 bao gồm mẫu 16x16 của thành phần luma (ánh sáng) và mẫu 8x8 cho từng thành phần chroma (sắc độ). Ngoài ra, trong ví dụ được minh họa trong HÌNH 3, vị trí tương đối của mẫu chroma (sắc độ) đối với mẫu luma dành cho các khối video bên cạnh CU 16x16 được minh họa. Đối với CU được định dạng theo định dạng 4:2:2, chiều rộng của mảng mẫu đối với thành phần luma sẽ gấp đôi chiều rộng của mảng mẫu đối với mỗi thành phần sắc độ, nhưng chiều cao của mảng mẫu đối với thành phần luma sẽ bằng chiều cao của mảng mẫu đối với mỗi thành phần sắc độ. Ngoài ra, đối với CU được định dạng theo định dạng 4:4:4, mảng mẫu của thành phần luma có cùng chiều rộng và chiều cao với mảng mẫu của mỗi thành phần sắc độ. Theo HÌNH 3, đối với mẫu luma, đường mẫu liền kề ngay phía trên khối video có thể được gọi là đường tham chiếu 0 (RL_0) và các đường mẫu tiếp theo ở trên có thể tương ứng được gọi là đường tham chiếu 1 (RL_1), đường tham chiếu 2 (RL_2) và đường tham chiếu 3 (RL_3). Tương tự, cột mẫu bên trái của khối video hiện tại có thể được phân loại là đường tham chiếu theo cách tương tự (nghĩa là đường mẫu liền kề ngay bên trái khối video có thể được gọi là đường tham chiếu 0 (RL_0)).

Cần lưu ý rằng đối với định dạng lấy mẫu, ví dụ: định dạng mẫu 4:2:0, loại vị trí chorma có thể được xác định. Nghĩa là, ví dụ đối với định dạng mẫu 4:2:0, các giá trị bù ngang và dọc biểu thị cách định vị không gian tương đối có thể được xác định cho mẫu sắc

độ (chroma) đối với mẫu luma. Bảng 2 trình bày định nghĩa về HorizontalOffsetC và VerticalOffsetC đối với 5 loại vị trí sắc độ được cung cấp trong JVET-N1001 và JVET-O2001. Ngoài ra, HÌNH 4A-4F minh họa các loại vị trí sắc độ được xác định trong JVET-N1001 và JVET-O2001 đối với định dạng mẫu 4:2:0.

ChromaLocType	HorizontalOffsetC	VerticalOffsetC
0	0	0,5
1	0,5	0,5
2	0	0
3	0,5	0
4	0	1
5	0,5	1

Bảng 2

Đối với các phương trình được sử dụng ở đây, có thể sử dụng các toán tử số học sau đây:

+ Phép cộng

– Phép trừ

* Phép nhân, bao gồm cả phép nhân ma trận

x^y Sự mũ hóa. Chỉ định x thành lũy thừa của y. Trong các ngữ cảnh khác, ký hiệu như vậy được dùng để ghi chỉ số trên không nhằm biểu diễn là lũy thừa.

/ Phép chia số nguyên với việc làm tròn kết quả về 0. Ví dụ: 7 / 4 và –7 / –4 được rút gọn thành 1 và –7 / 4 và 7 / –4 được rút gọn thành –1.

÷ Được dùng để biểu thị phép chia trong phương trình toán học mà không có ý định rút gọn hoặc làm tròn.

x/y Được dùng để biểu thị phép chia trong phương trình toán học mà không có ý định rút gọn hoặc làm tròn.

x % y Phép chia lấy dư. Phần dư của x chia cho y chỉ xác định cho các số nguyên x và y, với x >= 0 và y > 0.

Ngoài ra, có thể sử dụng các toán tử logic sau đây:

x && y Boolean logic “and” (và) của x và y

x | | y Boolean logic “or” (hoặc) của x và y

! Boolean logic "not" (không)

x ? y : z Nếu x là TRUE hoặc không bằng 0, ước lượng giá trị của y; nếu không, sẽ ước lượng giá trị của z.

Ngoài ra, có thể sử dụng các toán tử quan hệ sau đây:

> Lớn hơn

>= Lớn hơn hoặc bằng

< Nhỏ hơn

<= Nhỏ hơn hoặc bằng

== Bằng

!= Không bằng

Ngoài ra, có thể sử dụng các toán tử thao tác bit sau:

& Toán tử thao tác bit "and" (và). Khi xử lý với các đối số nguyên, xử lý trên phép biểu diễn phần bù 2 của giá trị nguyên. Khi xử lý trên đối số nhị phân chứa ít bit hơn đối số khác, đối số ngắn hơn được mở rộng bằng cách thêm nhiều bit có giá trị cao hơn bằng 0.

Toán tử thao tác bit "or" (hoặc). Khi xử lý với các đối số nguyên, xử lý trên phép biểu diễn phần bù 2 của giá trị nguyên. Khi xử lý trên đối số nhị phân chứa ít bit hơn đối số khác, đối số ngắn hơn được mở rộng bằng cách thêm nhiều bit có giá trị cao hơn bằng 0.

^ Một toán tử thao tác bit "exclusive or" (bao hàm hoặc). Khi xử lý với các đối số nguyên, xử lý trên phép biểu diễn phần bù 2 của giá trị nguyên. Khi xử lý trên đối số nhị phân chứa ít bit hơn đối số khác, đối số ngắn hơn được mở rộng bằng cách thêm nhiều bit có giá trị cao hơn bằng 0.

x >> yDịch chuyển số học sang phải của phép biểu diễn số nguyên bù 2 của x bằng y chữ số nhị phân. Hàm này chỉ được xác định cho các giá trị nguyên không âm của y. Các bit được dịch chuyển thành các bit có giá trị cao nhất (MSB) do phép dịch chuyển sang phải có giá trị bằng MSB của x trước khi thực hiện phép chuyển dịch.

x << y Dịch chuyển số học sang trái của phép biểu diễn số nguyên bù 2 của x bằng y chữ số nhị phân. Hàm này chỉ được xác định cho các giá trị nguyên không âm của y. Các bit được dịch chuyển thành các bit có giá trị thấp nhất (LSB) do phép dịch sang trái có giá trị bằng 0.

Ngoài ra, có thể sử dụng các toán tử gán sau đây:

= Toán tử gán

+ + Gia tăng, nghĩa là *x*+ + tương đương với *x* = *x* + 1; khi được sử dụng trong chỉ số mảng, ước lượng giá trị của biến trước khi thực hiện phép toán tăng.

– – Giảm, nghĩa là *x*– – tương đương với *x* = *x* – 1; khi được sử dụng trong chỉ số mảng, ước lượng giá trị của biến trước khi thực hiện phép toán giảm.

+= Tăng theo số lượng được xác định, nghĩa là x += 3 tương đương với x = x + 3 và x += (–3) tương đương với x = x + (–3).

–= Giảm theo số lượng được xác định, nghĩa là x –= 3 tương đương với x = x – 3 và x –= (–3) tương đương với x = x – (–3).

Ngoài ra, có thể sử dụng các hàm toán học đã xác định sau đây:

$$\text{Abs}(x) = \begin{cases} x & ; \ x >= 0 \\ -x & ; \ x < 0 \end{cases}$$

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

Log2(x) logarit cơ số 2 của x;

$$\text{Min}(x,y) = \begin{cases} x & ; \ x <= y \\ y & ; \ x > y \end{cases}$$

$$\text{Max}(x,y) = \begin{cases} x & ; \ x >= y \\ y & ; \ x < y \end{cases}$$

$$\text{Clip3}(x,y,z) = \begin{cases} x & ; \ z < x \\ y & ; \ z > y \\ z & ; \ \text{nếu không} \end{cases}$$

HÌNH 5 là sơ đồ khối minh họa ví dụ về hệ thống có thể được tạo cấu hình để mã hóa (nghĩa là mã hóa và/hoặc giải mã) dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này. Hệ thống 100 biểu diễn ví dụ về hệ thống có thể thực hiện mã hóa video bằng kỹ thuật phân vùng được mô tả theo một hoặc nhiều kỹ thuật của sáng chế này. Như được minh họa trong HÌNH 5, hệ thống 100 bao gồm thiết bị nguồn 102, môi trường truyền thông 110 và thiết bị đích 120. Theo ví dụ được minh họa trên HÌNH 5, thiết bị nguồn 102 có thể bao gồm

thiết bị bất kỳ được tạo cấu hình để mã hóa dữ liệu video và phát dữ liệu video đã mã hóa đến môi trường truyền thông 110. Thiết bị đích 120 có thể bao gồm bất kỳ thiết bị nào được định cấu hình để nhận dữ liệu video đã mã hóa qua phương tiện truyền thông 110 và để giải mã dữ liệu video đã mã hóa. Thiết bị nguồn 102 và/hoặc thiết bị đích 120 có thể bao gồm các thiết bị tính toán được trang bị để truyền thông có dây và/hoặc không dây, và có thể bao gồm bộ chuyển đổi tín hiệu, máy ghi video kỹ thuật số, TV, máy tính để bàn, máy tính xách tay hoặc máy tính bảng, máy chơi game, thiết bị di động, bao gồm, ví dụ như điện thoại "thông minh", điện thoại di động, thiết bị chơi trò chơi cá nhân và thiết bị chụp hình y tế.

Phương tiện truyền thông 110 có thể bao gồm bất kỳ phương tiện truyền thông không dây và có dây kết hợp nào, và/hoặc các thiết bị lưu trữ. Phương tiện truyền thông 110 có thể bao gồm cáp đồng trục, cáp quang, cáp đôi xoắn, thiết bị thu và phát không dây, bộ định tuyến, bộ chuyển mạch, bộ lặp tín hiệu, trạm gốc hoặc bất kỳ thiết bị nào khác có thể hữu ích trong việc tạo điều kiện liên lạc giữa các thiết bị và địa điểm khác nhau. Phương tiện truyền thông 110 có thể bao gồm một hoặc nhiều mạng. Ví dụ: phương tiện truyền thông 110 có thể bao gồm một mạng được định cấu hình để cho phép truy cập vào World Wide Web, chẳng hạn như Internet. Một mạng có thể hoạt động dựa theo một hoặc nhiều giao thức viễn thông kết hợp. Giao thức viễn thông có thể bao gồm các khía cạnh riêng và/hoặc có thể bao gồm các giao thức viễn thông đã chuẩn hóa. Ví dụ về các giao thức viễn thông đã chuẩn hóa bao gồm tiêu chuẩn Phát sóng số (DVB), tiêu chuẩn của Ủy ban Hệ thống Truyền hình Tiên tiến (ATSC), tiêu chuẩn Dịch vụ phát sóng kỹ thuật số tích hợp (ISDB), tiêu chuẩn Đặc điểm giao diện dữ liệu qua dịch vụ cáp (DOCSIS), tiêu chuẩn Hệ thống thông tin di động toàn cầu (GSM), tiêu chuẩn Đa truy nhập phân chia theo mã (CDMA), tiêu chuẩn Dự án đối tác thế hệ thứ 3 (3GPP), tiêu chuẩn của Viện tiêu chuẩn Viễn thông Châu Âu (ETSI), tiêu chuẩn Giao thức Internet (IP), tiêu chuẩn Giao thức ứng dụng không dây (WAP) và các tiêu chuẩn của Viện kỹ sư điện và điện tử (IEEE).

Thiết bị lưu trữ có thể bao gồm bất kỳ loại thiết bị hoặc phương tiện lưu trữ nào có khả năng lưu trữ dữ liệu. Phương tiện lưu trữ có thể bao gồm phương tiện hữu hình hoặc không chuyển tiếp, có thể đọc được bằng máy tính. Phương tiện có thể đọc được bằng máy tính có thể bao gồm đĩa quang, bộ nhớ flash, bộ nhớ từ tính hoặc bất kỳ phương tiện lưu trữ kỹ thuật số phù hợp nào khác. Trong một số ví dụ, thiết bị nhớ hoặc các phần của thiết bị nhớ có thể được mô tả là bộ nhớ không khả biến và trong các ví dụ khác thì các phần của thiết bị nhớ có thể được mô tả là bộ nhớ khả biến. Bộ nhớ khả biến, ví dụ, có thể bao gồm bộ nhớ truy

cập ngẫu nhiên (RAM), bộ nhớ truy cập ngẫu nhiên động (DRAM) và bộ nhớ truy cập ngẫu nhiên tĩnh (SRAM). Bộ nhớ không khả biến, ví dụ, có thể bao gồm đĩa cứng từ tính, đĩa quang, đĩa mềm, bộ nhớ flash hoặc các dạng bộ nhớ khả lập điện tử (EPROM) hoặc bộ nhớ có thể xóa và lập trình điện tử (EEPROM). Thiết bị lưu trữ có thể bao gồm thẻ nhớ (ví dụ: thẻ nhớ Kỹ thuật số bảo mật (SD)), ổ đĩa cứng gắn trong/di động và/hoặc ổ đĩa thể rắn gắn trong/di động. Có thể lưu trữ dữ liệu trên thiết bị lưu trữ theo định dạng tệp xác định.

Tham chiếu lần nữa đến HÌNH 5, thiết bị nguồn 102 bao gồm nguồn video 104, bộ mã hóa video 106 và giao diện 108. Nguồn video 104 có thể bao gồm bất kỳ thiết bị nào được định cấu hình để quay và/hoặc lưu trữ dữ liệu video. Ví dụ: nguồn video 104 có thể bao gồm một máy quay video và một thiết bị lưu trữ được ghép nối để hoạt động với nhau. Bộ mã hóa video 106 có thể bao gồm bất kỳ thiết bị nào được tạo cấu hình để thu dữ liệu video và tạo chuỗi bit theo chuẩn biểu diễn dữ liệu video đó. Chuỗi bit tương thích có thể đề cập đến chuỗi bit mà từ đó bộ giải mã video có thể thu và tái tạo dữ liệu video. Có thể xác định các khía cạnh của chuỗi bit tương thích theo một chuẩn mã hóa video. Khi tạo chuỗi bit tương thích, bộ mã hóa video 106 có thể nén dữ liệu video. Nén có thể mất dữ liệu (có thể nhận thấy hoặc không) hoặc không mất dữ liệu. Giao diện 108 có thể bao gồm bất kỳ thiết bị nào được tạo cấu hình để thu chuỗi bit video tương thích và phát và/hoặc lưu trữ chuỗi bit video tương thích đến môi trường truyền thông. Giao diện 108 có thể bao gồm thẻ giao diện mạng, chẳng hạn như thẻ Ethernet, và có thể bao gồm bộ thu phát quang, bộ thu phát tần số vô tuyến hoặc bất kỳ loại thiết bị nào khác có thể gửi và/hoặc nhận thông tin. Ngoài ra, giao diện 108 có thể bao gồm giao diện hệ thống máy tính mà có thể cho phép lưu trữ chuỗi bit video tương thích trên thiết bị lưu trữ. Ví dụ: giao diện 108 có thể bao gồm một chipset hỗ trợ giao thức bus Kết nối thành phần ngoại vi (PCI) và Kết nối thành phần ngoại vi tốc độ cao (PCIe), giao thức bus riêng, giao thức Bus nối tiếp đa năng (USB), I^2C hoặc bất kỳ cấu trúc vật lý và logic nào khác có thể được dùng để kết nối các thiết bị ngang hàng với nhau.

Tham chiếu lần nữa đến HÌNH 5, thiết bị đích 120 bao gồm giao diện 122, bộ giải mã video 124 và màn hình 126. Giao diện 122 có thể bao gồm thiết bị bất kỳ được tạo cấu hình để thu chuỗi bit video tương thích từ môi trường truyền thông. Giao diện 108 có thể bao gồm thẻ giao diện mạng, chẳng hạn như thẻ Ethernet và có thể bao gồm bộ thu phát quang, bộ thu phát tần số vô tuyến hoặc bất kỳ loại thiết bị nào khác có thể thu và/hoặc gửi thông tin. Ngoài ra, giao diện 122 có thể bao gồm một giao diện hệ thống máy tính cho phép truy xuất chuỗi bit video tương thích từ thiết bị lưu trữ. Ví dụ: giao diện 122 có thể

bao gồm chipset hỗ trợ giao thức bus PCI và PCIe, giao thức bus riêng, giao thức USB, I^2C, hoặc bất kỳ cấu trúc vật lý và logic nào khác có thể dùng để kết nối các thiết bị ngang hàng với nhau. Bộ giải mã video 124 có thể bao gồm bất kỳ thiết bị nào được tạo cấu hình để thu chuỗi bit tương thích và/hoặc các biến thể có thể được chấp nhận của chúng và tái tạo dữ liệu video từ đó. Màn hình 126 có thể bao gồm bất kỳ thiết bị nào được định cấu hình để hiển thị dữ liệu video. Màn hình 126 có thể bao gồm một trong nhiều thiết bi hiển thị như màn hình tinh thể lỏng (LCD), màn hình plasma, màn hình đi-ốt phát quang hữu cơ (OLED) hoặc một loại màn hình khác. Màn hình 126 có thể bao gồm màn hình Độ nét cao (HD) hoặc màn hình Độ nét cực cao (UHD). Cần lưu ý rằng, mặc dù trong ví dụ minh họa trong HÌNH 5, bộ giải mã video 124 được mô tả là xuất dữ liệu ra màn hình 126, nhưng có thể tạo cấu hình bộ giải mã video 124 để xuất dữ liệu video tới nhiều loại thiết bị và/hoặc thành phần phụ của chúng. Ví dụ: có thể tạo cấu hình bộ giải mã video 124 để xuất dữ liệu video tới bất kỳ phương tiện truyền thông nào, như được mô tả trong tài liệu này.

HÌNH 6 là sơ đồ khối minh họa ví dụ về bộ mã hóa video 200 có thể triển khai các kỹ thuật mã hóa dữ liệu video được mô tả trong tài liệu này. Cần lưu ý rằng, dù bộ mã hóa video điển hình 200 được minh họa là có các khối chức năng riêng biệt, nhưng hình ảnh minh họa đó là nhằm mục đích mô tả và không giới hạn bộ mã hóa video 200 và/hoặc các thành phần phụ của chúng đối với một kiến trúc phần cứng hoặc phần mềm cụ thể. Có thể hiện thực hóa các chức năng của bộ mã hóa video 200 bằng cách triển khai kết hợp phần cứng, chương trình cơ sở và/hoặc phần mềm bất kỳ. Theo một ví dụ, có thể tạo cấu hình bộ mã hóa video 200 để mã hóa dữ liệu video theo các kỹ thuật được mô tả trong tài liệu này. Bộ mã hóa video 200 có thể thực hiện mã hóa dự đoán nội ảnh và mã hóa dự đoán liên khung các vùng hình ảnh, và vì vậy, có thể được gọi là bộ mã hóa video lai. Theo ví dụ được minh họa trong HÌNH 6, bộ mã hóa video 200 thu các khối video nguồn. Trong một số ví dụ, các khối video nguồn có thể bao gồm các vùng hình ảnh đã được phân chia theo cấu trúc mã hóa. Ví dụ: dữ liệu video nguồn có thể bao gồm các khối macro, CTU, CB, các phân vùng của chúng và/hoặc một bộ mã hóa tương đương khác. Theo một số ví dụ, có thể tạo cấu hình bộ mã hóa video 200 để thực hiện phân vùng phụ thêm các khối video nguồn. Xin lưu ý rằng một số kỹ thuật được mô tả trong tài liệu này có thể áp dụng chung cho kỹ thuật mã hóa video, bất kể dữ liệu video nguồn được phân vùng như thế nào trước và/hoặc trong quá trình mã hóa. Theo ví dụ được minh họa trong HÌNH 6, bộ mã hóa video 200 bao gồm bộ cộng 202, bộ tạo hệ số biến đổi 204, bộ lượng tử

hóa hệ số 206, bộ xử lý lượng tử hóa/biến đổi ngược 208, bộ cộng 210, bộ xử lý dự đoán nội ảnh 212, bộ xử lý dự đoán liên khung 214, bộ lọc 216 và bộ mã hóa entropy 218.

Như được minh họa trong HÌNH 6, bộ mã hóa video 200 thu các khối video nguồn và xuất ra chuỗi bit. Bộ mã hóa video 200 có thể tạo ra dữ liệu dư bằng cách lấy khối video nguồn trừ đi khối video dự đoán. Bộ cộng 202 biểu diễn thành phần được tạo cấu hình để thực hiện phép toán trừ này. Trong một ví dụ, phép trừ các khối video xảy ra trong miền pixel. Bộ tạo hệ số biến đổi 204 áp dụng phép biến đổi, chẳng hạn như biến đổi cosin rời rạc (DCT), biến đổi sin rời rạc (DST) hoặc biến đổi tương tự về mặt khái niệm, cho khối dư hoặc các phân vùng phụ của chúng (ví dụ: có thể áp dụng bốn phép biến đổi 8x8 cho mảng 16x16 giá trị dư) để tạo ra một tập hợp hệ số biến đổi dư. Có thể tạo cấu hình bộ tạo hệ số biến đổi 204 để thực hiện bất kỳ và tất cả các tổ hợp phép biến đổi có trong tập hợp phép biến đổi lượng giác rời rạc. Như được mô tả ở trên, trong ITU-T H.265, TB bị giới hạn ở các kích thước sau 4x4, 8x8, 16x16 và 32x32. Theo một ví dụ, bộ tạo hệ số biến đổi 204 có thể được tạo cấu hình để thực hiện các phép biến đổi theo mảng có các kích thước là 4x4, 8x8, 16x16 và 32x32. Theo một ví dụ, bộ tạo hệ số biến đổi 204 có thể tiếp tục được tạo cấu hình để thực hiện các phép biến đổi theo mảng có các chiều khác. Đặc biệt, trong một số trường hợp, có thể hữu ích khi thực hiện phép biến đổi trên những mảng hình chữ nhật có các giá trị khác nhau. Theo một ví dụ, bộ tạo hệ số biến đổi 204 có thể được tạo cấu hình để thực hiện những phép biến đổi theo các kích thước mảng sau: 2x2, 2x4N, 4Mx2 và/hoặc 4Mx4N. Theo một ví dụ, phép biến đổi ngược MxN 2 chiều (2D) có thể được thực hiện dưới dạng biến đổi ngược điểm M 1 chiều (1D), sau đó là phép biến đổi ngược điểm N 1D. Theo một ví dụ, phép biến đổi ngược 2D có thể được thực hiện dưới dạng biến đổi dọc điểm N 1D, sau đó là biến đổi ngang điểm N 1D. Theo một ví dụ, phép biến đổi ngược 2D có thể được thực hiện dưới dạng biến đổi ngang điểm N 1D, sau đó là biến đổi dọc điểm N 1D. Bộ tạo hệ số biến đổi 204 có thể xuất hệ số biến đổi thành bộ lượng tử hóa hệ số 206.

Có thể tạo cấu hình bộ lượng tử hóa hệ số 206 để thực hiện lượng tử hóa hệ số biến đổi. Như đã mô tả ở trên, có thể thay đổi mức độ lượng tử hóa bằng cách điều chỉnh một thông số lượng tử hóa. Có thể tạo cấu hình thêm cho bộ lượng tử hóa hệ số 206 để xác định thông số lượng tử hóa (QP) và xuất dữ liệu QP (ví dụ: dữ liệu được dùng để xác định kích thước nhóm lượng tử hóa và/hoặc giá trị QP delta) mà có thể được bộ giải mã video sử dụng để tái tạo thông số lượng tử hóa nhằm thực hiện lượng tử hóa ngược trong quá trình giải mã video. Xin lưu ý rằng trong các ví dụ khác, có thể dùng một hoặc nhiều

thông số bổ sung hoặc thay thế có thể để xác định mức lượng tử hóa (ví dụ: hệ số tỷ lệ). Các kỹ thuật được mô tả trong tài liệu này có thể áp dụng chung để xác định mức lượng tử hóa cho hệ số biến đổi tương ứng với một thành phần của dữ liệu video dựa trên mức lượng tử hóa cho hệ số biến đổi tương ứng với một thành phần khác của dữ liệu video.

Tham chiếu lần nữa đến HÌNH 6, hệ số biến đổi lượng tử hóa được xuất đến bộ xử lý lượng tử hóa/biến đổi ngược 208. Có thể tạo cấu hình bộ xử lý lượng tử hóa/biến đổi ngược 208 để áp dụng lượng tử hóa ngược và biến đổi ngược để tạo dữ liệu tái tạo dư. Như được minh họa trong HÌNH 6, tại bộ cộng 210, dữ liệu tái tạo dư có thể được thêm vào khối video dự đoán. Theo cách này, có thể tái tạo một khối video được mã hóa và có thể dùng khối video tái tạo thu được để đánh giá chất lượng mã hóa cho một dự đoán, phép chuyển đổi và/hoặc lượng tử hóa nhất định. Có thể tạo cấu hình bộ mã hóa video 200 để thực hiện nhiều lượt mã hóa (ví dụ: thực hiện mã hóa trong khi thay đổi một hoặc nhiều dự đoán, thông số chuyển đổi và thông số lượng tử hóa). Có thể tối ưu hóa biến dạng tỷ lệ của chuỗi bit hoặc các thông số hệ thống khác dựa trên việc đánh giá các khối video được tái tạo. Ngoài ra, có thể lưu trữ và dùng các khối video tái tạo làm giá trị tham chiếu để dự đoán các khối tiếp theo.

Như được mô tả ở trên, có thể mã hóa khối video bằng chế độ dự đoán nội ảnh. Bộ xử lý dự đoán nội ảnh 212 có thể được tạo cấu hình để chọn chế độ dự đoán nội ảnh cho khối video được mã hóa. Có thể tạo cấu hình bộ xử lý dự đoán nội ảnh 212 để đánh giá khung hình và/hoặc vùng của khung hình, và xác định chế độ dự đoán nội ảnh sẽ dùng để mã hóa khối hiện tại. Như được minh họa trong HÌNH 6, bộ xử lý dự đoán nội ảnh 212 xuất dữ liệu dự đoán nội ảnh (ví dụ: phần tử cú pháp) đến bộ mã hóa entropy 218 và bộ tạo hệ số biến đổi 204. Như được mô tả ở trên, phép biến đổi được thực hiện trên dữ liệu dư có thể phụ thuộc vào chế độ. Như đã mô tả ở trên, các chế độ dự đoán nội ảnh khả thi có thể bao gồm chế độ dự đoán phẳng, chế độ dự đoán DC và chế độ dự đoán góc. Ngoài ra, theo một số ví dụ, chế độ dự đoán cho thành phần sắc độ (chroma) có thể được suy ra từ dự đoán nội ảnh đối với chế độ dự đoán luma (ánh sáng). Có thể tạo cấu hình bộ xử lý dự đoán liên khung 214 để thực hiện mã hóa dự đoán liên khung cho khối video hiện tại. Có thể tạo cấu hình bộ xử lý dự đoán liên khung 214 để thu các khối video nguồn và tính toán vector chuyển động cho PU của khối video Một vectơ chuyển động có thể chỉ báo độ dịch chuyển của PU (hoặc cấu trúc mã hóa tương tự) của khối video trong khung hình video hiện tại so với khối dự đoán trong khung hình tham chiếu. Mã hóa dự đoán liên khung có thể sử dụng một hoặc nhiều hình ảnh tham chiếu. Hơn nữa, dự đoán chuyển động có thể là

dự đoán đơn hướng (sử dụng một vectơ chuyển động) hoặc dự đoán hai hướng (sử dụng hai vectơ chuyển động). Có thể tạo cấu hình bộ xử lý dự đoán liên khung 214 để chọn khối dự đoán bằng cách tính toán chênh lệch pixel được xác định theo, ví dụ: tổng chênh lệch tuyệt đối (SAD), tổng chênh lệch bình phương (SSD) hoặc các số liệu chênh lệch khác. Như đã mô tả ở trên, có thể xác định và chỉ định vectơ chuyển động theo kỹ thuật dự đoán vectơ chuyển động. Ngoài ra, có thể tạo cấu hình bộ xử lý dự đoán liên khung 214 để thực hiện dự đoán vectơ chuyển động như được mô tả ở trên. Bộ xử lý dự báo liên khung 214 có thể được tạo cấu hình để tạo khối dự đoán bằng cách dùng dữ liệu dự đoán chuyển động. Ví dụ: bộ xử lý dự đoán liên khung 214 có thể xác định vị trí khối video dự đoán trong bộ đệm khung (không được minh họa trong HÌNH 6). Cần lưu ý rằng có thể tạo cấu hình thêm cho bộ xử lý dự đoán liên khung 214 để áp dụng một hoặc nhiều bộ lọc nội suy cho khối dư được tái tạo nhằm tính toán giá trị pixel số nguyên phụ để sử dụng trong ước tính chuyển động. Bộ xử lý dự đoán liên khung 214 có thể xuất dữ liệu dự đoán chuyển động cho vectơ chuyển động đã tính toán đến bộ mã hóa entropy 218. Như được minh họa trong HÌNH 6, bộ xử lý dự đoán liên khung 214 có thể thu khối video được tái tạo thông qua bộ lọc 216. Bộ mã hóa entropy 218 thu hệ số biến đổi lượng tử hóa và dữ liệu cú pháp dự đoán (nghĩa là dữ liệu dự đoán nội ảnh, dữ liệu dự đoán chuyển động, dữ liệu QP, v.v.). Cần lưu ý rằng theo một số ví dụ, bộ lượng tử hóa hệ số 206 có thể thực hiện quét ma trận bao gồm các hệ số biến đổi lượng tử hóa trước khi hệ số được xuất đến bộ mã hóa entropy 218. Theo các ví dụ khác, bộ mã hóa entropy 218 có thể thực hiện quét. Có thể tạo cấu hình bộ mã hóa entropy 218 để thực hiện mã hóa entropy theo một hoặc nhiều kỹ thuật được mô tả trong tài liệu này. Có thể tạo cấu hình bộ mã hóa entropy 218 để xuất chuỗi bit tương thích, nghĩa là chuỗi bit mà từ đó bộ giải mã video có thể thu và tái tạo dữ liệu video.

Tham chiếu lần nữa đến HÌNH 6, bộ lọc 216 có thể được tạo cấu hình để thực hiện tách khối, lọc Độ lệch thích ứng mẫu (SAO) và/hoặc lọc ALF như mô tả ở trên. Ngoài ra, bộ lọc 216 có thể được tạo cấu hình để thực hiện một hoặc nhiều kỹ thuật được mô tả trong tài liệu này nhằm giảm lỗi tái tạo theo tương quan giữa các thành phần. Như được mô tả ở trên, đối với bộ lọc ALF trong JEM, mỗi thành phần sử dụng tập hợp giá trị mẫu từ thành phần tương ứng làm đầu vào và dẫn xuất giá trị mẫu đầu ra theo cách độc lập với các thành phần khác. Bước lọc độc lập trên cơ sở từng thành phần có thể ít lý tưởng hơn, vì có thể có mối tương quan giữa các thành phần và/hoặc kênh dữ liệu video có thể hữu ích để giảm thiểu lỗi tái tạo. Ví dụ: theo HÌNH 8, HÌNH 8 minh họa ví dụ về khối nguồn luma 8x8 và khối nguồn sắc độ

4x4 tương ứng (nghĩa là theo định dạng lấy mẫu 4:2:0) và khối được tái tạo tương ứng, cũng như các lỗi tái tạo. Như được minh họa trong HÌNH 8, cả hai khối nguồn đều có một cạnh về đường chéo, điều này sẽ là điển hình trong trường hợp có kết cấu, cạnh hình dạng hoặc tương tự. Tuy nhiên, đối với thành phần sắc độ đã tái tạo, độ chính xác bị mất (ví dụ: do mức lượng tử hóa cao, v.v.) so với thành phần luma và cạnh không được khôi phục.

Theo các kỹ thuật trong tài liệu này, bộ lọc có thể được tạo cấu hình để dự đoán và/hoặc lọc thông tin trong kênh và/hoặc thành phần màu thứ nhất từ thông tin trong kênh và/hoặc thành phần màu thứ hai, điều này có thể giúp cải thiện hiệu quả mã hóa của kênh và/hoặc thành phần màu thứ nhất, vì độ chính xác của kênh và/hoặc thành phần màu được tăng lên với một số lượng bit nhỏ. HÌNH 7 minh họa ví dụ về bộ lọc thành phần chéo có thể được tạo cấu hình để mã hóa dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này. Như được minh họa trong HÌNH 7, bộ lọc thành phần chéo 300 bao gồm thiết bị xác định bộ lọc 302 và bộ sửa đổi mẫu 304. Cần lưu ý rằng bộ lọc thành phần chéo 300 minh họa ví dụ về bộ lọc thành phần chéo mà có thể có trong bộ mã hóa video. Các ví dụ về bộ lọc thành phần chéo tương ứng có thể có trong bộ giải mã video được mô tả chi tiết hơn bên dưới. Như được minh họa trong HÌNH 7, thiết bị xác định bộ lọc 302 và bộ sửa đổi mẫu 304 có thể thu thông tin thông số mã hóa (ví dụ: chế độ dự đoán nội ảnh) có sẵn tại thời điểm khối hiện tại được mã hóa/giải mã và dữ liệu khối video (như được minh họa trong HÌNH 7) tại bộ mã hóa video có thể bao gồm: Khối nguồn thành phần chéo; Khối tái tạo thành phần chéo; Lỗi tái tạo thành phần chéo; Khối nguồn thành phần hiện tại; Khối tái tạo thành phần hiện tại; và Lỗi tái tạo thành phần hiện tại. Nghĩa là, tham khảo đến ví dụ được minh họa trong HÌNH 8, khi khối tái tạo sắc độ sẽ được lọc, tất cả thông tin trong HÌNH 8 có thể có sẵn ở bộ lọc thành phần chéo 300. Vì vậy, thiết bị xác định bộ lọc 302 có thể dẫn xuất bộ lọc được sử dụng trên khối tái tạo sắc độ dựa trên dữ liệu video, và bộ sửa đổi mẫu 304 có thể thực hiện lọc theo bộ lọc đã dẫn xuất. Như được minh họa trong HÌNH 7, bộ sửa đổi mẫu 304 có thể xuất khối tái tạo đã sửa đổi đến bộ đệm hình ảnh tham chiếu (nghĩa là dưới dạng bộ lọc trong vòng) và xuất khối tái tạo đã sửa đổi đến đầu ra (ví dụ: màn hình). Ngoài ra, như được minh họa trong HÌNH 7, thiết bị xác định bộ lọc 302 có thể xuất ra dữ liệu lọc. Nghĩa là, dữ liệu bộ lọc xác định bộ lọc đã dẫn xuất có thể được báo hiệu đến bộ giải mã video. Những ví dụ về việc báo hiệu này được mô tả chi tiết hơn bên dưới. Cần lưu ý rằng đối với HÌNH 8, có thể có một số cách giảm lỗi tái tạo ở bộ mã hóa video, ví dụ như giảm lượng tử hóa và/hoặc thực hiện cải

thiện kỹ thuật dự đoán. Ngoài ra, trong một số trường hợp, bộ mã hóa video có thể trực tiếp báo hiệu lỗi tái tạo. Tuy nhiên, việc lọc thành phần chéo theo các kỹ thuật trong tài liệu này cung cấp phương pháp để giảm lỗi tái tạo trong khi báo hiệu một lượng thông tin tương đối nhỏ. Nghĩa là, ví dụ như các kỹ thuật lọc thành phần chéo được mô tả trong tài liệu này có thể cung cấp phương pháp để giảm lỗi tái tạo ở bộ giải mã video mà vẫn hiệu quả hơn các kỹ thuật giảm lỗi tái tạo khác. Ví dụ: có thể cần ít bit hơn để báo hiệu dữ liệu bộ lọc so với báo hiệu thông tin dư có độ chính xác cao hơn cho thành phần.

Do đó, bộ lọc thành phần chéo 300 có thể hoạt động bằng cách lấy thành phần màu thứ nhất và một hoặc nhiều thành phần màu thứ hai làm đầu vào và cung cấp thành phần màu thứ nhất nâng cao làm đầu ra. Cần lưu ý rằng tuy các ví dụ trong tài liệu này được mô tả liên quan đến thành phần luma, Cb và Cr, các kỹ thuật được mô tả trong tài liệu này thường có thể áp dụng cho các định dạng video khác (ví dụ: RGB) và các loại thông tin video khác như tia hồng ngoại, độ sâu, sự chênh lệch hoặc các đặc điểm khác.

Phương trình sau đây cung cấp ví dụ về mô hình của bộ lọc lấy làm giá trị mẫu đầu vào từ nhiều thành phần và xuất giá trị mẫu đã lọc $f_i(x, y)$ và do đó, theo một ví dụ, bộ lọc thành phần chéo 300 có thể thực hiện quy trình lọc dựa trên phương trình.

$$f_i(x,y) = \sum_{(x_0,y_0)\in S_{i,0}} I_0(g(x,y,i,0)+x_0, h(x,y,i,0)+y_0)c_0(x_0,y_0) + \sum_{(x_1,y_1)\in S_{i,1}} I_1(g(x,y,i,1)+x_1, h(x,y,i,1)+y_1)c_1(x_1,y_1) + \sum_{(x_2,y_2)\in S_{i,2}} I_2(g(x,y,i,2)+x_2, h(x,y,i,2)+y_2)c_2(x_2,y_2) + I_i(x,y)$$

Trong đó,

$f_i(x,y)$:

là đầu ra của thành phần i tại vị trí mẫu (x, y);

$S_{i,0}$; $S_{i,1}$; và $S_{i,2}$: Xác định tập hợp vị trí giá trị mẫu liên quan đến điểm gốc trong các thành phần tương ứng 0, 1, 2;

$g(x, y, i,0)$ và $h(x, y, i, 0)$, $g(x, y, i, 1)$ và $h(x, y, i, 1)$ và $g(x, y, i, 2)$ và $h(x, y, i, 2)$: Xác định điểm gốc của giá đỡ dựa trên x, y, i và thành phần đầu vào. Hàm g(), h() có thể còn phụ thuộc vào định dạng sắc độ, loại vị trí sắc độ, độ bao phủ màu, hình dạng bộ lọc;

c_0 (x_0, y_0), c_1 (x_0, y_0) và c_2 (x_0, y_0): Là giá trị hệ số bộ lọc cho vùng giá đỡ đối với từng thành phần;

I_0, I_1, và I_2: Là Giá trị mẫu đầu vào từ mỗi thành phần; và

I_i (x, y): là Giá trị mẫu của thành phần i tại vị trí mẫu (x, y) trước khi lọc.

Do đó, theo các kỹ thuật trong tài liệu này, bộ lọc thành phần chéo 300 có thể được tạo cấu hình để giảm lỗi tái tạo của thành phần hiện tại bằng cách thêm bước tinh chỉnh vào giá trị mẫu đã tái tạo của thành phần hiện tại dựa trên chức năng lọc đã dẫn xuất có giá trị mẫu đã tái tạo của các thành phần khác làm đầu vào. Theo một ví dụ, giá trị mẫu đã tái tạo của các thành phần khác được sử dụng làm đầu vào có thể được gọi là giá đỡ bộ lọc. HÌNH 9A-9F là sơ đồ khái niệm minh họa các ví dụ về các mẫu giá đỡ có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này. Theo ví dụ được minh họa trong HÌNH 9A-9F, đối với mỗi loại vị trí sắc độ định dạng mẫu 4:2:0 được cung cấp trong JVET-N1001 và JVET-O2001, các mẫu giá đỡ luma được minh họa cho mẫu màu sẽ được lọc. Nghĩa là, có thể sử dụng các mẫu giá đỡ 5x5, 5x6, 6x5, và/hoặc 6x6. Cần lưu ý, theo các ví dụ trong HÌNH 9A-9F, giá đỡ luma được xác định là đối xứng xung quanh giá trị mẫu sắc độ. Cần lưu ý rằng theo các ví dụ khác, giá đỡ luma có thể trải qua sự dịch pha tùy vào loại vị trí sắc độ trước khi được đưa vào giai đoạn lọc không phụ thuộc vào loại vị trí sắc độ. Như được mô tả chi tiết hơn bên dưới, đối với mỗi mẫu giá đỡ, hệ số bộ lọc có thể được xác định và báo hiệu. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, đối với các mẫu sắc độ có trong khối video, vị trí tương đối của giá đỡ cho từng mẫu có thể dựa trên định dạng mẫu. Ví dụ: trong một ví dụ, theo các kỹ thuật trong tài liệu này, đối với định dạng mẫu 4:2:0, khi mẫu sắc độ trong khối video ở vị trí sắc độ (x_C, y_C) tương ứng với giá đỡ có điểm gốc tại vị trí luma (x_L,y_L), điểm gốc của giá đỡ cho mẫu sắc độ trong khối video ở vị trí sắc độ (x_C+m, y_C+n) có thể ở vị trí luma (x_C+2m, y_C+2n); đối với định dạng mẫu 4:2:2, khi mẫu sắc độ trong khối video ở vị trí sắc độ (x_C, y_C) tương ứng với giá đỡ có điểm gốc tại vị trí luma (x_L,y_L), điểm gốc của giá đỡ cho mẫu sắc độ trong khối video ở vị trí sắc độ (x_C+m, y_C+n) có thể ở vị trí luma (x_C+2m, y_C+n); và đối với định dạng mẫu 4:4:4, khi mẫu sắc độ trong khối video ở vị trí sắc độ (x_C, y_C) tương ứng với giá đỡ có điểm gốc tại vị trí luma (x_L,y_L), điểm gốc của giá đỡ cho mẫu sắc độ trong khối

video ở vị trí sắc độ (x_C+m, y_C+n) có thể ở vị trí luma (x_C+m, y_C+n). Cần lưu ý rằng theo ví dụ này, sự dịch chuyển vị trí mẫu sắc độ tương ứng với sự dịch chuyển ở vị trí điểm gốc luma của giá đỡ mà ở đó, tỷ lệ giữa hai dịch chuyển là dựa trên định dạng sắc độ.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc áp dụng lọc thành phần chéo có thể dựa trên các đặc tính của mẫu có trong vùng giá đỡ bộ lọc. Ví dụ: theo một ví dụ, các giá trị mẫu luma trong vùng giá đỡ có thể được phân tích và việc có áp dụng lọc thành phần chéo hay không có thể dựa trên phân tích này. Ví dụ: theo một ví dụ, phương sai và/hoặc độ lệch của các mẫu trong vùng giá đỡ có thể được tính toán, và nếu phương sai và/hoặc độ lệch có một số đặc điểm nhất định, ví dụ như vùng này phẳng (nghĩa là phương sai nhỏ hơn ngưỡng) thì có thể không áp dụng lọc thành phần chéo cho vùng này. Theo một ví dụ, lựa chọn bộ lọc thành phần chéo (bao gồm việc có áp dụng bộ lọc hay không, và nếu có thì bộ lọc nào được áp dụng) có thể dựa trên chỉ số bộ lọc phân loại luma của mẫu luma tương ứng với mẫu sắc độ đang được đánh giá. Theo một ví dụ, chỉ số bộ lọc phân loại cho mẫu luma có thể được dẫn xuất như được mô tả trong JVET-O2001. Theo một ví dụ, có thể không áp dụng lọc thành phần chéo khi chỉ số bộ lọc phân loại luma được xác định có trong tập hợp con các chỉ số bộ lọc phân loại luma. Như được mô tả chi tiết hơn bên dưới, các giá trị của chỉ báo kiểm soát vùng cục bộ và/hoặc các phần tử cú pháp có thể được sử dụng để biểu thị/xác định xem có áp dụng lọc thành phần chéo cho vùng hay không, và nếu có thì bộ lọc thành phần nào sẽ được áp dụng. Theo một ví dụ, việc áp dụng lọc thành phần chéo có thể dựa trên đặc tính của các mẫu có trong vùng giá đỡ bộ lọc và/hoặc các giá trị của chỉ báo kiểm soát vùng cục bộ và/hoặc phần tử cú pháp. Nghĩa là, ví dụ như cách phân tích các mẫu giá đỡ luma có thể dựa trên chỉ báo kiểm soát vùng cục bộ và/hoặc phần tử cú pháp (ví dụ: nếu chỉ báo == 0, tính toán/đánh giá phương sai, nếu không thì tính toán/đánh giá chỉ số bộ lọc phân loại luma). Ngoài ra, theo một ví dụ, lựa chọn bộ lọc dựa trên giá trị của phần tử cú pháp và đặc tính của mẫu giá đỡ luma. Ví dụ: giá trị 0 cho phần tử cú pháp có thể biểu thị không áp dụng lọc thành phần chéo cho một vùng, giá trị 1 cho phần tử cú pháp và phương sai của giá đỡ luma lớn hơn ngưỡng có thể biểu thị là áp dụng bộ lọc có tập hợp hệ số bộ lọc thứ nhất, giá trị 1 cho phần tử cú pháp và phương sai của giá đỡ luma không lớn hơn ngưỡng có thể biểu thị là áp dụng bộ lọc có tập hợp hệ số bộ lọc thứ hai, giá trị 2 cho phần tử cú pháp và phương sai của giá đỡ luma lớn hơn ngưỡng có thể biểu thị là áp dụng bộ lọc có tập hợp hệ số bộ lọc thứ ba, giá trị 2 cho phần tử cú pháp và phương sai của giá đỡ luma không lớn hơn ngưỡng có thể biểu thị là áp dụng bộ lọc có tập hợp hệ số bộ lọc thứ tư, v.v.

Phụ lục của Đơn xin cấp bằng sáng chế Hoa Kỳ tạm thời số 62/865,933 nộp ngày 24 tháng 6 năm 2019, được kết hợp vào tài liệu này bằng viện dẫn, đề xuất ví dụ về các tập hợp dữ liệu tương ứng với việc triển khai bộ lọc thành phần chéo được mô tả trong tài liệu này. Nghĩa là, trong Phụ lục, tập hợp dữ liệu orgBlock đại diện cho giá trị mẫu của khối thành phần ban đầu 32x32 U; tập hợp dữ liệu preFilteringBlock đại diện cho giá trị mẫu của khối thành phần đã tái tạo 32x32 U; tập hợp dữ liệu orgError đại diện cho lỗi tái tạo giữa khối thành phần ban đầu 32x32 U và khối thành phần đã tái tạo 32x32 U; tập hợp dữ liệu bestSupportY đại diện cho giá trị mẫu của khối thành phần 67x68 Y, cung cấp giá đỡ bộ lọc để lọc khối thành phần đã tái tạo 32x32 U; tập hợp dữ liệu bestSupportU đại diện cho giá trị mẫu của khối thành phần 36x36 U, cung cấp giá đỡ để lọc khối thành phần đã tái tạo 32x32 U; tập hợp dữ liệu bestSupportV đại diện cho giá trị mẫu của khối thành phần 36x36 V, cung cấp giá đỡ để lọc khối thành phần đã tái tạo 32x32 U; tập hợp dữ liệu coeffY đại diện cho hệ số bộ lọc trong bộ lọc 5x6 đối với giá trị mẫu của khối giá đỡ thành phần 67x68 Y; tập hợp dữ liệu coeffU đại diện cho hệ số bộ lọc trong bộ lọc 5x5 đối với giá trị mẫu của khối giá đỡ thành phần 36x36 U; tập hợp dữ liệu coeffV đại diện cho hệ số bộ lọc trong bộ lọc 5x5 đối với giá trị mẫu của khối giá đỡ thành phần 36x36 V; tập hợp dữ liệu bestOutput đại diện cho giá trị mẫu của khối thành phần tái tạo được lọc 32x32 U; tập hợp dữ liệu bestError đại diện cho lỗi giữa khối thành phần ban đầu 32x32 U và khối thành phần tái tạo được lọc 32x32 U; tập hợp dữ liệu signedimprovement là bằng Abs(orgError) - Abs(bestError) và đại diện cho sự thay đổi trong lỗi tái tạo do quá trình lọc; và tập hợp dữ liệu cải thiện dương đại diện cho giá trị mẫu được tái tạo, trong đó giảm được lỗi tái tạo do quá trình lọc. Do đó, theo các kỹ thuật trong tài liệu này, có thể giảm lỗi tái tạo cho một hoặc nhiều hoặc phần lớn các mẫu bằng cách áp dụng bộ lọc thành phần chéo. Cần lưu ý rằng đối với loại nội dung video cụ thể, số lượng lỗi tái tạo được cải thiện theo mối quan hệ toán học có thể có nhiều kết quả khác nhau về cách cải thiện chất lượng hình ảnh cảm nhận được của video. Nghĩa là, ví dụ như giá trị tương đối nhỏ của signedimprovement có thể dẫn đến những cải tiến tương đối đáng kể là chất lượng hình ảnh.

Như được mô tả ở trên, bộ lọc thành phần chéo 300 thường có thể hoạt động bằng cách lấy thành phần màu thứ nhất và một hoặc nhiều thành phần màu thứ hai làm đầu vào và cung cấp thành phần màu thứ nhất nâng cao làm đầu ra. Nghĩa là, quy trình lọc do bộ lọc thành phần chéo 300 thực hiện có thể lấy giá trị mẫu luma đầu vào mà có thể được sử dụng để dự đoán sự chênh lệch giữa giá trị mẫu sắc độ tương ứng ban đầu và giá trị mẫu sắc độ đầu ra đã tinh chỉnh dựa trên dự đoán. Tham khảo lại ví dụ được minh họa trong HÌNH 8, HÌNH

10 là sơ đồ khái niệm minh họa ví dụ về việc giảm lỗi tái tạo bằng cách lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này. HÌNH 10 cung cấp ví dụ về vị trí giảm lỗi tái tạo bằng cách lấy giá trị trung bình của mẫu giá đỡ, và nếu giá trị trung bình lớn hơn 90 thì giá trị trung bình chia cho 10 sẽ được thêm vào mẫu tái tạo; và nếu giá trị trung bình không lớn hơn 90, giá trị trung bình chia cho 10 sẽ bị trừ khỏi mẫu tái tạo. Nghĩa là, theo ví dụ, bộ lọc dự đoán thường được mô tả như sau: nếu giá trị trung bình giá đỡ lớn hơn threshold1 thì thêm weight1 nhân với giá trị trung bình giá đỡ; nếu không thì thêm weight2 nhân với giá trị trung bình giá đỡ. Như đã minh họa trong ví dụ được minh họa ở HÌNH 10, lỗi tái tạo sắc độ sau lọc giúp giảm lỗi tái tạo. Do đó, theo các kỹ thuật trong tài liệu này, bước lọc thành phần chéo có thể giúp giảm lỗi tái tạo theo bộ lọc thành phần chéo được xác định theo hàm logic, ngưỡng, trọng số và những thứ tương tự.

Như được mô tả ở trên, JVET-N1001 và JVET-O2001 bao gồm bộ lọc tách khối, SAO và ALF, các kỹ thuật bộ lọc thành phần chéo được mô tả trong tài liệu này có thể được thực hiện ở nhiều điểm khác nhau trong chuỗi bộ lọc. Nghĩa là, ví dụ như ở các giai đoạn khác nhau của quy trình lọc trong vòng. HÌNH 11A-11D là các sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này. Theo HÌNH 11A-11D, bộ lọc SAO luma 402 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc SAO trên giá trị mẫu luma, Y, ví dụ như lọc SAO như được cung cấp trong JVET-N1001 hoặc JVET-O2001; Bộ lọc SAO Cb 404 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc SAO trên giá trị mẫu Cb sắc độ, ví dụ như lọc SAO như được cung cấp trong JVET-N1001 hoặc JVET-O2001; Bộ lọc SAO Cb 406 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc SAO trên giá trị mẫu Cb, ví dụ như lọc SAO như được cung cấp trong JVET-N1001 hoặc JVET-O2001; bộ lọc ALF luma 408 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc ALF trên giá trị mẫu luma, Y, ví dụ như lọc ALF như được cung cấp trong JVET-N1001 hoặc JVET-O2001; bộ lọc ALF sắc độ 410 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc ALF trên giá trị mẫu sắc độ, ví dụ như lọc ALF như được cung cấp trong JVET-N1001 hoặc JVET-O2001; bộ lọc tách khối luma 416 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc tách khối trên giá trị mẫu luma, Y, ví dụ như lọc tách khối như được cung cấp trong JVET-N1001 hoặc JVET-O2001; bộ lọc tách khối Cb 418 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc tách khối trên giá trị mẫu Cb, ví dụ như lọc tách khối như được cung cấp trong JVET-N1001 hoặc JVET-O2001; và bộ lọc tách khối Cr 420 đại diện cho bộ lọc được tạo cấu hình để thực hiện lọc tách khối trên

giá trị mẫu Cr, ví dụ như lọc tách khối như được cung cấp trong JVET-N1001 hoặc JVET-O2001. Ngoài ra, trong HÌNH 11A-11D, bộ lọc thành phần chéo Cb 412 đại diện cho ví dụ về bộ lọc thành phần chéo được tạo cấu hình để tạo sự tinh chỉnh Cb, ΔCb, theo một hoặc nhiều kỹ thuật được mô tả trong tài liệu này; và bộ lọc thành phần chéo Cr 414 đại diện cho ví dụ về bộ lọc thành phần chéo được tạocấu hình để tạo sự tinh chỉnh Cr, ΔCr, theo một hoặc nhiều kỹ thuật được mô tả trong tài liệu này. Do đó, như được minh họa trong HÌNH 11A-11D, quy trình lọc thành phần chéo theo các kỹ thuật trong tài liệu này có thể được áp dụng là các điểm khác nhau trong chuỗi lọc. Nghĩa là, đầu vào lọc thành phần chéo có thể được thu tại các điểm khác nhau trong chuỗi lọc và sự tinh chỉnh lọc thành phần chéo có thể được xuất ra tại các điểm khác nhau trong chuỗi lọc. Cần lưu ý rằng theo ví dụ được minh họa trong HÌNH 11B, đầu vào của bước tách khối luma được sử dụng làm đầu vào cho bước lọc mà có thể có lợi thế là giảm yêu cầu bộ đệm dòng. Cần lưu ý rằng theo ví dụ được minh họa trong HÌNH 11C, đầu ra của bước tách khối luma được sử dụng làm đầu vào cho bước lọc mà có thể có lợi thế là giảm yêu cầu bộ đệm dòng, cải thiện một chút hiệu quả mã hóa. Cần lưu ý rằng theo ví dụ được minh họa trong HÌNH 11D, đầu ra của luma ALF được sử dụng làm đầu vào cho bước lọc, mà có thể có lợi thế là cải thiện hiệu quả mã hóa.

Ngoài ra, các kỹ thuật lọc thành phần chéo được mô tả trong tài liệu này có thể bao gồm bước thực hiện phép toán cắt ở các điểm khác nhau trong chuỗi bộ lọc. Nghĩa là, ví dụ như ở các giai đoạn khác nhau của bộ lọc trong vòng. HÌNH 12A-12B là các sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này. Trong HÌNH 12A-12B, các phần tử đánh số thông thường được mô tả ở trên liên quan đến HÌNH 11A-11D và bộ cắt 422A-422D có thể được tạo cấu hình để thực hiện chức năng cắt dựa trên độ sâu bit đầu ra của thành phần tương ứng, ví dụ: Clip3(0, $2^{BitDepthC}$ - 1, *). Cần lưu ý rằng bộ cắt 422A-422D có thể được kích hoạt có chọn lọc dựa trên việc liệu các loại lọc cụ thể có được thực hiện hay không.

HÌNH 13A-13B là các sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này. HÌNH 13A-13B minh họa thêm rằng bước lọc thành phần chéo theo các kỹ thuật trong tài liệu này có thể được áp dụng như các điểm khác nhau trong chuỗi lọc. Theo HÌNH 13A-13C, các phần tử đánh số thông thường được mô tả ở trên.

Ngoài ra, cần lưu ý rằng theo một số trường hợp, có thể có nhiều hơn 3 thành phần dữ liệu video, ví dụ: YUV + độ sâu. Các kỹ thuật lọc thành phần chéo được mô tả trong tài

liệu này có thể áp dụng chung cho những trường hợp này. Theo một số trường hợp, việc xử lý trước giá trị mẫu đầu vào từ mỗi thành phần có thể được thực hiện trước phép toán lọc. Ví dụ, có thể tách các giá trị mẫu đầu vào. Ngoài ra, theo một ví dụ, khoảng cắt có thể khác nhau đối với từng hệ số và có thể được báo hiệu trong chuỗi bit. Cần lưu ý rằng theo một số ví dụ, phương trình sau cung cấp tùy chọn để xử lý trước các giá trị mẫu đầu vào:

$$I_j\,(g(x, y, i, j) + x_j, h(x, y, i, j) + y_j)$$

$$= \min(a, \max(b, I'_j(g(x, y, i, j) + x_j, h(x, y, i, j) + y_j) - derivedValue))$$

Ngoài ra, một tùy chọn khác để xử lý trước các giá trị mẫu đầu vào có thể như sau:

$$I_j\,(g(x, y, i, j) + x_j, h(x, y, i, j) + y_j)$$

$$= derivedValue + \min(a, \max(b, I'_j\,(g(x, y, i, j) + x_j, h\,(x, y, i, j) + y_j) - derivedValue))$$

Trong đó,

I'j(u,v): giá trị mẫu tại vị trí (u,v) trước khi xử lý

derivedValue : là giá trị được suy ra từ một tập hợp con của các giá trị trong I'$_j$(*,*), ví dụ: (a) $I_j(g(x, y, i, j) + 0, h\,(x, y, i, j) + 0)$, tại điểm gốc (b) $\frac{\left(I_j\left(g(x,y,i,j)+0,h(x,y,i,j)+0\right)+I_j\left(g(x,y,i,j)+1,h(x,y,i,j)+1\right)\right)}{2}$, quanh điểm gốc

b: là giá trị thu được trong chuỗi bit/được suy ra từ dữ liệu thu được trong chuỗi bit/được dẫn xuất từ tập hợp con các giá trị trong I'$_j$(*,*); và

b: là giá trị thu được trong chuỗi bit / được suy ra từ dữ liệu thu được trong chuỗi bit / được dẫn xuất từ tập hợp con các giá trị trong I'$_j$(*,*)

Theo một ví dụ, b có thể được suy ra từ a (ví dụ: b= –a) để giảm lượng tín hiệu cần thiết.

Ngoài ra, theo một ví dụ, sự suy rộng đầu vào được sử dụng trong phép toán lọc thành phần chéo có thể như sau:

$$f_i(x,y) = \sum_{\substack{(x_0,y_0)\in S_{i,0,0}\\(x_1,y_1)\in S_{i,1,0}\\(x_2,y_2)\in S_{i,2,0}\\(x_{c0},y_{c0})\in S_{c0}}} G_0\big(I_0(g(x,y,i,0)+x_0, h(x,y,i,0)+y_0), I_1(g(x,y,i,1)+x_1, h(x,y,i,1)$$

$$+ y_1), I_2(g(x,y,i,2)+x_2, h(x,y,i,1)+y_2)\big)c_0(x_{c0},y_{c0})$$

$$+ \sum_{\substack{(x_0,y_0)\in S_{i,0,1}\\(x_1,y_1)\in S_{i,1,1}\\(x_2,y_2)\in S_{i,2,1}\\(x_{c1},y_{c1})\in S_{c1}}} G_1\big(I_0(g(x,y,i,0)+x_0, h(x,y,i,0)+y_0), I_1(g(x,y,i,1)+x_1, h(x,y,i,1)$$

$$+ y_1), I_2(g(x,y,i,2)+x_2, h(x,y,i,1)+y_2)\big)c_1(x_{c1},y_{c1}) \ldots$$

$$+ \sum_{\substack{(x_0,y_0)\in S_{i,0,n}\\(x_1,y_1)\in S_{i,1,n}\\(x_2,y_2)\in S_{i,2,n}\\(x_{cn},y_{cn})\in S_{cn}}} G_n\big(I_0(g(x,y,i,0)+x_0, h(x,y,i,0)+y_0), I_1(g(x,y,i,1)+x_1, h(x,y,i,1)$$

$$+ y_1), I_2(g(x,y,i,2)+x_2, h(x,y,i,1)+y_2)\big)c_n(x_{cn},y_{cn}) + I_i(x,y)$$

Trong đó,

$G_i()$ là hàm được sử dụng để kết hợp các giá trị mẫu từ các thành phần và thu được giá trị dẫn xuất tương ứng với mỗi giá trị hệ số có chỉ số (x_{cj}, y_{cj}). Hàm $G_i()$ có thể phụ thuộc vào định dạng sắc độ, loại vị trí sắc độ, độ bao phủ màu, hình dạng bộ lọc

Theo một ví dụ, bước lọc thành phần chéo có thể được thực hiện theo như sau: Xác định vùng giá đỡ cho luma; Đối với 4:2:0, Tăng gấp đôi tần số lấy mẫu thành phần sắc độ sẽ được sử dụng làm đầu vào; Lấy giá trị giá đỡ được sử dụng cho thành phần sắc độ tương ứng trừ đi giá trị đã dẫn xuất (ví dụ 512 đối với sắc độ 10 bit, hoặc giá trị trung bình cục bộ); Sau đó, lấy tích số từng mẫu của giá trị mẫu luma và giá trị mẫu sắc độ tương ứng với vùng giá đỡ đã xác định; và Sử dụng tích số này làm một trong những yếu tố đầu vào cho phép toán lọc.

Ngoài ra, cần lưu ý rằng theo một số ví dụ, các kỹ thuật lọc thành phần chéo được mô tả trong tài liệu này có thể được thực hiện dựa trên dự đoán hoặc phần dư. Theo một ví dụ, nếu kỹ thuật mã hóa trường được sử dụng thay vì mã hóa lũy tiến thì đối với mẫu giá đỡ luma: theo một ví dụ, giá trị mẫu từ một trong các trường luma tương ứng có thể được sử dụng và theo một ví dụ khác, giá trị mẫu từ cả hai trường luma có thể được sử dụng.

Như đã mô tả ở trên, đối với từng mẫu giá đỡ, hệ số bộ lọc có thể được xác định và báo hiệu. Nghĩa là, ví dụ như đối với các hệ số bộ lọc 5x5, 5x6, 6x6 và/hoặc 6x6 có thể được báo hiệu. HÌNH 14A-14C minh họa ví dụ về hệ số bộ lọc báo hiệu cho bộ lọc 5x5. HÌNH 14D-14F minh họa ví dụ về hệ số bộ lọc báo hiệu cho bộ lọc 5x6 (và tương tự, 6x5).

HÌNH 15A-15D minh họa ví dụ về hệ số bộ lọc báo hiệu cho 6x6. Theo mỗi HÌNH 14A-15D, hệ số bộ lọc tương ứng đối với bộ lọc được biểu thị bằng C_N. Do đó, trong các trường hợp, cùng một giá trị C_N được cung cấp nhiều vị trí của cùng một bộ lọc thì hệ số bộ lọc sẽ giống nhau, nghĩa là dùng chung. Theo cách này, số lượng hệ số bộ lọc được báo hiệu cho bộ lọc bị giảm. Ví dụ: trong HÌNH 14D, 14 hệ số bộ lọc được báo hiệu cho 18 vị trí giá đỡ.

Theo một ví dụ, có thể mong muốn giới hạn số lượng bộ đệm dòng trong cấu trúc mà ở đó các mẫu được xử lý theo từng CTU. Nghĩa là, ví dụ như đường biên dòng ảo cung cấp vị trí cho từng CTU, các mẫu ở trên VB ngang có thể được xử lý trước khi có CTU thấp hơn, nhưng các mẫu ở dưới VB ngang không thể xử lý cho đến khi có CTU thấp hơn. JVET-N1001 và JVET-O2001 xác định đường biên dòng ảo nằm ngang (VB) cho ALF luma và SAO luma. Theo các kỹ thuật trong tài liệu này, VB này có thể được tái sử dụng cho bộ lọc luma-đầu vào-sắc độ-đầu ra được xác định trong tài liệu này. Ngoài ra, VB dọc có thể được tái sử dụng cho bộ lọc luma-đầu vào-sắc độ-đầu ra được xác định trong tài liệu này và/hoặc tập hợp con của VB có thể được tái sử dụng cho bộ lọc luma-đầu vào-sắc độ-đầu ra được xác định trong tài liệu này. Ngoài ra, có hai trường hợp được xác định, trong đó các mẫu giá đỡ trong thành phần luma có thể được dẫn xuất/sửa đổi: khi mẫu luma được xác định trước (tương ứng với mẫu sắc độ được giải mã, ví dụ như dựa trên loại vị trí sắc độ) ở trên VB và giá đỡ trải dài trên toàn bộ VB; và khi mẫu luma được xác định trước (tương ứng với mẫu sắc độ đang được giải mã, ví dụ như dựa trên loại vị trí sắc độ) ở dưới VB và giá đỡ trải dài trên toàn bộ VB. Theo ví dụ, mẫu được xác định trước là mẫu ở vị trí tương ứng với hệ số C6 cho giá đỡ luma 5x6 được minh họa trong HÌNH 14D. HÌNH 16A-16D là các sơ đồ khái niệm minh họa ví dụ về bộ đệm dòng ảo mà có thể được sử dụng để lọc thành phần chéo theo một hoặc nhiều kỹ thuật của sáng chế này. Theo HÌNH 16A, các mẫu ở dưới VB ngang thu được bằng cách sao chép các mẫu ở trên và gần nhất với đường biên dòng ảo và trong cùng một cột. Theo HÌNH 16B, các mẫu ở dưới VB nằm ngang thu được bằng cách sao chép các mẫu ở dưới và gần nhất với đường biên dòng ảo và trong cùng một cột. Theo ví dụ, VB luma là bốn mẫu từ đường biên ngang CTU. Theo một ví dụ, mỗi CTU, SAO và ALF có thể xử lý mẫu bên trái của VB dọc trước khi có CTU bên phải, nhưng không thể xử lý mẫu bên phải của VB dọc cho đến khi có CTU bên phải. Ví dụ về các sửa đổi khi giá đỡ kéo dài qua đường biên ảo dọc (VB) được minh họa trong các HÌNH 16C-16D. Theo HÌNH 16C, các mẫu ở bên phải của VB dọc thu được bằng cách sao chép các mẫu ở bên trái và gần nhất với đường biên dòng ảo và trong cùng một hàng. Theo HÌNH 16D, các mẫu ở bên trái của VB dọc thu được bằng cách sao chép các mẫu

ở bên phải và gần nhất với đường biên dòng ảo và trong cùng một hàng. Theo ví dụ, VB luma là bốn mẫu từ đường biên dọc CTU. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, đối với mẫu tạo cho VB ngang, người ta có thể coi trục tung và trục hoành đi qua tâm của vùng giá đỡ. Có thể thu được các mẫu đang được sao chép bằng cách sao chép các mẫu trong cột có cùng khoảng cách với trục tung, nhưng ở phía đối diện của trục tung. Theo một ví dụ, mẫu đang được sao chép có thể được sao chép từ hàng có cùng khoảng cách với trục hoành nhưng ở phía đối diện. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, đối với mẫu tạo cho VB dọc, người ta có thể coi trục tung và trục hoành đi qua tâm của vùng giá đỡ. Có thể thu được các mẫu đang được sao chép bằng cách sao chép các mẫu trong hàng có cùng khoảng cách với trục tung, nhưng ở phía đối diện của trục hoành. Theo một ví dụ, mẫu đang được sao chép có thể được sao chép từ cột có cùng khoảng cách với trục tung nhưng ở phía đối diện. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, đối với các mẫu tạo cho VB, có thể thu được các mẫu bằng cách chèn đối xứng. Nghĩa là, đối với mẫu VB ngang có thể được sao chép từ cùng một cột và ở cùng một khoảng cách mẫu từ VB và đối với VB dọc, các mẫu có thể được sao chép từ cùng một hàng và ở cùng một khoảng cách mẫu từ VB. Nghĩa là, giá trị mẫu được phản chiếu về VB. Cần lưu ý rằng phần 8.8.5.2 Quy trình lọc khối cây mã hóa đối với các mẫu luma của JVET-O2001 cung cấp kiểu chèn cho các mẫu luma trên toàn bộ đường biên ảo để sử dụng đối với quy trình ALF. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, kiểu chèn tương tự có thể được sử dụng để lọc thành phần chéo. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, khi một hoặc nhiều mẫu giá đỡ không khả dụng, ví dụ: do VB, tính năng lọc thành phần chéo có thể bị vô hiệu hóa.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, bước lọc thành phần chéo bao gồm bước chia tỷ lệ đầu ra về lọc thành phần chéo trước khi cộng đầu ra này vào đầu ra ALF sắc độ tương ứng. Nghĩa là, phép toán chia tỷ lệ có thể được sử dụng để chuyển đổi hệ số bộ lọc thành số nguyên, ví dụ như sau:

```
for( int i = 0; i < numCoeff; i++)
    {
      ký hiệu int = filterCoeff[i] > 0 ? 1 : -1;
      filterCoeffQuant[i] = int( filterCoeff[i] * sign * hệ số + 0,5 ) * sign;
    }
```

trong đó, theo một ví dụ, hệ số = $2^{BitDepthC}$, theo một hệ số điển hình khác = $2^{(BitDepthC - 1)}$), theo hệ số điển hình khác = $2^{(8-1)}$

Theo một ví dụ, hệ số tỷ lệ có thể được sử dụng để điều chỉnh đầu ra của lọc thành phần chéo như sau:

$$f'_i(x,y) = \sum_{(x_0,y_0)\in S_{i,0}} I_0(g(x,y,i,0)+x_0, h(x,y,i,0)+y_0) filterCoeffQuant[MapTol(x_0,y_0)] f_i(x,y)$$
$$= f'_i(x,y) \div \text{hệ số} + I_i(x,y)$$

Cần lưu ý rằng nếu hệ số = 2^x thì điều này tương ứng với số nguyên dịch chuyển sang phải làm tròn

$$(f_i(x,y) + 2^{(x-1)}) >> x$$

Như được mô tả ở trên, dữ liệu bộ lọc xác định bộ lọc đã dẫn xuất có thể được báo hiệu đến bộ giải mã video. Theo một ví dụ, có thể có ba khía cạnh chính của dữ liệu bộ lọc báo hiệu: bật/tắt bộ lọc; kiểm soát cục bộ công cụ, ví dụ như kích hoạt công cụ ở một số vùng không gian nhưng không phải những vùng khác; và báo hiệu các bộ lọc cụ thể. Theo một ví dụ, tập hợp thông số, chẳng hạn như Tập hợp thông số chuỗi có thể bao gồm chỉ báo có điều kiện để kích hoạt/vô hiệu hóa bộ lọc. Theo một ví dụ, chỉ báo có thể biểu thị liệu một trong các bộ lọc có được kích hoạt hay không, ví dụ như ALF và bộ lọc thành phần chéo. Theo một ví dụ, có thể sử dụng tín hiệu hệ số bộ lọc cấp độ mảnh. Theo một ví dụ khác, kim báo đến APS chứa dữ liệu hệ số bộ lọc tương ứng có thể được gửi trong phần đầu mảnh. Bảng 3-4 minh họa ví dụ về cú pháp mà có thể được đưa vào phần đầu mảnh để báo hiệu hệ số bộ lọc theo ví dụ này.

slice_header( ) {	**Bộ mô tả**
...	
if(sps_cross_component_alf_enabled flag) {	
slice_cross_component_alf_cb_enabled_flag	u(1)
if(slice_cross_component_alf_cb_enabled_flag) {	

slice_cross_component_alf_cb_aps_id	U(5)
slice_cross_component_alf_cb_log2_control_size_minus4	ue(v)
}	
slice_cross_component_alf_cr_enabled flag	u(1)
if(slice_cross_component_alf_cr_enabled_flag) {	
slice_cross_component_alf_cr_aps_id	u(5)
slice_cross_component_alf_cr_log2_control_size_minus4	ue(v)
}	
}	
...	
}	

Bảng 3

alf_data(adaptation_parameter_set_id) {	**Bộ mô tả**
if (sps_alf_enabled_flag) {	
alf_luma_filter_signal_flag	u(1)
if (chroma_format_idc != 0) // đơn sắc	
alf_chroma_filter_signal_flag	u(1)
}	
if (sps_cross_component_alf_enabled_flag) {	
alf_cross_component_cb_filter_signal_flag	u(1)
alf_cross_component_cr_filter_signal_flag	u(1)
}	
if(alf_luma_filter_signal_flag) {	
...	
}	
if(alf_chroma_filter_signal_flag) {	
...	
}	
if(alf_cross_component_cb_filter_signal_flag) {	
alf_cross_component_cb_min_eg_order_minus1	ue(v)
for(i = 0; i < 3; i++)	
alf_cross_component_cb_eg_order_increase_flag[i]	u(1)
for (j = 0; j < 14; j++) {	

alf_cross_component_cb_coeff_delta_abs[j]	uek(v)
if(alf_cross_component_cb_coeff_delta_abs[j])	
alf_cross_component_cb_coeff_delta_sign[j]	u(1)
}	
}	
if(alf_cross_component_cr_filter_signal_flag) {	
alf_cross_component_cr_min_eg_order_nimus1	ue(v)
for(i = 0; i < 3; i++)	
alf_cross_component_cr_eg_order_increase_flag[i]	u(1)
for (j = 0; j < 14;j++) {	
alf_cross_component_cr_coeff_delta_abs[j]	uek(v)
if(alf_cross_component_cr_coeff_delta_abs[j])	
alf_cross_component_cr_coeff_delta_sign[j]	u(1)
}	
}	
}	

Bảng 4

Đối với Bảng 3-4, theo một ví dụ, ngữ nghĩa học có thể dựa trên điểm sau:

slice_cross_component_alf_cb_enabled_flag bằng 0 xác định rằng bộ lọc Cb thành phần chéo không được áp dụng cho thành phần màu Cb. slice_cross_component_alf_cb_enabled_flag bằng 1 biểu thị rằng bộ lọc vòng thích ứng thành phần chéo được áp dụng cho thành phần màu Cb.

slice_cross_component_alf_cr_enabled_flag bằng 0 xác định rằng bộ lọc Cr thành phần chéo không được áp dụng cho thành phần màu Cr. slice_cross_component_alf_cb_enabled_flag bằng 1 biểu thị rằng bộ lọc vòng thích ứng thành phần chéo được áp dụng cho thành phần màu Cr.

slice_cross_component_alf_cb_aps_id xác định adaptation_parameter_set_id mà thành phần màu Cb của mảnh đề cập đến. Khi slice_cross_component_alf_cb_aps_id không hiện diện, nó được suy ra bằng slice_alf_aps_id_luma[0]. TemporalId của bộ ALF APS NAL có adaptation_parameter_set_id bằng slice_cross_component_alf_cb_aps_id phải nhỏ hơn hoặc bằng với TemporalId của bộ NAL mảnh được mã hóa.

slice_cross_component_alf_cr_aps_id xác định adaptation_parameter_set_id mà thành phần màu Cb của mảnh được đề cập đến Khi slice_cross_component_alf_cr_aps_id không hiện diện, nó sẽ được suy ra bằng slice_alf_aps_id_luma[0]. TemporalId của bộ ALF APS NAL có adaptation_parameter_set_id equal to slice_cross_component_alf_cr_aps_id phải nhỏ hơn hoặc bằng với TemporalId của bộ NAL mảnh được mã hóa.

slice_cross_component_alf_cb_log2_control_size_minus4 xác định giá trị của các kích thước khối vuông với số lượng mẫu như sau:

$$AlfCCSamplesCbW = AlfCCSamplesCbH = 2^{(\ slice\_cross\_component\_alf\_cb\_log2\_control\_size\_minus4 + 4\ )}$$

slice_cross_component_alf_cb_log2_control_size_minus4 sẽ nằm trong khoảng từ 0 đến 3, bao gồm cả hai khoảng này.

slice_cross_component_alf_cr_log2_control_size_minus4 xác định giá trị của các kích thước khối vuông với số lượng mẫu như sau:

$$AlfCCSamplesCrW = AlfCCSamplesCrH = 2^{(\ slice\_cross\_component\_alf\_cr\_log2\_control\_size\_minus4 + 4\ )}$$

slice_cross_component_alf_cr_log2_control_size_minus4 sẽ nằm trong khoảng từ 0 đến 3, bao gồm cả hai khoảng này.

alf_luma_filter_signal_flag bằng 1 cho thấy rằng tập hợp bộ lọc luma được báo hiệu. alf_luma_filter_signal_flag bằng 0 cho thấy rằng tập hợp bộ lọc luma không được báo hiệu. Khi alf_luma_filter_signal_flag không hiện diện, nó sẽ được suy ra là bằng 0.

alf_chroma_filter_signal_flag bằng 1 cho thấy rằng bộ lọc sắc độ được báo hiệu. alf_chroma_filter_signal_flag bằng 0 cho thấy rằng bộ lọc sắc độ không được báo hiệu. Khi alf_chroma_filter_signal_flag không hiện diện, nó sẽ được suy ra là bằng 0.

alf_cross_component_cb_filter_signal_flag bằng 1 xác định rằng tập hợp bộ lọc Cb thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 xác định rằng tập hợp bộ lọc Cb thành phần chéo không được báo hiệu. Khi alf_cross_component_cb_filter_signal_flag không hiện diện, nó sẽ được suy ra là bằng 0.

alf_cross_component_cr_filter_signal_flag bằng 1 xác định rằng tập hợp bộ lọc Cr thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 xác định rằng tập hợp bộ lọc Cr thành phần chéo không được báo hiệu. Khi alf_cross_component_cr_filter_signal_flagkhông hiện diện, nó sẽ được suy ra bằng 0.

alf_cross_component_cb_min_eg_order_minus1 cộng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cb thành phần chéo. Giá trị alf_cross_component_cb_min_eg_order_minus1 sẽ trong khoảng từ 0 đến 9, bao gồm cả hai khoảng này. Cần lưu ý rằng theo một số ví dụ, khoảng này có thể bị thay đổi.

alf_cross_component_cr_min_eg_order_minus1 cộng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cr thành phần chéo. Giá trị alf_cross_component_cb_min_eg_order_minus1 sẽ trong khoảng từ 0 đến 9, bao gồm cả hai khoảng này. Cần lưu ý rằng theo một số ví dụ, khoảng này có thể bị thay đổi.

alf_cross_component_cb_eg_order_increase_flag[i] bằng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cb thành phần chéo được tăng thêm 1. alf_cross_component_cb_eg_order_increase_flag[i] bằng 0 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cb thành phần chéo không được tăng thêm 1.

Bậc expGoOrderCb[i] của mã Golomb hàm số mũ được sử dụng để giải mã các giá trị của alf_cross_component_cb_coeff_delta_abs[j] được dẫn xuất như sau:

expGoOrderCb[i] = (i = = 0 ? alf_cross_component_cb_min_eg_order_minus1 + 1 :
expGoOrderCb [i – 1]) + alf_cross_component_cb_eg_order_increase_flag[i]

alf_cross_component_cr_eg_order_increase_flag[i] bằng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cr thành phần chéo được tăng thêm 1. alf_cross_component_cr_eg_order_increase_flag[i] bằng 0 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cr thành phần chéo không được tăng thêm 1.

Bậc expGoOrderCr[i] của mã Golomb hàm số mũ được sử dụng để giải mã các giá trị của alf_cross_component_cb_coeff_delta_abs[j] được dẫn xuất như sau:

expGoOrderCr[i] = (i = = 0 ? alf_cross_component_cr_min_eg_order_minus1 + 1 : expGoOrderCr[i – 1]) + alf_cross_component_cr_eg_order_increase_flag[i]

alf_cross_component_cb_coeff_delta_abs[j] xác định trị số tuyệt đối của delta hệ số thứ j của bộ lọc Cb thành phần chéo được báo hiệu. Khi alf_luma_cross_component_cb_coeff_delta_abs[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

golombOrderIdxCb[ ] = {0,2,2,2,1,2,2,2,2,2,2,1,2,1} [đây có thể là Hệ số phân loại thành 3 loại, mỗi loại sử dụng cùng một bậc k mã Golomb hàm số mũ]

k= expGoOrderCb[golombOrderIdxCb[j]]

alf_cross_component_cr_coeff_delta_abs[j] xác định trị số tuyệt đối của delta hệ số thứ j của bộ lọc Cr thành phần chéo được báo hiệu. Khi alf_luma_cross_component_cr_coeff_delta_abs[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

golombOrderIdxCr[ ] = {0,1,2,1,0,1,2,2,2,2,2,1,2,1} [đây có thể là Hệ số phân loại thành 3 loại, mỗi loại sử dụng cùng một mã Golomb hàm số mũ bậc k]

k = expGoOrderCr[golombOrderIdxCr[j]]

alf_cross_component_cb_coeff_sign[j] cho thấy tín hiệu của hệ số bộ lọc Cb thành phần chéo thứ j như sau:

- Nếu alf_cross_component_cb_coeff_sign[j] bằng 0, hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị dương.

- Nếu không (alf_cross_component_cb_coeff_sign[j] bằng 1), hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_cb_coeff_sign[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cb thành phần chéo $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id] với các phần tử $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][j], với j = 0..13 được dẫn xuất như sau:

$$AlfCCCoeff_{Cb}[\ adaptation\_parameter\_set\_id\ ][\ j\ ] = alf\_cross\_component\_cb\_coeff\_abs[\ j\ ] * (1 - 2 * alf\_cross\_component\_cb\_coeff\_sign[\ j\ ])$$

Cần có sự tương thích chuỗi bit mà các giá trị của $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][j] với j = 0..13 sẽ nằm trong khoảng từ $-2^{10} - 1$ đến $2^{10} - 1$, bao gồm cả hai khoảng này.

alf_cross_component_cr_coeff_sign[j] cho thấy tín hiệu của hệ số bộ lọc Cr thành phần chéo thứ j như sau:

- Nếu alf_cross_component_cr_coeff_sign[j] bằng 0, hệ số bộ lọc Cr thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_cr_coeff_sign[j] bằng 1), hệ số bộ lọc Cr thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_cr_coeff_sign[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cr thành phần chéo $AifCCCoeff_{Cr}$[adaptation_parameter_set_id] có các phần tử $AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][j], với j = 0..13 được dẫn xuất như sau:

$$AlfCCCoeff_{Cr}[\ adaptation\_parameter\_set\_id\ ][\ j\ ] = alf\_cross\_component\_cr\_coeff\_abs[\ j\ ] * (1 - 2 * alf\_cross\_component\_cr\_coeff\_sign[\ j\ ])$$

Cần có sự tương thích chuỗi bit mà các giá trị của AlfCCCoeff$_{Cr}$[adaptation_parameter_set_id][j] với j = 0..13 sẽ nằm trong khoảng từ $-2^{10} - 1$ đến $2^{10} - 1$, bao gồm cả hai khoảng này.

Cần lưu ý rằng khoảng từ $-2^{10} - 1$ đến $2^{10} - 1$ có thể thay đổi. Cần lưu ý rằng theo một số ví dụ, khoảng này có thể phụ thuộc vào độ sâu bit của luma (ánh sáng)/sắc độ (chroma) hoặc tập hợp con của chúng.

Đối với Bảng 4, theo một ví dụ, cấu trúc cú pháp alf_data() được cung cấp trong Bảng 5A có thể được sử dụng.

alf_data(adaptation_parameter_set_id) {	**Bộ mô tả**
...	
if (sps_cross_component_alf_enabled_flag) {	
alf_cross_component_cb_filter_signal_flag	u(1)
alf_cross_component_cr_filter_signal_flag	u(1)
}	
if(alf_luma_filter_signal_flag) {	
...	
}	
if(alf_chroma_filter_signal_flag) {	
...	
}	
if(alf_cross_component_cb_filter_signal_flag) {	
alf_cross_component_cb_min_eg_order_minus1	ue(v)
for(i = 0; i < 3; i++)	
alf_cross_component_cb_eg_order_increase_flag[i]	u(1)
for (j = 0; j < 14;j++) {	
alf_cross_component_cb_coeff_abs[j]	uek(v)
if(alf_cross_component_cb_coeff_abs[j])	
alf_cross_component_cb_coeff_sign[j]	u(1)
}	
}	
if(alf_cross_component_cr_filter_signal_flag) {	
alf_cross_component_cr_min_eg_order_minus1	ue(v)

for(i = 0; i < 3; i++)	
alf_cross_component_cr_eg_order_increase_flag[i]	u(1)
for(j = 0; j < 14; j++) {	
alf_cross_component_cr_coeff_abs[j]	uek(v)

if(alf_cross_component_cr_coeff_abs[j])	
alf_cross_component_cr_coeff_sign[j]	u(1)
}	
}	
}	

Bảng 5A

Đối với Bảng 5A, theo một ví dụ, ngữ nghĩa học có thể dựa trên điểm sau:

alf_luma_filter_signal_flag bằng 1 cho thấy rằng tập hợp bộ lọc luma được báo hiệu. alf_luma_filter_signal_flag bằng 0 cho thấy rằng tập hợp bộ lọc luma không được báo hiệu. Khi alf_luma_filter_signal_flag không hiện diện, nó sẽ được suy ra là bằng 0.

alf_chroma_filter_signal_flag bằng 1 cho thấy rằng bộ lọc sắc độ được báo hiệu. alf_chroma_filter_signal_flag bằng 0 cho thấy rằng bộ lọc sắc độ không được báo hiệu. Khi alf_chroma_filter_signal_flag không hiện diện, nó sẽ được suy ra là bằng 0.

alf_cross_component_cb_filter_signal_flag bằng 1 cho thấy rằng tập hợp bộ lọc Cb thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 cho thấy rằng tập hợp bộ lọc Cb thành phần chéo không được báo hiệu. Khi alf_cross_component_cb_filter_signal_flag không hiện diện, nó sẽ được suy ra là bằng 0.

alf_cross_component_cr_filter_signal_flag bằng 1 cho thấy rằng tập hợp bộ lọc Cr thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 cho thấy rằng tập hợp bộ lọc Cr thành phần chéo không được báo hiệu. Khi alf_cross_component_cr_filter_signal_flagkhông hiện diện, nó sẽ được suy ra bằng 0:

alf_cross_component_cb_min_eg_order_minus1 cộng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cb thành phần chéo. Giá trị alf_cross_component_cb_min_eg_order_minus1 sẽ trong khoảng từ 0 đến 9, bao gồm cả hai khoảng này. Cần lưu ý rằng theo một số ví dụ, khoảng này có thể bị thay đổi.

alf_cross_component_cr_min_eg_order_minus1 cộng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cr thành phần chéo. Giá trị alf_cross_component_cb_min_eg_order_minus1 sẽ trong khoảng từ 0 đến 9, bao gồm cả hai khoảng này. Cần lưu ý rằng theo một số ví dụ, khoảng này có thể bị thay đổi.

alf_cross_component_cb_eg_order_increase_flag[i] bằng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cb thành phần chéo được tăng thêm 1. alf_cross_component_cb_eg_order_increase_flag[i] bằng 0 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cb thành phần chéo không được tăng thêm 1.

Bậc expGoOrderCb[i] của mã Golomb hàm số mũ được sử dụng để giải mã các giá trị của alf_cross_component_cb_coeff_abs[j] được dẫn xuất như sau:

expGoOrderCb[i] = (i = = 0 ? alf_cross_component_cb_min_eg_order_minus1 + 1 :
expGoOrderCb[i – 1]) + alf_cross_component_cb_eg_order_increase_flag[i]

Theo một ví dụ, có thể dùng giá trị được xác định trước tương ứng với bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cb thành phần chéo.

Theo một ví dụ, một giá trị tương ứng với bậc tối thiểu của mã Golomb hàm số mũ cho thành phần chéo được sử dụng cho tất cả hệ số bộ lọc Cb có thể được báo hiệu trong chuỗi bit.

alf_cross_component_cr_eg_order_increase_flag[i] bằng 1 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cr thành phần chéo được tăng thêm 1. alf_cross_component_cr_eg_order_increase_flag[i] bằng 0 cho thấy bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cr thành phần chéo không được tăng thêm 1.

Bậc expGoOrderCr[i] của mã Golomb hàm số mũ được sử dụng để giải mã các giá trị của alf_cross_component_cb_coeff_abs[j] được dẫn xuất như sau:

expGoOrderCr[i] = (i = = 0 ? alf_cross_component_cr_min_eg_order_minus1 + 1 :
expGoOrderCr[i – 1]) + alf_cross_component_cr_eg_order_increase_flag[i]

Theo một ví dụ, giá trị được xác định trước tương ứng với bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu hệ số bộ lọc Cr thành phần chéo được sử dụng.

Theo ví dụ, một giá trị tương ứng với bậc tối thiểu của mã Golomb hàm số mũ đối với thành phần chéo được sử dụng cho tất cả hệ số bộ lọc Cr được báo hiệu trong chuỗi bit.

alf_cross_component_cb_coeff_abs[j] xác định trị số tuyệt đối của hệ số thứ j trong bộ lọc Cb thành phần chéo được báo hiệu. Khi alf_cross_component_cb_coeff_abs[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

golombOrderIdxCb[ ] = {0,2,2,2,1,2,2,2,2,2,2,1,2,1} [đây có thể là Hệ số phân loại thành 3 loại, mỗi loại sử dụng cùng một bậc k mã Golomb hàm số mũ]

k= expGoOrderCb[golombOrderIdxCb[j]]

Theo ví dụ, kỹ thuật mã hóa ue(v) có thể được sử dụng để báo hiệu giá trị của phần tử cú pháp **alf_cross_component_cb_coeff_abs[j]**

alf_cross_component_cr_coeff_abs[j] cho thấy trị số tuyệt đối của hệ số thứ j của bộ lọc Cr thành phần chéo được báo hiệu. Khi alf_cross_component_cr_coeff_abs[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

golombOrderIdxCr[ ] = {0,1,2,1,0,1,2,2,2,2,2,1,2,1}[đây có thể là Hệ số phân loại thành 3 loại, mỗi loại sử dụng cùng một mã Golomb hàm số mũ bậc k]

k = expGoOrderCr[golombOrderIdxCr[j]]

Theo ví dụ, kỹ thuật mã hóa ue(v) được sử dụng để báo hiệu giá trị của phần tử cú pháp **alf_cross_component_cr_coeff_abs**[j]

alf_cross_component_cb_coeff_sign[j] cho thấy tín hiệu của hệ số bộ lọc Cb thành phần chéo thứ j như sau:

- Nếu alf_cross_component_cb_coeff_sign[j] bằng 0, hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_cb_coeff_sign[j] bằng 1), hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_cb_coeff_sign[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cb thành phần chéo $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id] với các phần tử $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][j], với j = 0..13 được dẫn xuất như sau:

$AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][j] = alf_cross_component_cb_coeff_abs[j] * (1 – 2 * alf_cross_component_cb_coeff_sign[j])

Cần có sự tương thích chuỗi bit mà các giá trị của $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][j] với j = 0..13 sẽ nằm trong khoảng từ $-2^{10} - 1$ đến $2^{10} - 1$, bao gồm cả hai khoảng này.

alf_cross_component_cr_coeff_sign[j] cho thấy tín hiệu của hệ số bộ lọc Cr thành phần chéo thứ j như sau:

- Nếu alf_cross_component_cr_coeff_sign[j] bằng 0, hệ số bộ lọc Cr thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_cr_coeff_sign[j] bằng 1), hệ số bộ lọc Cr thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_cr_coeff_sign[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cr thành phần chéo $AlfCCCoeff_{Cr}$[adaptation_parameter_set_id] với các phần tử $AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][j], với j = 0..13 được dẫn xuất như sau:

$AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][j] = alf_cross_component_cr_coeff_abs[j] * (1 – 2 * alf_cross_component_cr_coeff_sign[j])

Cần có sự tương thích chuỗi bit mà các giá trị của $AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][j] với j = 0..13 sẽ nằm trong khoảng từ $-2^{10} - 1$ đến $2^{10} - 1$, bao gồm cả hai khoảng này.

Theo một ví dụ, cấu trúc cú pháp alf_data() được cung cấp trong Bảng 5B có thể được sử dụng. Cần lưu ý rằng trong Bảng 5B, khi hệ số được báo hiệu trong APS, số lượng bộ lọc được báo hiệu bằng cách sử dụng mã hóa trừ một.

alf_data(adaptation_parameter_set_id) {	**Bộ mô tả**
...	
if (sps_cross_component_alf_enabled_flag) {	
alf_cross_component_cb_filter_signal_flag	u(1)
alf_cross_component_cr_filter_signal_flag	u(1)
}	
if(alf_luma_filter_signal_flag) {	
...	
}	
if(alf_chroma_filter_signal_flag) {	
...	
}	
if(alf_cross_component_cb_filter_signal_flag) {	
alf_cross_component_cb_fllters_signalled_minus1	ue(v)
for(k = 0; k < (alf_cross_component_cb_filters_signalled_minus1+1); k++) {	
alf_cross_component_cb_min_eg_order_minus1	ue(v)
for(i = 0; i < 3; i++)	
alf_cross_component_cb_eg_order_increase_flag[k][i]	u(1)
for (j = 0; j < 14;j++) {	
alf_cross_component_cb_coeff_abs[k][j]	uek(v)
if(alf_cross_component_cb_coeff_abs[k][j])	
alf_cross_component_cb_coeff_sign[k][j]	u(1)
}	
}	

}	
if(alf_cross_component_cr_filter_signal_flag) {	
alf_cross_component_cr_filters_signalled_minus1	ue(v)
for(k= 0; k < (alf_cross_component_cr_filters_signalled_minus1 + 1); k++) {	

alf_cross_component_cr_min_eg_order_minus1	ue(v)
for(i= 0; i < 3; i++)	
alf_cross_component_cr_eg_order_increase_flag[k][i]	u(1)
for (j = 0; j < 14; j++) {	
alf_cross_component_cr_coeff_abs[k][j]	uek(v)
if(alf_cross_component_cr_coeff_abs[k][j])	
alf_cross_component_cr_coeff_sign[k][j]	u(1)
}	
}	
}	
}	

Bảng 5B

Đối với Bảng 5B, theo một ví dụ, ngữ nghĩa học có thể dựa trên điểm sau:

alf_cross_component_cb_filter_signal_flag bằng 1 cho thấy rằng tập hợp bộ lọc Cb thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 cho thấy rằng tập hợp bộ lọc Cb thành phần chéo không được báo hiệu. Khi alf_cross_component_cb_filter_signal_flag không hiện diện, nó sẽ được suy ra là bằng 0.

alf_cross_component_cr_filter_signal_flag bằng 1 cho thấy rằng tập hợp bộ lọc Cr thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 cho thấy rằng tập hợp bộ lọc Cr thành phần chéo không được báo hiệu. Khi alf_cross_component_cr_filter_signal_flagkhông hiện diện, nó sẽ được suy ra bằng 0.

alf_cross_component_cb_filters_signalled_minus1 cộng 1 xác định số lượng tập hợp bộ lọc Cb thành phần chéo mà các hệ số được báo hiệu. Giá trị của alf_cross_component_cb_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến NumCcAlfCbFilters – 1, bao gồm cả hai khoảng này.

NumCcAlfCbFilters đại diện cho số lượng tối đa tập hợp bộ lọc Cb thành phần chéo được phép trong chuỗi video. Theo một ví dụ, NumCcAlfCbFilters có thể được đặt thành giá trị kiểu số nguyên không âm được xác định trước. Theo một ví dụ, NumCcAlfCbFilters được báo hiệu trong tập hợp thông số, ví dụ như SPS, PPS.

alf_cross_component_cb_min_eg_order_minus1[k] cộng 1 xác định bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu tập hợp hệ số bộ lọc Cb thành phần chéo thứ k. Giá trị của alf_cross_component_cb_min_eg_order_minus1[k] sẽ trong khoảng từ 0 đến 9, bao gồm cả hai khoảng này.

alf_cross_component_cb_eg_order_increase_flag[k][i] bằng 1 xác định rằng bậc tối thiểu của mã exp-Golomb để báo hiệu tập hợp hệ số bộ lọc Cb thành phần chéo thứ k được tăng thêm 1. alf_cross_component_cb_eg_order_increase_flag[k][i] bằng 0 xác định rằng bậc tối thiểu của mã exp-Golomb để báo hiệu tập hợp hệ số bộ lọc Cb thành phần chéo thứ k không được tăng thêm 1.

Bậc expGoOrderCb[k][i] của mã Golomb hàm số mũ được sử dụng để giải mã các giá trị của alf_cross_component_cb_coeff_abs[k][j] được dẫn xuất như sau:

expGoOrderCb[k][i] = (i = = 0 ? alf_cross_component_cb_min_eg_order_minus1[k] + 1 : expGoOrderCb[k][i – 1]) + alf_cross_component_cb_eg_order_increase_flag[k][i]

alf_cross_component_cb_coeff_abs[k][j] xác định trị số tuyệt đối của hệ số thứ j trong tập hợp bộ lọc Cb thành phần chéo thứ k được báo hiệu. Khi alf_cross_component_cb_coeff_abs[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

golombOrderIdxCb[ ] = {0,2,2,2,1,2,2,2,2,2,2,1,2,1} [Lưu ý: những số này có thể phân loại hệ số thành 3 loại, mỗi loại sử dụng cùng một mã Golomb hàm số mũ bậc 1]

k = expGoOrderCb[golombOrderIdxCb[j]]

alf_cross_component_cb_coeff_sign[k][j] xác định ký hiệu của hệ số thứ j trong tập hợp hệ số bộ lọc Cb thành phần chéo thứ k như sau:

- Nếu alf_cross_component_cb_coeff_sign[k][j] bằng 0, hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_cb_coeff_sign[k][j] bằng 1), hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_cb_coeff_sign[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cb thành phần chéo $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][k] có các phần tử $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][k][j], với j = 0..13 được dẫn xuất như sau:

$AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][k][j] =
alf_cross_component_cb_coeff_abs[k][j] * (1 – 2 *
alf_cross_component_cb_coeff_sign[k][j])

Cần có sự tương thích chuỗi bit mà các giá trị của $AlfCCCoeff_{Cb}$[adaptation_parameter_set_id][k][j] với j = 0..13 sẽ nằm trong khoảng từ -2^7 đến $2^7 - 1$, bao gồm cả hai khoảng này.

alf_cross_component_cr_filters_signalled_minus1 cộng 1 xác định số lượng tập hợp bộ lọc Cr thành phần chéo mà các hệ số được báo hiệu. Giá trị của alf_cross_component_cr_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến NumCcAlfCrFilters – 1, bao gồm cả hai khoảng này.

NumCcAlfCrFilters đại diện cho số lượng tối đa tập hợp bộ lọc Cr thành phần chéo được phép trong chuỗi video. Theo một ví dụ, NumCcAlfCrFilters có thể được đặt thành giá trị kiểu số nguyên không âm được xác định trước. Theo một ví dụ, NumCcAlfCrFilters được báo hiệu trong tập hợp thông số, ví dụ như SPS, PPS.

alf_cross_component_cr_min_eg_order_minus1[k] cộng 1 xác định bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu tập hợp hệ số bộ lọc Cr thành phần chéo

thứ k. Giá trị của alf_cross_component_cb_min_eg_order_minus1[k] sẽ trong khoảng từ 0 đến 9, bao gồm cả hai khoảng này.

alf_cross_component_cr_eg_order_increase_flag[k][i] bằng 1 xác định rằng bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu tập hợp hệ số bộ lọc Cr thành phần chéo thứ k được tăng thêm 1. alf_cross_component_cr_eg_order_increase_flag[k][i] bằng 0 xác định rằng bậc tối thiểu của mã Golomb hàm số mũ để báo hiệu tập hợp hệ số bộ lọc Cr thành phần chéo thứ k không được tăng thêm 1.

Bậc expGoOrderCr[k][i] của mã exp-Golomb được sử dụng để giải mã các giá trị của alf_cross_component_cb_coeff_abs[k][j] được suy ra từ:

expGoOrderCr[k][i] = (i = = 0 ? alf_cross_component_cr_min_eg_order_minus1[k] + 1 : expGoOrderCr[k][i – 1]) + alf_cross_component_cr_eg_order_increase_flag[k][i]

alf_cross_component_cr_coeff_abs k[j] xác định trị số tuyệt đối của hệ số thứ j trong tập hợp bộ lọc Cr thành phần chéo thứ k được báo hiệu. Khi alf_cross_component_cr_coeff_abs[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

golombOrderIdxCr[ ] = {0,1,2,1,0,1,2,2,2,2,2,1,2,1}[Lưu ý: những số này có thể phân loại hệ số thành 3 loại, mỗi loại sử dụng cùng một mã Golomb hàm số mũ bậc k]

k= expGoOrderCr[golombOrderIdxCr[j]]

alf_cross_component_cr_coeff_sign[k][i] xác định ký hiệu của hệ số thứ j trong tập hợp hệ số bộ lọc Cr thành phần chéo thứ k như sau:

- Nếu alf_cross_component_cr_coeff_sign[k][j] bằng 0, hệ số bộ lọc Cr thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_cr_coeff_sign[k][j] bằng 1), hệ số bộ lọc Cr thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_cr_coeff_sign[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cr thành phần chéo $AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][k] với các phần tử $AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][k][j], với j = 0..13 được suy ra từ:

$AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][k][j] =
alf_cross_component_cr_coeff_abs[k][j] * (1 – 2 *
alf_cross_component_cr_coeff_sign[k][j])

Cần có sự tương thích chuỗi bit mà các giá trị của $AlfCCCoeff_{Cr}$[adaptation_parameter_set_id][k][j] với j = 0..13 sẽ nằm trong khoảng từ -2^7 đến $2^7 - 1$, bao gồm cả hai khoảng này.

Đối với Bảng 5B, theo một ví dụ, đối với các phần tử cú pháp **alf_cross_component_cb_coeff_abs**[k][i] và**alf_cross_component_cr_coeff_abs**[k][i], k của uek(v) đại diện cho việc mã hóa golomb theo cấp số nhân bậc k, có thể tương ứng với giá trị đã xác định trước. Do đó, "k" không cần phải được báo hiệu trong chuỗi bit. Theo một ví dụ, trong trường hợp này, cấu trúc cú pháp alf_data() được cung cấp trong Bảng 5C có thể được sử dụng.

alf_data(adaptation_parameter_set_id) {	**Bộ mô tả**
...	
if (sps_cross_component_alf_enabled_flag) {	
alf_cross_component_cb_filter_signal_flag	u(1)
alf_cross_component_cr_filter_signal_flag	u(1)
}	
if(alf_luma_filter_signal_flag) {	
...	
}	
if(alf_chroma_filter_signal_flag) {	
...	
}	

if(alf_cross_component_cb_filter_signal_flag) {	
alf_cross_component_cb_filters_signalled_minus1	ue(v)
for(k= 0; k< (alf_cross_component_cb_filters_signalled_minus1+1); k++) {	
for (j = 0; j < 14; j++) {	
alf_cross_component_cb_coeff_abs[k][j]	uek(v)
if(alf_cross_component_cb_coeff_abs[k][j])	
alf_cross_component_cb_coeff_sign[k][j]	u(1)
}	
}	
}	
if(alf_cross_component_cr_filter_signal_flag) {	
alf_cross_component_cr_filters_signalled_minus1	ue(v)
for(k= 0; k< (alf_cross_component_cr_filters_signalled_minus1+1); k++) {	
for (j = 0; j < 14;j++) {	
alf_cross_component_cr_coeff_abs[k][j]	uek(v)
if(alf_cross_component_cr_coeff_abs[k][j])	
alf_cross_component_cr_coeff_sign[k] [j]	u(1)
}	
}	
}	
}	

Bảng 5C

Đối với Bảng 5C, theo một ví dụ, ngữ nghĩa học có thể dựa trên ngữ nghĩa học được cung cấp ở trên đối với Bảng 5B. Đối với các phần tử cú pháp dành cho phần tử cú pháp **alf_cross_component_cb_coeff_abs**[k][i] và **alf_cross_component_cr_coeff_abs**[k][i], theo một ví dụ, ngữ nghĩa học có thể dựa trên những điểm sau:

alf_cross_component_cb_coeff_abs[k][j] xác định trị số tuyệt đối của hệ số thứ j trong tập hợp bộ lọc Cb thành phần chéo thứ k được báo hiệu. Khi alf_cross_component_cb_coeff_abs[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

alf_cross_component_cr_coeff_abs k[j] xác định trị số tuyệt đối của hệ số thứ j trong tập hợp bộ lọc Cr thành phần chéo thứ k được báo hiệu. Khi alf_cross_component_cr_coeff_abs[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

Cần lưu ý rằng trong các ví dụ trên, giá trị hệ số được biểu thị bằng cách sử dụng phần tử cú pháp biểu thị ký hiệu (ví dụ:**alf_cross_component_cr_coeff_sign**) của hệ số và phần tử cú pháp biểu thị trị số tuyệt đối (ví dụ:**alf_cross_component_cr_coeff_abs**) của hệ số đó. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, trị số tuyệt đối của hệ số có thể được biểu thị bằng cách sử dụng một hoặc nhiều chỉ báo biểu thị liệu trị số tuyệt đối của hệ số có lớn hơn một giá trị cụ thể hay không (ví dụ: lớn hơn 1, lớn hơn 2 chỉ báo, v.v.) và phần tử cú pháp biểu thị trị số tuyệt đối của hệ số dựa trên giá trị của một hoặc nhiều chỉ báo. Ví dụ: theo một ví dụ, việc mã hóa một giá trị hệ số cụ thể có thể dựa trên ngữ nghĩa học sau:

alf_cross_component_coeff_abs_greater_than_N_flag[k][j] xác định xem trị số tuyệt đối của hệ số thứ j trong tập hợp bộ lọc thành phần chéo thứ k được báo hiệu có lớn hơn N hay không. Theo một ví dụ, hệ số phải nằm trong khoảng từ -2^7 đến 2^{7-1}, bao gồm cả 2 khoảng này và N có thể là 64.

alf_cross_component_coeff_abs[k][j] xác định trị số tuyệt đối của hệ số thứ j trong tập hợp bộ lọc thành phần chéo thứ k được báo hiệu. Khi alf_cross_component_coeff_abs[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

alf_cross_component_coeff_sign[k][i] xác định ký hiệu của hệ số thứ j trong tập hợp hệ số bộ lọc thành phần chéo thứ k như sau:

- Nếu alf_cross_component_coeff_sign[k][j] bằng 0, hệ số bộ lọc thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_coeff_sign[k][j] bằng 1), hệ số bộ lọc thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_coeff_sign[k][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc thành phần chéo AlfCCCoeff [adaptation_parameter_set_id] [k] với các phần tử AlfCCCoeff [adaptation_parameter_set_id][k][j] được dẫn xuất như sau:

AlfCCCoeff[adaptation_parameter_set_id][k][j] =
(N*alf_cross_component_coeff_abs_greater_than_N_flag[k][j]
+alf_cross_component_coeff_abs[k][j]) * (1 – 2 *
alf_cross_component_coeff_sign[k][j])

Theo một ví dụ, tập hợp hệ số bộ lọc thành phần chéo có thể được báo hiệu cho bộ lọc màu thành phần chéo (Cb và/hoặc Cr). Mỗi tập hợp hệ số bộ lọc thành phần chéo có thể được gán chỉ số bộ lọc thành phần chéo. Theo ví dụ, tập hợp hệ số bộ lọc thành phần chéo có thể được báo hiệu trong APS. Theo ví dụ, giá trị mẫu có thể được phân chia (ví dụ: được xác định bằng cách sử dụng các kỹ thuật tương tự như cách phân chia do tín hiệu chỉ báo kiểm soát hoặc bất kỳ cách phù hợp nào khác). Chỉ số bộ lọc có thể được báo hiệu cho từng phân vùng của giá trị mẫu, trong đó chỉ số bộ lọc xác định bộ lọc thành phần chéo sẽ được áp dụng cho các mẫu trong phân vùng. Việc phân chia có thể được truyền thông bằng cách dùng thông số như kích thước khối (có thể giống với thông số kích thước khối kiểm soát, hoặc có thể độc lập). Vùng phân chia có thể được dẫn xuất bằng cách dùng giá trị thông số, kích thước hình ảnh/mảnh/nhóm ô/MCTS. Theo ví dụ, tập hợp chỉ gồm hệ số bộ lọc thành phần chéo có thể được báo hiệu trong APS cho mỗi thành phần màu. Mã nhận dạng APS có thể được báo hiệu cho mỗi phân vùng của giá trị mẫu mà xác định bộ lọc thành phần chéo sẽ được áp dụng cho các mẫu trong phân vùng.

Theo một ví dụ, bộ đệm cho tập hợp con hoặc tất cả các tầng thời gian có thể được đặt lại, ví dụ như cho tập hợp các loại NALU (ví dụ: tương ứng với điểm truy cập ngẫu nhiên như IRAP), cho tập hợp loại mảnh (giả sử I-mảnh). Theo ví dụ, sự đặt lại có thể ngụ ý là phép toán làm rỗng. Theo ví dụ, sự đặt lại có thể ngụ ý là đặt bộ đệm thành tập hợp giá trị được xác định trước (ví dụ như 0, tập hợp giá trị cố định cho mỗi TemporalID). Phép toán đặt lại bộ đệm có thể ngụ ý là giá trị được lưu trữ trong bộ đệm trước bộ truy cập (đáp ứng tập hợp điều kiện được xác định trước, ví dụ như loại NALU biểu thị IRAP, loại mảnh bằng với mảnh I) có thể không có sẵn cho bộ truy cập đó, và sau đó là các bộ truy cập được mã hóa. Theo ví dụ, khi bộ đệm không chứa bất kỳ hệ số nào thì có thể không báo hiệu phần tử cú pháp biểu thị liệu hệ số bộ lọc từ bộ đệm có được sử dụng hay không và giá trị của phần tử này sẽ được suy ra.

Theo ví dụ, khi bộ đệm không chứa bất kỳ hệ số nào thì có thể báo hiệu giới hạn ở giá trị được xác định trước phần tử cú pháp biểu thị liệu hệ số bộ lọc từ bộ đệm có được sử dụng hay không.

Ngoài ra, theo một ví dụ, không nên sử dụng bộ lọc được báo hiệu cho hình ảnh đầu bằng cách sử dụng các hình ảnh sau của hình ảnh IRAP liên quan. Điều này là do phép toán truy cập ngẫu nhiên các hình ảnh đầu chuỗi bit có thể bị loại bỏ khỏi chuỗi bit. Như được mô tả ở trên, theo các kỹ thuật trong tài liệu này, bộ lọc thành phần chéo có thể được báo hiệu bằng cách: tái sử dụng bộ lọc trong bộ đệm tầng phụ thời gian tương ứng và/hoặc bộ lọc trong APS. Cần lưu ý rằng trong một số trường hợp, APS có thể được báo hiệu nằm ngoài dải băng, vì vậy trong một số trường hợp, tất cả những gì có thể cần để tương thích chuỗi bit là phải có sẵn APS được tham chiếu.

Như được mô tả ở trên, đối với HÌNH 9A-9F, hình dạng bộ lọc có thể được xác định dựa trên loại vị trí sắc độ. Cần lưu ý rằng, hình dạng bộ lọc có thể bao gồm các hình dạng cho nhiều loại bộ lọc khác nhau. Ngoài ra, hình dạng bộ lọc nói chung có thể được mô tả là đại diện cho giá đỡ và điểm gốc liên quan đến mẫu được lọc. Theo một ví dụ, hình dạng bộ lọc có thể được báo hiệu cho từng bộ lọc thành phần chéo trong tập hợp thông số: ví dụ như SPS, PPS, VUI. Theo một ví dụ, hình dạng bộ lọc cũng có thể được báo hiệu trong APS hoặc phần đầu mảnh, ví dụ như khi hình dạng bộ lọc ảnh hưởng đến số lượng hệ số bộ lọc được mã hóa. Theo một ví dụ, hình dạng bộ lọc có thể được sử dụng để xác định số lượng hệ số bộ lọc. Theo một ví dụ, hệ số bộ lọc có thể được sử dụng lại. Nghĩa là, theo một ví dụ, Bộ đệm vào trước-ra trước (First In First Out - FIFO) có thể được duy trì cho từng thành phần Cb/Cr, cho mỗi tầng thời gian. Theo một ví dụ, kích thước của bộ đệm FIFO là 1 cho mỗi thành phần, mỗi tầng thời gian. Theo ví dụ này, chỉ báo có thể được báo hiệu để biểu thị nếu hệ số bộ lọc trong bộ đệm FIFO tương ứng (tầng thời gian) sẽ được sử dụng hoặc thu được tập hợp hệ số mới. Khi thu được tập hợp hệ số bộ lọc mới, tập hợp này sẽ thay thế dung lượng của bộ đệm FIFO (tầng thời gian) tương ứng. Theo một ví dụ, kích thước của bộ đệm FIFO là 1 cho mỗi thành phần, mỗi tầng thời gian. Theo ví dụ này, hệ số bộ lọc trong bộ đệm FIFO thuộc cùng một tầng thời gian hoặc tầng thời gian thấp hơn có thể được tái sử dụng. Phần tử cú pháp (ví dụ: chỉ báo) được thu để biểu thị liệu hệ số bộ lọc ở một trong các bộ đệm FIFO có được tái sử dụng hay không, và nếu có (ví dụ: giá trị chỉ báo là 1) thì thu được chỉ số đến bộ đệm FIFO có hệ số sẽ được tái sử dụng. Phạm vi giá trị hợp lệ đối với chỉ số được xác định dựa trên ID tầng thời gian hiện tại (ví dụ: 0 đến ID tầng thời gian hiện tại). Có thể sử dụng kỹ thuật mã hóa ue(v) để báo hiệu giá trị như -

(Tầng thời gian của bộ đệm FIFO - Tầng thời gian hiện tại). Theo một ví dụ khác, có thể sử dụng đơn phân cắt ngắn (với giá trị tối đa của TU dựa trên ID tầng thời gian hiện tại).

Theo một ví dụ, tín hiệu kiểm soát cục bộ của bộ lọc thành phần chéo có thể bao gồm việc gửi tất cả chỉ báo kiểm soát mức khối Cb và Cr thành phần chéo cho mảnh trong CTU thứ nhất của mảnh. Theo một ví dụ, khi kích thước khối kiểm soát lớn hơn kích thước CTU thì chỉ báo kiểm soát có thể được gửi trong CTU được mã hóa thứ nhất của khối kiểm soát. Các CTU còn lại trong khối kiểm soát có thể suy ra giá trị tương tự như trong CTU được mã hóa thứ nhất của khối kiểm soát. Khi CTU được mã hóa thứ nhất trong khối kiểm soát thu được chỉ báo kiểm soát, giá trị của chỉ báo kiểm soát này có thể được suy ra là bằng 0. Theo một ví dụ, chỉ báo kiểm soát được gửi cho mỗi CTU. Theo một ví dụ, chỉ báo kiểm soát có thể được báo hiệu trong CTU thứ nhất của nhóm mảnh/ô. Ngoài ra, theo một ví dụ, bốn chỉ báo kiểm soát có thể được báo hiệu cho từng CTU của nhóm ô/mảnh. Theo một ví dụ, khi bốn khối kiểm soát tồn tại trong CTU, có thể có một số chỉ báo khối kiểm soát khác nhau cho các CTU một phần (ví dụ: ở đường biên) so với các CTU hoàn chỉnh. Theo một ví dụ, chỉ báo kiểm soát có thể được báo hiệu cho từng CTU của nhóm ô/mảnh. Theo một ví dụ, chỉ báo kiểm soát có thể được báo hiệu cho CTU thứ nhất của nhóm CTU, và cho các CTU còn lại trong nhóm mà có thể suy ra cùng một giá trị của chỉ báo.

Theo một ví dụ, hệ số bộ lọc thành phần chéo đối với bộ lọc thành phần chéo có trong tập hợp thông số độc lập của riêng nó, ví dụ như APS. Theo một ví dụ, hệ số bộ lọc thành phần chéo có trong tập hợp thông số (ví dụ như APS) khác với bộ lọc không có thành phần chéo (ví dụ: ALF trong JVET-N1001-v8). Theo một ví dụ, thay vì bộ lọc 5x6, phương án có thể sử dụng quy trình lọc ALF 7x7 được mô tả trong mệnh đề JVET-N1001-v8 về "Quy trình lọc khối cây mã hóa dành cho mẫu luma". Điều này sẽ làm giảm thêm số lượng hệ số cần được báo hiệu. Theo mô tả ở trên, phép toán lọc được áp dụng để lọc các mẫu trong thành phần và/hoặc kênh màu và tín hiệu mà có thể kích hoạt/vô hiệu hóa phép toán này trên cơ sở từng khung hình và từng vị trí được cung cấp. Theo một ví dụ, phép toán lọc cũng có thể được thực hiện sao cho có sẵn nhiều phép toán lọc tại mỗi khung và/hoặc vị trí, và tín hiệu đã mô tả có thể được sử dụng để phát nhiều bộ lọc và chọn trong số các bộ lọc có sẵn. Theo một ví dụ, phương án có thể sử dụng bộ lọc hình kim cương 3x4. Cần lưu ý rằng trong các ví dụ khác, các kích thước và hình dạng bộ lọc khác có thể được sử dụng (ví dụ: 3x3, 4x3, 4x4, v.v.). Theo một ví dụ, khi sử dụng bộ lọc hình kim cương 3x4 thì tám hệ số duy nhất có thể được sử dụng. Cần lưu ý rằng, trong các ví dụ khác, đối với mỗi kích thước và

hình dạng bộ lọc được mô tả trong tài liệu này, số lượng các hệ số duy nhất được sử dụng và/hoặc được báo hiệu có thể khác nhau để tối ưu hóa phần mào đầu báo hiệu. Theo một ví dụ, khi sử dụng bộ lọc hình kim cương 3x4, có thể xác định tối đa bốn bộ lọc duy nhất cho mỗi thành phần (ví dụ: trên cấp trình tự, hình ảnh, ô hoặc mảnh), và khi áp dụng bộ lọc thì một trong bốn bộ lọc có thể được chọn. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, chỉ báo kiểm soát vùng cục bộ có thể được các kênh sắc độ khác nhau dùng chung.

Như được mô tả ở trên, việc áp dụng lọc thành phần chéo có thể dựa trên các đặc tính của mẫu có trong vùng giá đỡ bộ lọc (ví dụ: phương sai và/hoặc độ lệch). Theo một ví dụ, chỉ báo kiểm soát dùng chung, và các đặc tính của mẫu có trong vùng giá đỡ bộ lọc có thể được sử dụng để xác định xem có áp dụng lọc thành phần chéo cho một hoặc cả hai kênh sắc độ hay không. Ví dụ, theo một ví dụ, đối với Cb, giá trị của một chỉ báo kiểm soát dùng chung có thể xác định xem có áp dụng lọc thành phần chéo hay không và đối với Cr, giá trị của chỉ báo kiểm soát dùng chung và ngoài ra, phương sai của giá đỡ có thể xác định xem có áp dụng lọc thành phần chéo hay không. Như đã mô tả ở trên, theo một ví dụ, khi một hoặc nhiều mẫu giá đỡ không khả dụng (ví dụ: do VB) thì tính năng lọc thành phần chéo có thể bị vô hiệu hóa. Do đó, thuộc tính của các mẫu có trong vùng giá đỡ bộ lọc để xác định xem liệu có áp dụng lọc thành phần chéo hay không có thể bao gồm tính khả dụng. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc có áp dụng lọc thành phần chéo hay không có thể dựa trên số lượng mẫu giá đỡ ngưỡng có sẵn hay không (ví dụ: có sẵn 50 % trở lên mẫu giá đỡ hay không). Tương tự, khi một hoặc nhiều mẫu giá đỡ không khả dụng, việc có áp dụng lọc thành phần chéo hay không sẽ dựa trên việc liệu quy trình đệm đã xác định có thể tạo ra giá trị cho từng mẫu giá đỡ không khả dụng hay không. Nghĩa là, trong trường hợp quy trình đệm đã xác định không cung cấp cơ chế tạo giá trị cho mẫu không có sẵn thì tính năng lọc thành phần chéo có thể bị vô hiệu hóa. Ngoài ra, tính năng lọc thành phần chéo có thể bị vô hiệu hóa nếu vị trí của vùng giá đỡ bộ lọc, theo mẫu giá đỡ đã xác định (ví dụ: mẫu trung tâm của giá đỡ hình thoi) thuộc một phạm vi VB đã xác định. Ví dụ: theo một ví dụ, nếu mẫu trung tâm của vùng giá đỡ bộ lọc hình thoi nằm bên dưới và ở trong 2 mẫu VB ngang thì tính năng lọc thành phần chéo có thể bị vô hiệu hóa.

Như được mô tả ở trên, trong JVET-N1001 và JVET-O2001, CTU được phân chia theo cấu trúc cây đa kiểu hình cộng với cây tứ phân (QTMT hoặc QT+MTT). Trong JVET-N1001 và JVET-O2001, đối với mảnh I, mỗi CTU có thể được chia thành các bộ mã hóa với mẫu luma 64x64 bằng cách tách cây tứ phân ẩn và các bộ mã hóa này có thể

là gốc của hai cấu trúc cú pháp cây mã hóa riêng biệt, nghĩa là một dành cho kênh luma và một cho kênh sắc độ. Trong cả hai trường hợp, giá trị chỉ báo kiểm soát vùng cục bộ có thể được báo hiệu/xác định trong cú pháp của tín hiệu cây phân chia. Nghĩa là, ví dụ như cú pháp và ngữ nghĩa học được sử dụng để phân chia các kênh sắc độ thành CU (nghĩa là cây mã hóa sắc độ) có thể bao gồm tín hiệu (ngầm và/hoặc rõ ràng) biểu thị giá trị chỉ báo kiểm soát vùng cục bộ. Ví dụ: JVET-N1001 và JVET-O2001, xác định biến số cbSubdiv = 2 * cqtDepth, cqtDepth là độ sâu cây tứ phân mã hóa hiện tại và độ sâu tại CTU là 0. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, nút cây nhỏ nhất có cbSubdiv thấp hơn hoặc bằng ngưỡng đã cho có thể đại diện cho nhóm báo hiệu chỉ báo kiểm soát nút mẹ, trong đó tất cả các khối tạo thành từ những lần phân tách tiếp theo đều thuộc cùng một nhóm báo hiệu chỉ báo kiểm soát. Nghĩa là, chỉ báo kiểm soát vùng cục bộ tại CU mẹ có thể được sử dụng để suy ra giá trị của chỉ báo kiểm soát vùng cục bộ tại bất kỳ CU con nào thu được. Bảng 6 và Bảng 7 minh họa ví dụ về cú pháp và gán biến số để xác định giá trị chỉ báo kiểm soát vùng cục bộ. Nghĩa là, trong Bảng 6, nút cây nhỏ nhất có cbSubdiv thấp hơn hoặc bằng ngưỡng đã cho có thể đại diện cho nhóm báo hiệu chỉ báo kiểm soát nút mẹ. Bảng 7 cung cấp cú pháp bộ mã hóa tương ứng.

// Tại cây mã hóa sắc độ
if(CC ALF được kích hoạt cho mảnh && cbSubdiv <= độ sâu báo hiệu kiểm soát CC ALF cho mảnh) {
Giá trị chỉ báo kiểm soát Cu CC Alf được đặt là 0
IsCuCcAlfControlFlagCoded = 0
CuCcAlfTopLeftX = x0
CuCcAlfTopLeftY = y0
}

Bảng 6

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**Bộ mô tả**
...	
if(CC ALF được kích hoạt cho mảnh && !IsCuCcAlfControlFlagCoded) {	
cross_component_chroma_alf_control_flag (dùng chung hoặc riêng)	ae(v)
}	
...	

Bảng 7

Theo ví dụ được minh họa trong Bảng 6, độ sâu báo hiệu kiểm soát ALF CC đối với mảnh đại diện cho ngưỡng. Theo một ví dụ, ngưỡng này có thể được báo hiệu cho mảnh, chuỗi, hoặc hình ảnh. Vị trí luma (xCtrlBlk, yCtrlBlk) xác định mẫu luma trên cùng bên trái của khối kiểm soát sắc độ hiện tại so với mẫu luma trên cùng bên trái của hình ảnh hiện tại. Các vị trí ngang và dọc xCtrlBlk và yCtrlBlk được đặt bằng nhau lần lượt là CuCcAlfTopLeftX và CuCcAlfTopLeftY. Ngoài ra, khối kiểm soát sắc độ hiện tại là vùng hình chữ nhật bên trong khối cây mã hóa có cùng Giá trị chỉ báo kiểm soát Alf CC (dùng chung hoặc riêng đối với thành phần sắc độ). Chiều rộng và chiều cao của nó bằng với chiều rộng và chiều cao của nút cây mã hóa mà vị trí mẫu luma ở trên cùng bên trái được gán cho các biến số CuCcAlfTopLeftX và CuAlfCcTopLeftY. Cần lưu ý rằng theo một ví dụ, có thể có tập hợp biến số riêng cho từng thành phần sắc độ.

Đối với Bảng 7, theo một ví dụ, khi **cross_component_chroma_alf_control_flag** (dùng chung hoặc riêng) không hiện diện, nó sẽ được suy ra bằng 0 và khi **cross_component_chroma_alf_control_flag** (dùng chung hoặc riêng) có hiện diện, biến số IsCuCcAlfControlFlagCoded được đặt thành 1. Ngoài ra, Giá trị chỉ báo kiểm soát Cu CC Alf được đặt thành **cross_component_chroma_alf_control_flag.** Như vậy, theo ví dụ này, miễn là điều kiện để bắt đầu nhóm báo hiệu chỉ báo kiểm soát mới là true (nghĩa là CC ALF được kích hoạt để mảnh và cuSubdiv không cao hơn giới hạn), chỉ báo nội bộ "IsCuCcAlfControlFlagCoded" được đặt thành 0 và điểm gốc nút cây hiện tại sẽ được lưu dưới dạng điểm gốc nhóm báo hiệu chỉ báo kiểm soát trong các biến số CuCcAlfTopLeftX, CuCcAlfFopLeftY. Sau đó, trong cú pháp bộ mã hóa, nếu "IsCuCcAlfControlFlagCoded" là 0, chỉ báo nhóm báo hiệu kiểm soát được mã hóa và chỉ báo "IsCuCcAlfControlFlagCoded" được đặt thành 1, ngăn không mã hóa các chỉ báo khối kiểm soát khác cho đến khi tìm thấy nhóm báo hiệu chỉ báo kiểm soát mới. Bộ mã hóa có thể kế thừa giá trị chỉ báo kiểm soát của chúng từ điểm gốc nút cây được mã hóa cuối cùng cho đến khi tìm thấy nhóm báo hiệu chỉ báo kiểm soát mới. Theo ví dụ, khi phần tử cú pháp vùng kiểm soát cục bộ hiện diện trong bộ mã hóa thì IsCuCcAlfControlFlagCoded được đặt bằng 1. Cần lưu ý rằng JVET-N1001 và JVET-O2001 cung cấp các nhóm thông số lượng tử hóa biểu thị độ sâu thấp nhất mà giá trị QP được báo hiệu. Theo một ví dụ, tín hiệu kiểm soát ALF CC có thể được căn chỉnh các nhóm thông số lượng tử hóa. Nghĩa là, các nút con trong nhóm thông số lượng tử hóa dùng chung giá trị QP và giá trị điều khiển

ALF CC. Ngoài ra, trong một trường hợp, giá trị của giá trị điều khiển ALF CC có thể dựa trên hoặc được dẫn xuất hoàn toàn từ giá trị QP (ví dụ: nếu giá trị QP nhỏ hơn ngưỡng, chỉ báo điều khiển ALF CC không được báo hiệu cho nhóm và được suy ra là 0).

Như được mô tả ở trên, tín hiệu lọc thành phần chéo có thể bao gồm tín hiệu của một bộ lọc cụ thể (ví dụ: hình dạng bộ lọc và/hoặc hệ số bộ lọc). Trong một ví dụ, theo các kỹ thuật trong tài liệu này, giá trị của phần tử cú pháp có thể biểu thị liệu có áp dụng lọc thành phần chéo cho vùng hay không, và nếu có thì áp dụng bộ lọc cụ thể nào cho vùng đó. Ví dụ: giá trị 0 có thể biểu thị không áp dụng lọc thành phần chéo cho vùng, giá trị 1 có thể biểu thị áp dụng bộ lọc có tập hợp hệ số bộ lọc thứ nhất, giá trị 2 có thể biểu thị áp dụng bộ lọc có tập hợp hệ số bộ lọc thứ hai, v.v. Theo một ví dụ, vùng này có thể là CTU. Bảng 8 minh họa ví dụ về cú pháp có trong cấu trúc cú pháp coding_tree_unit( ) theo các kỹ thuật trong tài liệu này. Nghĩa là, theo ví dụ được minh họa trong Bảng 8, các phần tử cú pháp **alf_cross_component_cb_idc** và **alf_cross_component_cr_idc** biểu thị liệu có áp dụng lọc thành phần chéo hay không và thời điểm áp dụng lọc thành phần chéo biểu thị bộ lọc.

coding_tree_unit() {	**Bộ mô tả**
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
xCtbC = xCtb / SubWidthC	
yCtbC = yCtb / SubHeightC	
...	
if(slice_cross_component_alf_cb_enabled_flag) {	
xStartC = (xCtbC >> AlfCCSamplesCbLog2W) << AlfCCSamplesCbLog2W	
yStartC = (yCtbC >> AlfCCSamplesCbLog2H) << AlfCCSamplesCbLog2H	
if (xCtbC == xStartC && yCtbC == yStartC) { // ít nhất một chỉ báo sẽ được mã hóa trong CTU và khi vùng kiểm soát cục bộ không trải dài qua các CTU thì có thể bỏ qua kiểm tra này	
xEndC = xStartC + (1 << CtbLog2SizeY) / SubWidthC	
xEndC = (xEndC >= PicWidthInChromaSamples) ? PicWidthInChromaSamples : xEndC // không vượt quá chiều rộng hình ảnh	
yEndC = yStartC + (1 << CtbLog2SizeY) / SubHeightC	
yEndC = (yEndC >= PicHeightInChromaSamples) ? PicHeightInChromaSamples : yEndC // không vượt quá chiều cao hình ảnh	

for (yC = yStartC; yC < yEndC; yC += AlfCCSamplesCbLog2H) {	
for (xC = xStartC; xC < xEndC; xC += AlfCCSamplesCbLog2W) {	
alf_cross_component_cb_idc[xC >> AlfCCSamplesCbLog2W][yC >> AlfCCSamplesCbLog2H]	ae(v)
}	
}	
} else if (UnavailableCb(xStartC, yStartC)){// ví dụ ngoài mảnh, ngoài ô	
alf_cross_component_cb_idc[xC >> AlfCCSamplesCbLog2W][yC >> AlfCCSamplesCbLog2H]	ae(v)
} // Khi vùng kiểm soát cục bộ không trải dài qua các CTU và đường biên mảnh/ô/khối dạng gạch được căn chỉnh với đường biên CTU thì có thể bỏ qua nhánh "else if" này	
}	
if(slice_cross_component_alf_cr_enabled_flag) {	
xStartC = (xCtbC >> AlfCCSamplesCrLog2W) << AlfCCSamplesCrLog2W	
yStartC = (yCtbC >> AlfCCSamplesCrLog2H) << AlfCCSamplesCrLog2H	
if (xCtbC == xStartC && yCtbC == yStartC) { // ít nhất một chỉ báo sẽ được mã hóa trong CTU và Khi vùng kiểm soát cục bộ không trải dài qua các CTU thì có thể bỏ qua kiểm tra này	
xEndC = xStartC + (1 << CtbLog2SizeY) / SubWidthC	
xEndC = (xEndC >= PicWidthInChromaSamples) ? PicWidthInChromaSamples : xEndC // không vượt quá chiều rộng hình ảnh	
yEndC = yStartC + (1 << CtbLog2SizeY) / SubHeightC	
yEndC = (yEndC >= PicHeightInChromaSamples) ? PicHeightInChromaSamples : yEndC // không vượt quá chiều cao hình ảnh	
for (yC = yStartC; yC < yEndC; yC += AlfCCSamplesCrLog2H) {	
for (xC = xStartC; xC < xEndC; xC += AlfCCSamplesCrLog2W) {	
alf_cross_component_cr_idc[xC >> AlfCCSamplesCrLog2W][yC >> AlfCCSamplesCrLog2H]	ae(v)
}	
}	
} else if (UnavailableCr(xStartC, yStartC)){ // ví dụ ngoài mảnh, ngoài ô	

alf_cross_component_cr_idc[xC >> AlfCCSamplesCrLog2W][yC >> AlfCCSamplesCrLog2H]	ae(v)
} // Khi vùng kiểm soát cục bộ không trải dài qua các CTU và đường biên mảnh/ô/khối dạng gạch được căn chỉnh với đường biên CTU thì có thể bỏ qua nhánh "else if" này	
}	
...	
if(slice_type = = I && qtbtt_dual_tree_intra_flag)	
dual_tree_implicit_qt_split (xCtb, yCtb, CtbSizeY, 0)	
khác	
coding_tree(xCtb, yCtb, CtbSizeY, CtbSizeY, 1, 1, 0, 0, 0, 0, 0, SINGLE_TREE, MODE_TYPE_ALL)	
}	

Bảng 8

Đối với Bảng 8, theo một ví dụ, ngữ nghĩa học có thể dựa trên điểm sau:

PicWidthInChromaSamples = pic_width_in_luma_samples / SubWidthC

PicHeightInChromaSamples = pic_height_in_luma_samples / SubHeightC

AlfCCSamplesCbLog2W = AlfCCSamplesCbLog2H =
slice_cross_component_alf_cb_log2_control_size_minus4 + 4

AlfCCSamplesCrLog2W = AlfCCSamplesCrLog2H =
slice_cross_component_alf_cr_log2_control_size_minus4 + 4

Ngoài ra, cần lưu ý:

Theo ví dụ, xStartC tương ứng với cạnh trên cùng của CTU. Theo ví dụ, yStartC tương ứng với cạnh bên trái của CTU.

Khi các vùng kiểm soát cục bộ không trải rộng trên các CTU thì có thể bỏ qua kiểm tra (xCtbC == xStartC && yCtbC == yStartC).

Theo một ví dụ, (xEndC >= PicWidthInChromaSamples) và (yEndC >= PicHeightInChromaSamples) cận trên được cung cấp bởi PicWidthInChromaSamples và

thay vào đó, PicHeightInChromaSample có thể lần lượt tương ứng với đường biên mảnh/ô/khối dạng gạch/CTU bên phải và ở dưới cùng.

alf_cross_component_cb_idc[xC >> AlfCCSamplesCbLog2W][yC >> AlfCCSamplesCbLog2H] bằng 0 biểu thị rằng bộ lọc Cb thành phần chéo không được áp dụng cho khối các mẫu sắc độ Cb có vị trí sắc độ trên cùng bên trái (xC, yC). alf_cross_component_cb_idc[xC >> AlfCCSampiesCbLog2W][yC >> AlfCCSamplesCbLog2H] bằng m, trong đó m lớn hơn 0 biểu thị rằng tập hợp bộ lọc Cb thành phần chéo thứ k=(m-1) được áp dụng cho khối các mẫu sắc độ Cb, với vị trí sắc độ ở trên cùng bên trái (xC, yC)

alf_cross_component_cr_idc[xC >> AlfCCSamplesCrLog2W] [yC >> AlfCCSamplesCrLog2H] bằng 0 biểu thị rằng bộ lọc Cr thành phần chéo không được áp dụng cho khối các mẫu sắc độ Cr có vị trí sắc độ ở trên cùng bên trái (xC, yC). alf_cross_component_cr_idc[xC >> AlfCCSamplesCrLog2W][yC >> AlfCCSamplesCrLog2H] bằng m, trong đó m lớn hơn 0 biểu thị rằng tập hợp bộ lọc Cr thành phần chéo thứ k=(m-1) được áp dụng cho khối các mẫu sắc độ Cr, với vị trí sắc độ ở trên cùng bên trái (xC, yC)

Hàm UnavailableCb(xC,yC) trả về TRUE khi **alf_cross_component_cb_idc[** xC >> AlfCCSamplesCbLog2W][yC >> AlfCCSamplesCbLog2H] tương ứng với vị trí mẫu sắc độ (xC,yC) không có sẵn, ví dụ như khi (xC,yC) thuộc mảnh/ô khác so với CTU hiện tại. Nếu không, nó sẽ trả về FALSE

Hàm UnavailableCr(xC,yC) trả về TRUE khi **alf_cross_component_cr_idc[** xC >> AlfCCSamplesCrLog2W][yC >> AlfCCSamplesCrLog2H] tương ứng với vị trí mẫu sắc độ (xC,yC) không có sẵn, ví dụ như khi (xC,yC) thuộc mảnh/ô khác so với CTU hiện tại. Nếu không, nó sẽ trả về FALSE.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, nhị phân hóa của **alf_cross_component_cb_idc** và/hoặc **alf_cross_component_cr_idc** có thể là Truncated Rice, TR, nhị phân hóa có giá trị cực đại, cMax, tùy thuộc vào số lượng tập hợp hệ số bộ lọc được báo hiệu và cRiceParam là bằng 0. Bảng 9 minh họa ví dụ về nhị phân hóa Truncated Rice đối với **alf_cross_component_cb_idc** và/hoặc **alf_cross_component_cr_idc**:

Giá trị chỉ báo	**Nhị phân hóa TR (cRiceParam bằng 0)**
0	0

1	10
2	110
3	1110
...	...
cMax-1	1....10 chứa (cMax-1) 1's
cMax	1....11 chứa cMax 1 's

Bảng 9

Trong đó,

Đối với CC ALF Cb cMax được đặt thành (alf_cross_component_cb_filters_signalled_minus1 cộng 1) tương ứng

Đối với CC ALF Cr cMax được đặt thành (alf_cross_component_cr_filters_signalled_minus1 cộng 1) tương ứng

Theo một ví dụ, đối với **alf_cross_component_cb_idc** và/hoặc **alf_cross_component_cr_idc** chỉ bin thứ nhất có thể được mã hóa theo ngữ cảnh (nghĩa là các bin khác có thể không được mã hóa). Theo một ví dụ, dẫn xuất của ngữ cảnh cho bin thứ nhất có thể như sau:

Đầu vào cho quy trình này là vị trí luma (x0, y0) xác định mẫu luma ở trên cùng bên trái của khối luma hiện tại so với mẫu trên cùng bên trái của hình ảnh hiện tại, thành phần màu cIdx, độ sâu cây tứ phân mã hóa hiện tại cqtDepth, loại kênh cây kép chType, chiều rộng và chiều cao của khối mã hóa hiện tại trong các mẫu luma cbWidth và cbHeight, cũng như các biến số allowSplitBtVer, allowSplitBtHor, allowSplitTtVer, allowSplitTtHor và allowSplitQt như được dẫn xuất trong ngữ nghĩa học cây mã hóa.

Đầu ra của quy trình này là ctxInc.

Vị trí (xNbL, yNbL) được đặt bằng nhau (x0 – 1, y0) và quy trình dẫn xuất đối với tính khả dụng khối lân cận như đã xác định được dẫn ra với vị trí (xCurr, yCurr) được đặt bằng nhau (x0, y0), vị trí lân cận (xNbY, yNbY) được đặt bằng nhau (xNbL, yNbL), checkPredModeY được đặt là FALSE, cũng như cIdx làm đầu vào và đầu ra được gán cho availableL.

Vị trí (xNbA, yNbA) được đặt bằng nhau (x0, y0 – 1) và quy trình dẫn xuất đối với tính khả dụng khối lân cận như đã xác định được dẫn ra với vị trí (xCurr, yCurr)

được đặt bằng nhau (x0, y0), vị trí lân cận (xNbY, yNbY) được đặt bằng nhau (xNbA, yNbA), checkPredModeY được đặt là FALSE, cũng như cIdx làm đầu vào và đầu ra được gán cho availableA.

Với condL và condA được xác định trong Bảng 10, việc gán ctxInc được xác định như sau:

Đối với phần tử cú pháp alf_cross_component_cb_idc[x0][y0] và alf_cross_component_cr_idc[x0][y0]: ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3

Phần tử cú pháp	condL	condA	ctxSetIdx
alf_cross_component_cb_idc[x 0][y0]	alf_cross_component_cb_idc[xNbL][yNbL] = 0	alf_cross_component_cb_idc[xNbA][yNbA]==0	0
alf_cross_component_cr_idc[x0][y0]	alf_cross_component_cr_idc[xNbL][yNbL] == 0	alf_cross_component_cr_idc[xNbA][yNbA] == 0	0

Bảng 10

Đối với Bảng 17, theo một ví dụ, mỗi thử nghiệm "== 0" có thể được thay thế bằng "!=0."

Theo một ví dụ, đối với **alf_cross_component_cb_idc** và/hoặc**alf_cross_component_cr_idc** tất cả các bin (ngăn) có thể được mã hóa theo ngữ cảnh và mỗi bin có thể sử dụng một tập hợp ngữ cảnh riêng như được biểu thị trong Bảng 11.

Phần tử cú pháp	binIdx					
	0	1	2	3	4	>= 5
alf_cross_component_cb_idc[ ][ ]	0..2	3..5	6..8	9..12	13..15	3*binIdx .. (4*binIdx-1)
alf_cross_component_cr_idc[ ][ ]	0..2	3..5	6..8	9..12	13..15	3*binIdx .. (4*binIdx-1)

Bảng 11

Đối với Bảng 11, theo một ví dụ, dẫn xuất của ngữ cảnh có thể là như sau:

Đầu vào cho quy trình này là vị trí luma (x0, y0) xác định mẫu luma ở trên cùng bên trái của khối luma hiện tại so với mẫu trên cùng bên trái của hình ảnh hiện tại, thành

phần màu cIdx, độ sâu cây tứ phân mã hóa hiện tại cqtDepth, loại kênh cây kép chType, chiều rộng và chiều cao của khối mã hóa hiện tại trong các mẫu luma cbWidth và cbHeight, cũng như các biến số allowSplitBtVer, allowSplitBtHor, allowSplitTtVer, allowSplitTtHor và allowSplitQt như được dẫn xuất trong ngữ nghĩa học cây mã hóa.

Đầu ra của quy trình này là ctxInc.

Vị trí (xNbL, yNbL) được đặt bằng nhau (x0 – 1, y0) và quy trình dẫn xuất đối với tính khả dụng khối lân cận như đã xác định được dẫn ra với vị trí (xCurr, yCurr) được đặt bằng nhau (x0, y0), vị trí lân cận (xNbY, yNbY) được đặt bằng nhau (xNbL, yNbL), checkPredModeY được đặt là FALSE, cũng như cIdx làm đầu vào và đầu ra được gán cho availableL.

Vị trí (xNbA, yNbA) được đặt bằng nhau (x0, y0 – 1) và quy trình dẫn xuất đối với tính khả dụng khối lân cận như đã xác định được dẫn ra với vị trí (xCurr, yCurr) được đặt bằng nhau (x0, y0), vị trí lân cận (xNbY, yNbY) được đặt bằng nhau (xNbA, yNbA), checkPredModeY được đặt là FALSE, cũng như cIdx làm đầu vào và đầu ra được gán cho availableA.

Với condL và condA được xác định trong Bảng 12, việc gán ctxInc được xác định như sau:

Đối với phần tử cú pháp, alf_cross_component_cb_idc[x0][y0] và alf_cross_component_cr_idc[x0][y0]: ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3

Phần tử cú pháp	condL	condA	ctxSetIdx
alf_cross_component_cb_idc[x0][y0] chỉ số bin binIdx	(alf_cross_component_cb_idc[xNbL][yNbL] & (1 << binIdx)) == 0	(alf_cross_component_cb_idc[xNbA][yNbA] & (1 << binIdx)) == 0	binIdx
alf_cross_component_cr_idc[x0] [y0] chỉ số binIdx	(alf_cross_component_cr_idc[xNbL][yNbL] == & (1 << binIdx)) == 0	(alf_cross_component_cr_idc[xNbA][yNbA] & (1 << binIdx)) = 0	binIdx

Bảng 12

Theo một ví dụ, đối với **alf_cross_component_cb_idc** và/hoặc **alf_cross_component_cr_idc** tất cả bin có thể được mã hóa theo ngữ cảnh và tất cả bin sử dụng cùng một ngữ cảnh, nghĩa là cùng một ngữ cảnh cho từng binIdx. Trong một số trường hợp, không áp dụng tập hợp con của binIdx, ví dụ như NumCcAlfCbFilters là 3, binIdx>3 không áp dụng cho alf_cross_component_cb_idc[ ][ ], NumCcAlfCrFilters là 3, binIdx>3 không áp dụng cho alf_cross_component_cr_idc[ ][ ].

Theo một ví dụ, đối với **alf_cross_component_cb_idc** và/hoặc **alf_cross_component_cr_idc** tất cả bin có thể được mã hóa theo ngữ cảnh và mỗi bin có thể sử dụng cùng một tập hợp ngữ cảnh như được biểu thị trong Bảng 13. Đối với Bảng 13, cần lưu ý rằng trong một số trường hợp, tập hợp con của binIdx không được áp dụng, ví dụ như NumCcAlfCbFilters là 3, binIdx>3 không áp dụng cho alf_cross_component_cb_idc[ ][ ], NumCcAlfCrFilters là 3, binIdx>3 không áp dụng cho alf_cross_component_cr_idc[ ][ ].

Phần tử cú pháp	**binIdx**					
	0	**1**	**2**	**3**	**4**	**>= 5**
alf_cross_component_cb_idc[ ][ ]	0..2	0..2	0..2	0..2	0..2	0..2
alf_cross_component_cr_idc[ ][ ]	0..2	0..2	0..2	0..2	0..2	0..2

Bảng 13

Đối với Bảng 13, theo một ví dụ, dẫn xuất của ngữ cảnh có thể là như sau:

Đầu vào cho quy trình này là vị trí luma (x0, y0) xác định mẫu luma ở trên cùng bên trái của khối luma hiện tại so với mẫu trên cùng bên trái của hình ảnh hiện tại, thành phần màu cIdx, độ sâu cây tứ phân mã hóa hiện tại cqtDepth, loại kênh cây kép chType, chiều rộng và chiều cao của khối mã hóa hiện tại trong các mẫu luma cbWidth và cbHeight, cũng như các biến số allowSplitBtVer, allowSplitBtHor, allowSplitTtVer, allowSplitTtHor và allowSplitQt như được dẫn xuất trong ngữ nghĩa học cây mã hóa.

Đầu ra của quy trình này là ctxInc.

Vị trí (xNbL, yNbL) được đặt bằng nhau (x0 – 1, y0) và quy trình dẫn xuất đối với tính khả dụng khối lân cận như đã xác định được dẫn ra với vị trí (xCurr, yCurr) được đặt bằng nhau (x0, y0), vị trí lân cận (xNbY, yNbY) được đặt bằng nhau (xNbL, yNbL), checkPredModeY được đặt là FALSE, cũng như cIdx làm đầu vào và đầu ra được gán cho availableL.

Vị trí (xNbA, yNbA) được đặt bằng nhau (x0, y0 – 1) và quy trình dẫn xuất đối với tính khả dụng khối lân cận như đã xác định được dẫn ra với vị trí (xCurr, yCurr) được đặt bằng nhau (x0, y0), vị trí lân cận (xNbY, yNbY) được đặt bằng nhau (xNbA, yNbA), checkPredModeY được đặt là FALSE, cũng như cIdx làm đầu vào và đầu ra được gán cho availableA.

Với condL và condA được xác định trong Bảng 14, việc gán ctxInc được xác định như sau:

Đối với phần tử cú pháp alf_cross_component_cb_idc[x0][y0] và alf_cross_component_cr_idc[x0][y0]: ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3

Phần tử cú pháp	condL	condA	ctxSetIdx
alf_cross_component_cb_idc[x0][y0] chỉ số bin binIdx	(alfcross_component_cb_idc[xNbL][yNbL] & (1 << binIdx)) == 0	(alf_cross_component_cb_idc[xNbA][yNbA] & (1 << binIdx)) == 0	0
alf_cross_component_cr_idc[x0][y0] chỉ số bin binIdx	(alf_cross_component_cr_idc[xNbL][yNbL] == & (1 << binIdx)) == 0	(alf_cross_component_cr_idc[xNbA][yNbA] & (1 << binIdx)) == 0	0

Bảng 14

Như được cung cấp ở trên, đối với ví dụ tương ứng với Bảng 8 và Bảng 9, phần tử cú pháp **alf_cross_component_cb_idc**[][] và **alf_cross_component_cr_idc**[][] được báo hiệu trong coding_tree_unit() và mã nhị phân của **alf_cross_component_cb_idc**[][] và **alf_cross_component_cr_idc**[][] phụ thuộc vào giá trị tương ứng của các phần tử cú pháp **alf_cross_component_cb_filters_signalled_minus1** và**alf_cross_component_cr_filters_signalled_minus1.** Nếu**alf_cross_component_cb_filters_signalled_minus1** và**alf_cross_component_cr_filters_signalled_minus1** có trong APS, sự thiếu hụt APS chứa **alf_cross_component_cb_filters_signalled_minus1** và**alf_cross_component_cr_filters_signalled_minus1** tương ứng có nghĩa là phần tử cú pháp **alf_cross_component_cb_idc**[][] và **alf_cross_component_cr_idc**[][] không thể phân tách được. Theo một ví dụ, để giảm thiểu/ngăn chặn xảy ra tình trạng phần tử cú pháp không

phân tách được, các phần tử cú pháp xác định số lượng tập hợp bộ lọc Cb/Cr thành phần chéo mà hệ số và/hoặc sự nhị phân hóa đối với **alf_cross_component_cb_idc**[][] và **alf_cross_component_cr_idc**[][] có thể được báo hiệu trong một cấu trúc cú pháp khác. Ví dụ, theo một ví dụ, các phần tử cú pháp **slice_alf_cross_component_cb_filters_signalled_minus 1** và**slice_alf_cross_component_cr_filters_signalled_minus1** dựa trên ngữ nghĩa sau đây có thể được báo hiệu trong phần đầu mảnh và giá trị của chúng có thể được dùng để nhị phân hóa **alf_cross_component_cb_idc**[][] và **alf_cross_component_cr_idc**[][]. Nghĩa là, khi nhị phân hóa TR, đối với**alf_cross_component_cb_idc**[][], cMax bằng **slice_alf_cross_component_cb_filters_signalled_minus1**+1 và đối với **alf_cross_component_cr_idc**[][], cMax bằng **slice_alf_cross_component_cr_filters_signalled_minus1**+1. Cần lưu ý rằng theo một ví dụ, **slice_alf_cross_component_cb_filters_signalled_minus1** có thể có trong cấu trúc cú pháp phần đầu mảnh ngay sau đây**slice_cross_component_alf_cb_log2_control_size_minus4,** và**slice_alf_cross_component_cr_filters_signalled_minus1** có thể có trong cấu trúc cú pháp phần đầu mảnh ngay sau đây **slice_cross_component_alf_cr_log2_control_size_minus4.**

slice_cross_component_alf_cb_signalled_minus1 cộng 1 xác định giá trị cực đại cho phần tử cú pháp alf_cross_component_cb_idc[][].

slice_cross_component_alf_cr_signalled_minus1 cộng 1 xác định giá trị cực đại cho phần tử cú pháp alf_cross_component_cr_idc[][].

Cần lưu ý rằng mặc dù việc báo hiệu số lượng bộ lọc trong phần đầu mảnh loại bỏ phần phụ thuộc phân tách nhưng có thể có lợi ích khi đặt giới hạn cho số lượng phần tử cú pháp bộ lọc có trong phần đầu mảnh. Theo một ví dụ, có thể cần số lượng phần tử cú pháp bộ lọc trong phần đầu mảnh giống với số lượng bộ lọc được báo hiệu trong APS. Ngoài ra, theo một ví dụ, có thể không cần số lượng phần tử cú pháp bộ lọc trong phần đầu mảnh giống với số bộ lọc trong APS.

Khi không có yêu cầu về số lượng phần tử cú pháp bộ lọc trong phần đầu mảnh giống với số lượng bộ lọc trong APS, thì (1) trường hợp số lượng bộ lọc được báo hiệu trong phần đầu mảnh nhỏ hơn hoặc bằng số bộ lọc được báo hiệu trong APS và (2) trường hợp số bộ lọc được báo hiệu trong phần đầu mảnh lớn hơn số bộ lọc được báo hiệu trong APS. Khi số bộ lọc được báo hiệu trong phần đầu mảnh lớn hơn số bộ lọc được báo hiệu trong APS, bộ lọc phải

được xác định để chỉ số bộ lọc vượt quá số bộ lọc trong APS. Ví dụ: nếu có hai bộ lọc được báo hiệu trong APS, nhưng tín hiệu trong phần đầu mảnh chỉ báo có bốn bộ lọc thì bộ giải mã video phải có hành động xác định nếu thu được chỉ số bộ lọc 3 hoặc 4. Theo một ví dụ, bộ giải mã có thể vô hiệu hóa việc xử lý CC-ALF ở vùng có chỉ số bộ lọc 3 hoặc 4 (có thể bao gồm việc sử dụng bộ lọc có tất cả giá trị chạm bằng không). Theo một ví dụ khác, bộ giải mã video có thể sử dụng bộ lọc đã có trong APS. Ví dụ: có thể sử dụng bộ lọc có chỉ số 2, nếu thu được chỉ số bộ lọc 3 hoặc 4. Theo một ví dụ khác, bộ lọc có chỉ số 1 có thể được sử dụng nếu thu được chỉ số bộ lọc 3 và bộ lọc có chỉ số 2 có thể được sử dụng nếu thu được chỉ số bộ lọc 4. Theo một ví dụ khác, các bộ lọc cố định đã biết ở bộ mã hóa và bộ giải mã có thể được sử dụng khi thu được chỉ số bộ lọc 3 và chỉ số bộ lọc 4. Trong trường hợp số bộ lọc được báo hiệu trong phần đầu mảnh nhỏ hơn hoặc bằng số bộ lọc được báo hiệu trong APS, thì không phải xác định thêm hành động nào của bộ giải mã video. Tuy nhiên, người ta khẳng định rằng việc cho phép điều kiện này có thể mang đến lợi ích về hiệu quả mã hóa. Ví dụ: mảnh thứ nhất có thể tham chiếu APS thứ nhất và sử dụng tất cả bộ lọc M từ APS và mảnh thứ hai cũng có thể tham chiếu APS thứ nhất nhưng chỉ sử dụng bộ lọc N thứ nhất trong APS. Nhờ đó, hiệu quả mã hóa sẽ được cải thiện vì (i) không cần thiết phải gửi APS thứ hai cho mảnh thứ hai và (ii) các bit cần để báo hiệu chỉ số bộ lọc sẽ ít hơn trong mảnh thứ hai do số lượng bộ lọc ít hơn.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc triển khai lọc thành phần chéo có thể dựa trên cú pháp và ngữ nghĩa sau đây. Đối với cú pháp và ngữ nghĩa sau, trong Bảng 15, các phần tử cú pháp **alf_cross_component_cb_filter_signal_flag, alf_cross_component_cr_filter_signal_flag, alf_cross_component_cb_coeff_abs, alf_cross_component_cb_coeff_sign, alf_cross_component_cr_coeff_abs,** và **alf_cross_component_cr_coeff_sign,** được thêm vào cấu trúc cú pháp alf_data() được cung cấp trong JVET-O2001. Cần lưu ý rằng cấu trúc cú pháp alf_data() trong JVET-O2001 được cung cấp trong cấu trúc cú pháp tập hợp thông số thích ứng. Trong Bảng 16, các phần tử cú pháp **slice_cross_component_alf_cb_enabled_flag, slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag, slice_cross_component_alf_cb_aps_id, slice_cross_component_alf_cb_log2_control_size_minus4, slice_cross_component_alf_cr_enabled_flag, slice_cross_component_alf_cr_reuse_temporal_layer_filter_flag, slice_cross_component_alf_cr_aps_id,** và

slice_cross_component_alf_cb_log2_control_size_minus4 được thêm vào cấu trúc cú pháp slice_header() được cung cấp trong JVET-O2001. Trong Bảng 17, các phần tử cú pháp **alf_cross_component_cb_flag** and **alf_cross_component_cr_flag** được thêm vào cấu trúc cú pháp coding_tree_unit() được cung cấp trong JVET-O2001.

alf_data( ) {	**Bộ mô tả**
alf_luma_filter_signal_flag	u(1)
alf_chroma_filter_signal_flag	u(1)
alf_cross_component_cb_filter_signal_flag	u(1)
alf_cross_component_cr_filter_signal_flag	u(1)
if(alf_luma_filter_signal_flag) {	
alf_luma_clip_flag	u(1)
alf_luma_num_filters_signalled_minus1	ue(v)
if(alf_luma_num_filters_signalled_minus1 > 0) {	
for(filtIdx = 0; filtIdx < NumAlfFilters; filtIdx++)	
alf_luma_coeff_delta_idx[filtIdx]	u(v)
}	
alf_luma_coeff_signalled_flag	u(1)
if(alf_luma_coeff_signalled_flag) {	
for(sfIdx = 0; sfIdx <= alf_luma_num_filters_signalled_minus1; sfIdx++)	
alf_luma_coeff_flag[sfIdx]	u(1)
}	
for(sfIdx = 0; sfIdx <= alf_luma_num_filters_signalled_minus1; sfIdx++) {	
if(alf_luma_coeff_flag[sfIdx]) {	
for (j = 0;j < 12;j++) {	
alf_luma_coeff_abs[sfIdx][j]	uek(v)
if(alf_luma_coeff_abs[sfIdx][j])	
alf_luma_coeff_sign[sfIdx][j]	u(1)
}	
}	
}	
if(alf_luma_clip_flag) {	

for(sfIdx = 0; sfIdx <= alf_luma_num_filters_signalled_minus1; sfIdx++) {	
if(aif_luma_coeff_flag[sfIdx]) {	
for(j = 0; j < 12; j++)	
alf_luma_clip_idx[sfIdx][j]	u(2)
}	
}	
}	
}	
if(alf_chroma_filter_signal_flag) {	
alf_chroma_num_alt_filters_minus1	ue(v)
for(altIdx = 0; altIdx <= alf_chroma_num_alt_filters_minus1; altIdx++) {	
alf_chroma_clip_flag[altIdx]	u(1)
for(j = 0; j < 6; j++) {	
alf_chroma_coeff_abs[altIdx][j]	uek(v)
if(alf_chroma_coeff_abs[altIdx][j] > 0)	
alf_chroma_coeff_sign[altIdx][j]	u(1)
}	
if(alf_chroma_clip_flag[altIdx]) {	
for(j = 0; j < 6; j++)	
alf_chroma_clip_idx[altIdx][j]	u(2)
}	
}	
}	
if (alfcross_component_cb_filter_signal_flag)	
for (j = 0;j < 14; j++)	
alf_cross_component_cb_coeff_abs[j]	uek(v)
if(alf_cross_component_cb_coeff_abs[j])	
alf_cross_component_cb_coeff_sign[j]	u(1)
if (alf_cross_component_cr_filter_signal_flag)	
for(j =0;j < 14; j++)	
alf_cross_component_cr_coeff_abs[j]	uek(v)
if(alf_cross_component_cr_coeff_abs[j])	

alf_cross_component_cr_coeff_sign[j]	u(1)
}	

Bảng 15

slice_header( ) {	**Bộ mô tả**
slice_pic_parameter_set_id	ue(v)
...	
if (slice_type != I) {	
...	
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
...	
}	
...	
slice_qp_delta	se(v)
...	
if(sps_sao_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(sps_alf_enabled_flag) {	
slice_alf_enabled_flag	u(1)
if(slice_alf_enabled_flag) {	
slice_num_alf_aps_ids_luma	u(3)
for(i = 0; i < slice_num_alf_aps_ids_luma; i++)	
slice_alf_aps_id_luma[i]	u(3)
if(ChromaArrayType != 0)	
slice_alf_chroma_idc	u(2)
if(slice_alf_chroma_idc)	
slice_alf_aps_id_chroma	u(3)
}	
if(ChromaArrayType != 0)	

slice_cross_component_alf_cb_enabled_flag	u(1)
if(slice_cross_component_alf_cb_enabled_flag) {	
slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag	u(1)
if (!slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag)	
slice_cross_component_alf_cb_aps_id	u(3)
slice_cross_component_alf_cb_log2_control_size_minus4	ue(v)
}	
if(ChromaArrayType != 0)	
slice_cross_component_alf_cr_enabled_flag	u(1)
if(slice_cross_component_alf_cr_enabled_flag) {	
slice_cross_component_alf_cr_reuse_temporal_layer_filter_flag	u(1)
if (!slice_cross_component_alf_cr_reuse_temporal_layer_filter_flag)	
slice_cross_component_alf_cr_aps_id	u(3)
slice_cross_component_alf_cr_log2_control_size_minus4	ue(v)
}	
}	
...	
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
if(sps_lmcs_enabled_flag) {	
slice_lmcs_enabled_flag	u(1)
if(slice_lmcs_enabled_flag) {	
slice_lmcs_aps_id	u(2)
if(ChromaArrayType != 0)	
slice_chroma_residual_scale_flag	u(1)
}	

}	
...	

byte_alignment()	
}	

Bảng 16

coding_tree_unit() {	Bộ mô tả
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
if(slice_sao_luma_flag | | slice_sao_chroma_flag)	
sao(xCtb >> CtbLog2SizeY, yCtb >> CtbLog2SizeY)	
if(slice_alf_enabled_flag){	
alf_ctb_flag[0][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
if(alf_ctb_flag[0][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]) {	
if(slice_num_alf_aps_ids_luma > 0)	
alf_ctb_use_first_aps_flag	ae(v)
if(!alf_ctb_use_first_aps_flag) {	
if(slice_num_alf_aps_ids_luma > 1)	
alf_use_aps_flag	ae(v)
if(alf_use_aps_flag) {	
if(slice_num_alf_aps_ids_luma > 2)	
alf_luma_prev_filter_idx_minus1	ae(v)
} else	
alf_luma_fixed_filter_idx	ae(v)
}	
}	
if(slice_alf_chroma_idc = = 1 | | slice_alf_chroma_idc = = 3) {	
alf_ctb_flag[1][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
if(alf_ctb_flag[1][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] && aps_alf_chroma_num_alt_filters_minus1 > 0)	

alf_ctb_filter_alt_idx[0][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
}	
if(slice_alf_chroma_idc = = 2 | | slice_alf_chroma_idc = = 3) {	
alf_ctb_flag[2][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
if(alf_ctb_flag[2][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] && aps_alf_chroma_num_alt_filters_minus1 > 0)	
alf_ctb_filter_alt_idx[1][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
}	
}	
xEnd = xCtb + (1 << CtbLog2SizeY)	
xEnd = (xEnd >= pic_width_in_luma_samples) ? pic_width_in_luma_samples : xEnd	
yEnd = yCtb + (1 << CtbLog2SizeY)	
yEnd = (yEnd >= pic_height_in_luma_samples) ? pic_height_in_luma_samples : yEnd	
if (slice_cross_component_alf_cb_enabled_flag)	
for (yL = yCtb; yL < yEnd; yL += CcAlfHeightCbL)	
for (xL = xCtb; xL < xEnd; xL += CcAifWidthCbL)	
alf_cross_component_cb_flag[xL / CcAlfWidthCbL][yL / CcAlfHeightCbL]	ae(v)
if (slice_cross_component_alf_cr_enabled_flag)	
for (yL = yCtb; yL < yEnd; yL += CcAlfHeightCrL)	
for (xL = xStartC; xL < xEndC; xL += CcAlfWidthCrL)	
alf_cross_component_cr_flag[xL / CcAlfWidthCrL][yL / CcAlfHeightCrL]	ae(v)
if(slice_type = = I && qtbtt_dual_tree_intra_flag)	
dual_tree_implicit_qt_split (xCtb, yCtb, CtbSizeY, 0)	
khác	
coding_tree(xCtb, yCtb, CtbSizeY, CtbSizeY, 1, 1, 0, 0, 0, 0, 0, SINGLE_TREE, MODE_TYPE_ALL)	
}	

Bảng 17

Đối với Bảng 15, theo một ví dụ, ngữ nghĩa học có thể dựa trên các điểm sau:

alf_luma_filter_signal_flag bằng 1 xác định rằng tập hợp bộ lọc luma được báo hiệu. alf_luma_filter_signal_flag bằng 0 xác định rằng tập hợp bộ lọc luma không được báo hiệu.

alf_chroma_filter_signal_flag bằng 1 xác định rằng bộ lọc sắc độ được báo hiệu. alf_chroma_filter_signal_flag bằng 0 xác định rằng bộ lọc sắc độ không được báo hiệu. Khi ChromaArrayType bằng 0 thì alf_chroma_filter_signal_flag sẽ bằng 0.

Biến NumAlfFilters xác định số lượng các bộ lọc vòng thích ứng khác nhau được đặt bằng 25.

alf_cross_component_cb_filter_signal_flag bằng 1 xác định rằng bộ lọc Cb thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 xác định rằng bộ lọc Cb thành phần chéo không được báo hiệu. Khi ChromaArrayType bằng 0 thì alf_cross_component_cb_filter_signal_flag sẽ bằng 0.

alf_cross_component_cr_filter_signal_flag bằng 1 xác định rằng bộ lọc Cr thành phần chéo được báo hiệu, alf_cross_component_cr_filter_signal_flag bằng 0 xác định rằng bộ lọc Cr thành phần chéo không được báo hiệu. Khi ChromaArrayType bằng 0 thì alf_cross_component_cr_filter_signal_flag sẽ bằng 0.

alf_luma_clip_flag bằng 0 xác định rằng quy trình lọc vòng thích ứng tuyến tính trên thành phần luma sẽ được áp dụng. alf_luma_clip_flag bằng 1 xác định rằng quy trình lọc vòng thích ứng không tuyến tính có thể được áp dụng trên thành phần luma.

alf_luma_num_filters_signalled_minus1 cộng 1 xác định số lớp bộ lọc vòng thích ứng mà hệ số luma có thể được báo hiệu. Giá trị của alf_luma_num_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến NumAlfFilters – 1, bao gồm cả hai khoảng này.

alf_luma_coeff_delta_idx[filtIdx] xác định các chỉ số của delta hệ số luma của bộ lọc vòng thích ứng được báo hiệu cho lớp bộ lọc được biểu thị bởi filtIdx, nằm trong khoảng từ 0 đến NumAlfFilters – 1. Khi alf_luma_coeff_delta_idx[filtIdx] không hiện diện, nó sẽ được suy ra là bằng 0. Độ dài các bit alf_luma_coeff_delta_idx[filtIdx] là Ceil(Log2(alf_luma_num_filters_signalled_minus1 + 1)).

alf_luma_coeff_signalled_flag bằng 1 biểu thị rằng alf_luma_coeff_flag[sfIdx] được báo hiệu. alf_luma_coeff_signalled_flag bằng 0 biểu thị rằng alf_luma_coeff_flag[sfIdx] không được báo hiệu.

alf_luma_coeff_flag[sfIdx] bằng 1 xác định rằng các hệ số của bộ lọc luma được biểu thị bởi sfIdx được báo hiệu. alf_luma_coeff_flag[sfIdx] bằng 0 xác định rằng tất cả hệ số bộ lọc của bộ lọc luma được biểu thị bởi sfIdx đều được đặt bằng 0. Khi không hiện diện, alf_luma_coeff_flag[sfIdx] được đặt bằng 1.

alf_luma_coeff_abs[sfIdx][j] xác định trị số tuyệt đối của hệ số thứ j trong bộ lọc luma đã báo hiệu được sfIdx chỉ báo. Khi alf_luma_coeff_abs[sfIdx][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

alf_luma_coeff_sign[sfIdx][j] xác định ký hiệu của hệ số luma thứ j trong bộ lọc được biểu thị bởi sfIdx như sau:

- Nếu alf_luma_coeff_sign[sfIdx] [j] bằng 0, hệ số bộ lọc luma tương ứng có giá trị dương.
- Nếu không (alf_luma_coeff_sign[sfIdx][j] bằng 1), hệ số bộ lọc luma tương ứng có giá trị âm.

Khi alf_luma_coeff_sign[sfIdx][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Biến filtCoeff[sfIdx][j] có sfIdx = 0..alf_luma_num_filters_signalled_minus1, j = 0..11 được khởi tạo như sau:

filtCoeff[sfIdx][j] = alf_luma_coeff_abs[sfIdx][j] * (1 – 2 * alf_luma_coeff_sign[sfIdx] [j])

Hệ số bộ lọc luma $AlfCoeff_L$[adaptation_parameter_set_id] có các phần tử $AlfCoeff_L$[adaptation_parameter_set_id][filtIdx][j], với filtIdx = 0..NumAlfFilters – 1 và j = 0..11 được dẫn xuất như sau:

$AlfCoeff_L$[adaptation_parameter_set_id][filtIdx][j] = filtCoeff[alf_luma_coeff_delta_idx[filtIdx]][j]

Hệ số bộ lọc cố định AlfFixFiltCoeff[i][j] với i = 0..64, j = 0..11 và lớp để lọc ánh xạ AlfClassToFiltMap[m][n], với m = 0..15 và n = 0..24 được dẫn xuất như sau:

AlfFixFiltCoeff=

{

{ 0, 0, 2, –3, 1, –4, 1, 7, –1, 1, –1, 5}

{ 0, 0, 0, 0, 0, –1, 0, 1, 0, 0, –1, 2}

{ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0}

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 1}

{ 2, 2, –7, –3, 0, –5, 13, 22, 12, –3, –3, 17}

{–1, 0, 6, –8, 1, –5, 1, 23, 0, 2, –5, 10}

{0, 0, –1, –1, 0, –1, 2, 1, 0, 0, –1, 4}

{ 0, 0, 3, –11, 1, 0, –1, 35, 5, 2, –9, 9}

{ 0, 0, 8, –8, –2, –7, 4, 4, 2, 1, –1, 25}

{ 0, 0, 1, –1, 0, –3, 1, 3, –1, 1, –1, 3}

{0, 0, 3, –3, 0, –6, 5, –1, 2, 1, –4, 21}

{– 7, 1, 5, 4, –3, 5, 11, 13, 12, –8, 11, 12}

{–5, –3, 6, –2, –3, 8, 14, 15, 2, –7, 11, 16}

{ 2, –1, –6, –5, –2, –2, 20, 14, –4, 0, –3, 25}

{ 3, 1, –8, –4, 0, –8, 22, 5, –3, 2, –10, 29}

{ 2, 1, –7, –1, 2, –11, 23, –5, 0, 2, –10, 29}

{–6, –3, 8, 9, –4, 8, 9, 7, 14, –2, 8, 9}

{ 2, 1, –4, –7, 0, –8, 17, 22, 1, –1, –4, 23}

{ 3, 0, –5, –7, 0, –7, 15, 18, –5, 0, –5, 27}

{ 2, 0, 0, –7, 1, –10, 13, 13, –4, 2, –7, 24}

{ 3, 3, –13, 4, –2, –5, 9, 21, 25, –2, –3, 12}

{–5, –2, 7, –3, –7, 9, 8, 9, 16, –2, 15, 12}

{ 0, –1, 0, –7, –5, 4, 11, 11, 8, –6, 12, 21}

{3, –2, –3, –8, –4, –1, 16, 15, –2, –3, 3, 26}

{ 2, 1, –5, –4, –1, –8, 16, 4, –2, 1, –7, 33}

{ 2, 1, –4, –2, 1, –10, 17, –2, 0, 2, –11, 33}

{ 1, –2, 7, –15, –16, 10, 8, 8, 20, 11, 14, 11}

{ 2, 2, 3, –13, –13, 4, 8, 12, 2, –3, 16, 24}

{ 1, 4, 0, –7, –8, –4, 9, 9, –2, –2, 8, 29}

{ 1, 1, 2, –4, –1, –6, 6, 3, –1, –1, –3, 30}

{–7, 3, 2, 10, –2, 3, 7, 11, 19, –7, 8, 10}

{ 0, –2, –5, –3, –2, 4, 20, 15, –1, –3, –1, 22}

{ 3, –1, –8, –4, –1, –4, 22, 8, –4, 2, –8, 28}

{ 0, 3, –14, 3, 0, 1, 19, 17, 8, –3, –7, 20}

{ 0, 2, –1, –8, 3, –6, 5, 21, 1, 1, –9, 13}

{–4, –2, 8, 20, –2, 2, 3, 5, 21, 4, 6, 1}

{ 2, –2, –3, –9, –4, 2, 14, 16, 3, –6, 8, 24}

{ 2, 1, 5, –16, –7, 2, 3, 11, 15, –3, 11, 22}

{ 1, 2, 3, –11, –2, –5, 4, 8, 9, –3, –2, 26}

{ 0, –1, 10, –9, –1, –8, 2, 3, 4, 0, 0, 29}

{ 1, 2, 0, –5, 1, –9, 9, 3, 0, 1, –7, 20}

{–2, 8, –6, –4, 3, –9, –8, 45, 14, 2, –13, 7}

{ 1, –1, 16, –19, –8, –4, –3, 2, 19, 0, 4, 30}

{ 1, 1, –3, 0, 2, –11, 15, –5, 1, 2, –9, 24}

{ 0, 1, –2, 0, 1, –4, 4, 0, 0, 1, –4, 7}

{ 0, 1, 2, –5, 1, –6, 4, 10, –2, 1, –4, 10}

{ 3, 0, –3, –6, –2, –6, 14, 8, –1, –1, –3, 31}

{ 0, 1, 0, –2, 1, –6, 5, 1, 0, 1, –5, 13}

{ 3, 1, 9, –19, –21, 9, 7, 6, 13, 5, 15, 21}

{ 2, 4, 3, –12, –13, 1, 7, 8, 3, 0, 12, 26}

{ 3, 1, –8, –2, 0, –6, 18, 2, –2, 3,–10, 23}

{ 1, 1, –4, –1, 1, –5, 8, 1, –1, 2, –5, 10}

{ 0, 1, –1, 0, 0, –2, 2, 0, 0, 1, –2, 3}

{ 1, 1, –2, –7, 1, –7, 14, 18, 0, 0, –7, 21}

{ 0, 1, 0, –2, 0, –7, 8, 1, –2, 0, –3, 24}

{ 0, 1, 1, –2, 2,–10, 10, 0, –2, 1, –7, 23}

{ 0, 2, 2, –11, 2, –4, –3, 39, 7, 1, –10, 9}

{ 1, 0, 13, –16, –5, –6, –1, 8, 6, 0, 6, 29}

{ 1, 3, 1, –6, –4, –7, 9, 6, –3, –2, 3, 33}

{ 4, 0, –17, –1, –1, 5, 26, 8, –2, 3, –15, 30}

{ 0, 1, –2, 0, 2, –8, 12, –6, 1, 1, –6, 16}

{ 0, 0, 0, –1, 1, –4, 4, 0, 0, 0, –3, 11}

{ 0, 1, 2, –8, 2, –6, 5, 15, 0, 2, –7, 9}

{ 1, –1, 12, –15, –7, –2, 3, 6, 6, –1, 7, 30}

},

AlfClassToFiltMap =

{

{ 8, 2, 2, 2, 3, 4, 53, 9, 9, 52, 4, 4, 5, 9, 2, 8, 10, 9, 1, 3, 39, 39, 10, 9, 52}

{ 11, 12, 13, 14, 15, 30, 11, 17, 18, 19, 16, 20, 20, 4, 53, 21, 22, 23, 14, 25, 26, 26, 27, 28, 10 }

{ 16, 12, 31, 32, 14, 16, 30, 33, 53, 34, 35, 16, 20, 4, 7,16, 21, 36, 18, 19, 21, 26, 37, 38, 39 }

{ 35, 11, 13, 14, 43, 35, 16, 4, 34, 62, 35, 35, 30, 56, 7, 35, 21, 38, 24, 40, 16, 21, 48, 57, 39 }

{ 11, 31, 32, 43, 44, 16, 4, 17, 34, 45, 30, 20, 20, 7, 5, 21, 22, 46, 40, 47, 26, 48, 63, 58, 10 }

{ 12, 13, 50, 51, 52, 11, 17, 53, 45, 9, 30, 4, 53, 19, 0, 22, 23, 25, 43, 44, 37, 27, 28, 10, 55 }

{30, 33, 62, 51, 44, 20, 41, 56, 34, 45, 20, 41, 41, 56, 5, 30, 56, 38, 40, 47, 11, 37, 42, 57, 8 }

{ 35, 11, 23, 32, 14, 35, 20, 4, 17, 18, 21, 20, 20, 20, 4, 16, 21, 36, 46, 25, 41, 26, 48, 49, 58 }

{ 12, 31, 59, 59, 3, 33, 33, 59, 59, 52, 4, 33, 17, 59, 55, 22, 36, 59, 59, 60, 22, 36, 59, 25, 55 }

{ 31, 25, 15, 60, 60, 22, 17, 19, 55, 55, 20, 20, 53, 19, 55, 22, 46, 25, 43, 60, 37, 28, 10, 55, 52 }

{ 12, 31, 32, 50, 51, 11, 33, 53, 19, 45, 16, 4, 4, 53, 5, 22, 36, 18, 25, 43, 26, 27, 27, 28, 10 }

{ 5, 2, 44, 52, 3, 4, 53, 45, 9, 3, 4, 56, 5, 0, 2, 5, 10, 47, 52, 3, 63, 39, 10, 9, 52}

{ 12, 34, 44, 44, 3, 56, 56, 62, 45, 9, 56, 56, 7, 5, 0, 22, 38, 40, 47, 52, 48, 57, 39, 10, 9}

{ 35, 11, 23, 14, 51, 35, 20, 41, 56, 62, 16, 20, 41, 56, 7, 16, 21, 38, 24, 40, 26, 26, 42, 57, 39 }

{ 33, 34, 51, 51, 52, 41, 41, 34, 62, 0, 41, 41, 56, 7, 5, 56, 38, 38, 40, 44, 37, 42, 57, 39, 10 }

{ 16, 31, 32, 15, 60, 30, 4, 17, 19, 25, 22, 20, 4, 53, 19, 21, 22, 46, 25, 55, 26, 48, 63, 58, 55 }

},

Cần có sự tương thích chuỗi bit mà các giá trị của $AlfCoeff_L$[adaptation_parameter_set_id][filtIdx][j], với filtIdx = 0..NumAlfFilters – 1, j = 0..11 sẽ nằm trong khoảng từ -2^7 đến $2^7 - 1$, bao gồm cả hai khoảng này.

alf_luma_clip_idx[sfIdx][j] xác định chỉ số cắt của giá trị cắt để sử dụng trước khi nhân với hệ số thứ j của bộ lọc luma đã báo hiệu được biểu thị bởi sfIdx. Cần có sự tương thích chuỗi bit mà các giá trị của alf_luma_clip_idx[sfIdx][j], với sfIdx = 0..alf_luma_num_filters_signalled_minus1 và j = 0.. 11 sẽ nằm trong khoảng từ 0 đến 3, bao gồm cả hai khoảng này.

Các giá trị cắt của bộ lọc luma $AlfClip_L$[adaptation_parameter_set_id] có các phần tử $AlfClip_L$[adaptation_parameter_set_id][filtIdx][j], với filtIdx = 0..NumAlfFilters – 1 và j = 0..11 được suy ra như được xác định trong Bảng 18, tùy vào bitDepth được đặt bằng $BitDepth_Y$ và clipIdx được đặt bằng alf_luma_clip_idx[alf_luma_coeff_delta_idx[filtIdx]][j].

alf_chroma_num_alt_filters_minus1 cộng 1 xác định số lượng bộ lọc thay thế cho thành phần sắc độ.

alf_chroma_clip_flag[altIdx] bằng 0 xác định rằng quy trình lọc vòng thích ứng tuyến tính được áp dụng trên thành phần sắc độ khi sử dụng bộ lọc sắc độ có chỉ số altIdx; alf_chroma_clip_flag[altIdx] bằng 1 xác định rằng quy trình lọc vòng thích ứng phi tuyến tính được áp dụng trên thành phần sắc độ khi sử dụng bộ lọc sắc độ có chỉ số altIdx. Khi không hiện diện, alf_chroma_clip_flag[altIdx] được suy ra bằng 0.

alf_chroma_coeff_abs[altIdx][j] xác định trị số tuyệt đối của hệ số bộ lọc sắc độ thứ j cho bộ lọc sắc độ thay thế có chỉ số altIdx. Khi alf_chroma_coeff_abs[altIdx][j] không hiện diện, nó sẽ được suy ra là bằng 0. Cần có sự tương thích chuỗi bit mà các giá trị của alf_chroma_coeff_abs[altIdx] [j] sẽ nằm trong khoảng từ 0 đến $2^7 – 1$, bao gồm cả hai khoảng này.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

alf_chroma_coeff_sign[altIdx][j] xác định ký hiệu của hệ số bộ lọc sắc độ thứ j cho bộ lọc sắc độ thay thế với chỉ số altIdx như sau:

- Nếu alf_chroma_coeff_sign[altIdx][j] bằng 0, hệ số bộ lọc sắc độ tương ứng có giá trị dương.
- Nếu không (alf_chroma_coeff_sign[altIdx] [j] bằng 1), hệ số bộ lọc sắc độ tương ứng có giá trị âm.

Khi alf_chroma_coeff_sign[altIdx][j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc sắc độ $AlfCoeff_C$[adaptation_parameter_set_id][altIdx] có các phần tử $AlfCoeff_C$[adaptation_parameter_set_id][altIdx][j], với altIdx = 0..alf_chroma_num_alt_filters_minus1, j = 0..5 được dẫn xuất như sau:

$$AlfCoeff_C[\ adaptation\_parameter\_set\_id\ ][\ altIdx\ ][\ j\ ] = alf\_chroma\_coeff\_abs[\ altIdx\ ][\ j\ ] * (\ 1 - 2 * alf\_chroma\_coeff\_sign[\ altIdx\ ][\ j\ ]\ )$$

Cần có sự tương thích chuỗi bit mà các giá trị của $AlfCoeff_C[\ adaptation\_parameter\_set\_id\ ][\ altIdx\ ][\ j\ ]$ với altIdx = 0..alf_chroma_num_alt_filters_minus1, j = 0..5 sẽ nằm trong khoảng từ $-2^7 - 1$ đến $2^7 - 1$, bao gồm cả hai khoảng này.

alf_cross_component_cb_coeff_sign[j] xác định trị số tuyệt đối của hệ số bộ lọc Cb thành phần chéo thứ j. Khi alf_cross_component_cb_coeff_abs[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

alf_cross_component_cb_coeff_sign[j] xác định ký hiệu của hệ số bộ lọc Cb thành phần chéo thứ j như sau:

- Nếu alf_cross_component_cb_coeff_sign[j] bằng 0, hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_cb_sign[j] bằng 1), hệ số bộ lọc Cb thành phần chéo tương ứng có giá trị âm.

Khi alf_cross_component_cb_coeff_sign[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cb thành phần chéo $CcAlfApsCoeff_{Cb}[\ adaptation\_parameter\_set\_id\ ]$ với phần tử $CcAlfApsCoeff_{Cb}[\ adaptation\_parameter\_set\_id\ ][\ j\ ]$, với j = 0..13 được suy ra như sau:

$$CcAlfApsCoeff_{Cb}[\ adaptation\_parameter\_set\_id\ ][\ j\ ] = alf\_cross\_component\_cb\_coeff\_abs[\ j\ ] * (\ 1 - 2 * alf\_cross\_component\_cb\_coeff\_sign[\ j\ ]\ )$$

Cần có sự tương thích chuỗi bit mà các giá trị của $CcAlfApsCoeff_{Cb}[\ adaptation\_parameter\_set\_id\ ][\ j\ ]$ với j = 0..13 sẽ nằm trong khoảng từ -2^7 đến $2^7 - 1$, bao gồm cả hai khoảng này.

alf_cross_component_cr_coeff_abs[j] xác định trị số tuyệt đối của hệ số bộ lọc Cr thành phần chéo thứ j. Khi alf_cross_component_cr_coeff_abs[j] không hiện diện, nó sẽ được suy ra là bằng 0. Bậc k của uek(v) nhị phân hóa Golomb hàm số mũ được dẫn xuất như sau:

alf_cross_component_cr_coeff_sign[j] xác định ký hiệu của hệ số bộ lọc Cr thành phần chéo thứ j như sau:

- Nếu alf_cross_component_cr_coeff_sign[j] bằng 0, hệ số bộ lọc Cr thành phần chéo tương ứng có giá trị dương.
- Nếu không (alf_cross_component_cr_sign[j] bằng 1), hệ số bộ lọc cross_component_Cr tương ứng có giá trị âm.

Khi alf_cross_component_cr_coeff_sign[j] không hiện diện, nó sẽ được suy ra là bằng 0.

Hệ số bộ lọc Cr thành phần chéo $CcAlfApsCoeff_{Cr}$[adaptation_parameter_set_id] có các phần tử $CcAlfApsCoeff_{Cr}$[adaptation_parameter_set_id][j], với j = 0..13 được dẫn xuất như sau:

$$CcAlfApsCoeff_{Cr}[\text{ adaptation\_parameter\_set\_id }][\text{ j }] = \text{alf\_cross\_component\_cr\_coeff\_abs[j]} * ((1 - 2 * \text{alf\_cross\_component\_cr\_coeff\_sign[j]})$$

Cần có sự tương thích chuỗi bit mà các giá trị của $CcAlfApsCoeff_{Cr}$[adaptation_parameter_set_id][j] với j = 0..13 sẽ nằm trong khoảng từ -2^7 đến $2^7 - 1$, bao gồm cả hai khoảng này.

alf_chroma_clip_idx[altIdx] [j] xác định chỉ số cắt của giá trị cắt để sử dụng trước khi nhân với hệ số thứ j của bộ lọc sắc độ thay thế có chỉ số altIdx. Cần có sự tương thích chuỗi bit mà các giá trị của alf_chroma_clip_idx[altIdx][j] với altIdx = 0..alf_chroma_num_alt_filters_minus1, j = 0..5 sẽ nằm trong khoảng từ 0 đến 3, bao gồm cả hai khoảng này.

Các giá trị cắt của bộ lọc sắc độ $AlfClip_C$[adaptation_parameter_set_id][altIdx] với phần tử $AlfClip_C$[adaptation_parameter_set_id][altIdx][j], với altIdx =

0..alf_chroma_num_alt_filters_minus1, j = 0..5 được suy ra như được xác định trong Bảng 18, tùy vào onbitDepth được đặt bằng BitDepthC và clipIdx được đặt bằng alf_chroma_clip_idx[altIdx][j].

bitDepth	clipIdx			
	0	1	2	3
8	255	64	16	4
9	511	108	23	5
10	1023	181	32	6
11	2047	304	45	7
12	4095	512	64	8
13	8191	861	91	10
14	16383	1448	128	11
15	32767	2435	181	13
16	65535	4096	256	16

Bảng 18

Ngoài ra, đối với Bảng 15, cần lưu ý rằng JVET-2001 đề xuất cú pháp và ngữ nghĩa học sau cho cấu trúc cú pháp adaptation_parameter_set:

adaptation_parameter_set_rbsp() {	**Bộ mô tả**
adaptation_parameter_set_id	u(5)
aps_params_type	u(3)
if(aps_params_type = = ALF_APS)	
alf_data()	
else if(aps_params_type = = LMCS_APS)	
lmcs_data()	
else if(aps_params_type = = SCALING_APS)	
scaling_list_data()	
aps_extension_flag	u(1)
if(aps_extension_flag)	
while(more_rbsp_data( ))	
aps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

Bảng 19

Mỗi APS RBSP sẽ có sẵn cho quá trình giải mã trước khi được tham chiếu, có trong ít nhất một khối truy cập với TemporalId nhỏ hơn hoặc bằng TemporalId của đơn vị NAL mảnh đã mã hóa tham chiếu đến nó hoặc được cung cấp thông qua các phương tiện bên ngoài.

Đặt aspLayerId là nuh_layer_id của đơn vị APS NAL. Nếu lớp có nuh_layer_id bằng với aspLayerId là một lớp riêng (tức là, vps_independent_layer_flag[GeneralLayerIdx[aspLayerId]] bằng 1), đơn vị APS NAL chứa APS RBSP sẽ có nuh_layer_id bằng với nuh_layer_id của đơn vị NAL mảnh được mã hóa tham chiếu đơn vị này. Mặt khác, đơn vị APS NAL chứa APS RBSP sẽ có nuh_layer_id bằng với nuh_layer_id của đơn vị NAL mảnh đã mã hóa tham chiếu nó hoặc bằng với nuh_layer_id của lớp phụ thuộc trực tiếp trong lớp chứa đơn vị NAL mảnh đã mã hóa tham chiếu nó.

Tất cả đơn vị APS NAL với một giá trị adaptation_parameter_set_id cụ thể và một giá trị aps_params_type cụ thể trong khối truy cập sẽ có cùng nội dung.

adaptation_parameter_set_id cung cấp mã nhận dạng cho APS để tham chiếu theo các phần tử cú pháp khác. Khi aps_params_type bằng ALF_APS hoặc SCALING_APS, giá trị của adaptation_parameter_set_id sẽ nằm trong khoảng từ 0 đến 7, bao gồm cả 2 số này.

Khi aps_params_type bằng LMCS_APS, giá trị của adaptation_parameter_set_id sẽ nằm trong khoảng từ 0 đến 3, bao gồm cả hai số này.

aps_params_type xác định loại thông số APS có trong APS như được xác định trong Bảng 20. Khi aps_params_type bằng 1 (LMCS_APS), giá trị của adaptation_parameter_set_id sẽ nằm trong khoảng từ 0 đến 3, bao gồm cả 2 số này.

aps_params_type	Tên của aps_params_type	Loại thông số APS
0	ALF_APS	Thông số ALF
1	LMCS_APS	Thông số LMCS
2	SCALING_APS	Thông số trong danh sách theo tỷ lệ
3..7	Dành riêng	Dành riêng

Bảng 20

LƯU Ý – Mỗi loại APS sử dụng không gian giá trị riêng cho adaptation_parameter_set_id.

LƯU Ý – Đơn vị APS NAL (có giá trị adap_parameter_set_id cụ thể và giá trị aps_params_type cụ thể) có thể được chia sẻ trên các hình ảnh, và các mảnh khác nhau trong hình ảnh có thể tham chiếu đến các ALF APS khác nhau.

pps_extension_flag bằng 0 xác định rằng không có phần tử cú pháp aps_extension_data_flag trong cấu trúc cú pháp APS RBSP. aps_extension_flag bằng 1 xác định rằng có phần tử cú pháp aps_extension_data_flag trong cấu trúc cú pháp APS RBSP.

aps_extension_data_flag có thể có giá trị bất kỳ. Sự hiện diện và giá trị của nó không ảnh hưởng đến sự tương thích của bộ giải mã đối với các cấu hình được quy định trong trường hợp của Bản đặc tả này. Bộ giải mã tuân theo trường hợp của Bản đặc tả này sẽ bỏ qua tất cả các phần tử cú pháp aps_extension_data_flag.

Đối với Bảng 15, theo một ví dụ, các phần tử cú pháp **alf_chroma_filter_signal_flag, alf_cross_component_cb_filter_signal_flag,** và/hoặc **alf_cross_component_cr_filter_signal_flag** chỉ có thể được báo hiệu có điều kiện khi ChormaArrayType không bằng 0 và khi không có giá trị suy ra của chúng. Sự báo hiệu có điều kiện giúp tiết kiệm bit. Nghĩa là, theo một ví dụ, Bảng 15 có thể được sửa đổi như sau:

alf_data() {	**Bộ mô tả**
alf_luma_filter_signal_flag	u(1)
if(ChromaArrayType != 0) {	
alf_chroma_filter_signal_flag	u(1)
alf_cross_component_cb_filter_signal_flag	u(1)
alf_cross_component_cr_filter_signal_flag	u(1)
}	
...	
}	

Với ngữ nghĩa sau:

alf_chroma_filter_signal_flag bằng 1 xác định rằng bộ lọc sắc độ được báo hiệu. alf_chroma_filter_signal_flag bằng 0 xác định rằng bộ lọc sắc độ không được báo hiệu. Khi không hiện diện, alf_chroma_filter_signal_flag được suy ra bằng 0.

Biến NumAlfFilters xác định số lượng các bộ lọc vòng thích ứng khác nhau được đặt bằng 25.

alf_cross_component_cb_filter_signal_flag bằng 1 xác định rằng bộ lọc Cb thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 xác định rằng bộ lọc Cb thành phần chéo không được báo hiệu. Khi không hiện diện, alf_cross_component_cb_filter_signal_flag được suy ra bằng 0.

alf_cross_component_cr_filter_signal_flag bằng 1 xác định rằng bộ lọc Cr thành phần chéo được báo hiệu. alf_cross_component_cr_filter_signal_flag bằng 0 xác định rằng bộ lọc Cr thành phần chéo không được báo hiệu. Khi không hiện diện, alf_cross_component_cr_filter_signal_flag được suy ra bằng 0.

Đối với Bảng 16, theo một ví dụ, ngữ nghĩa học có thể dựa trên các điểm sau:

slice_pic_parameter_set_id xác định giá trị của pps_pic_parameter_set_id đối với PPS đang được sử dụng. Giá trị của slice_pic_parameter_set_id sẽ nằm trong khoảng từ 0 đến 63, bao gồm cả 2 số này.

Cần có sự tương thích chuỗi bit mà giá trị TemporalId của hình ảnh hiện tại phải lớn hơn hoặc bằng với giá trị TemporalId của PPS có pps_pic_parameter_set_id bằng với slice_pic_parameter_set_id.

cabac_init_flag định rõ phương pháp xác định bảng khởi tạo được sử dụng trong quá trình khởi tạo cho các biến ngữ cảnh. Khi cabac_init_flag không hiện diện thì giá trị này được suy ra là bằng 0.

slice_qp_delta xác định giá trị ban đầu của Qp_Y sẽ được sử dụng cho các khối mã hóa trong mảnh cho đến khi được sửa đổi bởi giá trị của CuQpDeltaVal trong lớp đơn vị mã hóa. Giá trị ban đầu của thông số lượng tử hóa Qp_Y cho mảnh, $SliceQp_Y$ được dẫn xuất như sau:

$$SliceQp_Y = 26 + init\_qp\_minus26 + slice\_qp\_delta$$

Giá trị của $SliceQp_Y$ sẽ nằm trong khoảng từ $-QpBdOffset_Y$ đến +63, bao gồm cả 2 khoảng này.

Trong đó,

init_qp_minus26 cộng 26 xác định giá trị ban đầu của $SliceQp_Y$ cho mỗi mảnh tham chiếu đến PPS. Giá trị ban đầu của $SliceQp_Y$ được sửa đổi tại lớp của mảnh khi giá

trị khác 0 của slice_qp_delta được giải mã. Giá trị của init_qp_minus26 sẽ nằm trong khoảng –(26 + QpBdOffsetY) đến +37.

slice_sao_luma_flag bằng 1 xác định rằng SAO được kích hoạt đối với thành phần luma trong mảnh hiện tại; slice_sao_luma_flag bằng 0 xác định rằng SAO bị vô hiệu hóa đối với thành phần luma trong mảnh hiện tại. Khi slice_sao_luma_flag không hiện diện thì giá trị này được suy ra bằng 0.

slice_sao_chroma_flag bằng 1 xác định rằng SAO được kích hoạt đối với thành phần sắc độ trong mảnh hiện tại; slice_sao_chroma_flag bằng 0 xác định rằng SAO bị vô hiệu hóa đối với thành phần sắc độ trong mảnh hiện tại. Khi slice_sao_chroma_flag không hiện diện thì giá trị này được suy ra bằng 0.

slice_alf_enabled_flag bằng 1 xác định rằng bộ lọc vòng thích ứng được kích hoạt và có thể được áp dụng cho thành phần màu Y, Cb hoặc Cr trong mảnh. slice_alf_enabled_flag bằng 0 xác định rằng bộ lọc vòng thích ứng bị vô hiệu hóa đối với tất cả các thành phần màu trong mảnh.

slice_num_alf_aps_ids_luma xác định số lượng ALF APS mà mảnh tham chiếu đến. Giá trị của slice_num_alf_aps_ids_luma sẽ nằm trong khoảng từ 0 đến 63, bao gồm cả hai số này.

slice_alf_aps_id_luma[i] xác định adaptation_parameter_set_id của ALF APS thứ i mà thành phần luma của mảnh tham chiếu đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với ALF_APS và adaptation_parameter_set_id bằng với slice_alf_aps_id_luma[i] phải nhỏ hơn hoặc bằng TemporalId của đơn vị NAL mảnh được mã hóa.

Đối với các mảnh và mảnh nội bộ trong hình ảnh IRAP, slice_alf_aps_id_luma[i] không tham chiếu đến ALF APS liên kết với các hình ảnh khác thay vì hình ảnh chứa mảnh nội bộ hoặc hình ảnh IRAP.

slice_alf_chroma_idc bằng 0 xác định rằng bộ lọc vòng thích ứng không được áp dụng cho thành phần màu Cb và Cr. slice_alf_chroma_idc bằng 1 biểu thị rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cb, slice_alf_chroma_idc bằng 2 biểu thị rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cr. slice_alf_chroma_idc bằng 3 biểu thị

rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cb và Cr. Khi slice_alf_chroma_idc không hiện diện thì giá trị này được suy ra bằng 0.

slice_alf_aps_id_chroma xác định adaptation_parameter_set_id của ALF APS mà thành phần sắc độ của mảnh tham chiếu đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với ALF_APS và adaptation_parameter_set_id bằng với slice_alf_aps_id_chroma phải nhỏ hơn hoặc bằng với TemporalId của đơn vị NAL mảnh được mã hóa.

Đối với các mảnh và mảnh nội bộ trong hình ảnh IRAP, slice_alf_aps_id_chroma sẽ không tham chiếu đến ALF APS liên kết với các hình ảnh khác thay vì hình ảnh chứa mảnh nội bộ hoặc hình ảnh IRAP.

slice_cross_component_alf_cb_enabled_flag bằng 0 xác định rằng bộ lọc Cb thành phần chéo không được áp dụng cho thành phần màu Cb. slice_cross_component_alf_cb_enabled_flag bằng 1 biểu thị rằng bộ lọc Cb thành phần chéo được áp dụng cho thành phần màu Cb. Khi slice_cross_component_alf_cb_enabled_flag không hiện diện thì giá trị này được suy ra bằng 0.

slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag bằng 1 xác định rằng hệ số bộ lọc Cb thành phần chéo, với j=0..13, bao gồm cả 2 số này được đặt bằng với $CcAlfTemporalCoeff_{Cb}[\ TemporalId\ ][\ j\ ]$.

slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag bằng 0 và slice_cross_component_alf_cb_enabled_flag bằng 1 xác định rằng phần tử cú pháp slice_cross_component_alf_cb_aps_id có trong phần đầu mảnh hiện tại.

Khi slice_cross_component_alf_cb_enabled_flag bằng 1 và slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag bằng 0, các phần tử của $CcAlfTemporalCoeff_{Cb}[\ TemporalId\ ][\ j\ ]$ và $CcAlfCoeff_{Cb}[\ j\ ]$, với j = 0..13 được dẫn xuất như sau:

$$CcAlfTemporalCoeff_{Cb}[\ TemporalId\ ][\ j\ ] = CcAlfApsCoeff_{Cb}[\ slice\_cross\_component\_alf\_cb\_aps\_id\ ][\ j\ ]$$

$$CcAlfCoeff_{Cb}[\ j\ ] = CcAlfApsCoeff_{Cb}[\ slice\_cross\_component\_alf\_cb\_aps\_id\ ][\ j\ ]$$

Khi slice_cross_component_alf_cb_enabled_flag bằng 1 và slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag bằng 1, các phần tử của $CcAlfCoeff_{Cb}[j]$ với j = 0..13 được dẫn xuất như sau:

$$CcAlfCoeff_{Cb}[j] = CcAlfTemporalCoeff_{Cb}[TemporalId][j]$$

Cần lưu ý rằng theo một số ví dụ, slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag có thể được báo hiệu có điều kiện với if(slice_type != I) và/hoặc loại đơn vị NAL là không IRAP và không GDR, cũng như được suy ra bằng 0 khi không hiện diện.

Loại đơn vị NAL là không IRAP và không GDR có thể được biểu thị bằng câu lệnh điều kiện sau:

```
if( nal_unit_type ! = IDR_W_RADL && nal_unit_type ! = IDR_N_LP &&
    nal_unit_type ! = CRA_NUT && nal_unit_type ! = GDR_NUT )
```

Loại đơn vị NAL là không IRAP, không GDR và if (slice_type != I) có thể được biểu thị bằng câu lệnh điều kiện sau:

```
if( nal_unit_type ! = IDR_W_RADL && nal_unit_type ! = IDR_N_LP &&
nal_unit_type ! = CRA_NUT && nal_unit_type ! = GDR_NUT && slice_type != I )
```

slice_cross_component_alf_cb_aps_id xác định adaptation_parameter_set_id mà thành phần màu Cb của mảnh đề cập đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với ALF_APS và adaptation_parameter_set_id bằng với slice_cross_component_alf_cb_aps_id phải nhỏ hơn hoặc bằng với TemporalId của đơn vị NAL mảnh được mã hóa.

Đối với các mảnh và mảnh nội bộ trong hình ảnh IRAP, slice_cross_component_alf_cb_aps_id sẽ không tham chiếu đến ALF APS liên kết với các hình ảnh khác thay vì hình ảnh chứa mảnh nội bộ hoặc hình ảnh IRAP.

Khi slice_cross_component_alf_cb_enabled_flag bằng 1, yêu cầu về tương thích chuỗi bit mà (đối với tất cả các mảnh của hình ảnh hiện tại) ALF APS được tham chiếu đến theo slice_cross_component_alf_cb_aps_id sẽ giống nhau.

slice_cross_component_alf_cb_log2_control_size_minus4 xác định kích thước khối điều khiển theo số lượng mẫu sắc độ cho thành phần màu Cb. slice_cross_component_alf_cb_log2_control_size_minus4 sẽ nằm trong khoảng từ 0 đến Min(Log2(CtbWidthC), Log2(CtbHeightC)) – 4, bao gồm cả 2 giá trị này.

Các biến CcAlfWidthCbL và CcAlfHeightCbL được dẫn xuất như sau:

CcAlfWidthCbL = (1 << (slice_cross_component_alf_cb_log2_control_size_minus4 + 4)) * SubWidthC

CcAlfHeightCbL = (1 <<(slice_cross_component_alf_cb_log2_control_size_minus4 + 4)) * SubHeightC

slice_cross_component_alf_cr_enabled_flag bằng 0 xác định rằng bộ lọc Cr thành phần chéo không được áp dụng cho thành phần màu Cr. slice_cross_component_alf_cb_enabled_flag bằng 1 biểu thị rằng bộ lọc vòng thích ứng thành phần chéo được áp dụng cho thành phần màu Cr. Khi slice_cross_component_alf_cr_enabled_flag không hiện diện thì giá trị này được suy ra bằng 0.

Đối với Bảng 16, theo một ví dụ về phần tử cú pháp **slice_cross_component_alf_cb_enabled_flag** và **slice_cross_component_alf_cr_enabled_flag** có thể được nhóm thành một câu lệnh if(ChromaArrayType != 0){}.

slice_cross_component_alf_cr_reuse_temporal_layer_filter_flag bằng 1 xác định rằng hệ số bộ lọc Cr thành phần chéo với j=0..13, bao gồm cả 2 số này, được đặt bằng $AlfCCTemporalCoeff_{Cr}$[TemporalId][j].

slice_cross_component_alf_cr_reuse_temporal_layer_filter_flag bằng 0 và slice_cross_component_alf_cr_enabled_flag bằng 1 xác định rằng phần tử cú pháp slice_cross_component_alf_cr_aps_id có trong phần đầu mảnh hiện tại.

Khi slice_cross_component_alf_cr_enabled_flag bằng 1 và slice_cross_component_alf_cr_reuse_temporal_layer_filter_flag bằng 0, các phần tử của $CcAlfTemporalCoeff_{Cr}[\ TemporalId\ ][\ j\ ]$ và $CcAlfCoeff_{Cr}[\ j\ ]$, với j = 0..13 được dẫn xuất như sau:

$$CcAlfTemporalCoeff_{Cr}[\ TemporalId\ ][\ j\ ] = CcAlfApsCoeff_{Cr}[\ slice\_cross\_component\_alf\_cr\_aps\_id\ ][\ j\ ]$$

$$CcAlfCoeff_{Cr}[\ j\ ] = CcAlfApsCoeff_{Cr}[\ slice\_cross\_component\_alf\_cr\_aps\_id\ ][\ j\ ]$$

Khi slice_cross_component_alf_cb_enabled_flag bằng 1 và slice_cross_component_alf_cb_reuse_temporal_layer_filter_flag bằng 1, các phần tử của $CcAlfCoeff_{Cr}[\ j\ ]$ với j = 0..13 được dẫn xuất như sau:

$$CcAlfCoeff_{Cr}[\ j\ ] = CcAlfTemporalCoeff_{Cr}[\ TemporalId\ ]\ [\ j\ ]$$

Cần lưu ý rằng theo một số ví dụ, slice_cross_component_alf_cr_reuse_temporal_layer_filter_flag có thể được báo hiệu có điều kiện bằng if(slice_type != I) và/hoặc loại đơn vị NAL là không IRAP và không GDR, cũng như được suy ra bằng 0 khi không hiện diện.

slice_cross_component_alf_cr_aps_id xác định adaptation_parameter_set_id mà thành phần màu Cr của mảnh đề cập đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với ALF_APS và adaptation_parameter_set_id bằng với slice_cross_component_alf_cr_aps_id phải nhỏ hơn hoặc bằng với TemporalId của đơn vị NAL mảnh được mã hóa.

Đối với các mảnh và mảnh nội bộ trong hình ảnh IRAP, slice_cross_component_alf_cr_aps_id sẽ không tham chiếu đến ALF APS liên kết với các hình ảnh khác thay vì hình ảnh chứa mảnh nội bộ hoặc hình ảnh IRAP.

Khi slice_cross_component_alf_cr_enabled_flag bằng 1, yêu cầu về tương thích chuỗi bit mà (đối với tất cả các mảnh của hình ảnh hiện tại) ALF APS được tham chiếu đến theo slice_cross_component_alf_cr_aps_id sẽ giống nhau.

slice_cross_component_alf_cr_log2_control_size_minus4 xác định kích thước khối điều khiển theo số lượng mẫu sắc độ cho thành phần màu Cr. slice_cross_component_alf_cr_log2_control_size_minus4 sẽ nằm trong khoảng từ 0 đến Min(Log2(CtbWidthC), Log2(CtbHeightC)) – 4, bao gồm cả 2 giá trị này.

Các biến CcAlfWidthCrL và CcAlfHeightCrL được suy ra như sau:

CcAlfWidthCrL = (1 << (slice_cross_component_alf_cr_log2_control_size_minus4 + 4))*SubWidth

CCcAlfHeightCrL = (1 <<(slice_cross_component_alf_cr_log2_control_size_minus4 + 4))*SubHeightC

deblocking_filter_override_flag bằng 1 xác định rằng thông số tách khối có trong phần đầu mảnh. deblocking_filter_override_flag bằng 0 xác định rằng thông số tách khối không có trong phần đầu mảnh. Khi không hiện diện, giá trị của deblocking_filter_override_flag được suy ra bằng 0.

slice_deblocking_filter_disabled_flag bằng 1 xác định rằng hoạt động của bộ lọc tách khối không được áp dụng cho mảnh hiện tại. slice_deblocking_filter_disabled_flag bằng 0 xác định rằng hoạt động của bộ lọc tách khối không được áp dụng cho mảnh hiện tại. Khi slice_deblocking_filter_disabled_flag không hiện diện thì sẽ được suy ra bằng pps_deblocking_filter_disabled_flag.

slice_beta_offset_div2 và **slice_tc_offset_div2** xác định độ lệch thông số tách khối cho β và tC (chia cho 2) đối với mảnh hiện tại. Giá trị của slice_beta_offset_div2 và slice_tc_offset_div2 sẽ nằm trong khoảng từ –6 đến 6, bao gồm cả 2 số này. Khi không

hiện diện thì các giá trị slice_beta_offset_div2 và slice_tc_offset_div2 được suy ra lần lượt bằng pps_beta_offset_div2 và pps_tc_offset_div2.

sps_lmcs_enabled_flag bằng 1 xác định rằng sự ánh xạ luma với bước chia tỷ lệ sắc độ được kích hoạt cho mảnh hiện tại. slice_lmcs_enabled_flag bằng 0 xác định rằng sự ánh xạ luma với bước chia tỷ lệ sắc độ không được kích hoạt cho mảnh hiện tại. Khi sslice_lmcs_enabled_flag không hiện diện thì giá trị này được suy ra bằng 0.

slice_lmcs_aps_id xác định adaptation_parameter_set_id của LMCS APS mà mảnh tham chiếu đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với LMCS_APS và adaptation_parameter_set_id bằng với slice_lmcs_aps_id phải nhỏ hơn hoặc bằng với TemporalId của đơn vị NAL mảnh được mã hóa.

Khi hiện diện, giá trị của slice_lmcS_aps_id sẽ giống nhau đối với tất cả các mảnh của hình ảnh.

slice_chroma_residual_scale_flag bằng 1 xác định rằng bước chia tỷ lệ dư sắc độ được kích hoạt cho mảnh hiện tại. slice_chroma_residual_scale_flag bằng 0 xác định rằng bước chia tỷ lệ dư sắc độ không được kích hoạt cho mảnh hiện tại. Khi slice_chroma_residual_scale_flag không hiện diện thì giá trị này được suy ra bằng 0.

Như được cung cấp ở trên, **slice_cross_component_alf_cb_log2_control_size_minus4** và**slice_cross_component_alf_cr_log2_control_size_minus4** giới hạn kích thước khối tối đa của chỉ báo bộ lọc điều khiển cục bộ (nghĩa là đến Min(Log2(CtbWidthC), Log2(CtbHeightC))). Theo một ví dụ, kích thước khối tối đa của chỉ báo bộ lọc điều khiển cục bộ có thể bị giới hạn ở Min(Floor(Log2(CtbWidthC)), Floor(Log2(CtbHeightC))). Ngoài ra, theo một ví dụ, kích thước khối tối đa của chỉ báo điều khiển cục bộ có thể bị giới hạn để các khối điều khiển không thể kéo dài qua hơn một CTU. Điều này có thể khiến quá trình xử lý trên đường biên Ô/Mảnh đơn giản hơn vì Ô/Mảnh được mô tả theo đơn vị CTU. Theo một ví dụ, giới hạn kích thước khối tối đa được dẫn xuất dựa trên kích thước CTU tối đa (khối và/hoặc chiều rộng) và/hoặc định dạng sắc độ.

Cần lưu ý rằng JVET-O2001 đề xuất cú pháp và ngữ nghĩa học sau dành cho phần tử cú pháp **non_reference_picture_flag:**

slice_header( ) {	**Bộ mô tả**

slice_pic_parameter_set_id	ue(v)
if(rect_slice_flag | | NumBricksInPic > 1)	
slice_address	u(v)
if(!rect_slice_flag && !single_brick_per_slice_flag)	
num_bricks_in_slice_minus1	ue(v)
non_reference_picture_flag	u(1)
slice_type	ue(v)
...	
}	

non_reference_picture_flag bằng 1 xác định hình ảnh chứa mảnh không bao giờ được sử dụng làm hình ảnh tham chiếu. non_reference_picture_flag bằng 0 xác định hình ảnh chứa mảnh có thể hoặc không thể được sử dụng làm hình ảnh tham chiếu.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, ngữ nghĩa học của phần tử cú pháp **slice_alf_aps_id_luma**[i], **slice_alf_chroma_idc** và **slice_alf_aps_id_chroma** có thể được dựa trên việc:

slice_alf_aps_id_luma[i] xác định adaptation_parameter_set_id của ALF APS thứ i mà thành phần luma của mảnh tham chiếu đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với ALF_APS và adaptation_parameter_set_id bằng với slice_alf_aps_id_luma[i] phải nhỏ hơn hoặc bằng TemporalId của đơn vị NAL mảnh được mã hóa.

Đối với các mảnh và mảnh nội bộ trong hình ảnh IRAP, slice_alf_aps_id_luma[i] không tham chiếu đến ALF APS liên kết với các hình ảnh khác thay vì hình ảnh chứa mảnh nội bộ hoặc hình ảnh IRAP.

Đối với i trong khoảng từ 0 đến slice_num_alf_aps_ids_luma, bao gồm cả 2 giá trị này, cần có sự tương thích chuỗi bit mà alf_luma_filter_signal_flag bằng 1 đối với ALF APS có adap_parameter_set_id bằng slice_alf_aps_id_luma [i].

slice_alf_chroma_idc bằng 0 xác định rằng bộ lọc vòng thích ứng không được áp dụng cho thành phần màu Cb và Cr. slice_alf_chroma_idc bằng 1 biểu thị rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cb. slice_alf_chroma_idc bằng 2 biểu thị rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cr, slice_alf_chroma_idc bằng 3 biểu thị

rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cb và Cr. Khi slice_alf_chroma_idc không hiện diện thì giá trị này được suy ra bằng 0.

slice_alf_aps_id_chroma xác định adaptation_parameter_set_id của ALF APS mà thành phần sắc độ của mảnh tham chiếu đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với ALF APS và adaptation_parameter_set_id bằng với slice_alf_aps_id_chroma phải nhỏ hơn hoặc bằng với TemporalId của đơn vị NAL mảnh được mã hóa.

Đối với các mảnh và mảnh nội bộ trong hình ảnh IRAP, slice_alf_aps_id_chroma sẽ không tham chiếu đến ALF APS liên kết với các hình ảnh khác thay vì hình ảnh chứa mảnh nội bộ hoặc hình ảnh IRAP.

Cần có sự tương thích chuỗi bit mà, alf_chroma_filter_signal_flag bằng 1 đối với ALF APS có adaplation_parameter_set_id bằng slice_alf_aps_id_chroma.

Ngoài ra, JVET-O2001 bao gồm các phần tử cú pháp sau trong cấu trúc cú pháp của tập hợp thông số trình tự:

sps_sao_enabled_flag bằng 1 cho thấy rằng quy trình độ lệch thích ứng mẫu được áp dụng cho hình ảnh được tái tạo sau quy trình bộ lọc tách khối. sps_sao_enabled_flag bằng 0 cho thấy rằng quy trình độ lệch thích ứng mẫu không được áp dụng đối với hình ảnh được tái tạo sau quy trình bộ lọc tách khối.

sps_alf_enabled_flag bằng 0 cho thấy rằng bộ lọc vòng thích ứng đã tắt. sps_alf_enabled_flag equal 1 cho thấy rằng bộ lọc vòng thích ứng đã bật.

sps_lmcs_enabled_flag bằng 1 cho thấy ánh xạ luma với bước chia tỷ lệ sắc độ được sử dụng trong CVS. sps_lmcs_enabled_flag bằng 0 cho thấy ánh xạ luma với bước chia tỷ lệ sắc độ không được sử dụng trong CVS.

loop_filter_across_subpic_enabled_flag[i] bằng 1 xác định rằng hoạt động lọc trong vòng có thể được thực hiện trên đường biên của ảnh phụ thứ i trong từng hình ảnh được mã hóa trong CVS. loop_filter_across_subpic_enabled_flag[i] bằng 0 xác định rằng hoạt động lọc trong vòng có thể được thực hiện trên đường biên của ảnh phụ thứ i trong từng hình ảnh được mã hóa trong CVS. Khi không hiện diện thì giá trị loop_filter_across_subpic_enabled_pic_flag[i] được suy ra là bằng 1.

Và các phần tử cú pháp sau trong cấu trúc cú pháp của tập hợp thông số hình ảnh:

cabac_init_present_flag bằng 1 chỉ định rằng cabac_init_flag hiện diện trong các phần đầu mảnh tham chiếu đến PPS. cabac_init_present_flag bằng 0 chỉ định rằng cabac_init_flag không hiện diện trong các phần đầu mảnh tham chiếu đến PPS.

deblocking_filter_override_enabled_flag bằng 1 cho biết deblocking_filter_override_flag hiện diện trong phần đầu mảnh đối với các hình ảnh tham chiếu đến PPS. deblocking_filter_override_enabled_flag bằng 0 cho biết deblocking_filter_override_flag hiện diện trong các phần đầu mảnh đối với các hình ảnh tham chiếu đến PPS. Khi không hiện diện, giá trị của deblocking_filter_override_enabled_flag được suy ra bằng 0.

loop_filter_across_bricks_enabled_flag bằng 1 xác định rằng hoạt động lọc trong vòng có thể được thực hiện trên đường biên khối trong hình ảnh tham chiếu đến PPS. loop_filter_across_bricks_enabled_flag bằng 0 xác định rằng hoạt động lọc trong vòng có thể không được thực hiện trên đường biên khối trong hình ảnh tham chiếu đến PPS. Các hoạt động lọc trong vòng bao gồm bộ lọc tách khối, bộ lọc độ lệch thích ứng mẫu và các hoạt động lọc vòng thích ứng. Khi không hiện diện, thì giá trị loop_filter_across_bricks_enabled_flag được suy ra là bằng 1.

loop_filter_across_slices_enabled_flag bằng 1 chỉ định rằng các hoạt động lọc trong vòng có thể được thực hiện qua các biên của mảnh trong ảnh tham chiếu đến PPS. loop_filter_across_slice_enabled_flag bằng 0 chỉ định rằng các hoạt động lọc trong vòng không được thực hiện qua các biên của mảnh trong ảnh tham chiếu đến PPS. Các hoạt động lọc trong vòng bao gồm bộ lọc tách khối, bộ lọc độ lệch thích ứng mẫu và các hoạt động lọc vòng thích ứng. Khi không hiện diện, thì giá trị loop_filter_across_slices_enabled_flag được suy ra là bằng 0.

Hơn nữa, đối với Bảng 16, các biến ChromaArrayType, SubWidthC, và SubHeightC có thể được suy ra như được cung cấp trong Bảng 21:

chroma_ format_idc	separate_ colour_plane_ flag	Định dạng sắc độ	Loại ChromaArray	SubWidthC	SubHeightC
0	0	Đơn sắc	0	1	1
1	0	4:2:0	1	2	2
2	0	4:2:2	2	2	1
3	0	4:4:4	3	1	1

3	1	4:4:4	0	1	1

Bảng 21

Ngoài ra, đối với Bảng 16, JVET-O2001 bao gồm các phần tử cú pháp sau trong cấu trúc cú pháp của tập hợp thông số trình tự:

log2_ctu_size_minus5 cộng 5 xác định kích thước khối cây mã hóa luma của mỗi CTU. Cần có sự tương thích chuỗi bit mà giá trị log2_ctu_size_minus5 nhỏ hơn hoặc bằng 2.

log2_min_luma_coding_block_size_minus2 cộng 2 xác định kích thước khối mã hóa luma tối thiểu.

Các biến CtbLog2SizeY, CtbSizeY, MinCbLog2SizeY, MinCbSizeY, IbcBufWidthY, IbcBufWidthC và Vsize được dẫn xuất như sau:

CtbLog2SizeY = log2_ctu_size_minus5 + 5

CtbSizeY = 1 << CtbLog2SizeY

MinCbLog2SizeY = log2_min_luma_coding_block_size_minus2 + 2

MinCbSizeY = 1 << MinCbLog2SizeY

IbcBufWidthY = 128 * 128 / CtbSizeY

IbcBufWidthC = IbcBufWidthY / SubWidthC

VSize = Min(64, CtbSizeY)

Các biến CtbWidthC và CtbHeightC xác định chiều rộng và chiều cao tương ứng của mảng cho từng CTB sắc độ có thể được suy ra như sau:

- Nếu chroma_format_idc bằng 0 (đơn sắc) hoặc separate_colour_plane_flag bằng 1 thì CtbWidthC và CtbHeightC đều bằng 0.
- Nếu không, CtbWidthC và CtbHeightC được suy ra như sau:

  CtbWidthC = CtbSizeY / SubWidthC

  CtbHeightC = CtbSizeY / SubHeightC

Đối với Bảng 17, theo một ví dụ, ngữ nghĩa học có thể dựa trên điểm sau:

CTU là nút gốc của cấu trúc cây mã hóa.

Mảng IsAvailable[cIdx][x][y] xác định xem có sẵn mẫu tại (x, y) để sử dụng trong quá trình dẫn xuất cho tính khả dụng của khối lân cận như đã xác định hay không, được khởi tạo như sau cho cIdx = 0..2, x = 0..CtbSizeY – 1 và y = O..CtbSizeY – 1:

IsAvailable[cIdx][x][y] = FALSE

Mảng IsInSmr[x][y] xác định xem mẫu tại (x, y) có nằm bên trong vùng danh sách ứng viên hợp nhất dùng chung hay không, được khởi tạo như sau cho x = 0..CtbSizeY – 1 và y = 0..CtbSizeY – 1:

IsInSmr[x][y] = FALSE

alf_ctb_flag[cIdx][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] bằng 1 xác định rằng bộ lọc vòng thích ứng được áp dụng cho khối cây mã hóa của thành phần màu được biểu thị bởi cIdx của đơn vị cây mã hóa tại vị trí luma (xCtb, yCtb). alf_ctb_flag[cIdx][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] bằng 0 xác định rằng bộ lọc vòng thích ứng không được áp dụng cho khối cây mã hóa của thành phần màu được biểu thị bởi cIdx của đơn vị cây mã hóa tại vị trí luma (xCtb, yCtb).

Khi alf_ctb_flag[cIdx][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] không hiện diện thì giá trị này được suy ra bằng 0.

alf_ctb_use_first_aps_flag bằng 1 xác định rằng thông tin bộ lọc trong APS có adaptive_parameter_set_id bằng slice_alf_aps_id_luma[0] được sử dụng. alf_ctb_use_first_aps_flag bằng 0 xác định rằng CTB luma không sử dụng thông tin bộ lọc trong APS có adaptive_parameter_set_id bằng slice_alf_aps_id_luma[0]. Khi alf_ctb_use_first_aps_flag không hiện diện thì sẽ được suy ra bằng 0.

alf_use_aps_flag bằng 0 xác định rằng một trong các tập hợp bộ lọc cố định được áp dụng cho CTB luma. alf_use_aps_flag bằng 1 xác định rằng tập hợp bộ lọc từ APS được áp dụng cho CTB luma. Khi alf_use_aps_flag không hiện diện thì giá trị này được suy ra là bằng 0.

alf_luma_prev_filter_idx_minus1 cộng 1 xác định bộ lọc trước đó được áp dụng cho CTB luma. Giá trị alf_luma_prev_filter_idx_minus1 sẽ nằm trong khoảng từ 0 đến slice_num_alf_aps_ids_luma – 2, bao gồm cả hai giá trị này. Khi alf_luma_prev_filter_idx_minus1 không hiện diện thì giá trị này được suy ra bằng 0.

Biến AlfCtbFiltSetIdxY[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] xác định chỉ số tập hợp bộ lọc cho CTB luma tại vị trí (xCtb, yCtb) được suy ra như sau:

– Nếu alf_ctb_use_first_aps_flag bằng 1, AlfCtbFiltSetIdxY[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] được đặt bằng 16.

– Mặt khác, nếu alf_use_aps_flag bằng 0, AlfCtbFiltSetIdxY[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] sẽ được đặt bằng alf_luma_fixed_filter_idx.

– Nếu không thì AlfCtbFiltSetIdxY[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] sẽ được đặt bằng 17 + alf_luma_prev_filter_idx_minus1.

alf_luma_fixed_filter_idx xác định bộ lọc cố định được áp dụng cho CTB luma. Giá trị alf_luma_fixed_filter_idx sẽ nằm trong khoảng từ 0 đến 15, bao gồm cả hai giá trị này.

alf_ctb_filter_alt_idx[chromaIdx][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] xác định chỉ số của bộ lọc sắc độ thay thế được áp dụng cho khối cây mã hóa của thành phần sắc độ có chromaIdx bằng 0 đối với Cb và chromaIdx bằng 1 đối với Cr, của đơn vị cây mã hóa tại vị trí luma (xCtb, yCtb). Khi alf_ctb_filter_alt_idx[chromaIdx][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] không hiện diện, nó sẽ được suy ra bằng 0.

alf_cross_component_cb_flag[xL / CcAlfWidthCbL][yL / CcAlfHeightCbL] bằng 0 biểu thị rằng bộ lọc Cb thành phần chéo không được áp dụng cho khối của các mẫu thành phần màu Cb tại vị trí luma (xL, yL). alf_cross_component_cb_flag[xL / CcAlfWidthCbL][yL/CcAlfHeightCbL] bằng 1 biểu thị rằng bộ lọc Cb thành phần chéo được áp dụng cho khối của các mẫu thành phần màu Cb tại vị trí luma (xL, yL)

alf_cross_component_cr_flag[xL / CcAlfWidthCbL][yL / CcAlfHeightCbL] bằng 0 biểu thị rằng bộ lọc Cr thành phần chéo không được áp dụng cho khối của các mẫu thành phần màu Cr tại vị trí luma (xL, yL). alf_cross_component_cr_flag[xL / CcAlfWidthCbL] [yL / CcAlfHeightCbL] bằng 1 biểu thị rằng bộ lọc Cr thành phần chéo được áp dụng cho khối của các mẫu thành phần màu Cr tại vị trí luma (xL, yL)

Cần lưu ý rằng theo một ví dụ, [xL / CcAlfWidthCbL][yL / CcAlfHeightCbL] có thể được biểu thị bằng cách sử dụng phép toán dịch chuyển /.

Ngoài ra, đối với Bảng 17, JVET-O2001 bao gồm các phần tử cú pháp sau trong cấu trúc cú pháp của tập hợp thông số hình ảnh:

pic_width_in_luma_samples xác định chiều rộng của mỗi hình ảnh được giải mã tham chiếu đến PPS theo đơn vị mẫu luma. pic_width_in_luma_samples không được bằng 0 mà phải là bội số nguyên của Max(8, MinCbSizeY) và phải nhỏ hơn hoặc bằng pic_width_max_in_luma_samples.

Khi subpics_present_flag bằng 1, giá trị pic_width_in_luma_samples phải bằng với pic_width_max_in_luma_samples.

pic_height_in_luma_samples xác định chiều cao của mỗi hình ảnh được giải mã tham chiếu đến PPS theo đơn vị mẫu luma. pic_height_in_luma_samples không được bằng 0 và phải là bội số nguyên của Max(8, MinCbSizeY) và phải nhỏ hơn hoặc bằng pic_height_max_in_luma_samples.

Khi subpics_present_flag bằng 1, giá trị pic_height_in_luma_samples phải bằng pic_height_max_in_luma_samples.

Đặt refPicWidthInLumaSamples và refPicHeightInLumaSamples lần lượt là pic_width_in_luma_samples và pic_height_in_luma_samples, trong hình ảnh tham chiếu của hình ảnh hiện tại tham chiếu đến PPS này. Cần có sự tương thích chuỗi bit mà tất cả các điều kiện sau đều được thỏa mãn:

– pic_width_in_luma_samples * 2 sẽ lớn hơn hoặc bằng refPicWidthInLumaSamples.

– pic_height_in_luma_samples * 2 sẽ lớn hơn hoặc bằng refPicHeightInLumaSamples.

– pic_width_in_luma_samples sẽ nhỏ hơn hoặc bằng refPicWidthInLumaSamples * 8.

– pic_height_in_luma_samples sẽ nhỏ hơn hoặc bằng refPicHeightInLumaSamples * 8.

Các biến PicWidthInCtbsY, PicHeightInCtbsY, PicSizeInCtbsY, PicWidthInMinCbsY, PicHeightInMinCbsY, PicSizeInMinCbsY, PicSizeInSamplesY, PicWidthInSamplesC và PicHeightInSamplesC được suy ra như sau:

PicWidthInCtbsY = Ceil(pic_width_in_luma_samples ÷ CtbSizeY)

PicHeightInCtbsY = Ceil(pic_height_in_luma_samples ÷ CtbSizeY)

PicSizeInCtbsY = PicWidthInCtbsY * PicHeightInCtbsY

PicWidthInMinCbsY = pic_width_in_luma_samples / MinCbSizeY

PicHeightInMinCbsY = pic_height_in_luma_samples / MinCbSizeY

PicSizeInMinCbsY = PicWidthInMinCbsY * PicHeightInMinCbsY

PicSizeInSamplesY = pic_width_in_luma_samples * pic_height_in_luma_samples

PicWidthInSamplesC = pic_width_in_luma_samples / SubWidthC

PicHeightInSamplesC = pic_height_in_luma_samples / SubHeightC

Ngoài ra, JVET-O2001 bao gồm các phần tử cú pháp sau trong cấu trúc cú pháp của tập hợp thông số hình ảnh:

pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1 xác định rằng hoạt động lọc trong vòng bị vô hiệu hóa trên đường biên ảo trong hình ảnh tham chiếu đến PPS. pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 0 xác định rằng không áp dụng việc vô hiệu hóa hoạt động lọc trong vòng đó trong hình ảnh tham chiếu đến PPS. Các hoạt động lọc trong vòng bao gồm bộ lọc tách khối, bộ lọc độ lệch thích ứng mẫu và các hoạt động lọc vòng thích ứng. Khi không hiện diện, thì giá trị pps_loop_filter_across_virtual_boundaries_disabled_flag được suy ra là bằng 0.

pps_num_ver_virtual_boundaries chỉ định số lượng phần tử cú pháp pps_virtual_boundaries_pos_x[i] hiện diện trong PPS. Khi không hiện diện pps_num_ver_virtual_boundaries, thì giá trị này được suy ra là bằng 0.

pps_virtual_boundaries_pos_x[i] được sử dụng để tính toán giá trị PpsVirtualBoundariesPosX [i], xác định vị trí của đường biên ảo hướng dọc thứ i theo đơn vị mẫu luma. pps_virtual_boundaries_pos_x[i] sẽ nằm trong khoảng từ 1 đến Ceil(pic_width_in_luma_samples ÷ a 8) – 1, bao gồm cả 2 giá trị này.

Vị trí của đường biên ảo theo chiều dọc PpsVirtualBoundariesPosX[i] được suy ra như sau:

PpsVirtualBoundariesPosX[i] = pps_virtual_boundaries_pos_x[i] * 8

Khoảng cách giữa hai biên ảo theo chiều dọc bất kỳ phải lớn hơn hoặc bằng các mẫu luma CtbSizeY.

pps_num_hor_virtual_boundaries chỉ định số lượng phần tử cú pháp pps_virtual_boundaries_pos_y[i] hiện diện trong PPS. Khi pps_num_hor_virtual_boundaries không hiện diện, thì giá trị này được suy ra là bằng 0.

pps_virtual_boundaries_pos_y[i] được sử dụng để tính toán giá trị PpsVirtualBoundariesPosY[i], xác định vị trí của đường biên ảo hướng ngang thứ i theo đơn vị mẫu luma. pps_virtual_boundaries_pos_y[i] sẽ nằm trong khoảng từ 1 đến Ceil(pic_height_in_luma_samples ÷ 8) – 1, bao gồm cả 2 giá trị này.

Vị trí của biên ảo theo chiều ngang PpsVirtualBoundariesPosY[i] được suy ra như sau.

PpsVirtualBoundariesPosY[i] = pps_virtual boundaries_pos_y[i] * 8

Khoảng cách giữa hai biên ảo theo chiều ngang bất kỳ phải lớn hơn hoặc bằng các mẫu luma CtbSizeY.

Ví dụ, như đã cung cấp ở trên, đối với Bảng 5C và Bảng 15, một hoặc nhiều tập hợp bộ lọc Cb và Cr thành phần chéo của phần tử cú pháp bộ lọc có thể được báo hiệu, (ví dụ: trong alf_data()). Trong một ví dụ, theo các kỹ thuật trong tài liệu này, một hoặc nhiều tập hợp bộ lọc thành phần chéo có thể được xác định cho bộ giải mã (ví dụ: được lưu trữ trong bộ nhớ bộ giải mã). Nghĩa là, tập hợp bộ lọc thành phần chéo mặc định có thể được xác định. Ngoài ra, theo một ví dụ, tập hợp bộ lọc thành phần chéo có thể được lập chỉ số và giá trị chỉ số có thể được báo hiệu để biểu thị tập hợp bộ lọc thành phần chéo sẽ được áp dụng. Hơn nữa, tập hợp bộ lọc thành phần chéo mặc định có thể được dùng kết hợp với tập hợp bộ lọc thành phần chéo được báo hiệu. Nghĩa là, ví dụ, theo một ví dụ, cờ có thể được báo hiệu biểu thị nếu chỉ số được sử dụng để biểu thị tập hợp bộ lọc thành phần chéo sẽ được áp dụng hoặc nếu tập hợp bộ lọc thành phần chéo được báo hiệu. Ví dụ, tập hợp bộ lọc mặc định riêng có thể được xác định tương ứng cho Cb và Cr. Ngoài ra, cần lưu ý rằng theo một số ví dụ, các tập hợp bộ lọc cố định và bộ lọc hệ số được báo hiệu có thể có các kích thước và/hoặc hình dạng khác nhau. Theo các ví dụ này, quy trình lọc có thể mô tả việc lọc cho mỗi bộ lọc có hình dạng/kích thước cụ thể.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, ví dụ như đối với cú pháp và ngữ nghĩa học được cung cấp ở trên đối với Bảng 15-21, quy trình lọc vòng thích ứng có thể được thực hiện dựa trên những điểm sau:

Đầu vào của quy trình này là mảng mẫu hình ảnh tái tạo trước bộ lọc vòng thích ứng $recPicture_L$ và, khi ChromaArrayType không bằng 0, các mảng $recPicture_{Cb}$ và $recPicture_{Cr}$.

Đầu ra của quy trình này là mảng mẫu hình ảnh tái tạo đã sửa đổi sau bộ lọc vòng thích ứng $alfPicture_L$ và, khi ChromaArrayType không bằng 0, các mảng $ccAlfPicture_{Cb}$ và $ccAlfPicture_{Cr}$.

Giá trị mẫu trong mảng mẫu hình ảnh tái tạo đã sửa đổi sau bộ lọc vòng thích ứng $alfPicture_L$ và, khi ChromaArrayType không bằng 0, các mảng $alfPicture_{Cb}$ và $alfPicture_{Cr}$ ban đầu được đặt bằng giá trị mẫu trong mảng mẫu hình ảnh tái tạo trước bộ lọc vòng thích ứng $recPicture_L$ và, khi ChromaArrayType không bằng 0, các mảng $recPicture_{Cb}$ và $recPicture_{Cr}$ tương ứng.

Áp dụng các bước theo thứ tự sau đây:

- Đối với mọi đơn vị cây mã hóa có vị trí khối cây mã hóa luma (rx, ry), trong đó rx = 0..PicWidthInCtbsY – 1 và ry = 0..PicHeightInCtbsY – 1, các điểm sau sẽ được áp dụng:
  - Khi alf_ctb_flag[0][rx][ry] bằng 1, quy trình lọc khối cây mã hóa cho mẫu luma như đã xác định được dẫn ra với $recPicture_L$, $alfPicture_L$ và vị trí khối cây mã hóa luma (xCtb, yCtb) được đặt bằng (rx << CtbLog2SizeY, ry << CtbLog2SizeY) làm đầu vào, và đầu ra là $alfPicture_L$ hình ảnh được lọc đã sửa đổi.
  - Khi ChromaArrayType khác 0 và alf_ctb_flag[1][rx][ry] bằng 1, quy trình lọc khối cây mã hóa đối với mẫu sắc độ như được xác định được dẫn ra với recPicture được đặt là bằng $recPicture_{Cb}$, alfPicture được đặt bằng $alfPicture_{Cb}$, vị trí khối cây mã hóa sắc độ (xCtbC, yCtbC) được đặt bằng ((rx << CtbLog2SizeY) / SubWidthC, (ry << CtbLog2SizeY) / SubHeightC), và chỉ số bộ lọc sắc độ thay thế altIdx được đặt bằng alf_ctb_filter_alt_idx[0][rx][ry] là đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $alfPicture_{Cb}$.
  - Khi ChromaArrayType khác 0 và alf_ctb_flag[2][rx][ry] bằng 1, quy trình lọc khối cây mã hóa cho mẫu sắc độ như đã được xác định được dẫn ra với recPicture được đặt bằng $recPicture_{Cr}$, alfPicture được đặt bằng $alfPicture_{Cr}$, vị

trí cây mã hóa sắc độ (xCtbC, yCtbC) được đặt bằng ((rx << CtbLog2SizeY) / SubWidthC, (ry << CtbLog2SizeY) / SubHeightC), và chỉ số bộ lọc sắc độ thay thế altIdx được đặt bằng alf_ctb_filter_alt_idx[0][rx][ry] là đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $alfPicture_{Cr}$.

- Đối với mọi vị trí luma (rx, ry), trong đó rx = 0..pic_width_in_luma_samples / CcAlfWidthCbL – 1 và ry = 0..pic_height_in_luma_samples / CcAlfHeightCbL – 1, các điểm sau sẽ được áp dụng:
  - Khi ChromaArrayType không bằng 0 và alf_cross_component_cb_flag[rx][ry] bằng 1, quy trình lọc thành phần chéo cho khối của mẫu sắc độ như đã xác định được dẫn ra với $recPicture_L$ được đặt bằng $recPicture_L$, $alfPicture_C$ được đặt bằng $alfPicture_{Cb}$, khối sắc độ của vị trí mẫu (xC, yC) được đặt bằng

    (rx * CcAlfWidthCbL / SubWidthC, ry * CcAlfHeightCbL / SubHeightC), ccAlfWidth được đặt bằng CcAlfWidthCbL / SubWidthC, ccAlfHeight được đặt bằng CcAlfHeightCbL / SubHeightC, và hệ số bộ lọc thành phần chéo CcAlfCoeff[j] được đặt bằng $CcAlfCoeff_{Cb}$[j], với j = 0..13, là đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $ccAlfPicture_{Cb}$.
- Đối với mọi vị trí luma (rx, ry), trong đó rx = 0..pic_width_in_luma_samples / CcAlfWidthCrL – 1 và ry = 0..pic_height_in_luma_samples / CcAlfHeightCrL – 1, điểm sau sẽ được áp dụng:
  - Khi ChromaArrayType khác 0 và alf_cross_component_cr_flag[rx][ry] bằng 1, quy trình lọc thành phần chéo cho khối của mẫu sắc độ như đã xác định bên dưới được dẫn ra với $recPicture_L$ được đặt bằng $recPicture_L$, $alfPicture_C$ được đặt bằng $alfPicture_{Cr}$, khối sắc độ của vị trí mẫu (xC, yC) được đặt bằng

    (rx * CcAlfWidthCrL / SubWidthC, ry* CcAlfHeightCrL / SubHeightC), ccAlfWidth được đặt bằng CcAlfWidthCrL / SubWidthC, ccAlfHeight được đặt bằng CcAlfHeightCrL / SubHeightC, và hệ số bộ lọc thành phần chéo CcAlfCoeff[j] được đặt bằng $CcAlfCoeff_{Cr}$[j], với j = 0..13, là đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $ccAlfPicture_{Cr}$.

Quy trình lọc thành phần chéo cho khối mẫu sắc độ

Giá trị đầu vào của quy trình này là:

- mảng mẫu hình ảnh luma tái tạo $recPicture_L$ trước quá trình lọc vòng thích ứng luma,
- mảng mẫu hình ảnh sắc độ tái tạo đã lọc $alfPicture_C$,
- vị trí sắc độ (xC, yC) xác định mẫu của khối mẫu sắc độ hiện tại trên cùng bên trái so với mẫu của hình ảnh hiện tại trên cùng bên trái,
- chiều rộng ccAlfWidth của khối mẫu sắc độ
- chiều cao ccAlfHeight của khối mẫu sắc độ
- hệ số bộ lọc thành phần chéo CcAlfCoeff[j], với j = 0..13

Đầu ra của quy trình này là mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture. Vị trí luma khối cây mã hóa (xCtb, yCtb) được suy ra như sau:

$$xCtb = (((xC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY$$

$$yCtb = (((yC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY$$

Để suy ra các mẫu sắc độ tái tạo đã lọc ccAlfPicture[xC + x][yC + y], mỗi mẫu sắc độ tái tạo trong khối các mẫu sắc độ hiện tại $alfPicture_C$[xC + x][yC + y] với x = 0..ccAlfWidth – 1, y = 0..ccAlfHeight – 1, được lọc như sau:

- Vị trí luma (xL, yL) tương ứng với mẫu sắc độ hiện tại ở vị trí sắc độ (xC + x, yC + y) được đặt bằng ((xC + x) * SubWidthC, (yC + y) * SubHeightC)
- Vị trí luma (h_{xL+i}, v_{yL+j}) với i = -2..2, j = –2..3 bên trong mảng $recPicture_L$ được suy ra như sau:
  - Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và xL – PpsVirtualBoundariesPosX[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic\_width\_in\_luma\_samples - 1, xL + i)$$

– Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và PpsVirtualBoundariesPosX[n]– xL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{x+i} = Clip3(0, PpsVirtualBoundariesPosXf n] – 1, xL + i)$$

– Nếu không sẽ áp dụng điểm sau đây:

$$h_{x+i} = Clip3(0, pic\_width\_in\_luma\_samples – 1, xL + i)$$

– Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và yL – PpsVirtualBoundariesPosY[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n= 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic\_height\_in\_luma\_samples – 1, yL + j)$$

– Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và PpsVirtualBoundariesPosY[n] – yL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] – 1, yL + j)$$

– Nếu không sẽ áp dụng điểm sau đây:

$$v_{y+j} = Clip3(0, pic\_height\_in\_luma\_samples – 1, yL + j)$$

– Các biến clipLeftPos, clipRightPos, clipTopPos và clipBottomPos được suy ra bằng cách dẫn ra quy trình dẫn xuất vị trí biên ALF như được xác định bên dưới với (xCtb, yCtb) và (xL – xCtb, yL – yCtb) là đầu vào.

– Độ lệch vị trí mẫu hướng dọc yM2, yM2, yM1, yP1, yP2 và yP3 được quy định trong Bảng 22 theo vị trí mẫu luma hướng dọc yL, clipLeftPos và clipRightPos.

– Độ lệch vị trí mẫu hướng ngang xM1, xM2, xP1 và xP2 được quy định trong Bảng 23 theo vị trí mẫu luma hướng ngang xL, clipLeftPos và clipRightPos.

– Biến curr được suy ra như sau:

$$curr = alfPicture_C[xC + x, yC + y]$$

– Mảng của hệ số bộ lọc thành phần chéo f[j] được suy ra như sau, với j = 0..13:

$$f[j] = CcAlfCoeff[j]$$

– Biến sum được suy ra như sau:

$$\begin{aligned} sum = {} & f[0] * recPicture_L[h_x, v_{y+yM2}] + \\ & f[1] * recPicture_L[h_{x+xM1}, v_{y+yM1}] + \\ & f[2] * recPicture_L[h_x, v_{y+yM1}] + \\ & f[3] * recPicture_L[h_{x+xP1}, v_{y+yM1}] + \\ & f[4] * recPicture_L[h_{x+xM2}, v_y] + \\ & f[5] * recPicture_L[h_{x+xM1}, v_y] + \\ & f[6] * recPicture_L[h_x, v_y] + \\ & f[7] * recPicture_L[h_{x+xP1}, v_y] + \\ & f[4] * recPicture_L[h_{x+xP2}, v_y] + \\ & f[4] * recPicture_L[h_{x+xM2}, v_{y+yP1}] + \end{aligned}$$

f[8] * recPicture$_L$[h_{x+xM1}, v_{y+yP1}] +

f[9] * recPicture$_L$[h_x, v_{y+yP1}] +

f[10] * recPicture$_L$[h_{x+xP1}, v_{y+yP1}] +

f[4] * recPicture$_L$[h_{x+xP2}, v_{y+yP1}] +

f[11] * recPicture$_L$[h_{x+xM1}, v_{y+yP2}] +

f[12] * recPicture$_L$[h_x, v_{y+yP2}] +

f[13] * recPicture$_L$[h_{x+xP1}, v_{y+yP2}] +

f[0] * recPicture$_L$[h_x, v_{y+yP3}] +

sum = curr + (sum + 64) >> 7)

– Mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture[xC + x][yC + y] được suy ra như sau:

ccAlfPicture[xC + x][yC + y] = Clip3(0, (1 << BitDepth$_C$) – 1, sum)

Điều kiện	yM2	yM1	yP1	yP2	yP3
yL = = clipTopPos + 1	–1	–1	1	2	3
yL = = clipTopPos	0	0	1	2	3
yL = = clipBottomPos – 1	–2	–1	0	0	0
yL = = clipBottomPos – 2	–2	–1	1	1	1
yL = = clipBottomPos – 3	–2	–1	1	2	2
Nếu không	–2	–1	1	2	3

Bảng 22

Điều kiện	xM2	xM1	yP1	xP2
xL = = clipLeftPos + 1	–1	–1	1	2
xL = = clipLeftPos	0	0	1	2
xL = = clipRightPos – 1	–2	–1	0	0
xL = = clipRightPos – 2	–2	–1	1	1
Nếu không	–2	–1	1	2

Bảng 23

Dẫn xuất vị trí biên ALF

Giá trị đầu vào của quy trình này là:

- vị trí luma (xCtb, yCtb) xác định mẫu của khối cây mã hóa luma hiện tại trên cùng bên trái so với mẫu của hình ảnh hiện tại trên cùng bên trái,
- vị trí luma (x, y) xác định mẫu hiện tại so với mẫu trên cùng bên trái của khối cây mã hóa luma hiện tại.

Giá trị đầu ra của quy trình này là:

- vị trí đường biên dọc bên trái clipLeftPos,
- vị trí đường biên dọc bên phải clipRightPos,
- vị trí đường biên ngang bên trên clipTopPos,
- vị trí đường biên ngang bên dưới clipBottomPos.

Các biến clipLeftPos, clipRightPos, clipTopPos và clipBottomPos được đặt bằng –128.

Biến clipTopPos được sửa đổi như sau:

- Nếu đường biên dưới cùng của khối cây mã hóa hiện tại không phải là đường biên dưới cùng của hình ảnh và y – (CtbSizeY – 4) lớn hơn hoặc bằng 0, biến clipTopPos được đặt bằng yCtb + CtbSizeY – 4.
- Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY bằng 0, và yCtb + y – PpsVirtualBoundariesPosY[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 cho mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$clipTopPos = PpsVirtualBoundariesPosY[n]$$

- Mặt khác, nếu y nhỏ hơn 3 và đường biên trên cùng của khối cây mã hóa hiện tại không phải là đường biên trên cùng của hình ảnh, và một hoặc nhiều điều kiện sau là đúng thì biến clipTopPos được đặt bằng yCtb:
  - Nếu đường biên trên cùng của khối cây mã hóa hiện tại là đường biên trên cùng của khối (brick), và loop_filter_across_bricks_enabled_flag bằng 0.
  - Nếu đường biên trên cùng của khối cây mã hóa hiện tại là đường biên trên cùng của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.

- Nếu đường biên trên cùng của khối cây mã hóa hiện tại là đường biên trên cùng của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Biến clipBottomPos được sửa đổi như sau:

- Nếu đường biên dưới cùng của khối cây mã hóa hiện tại không phải là đường biên dưới cùng của hình ảnh và CtbSizeY – 4 – y lớn hơn 0 và nhỏ hơn 4, biến clipBottomPos được đặt bằng với yCtb + CtbSizeY – 4.
- Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, PpsVirtualBoundariesPosY[n] % CtbSizeY bằng 0, PpsVirtualBoundariesPosY[n] không bằng pic_height_in_luma_samples –1 hoặc 0, và PpsVirtualBoundariesPosY[n] – yCtb – y lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

clipBottomPos = PpsVirtualBoundariesPosY[n]

- Mặt khác, nếu CtbSizeY – y nhỏ hơn 4 và đường biên dưới cùng của khối cây mã hóa hiện tại không phải là đường biên dưới cùng của hình ảnh và một hoặc nhiều điều kiện sau là đúng, biến clipBottomPos được đặt bằng yCtb + CtbSizeY:
  - Nếu đường biên dưới cùng của khối cây mã hóa hiện tại là đường biên dưới cùng của khối, và loop_filter_across_bricks_enabled_flag bằng 0.
  - Nếu đường biên dưới cùng của khối cây mã hóa hiện tại là đường biên dưới cùng của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.
  - Nếu đường biên dưới cùng của khối cây mã hóa hiện tại là đường biên dưới cùng của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Biến clipLeftPos được sửa đổi như sau:

- Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY bằng 0, và xCtb + x – PpsVirtualBoundariesPosX[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 cho mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

clipLeftPos = PpsVirtualBoundariesPosX[n]

- Mặt khác, nếu x nhỏ hơn 3, đường biên bên trái của khối cây mã hóa hiện tại không phải là đường biên bên trái của hình ảnh và một hoặc nhiều điều kiện sau là đúng, thì biến clipLeftPos được đặt bằng xCtb:
  - Nếu đường biên bên trái của khối cây mã hóa hiện tại là đường biên bên trái của khối (brick), và loop_filter_across_bricks_enabled_flag bằng 0.
  - Nếu đường biên bên trái của khối cây mã hóa hiện tại là đường biên bên trái của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.
  - Nếu đường biên bên trái của khối cây mã hóa hiện tại là đường biên bên trái của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Biến clipRightPos được sửa đổi như sau:

- Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY bằng 0, và PpsVirtualBoundariesPosX[n]– xCtb – x lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

clipRightPos = PpsVirtualBoundariesPosX[n]

- Mặt khác, nếu CtbSizeY – x nhỏ hơn 4 và đường biên bên phải của khối cây mã hóa hiện tại không phải là đường biên bên phải của hình ảnh và một hoặc nhiều điều kiện sau là đúng, thì biến clipRightPos được đặt bằng xCtb + CtbSizeY:
  - Nếu đường biên bên phải của khối cây mã hóa hiện tại là đường biên bên phải của khối, và loop_filter_across_bricks_enabled_flag bằng 0.
  - Nếu đường biên bên phải của khối cây mã hóa hiện tại là đường biên bên phải của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.
  - Nếu đường biên bên phải của khối cây mã hóa hiện tại là đường biên bên phải của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Như được mô tả ở trên, phần tử cú pháp có thể được mã hóa entropy theo CABAC hoặc tương tự. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, sự nhị phân hóa **alf_cross_component_cb_flag** và/hoặc **alf_cross_component_cr_flag** có thể là độ dài cố định, FL, nhị phân hóa với giá trị cực đại, cMax = 1. Ngoài ra, trong một ví dụ, theo các kỹ thuật trong tài liệu này, đối với **alf_cross_component_cb_flag** và/hoặc **alf_cross_component_cr_flag** hai biến pStateIdx0 và pStateIdx1 tương ứng với các chỉ số trạng thái xác suất để mã hóa CABAC có thể được khởi tạo như sau:

Bảng 24 và Bảng 25 chứa các giá trị của biến 6 bit initValue được sử dụng để khởi tạo các biến ngữ cảnh được gán cho các phần tử cú pháp alf_cross_component_cb_flag and alf_cross_component_cr_flag. Trong 24 và Bảng 25, các giá trị của biến ctxIdx được ánh xạ đến initValue và shiftIdx. Cần lưu ý rằng trong 24 và Bảng 25, giá trị shiftIdx được sử dụng để suy ra giá trị shift cho quy trình chuyển tiếp trạng thái. Ngoài ra, giá trị EP biểu thị xác suất ngang nhau và tương ứng với giá trị 35 trong JVET-O2001.

Biến khởi tạo	**ctxIdx of alf_cross_component_cb_flag**		
	0	1	2
initValue	EP	EP	EP
shiftIdx	0	0	0

Bảng 24

Biến khởi tạo	**ctxIdx of alf_cross_component_cr_flag**		
	0	**1**	**2**
initValue	EP	EP	EP
shiftIdx	0	0	0

Bảng 25

Từ mục nhập trong bảng 6 bit initValue, hai biến 3 bit slopeIdx và offsetIdx được suy ra như sau:

slopeIdx = initValue >> 3

offsetIdx = initValue & 7

Các biến m và n (được sử dụng khi khởi tạo các biến ngữ cảnh) được suy ra từ slopeIdx và offsetIdx như sau:

m = slopeIdx – 4

n = (offsetIdx * 18) + 1

Hai giá trị được gán cho pStateIdx0 và pStateIdx1 để khởi tạo được suy ra từ SliceQpy, được suy ra như đã cung cấp ở trên:

Với các biến m và n, sự khởi tạo được xác định như sau:

preCtxState = Clip3(1, 127, ((m * (Clip3(0, 51, SliceQpy) – 16)) >> 1) + n)

Hai giá trị được gán cho pStateIdx0 và pStateIdx1 để khởi tạo được suy ra như sau.

pStateIdx0 = preCtxState << 3

pStateIdx1 = preCtxState << 7

ctxIdx mà cần khởi tạo cho mỗi kiểu trong số ba kiểu khởi tạo, được xác định bởi biến initType, được liệt kê trong Bảng 26. Đối với loại mảnh P và B, việc suy ra initType phụ thuộc vào giá trị của phần tử cú pháp cabac_init_flag. Biến initType được suy ra như sau:

if(slice_type = = I)

initType = 0

else if(slice_type = = P)

initType = cabac_init_flag ? 2 : 1

khác

initType = cabac_init_flag ? 1 : 2

Phần tử cú pháp	initType		
	0	1	2
alf_cross_component_cb_flag[ ][ ]	0	1	2
alf_cross_component_cr_flag[ ][ ]	0	1	2

Bảng 26

Từ Bảng 26, biến ctxIdxOffset được đặt bằng với initType. Biến ctxIdx được đặt bằng tổng của ctxInc và ctxIdxOffset.

Theo một ví dụ, việc gán ctxInc được xác định như sau với condL và condA được xác định trong Bảng 27:

– Đối với phần tử cú pháp alf_cross_component_cb_flag[x0 / CcAlfWidthCbL][y0 / CcAlfHeightCbL] và alf_cross_component_cr_flag[x0 / CcAlfWidthCrL][y0 / CcAlfHeightCrL]:

ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3

Phần tử cú pháp	condL	condA	ctxSetIdx
alf_cross_component_cb_flag[x0 / CcAlfWidthCbL][y0 / CcAlfHeightCbL]	alf_cross_component_cb_flag[xNbL / CcAlfWidthCbL][yNbL / CcAlfHeightCbL]	alf_cross_component_cb_flag[xNbA / CcAlfWidthCbL][yNbA / CcAlfHeightCbL]	0
alf_cross_component_cr_flag[x0 / CcAlfWidthCrL][y0 / CcAlfHeightCrL]	alf_cross_component_cr_flag[x NbL / CcAlfWidthCrL][yNbL / CcAlfHeightCrL]	alf_cross_component_cr_flag [xNbA / CcAlfWidthCrL][yNbA / CcAlfHeightCrL]	0

Bảng 27

Đối với Bảng 27, trong một ví dụ, mỗi phần tử cú pháp trong condL và/hoặc condA có thể được so sánh với 0, ví dụ: có thể thêm kiểm nghiệm "== 0".

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc lọc thành phần chéo có thể bao gồm bước sử dụng bộ lọc khuếch đại bằng không. Theo một ví dụ, tổng các hệ số dành cho bộ lọc khuếch đại bằng không là 0. Cần lưu ý rằng bộ lọc khuếch đại bằng không có tổng hệ số bằng 0 có thể cho hiệu quả mã hóa tốt hơn đối với một số hệ số được báo hiệu, vì hệ số bộ lọc không ký hiệu có thể được xác định và sử dụng. Nghĩa là, khi tổng các hệ số dành cho bộ lọc khuếch đại bằng không là 0, giá trị cho hệ số của bộ lọc có thể được suy ra nếu giá trị của hệ số bộ lọc còn lại được biết đến, và như vậy thì không cần phải báo hiệu rõ ràng một trong các hệ số, điều này sẽ tiết kiệm tốc độ bit. Ngoài ra, cần lưu ý rằng trong các ví dụ khác, bộ lọc có thể được tách thành hai hoặc nhiều tập hợp con của hệ số

và giá trị của hệ số trong mỗi tập hợp con có thể cần tính tổng thành một giá trị cụ thể (ví dụ: giá trị không nhất thiết bằng 0). Ví dụ: theo một ví dụ, bộ lọc có thể được chia (ví dụ: theo chiều ngang, chiều dọc hoặc về đường chéo) thành hai nửa có kích thước bằng nhau. Các hệ số trong nửa thứ nhất có thể được giới hạn để tính tổng thành một giá trị xác định trước (ví dụ: biểu diễn điểm cố định là 0,5) và các hệ số trong nửa thứ hai có thể được giới hạn để tính tổng trừ đi giá trị xác định trước. Giá trị định trước cũng có thể bằng 0.

HÌNH 20A-20B minh họa ví dụ về bộ lọc có gia lượng bằng không với tổng các hệ số bằng không. Nghĩa là, như được minh họa trong Hình 20A-20B, đối với số lượng mẫu giá đỡ bộ lọc, hệ số N, N-1 được báo hiệu và một hệ số được suy ra. Cần lưu ý rằng việc sử dụng bộ lọc có gia lượng bằng không với tổng các hệ số bằng 0 có thể được áp dụng cho mọi kích thước và hình dạng bộ lọc được mô tả trong tài liệu này.

Đối với HÌNH 20A, theo một ví dụ, quy trình lọc tương ứng có thể dựa trên những điểm sau:

Quy trình lọc thành phần chéo cho khối mẫu sắc độ

Giá trị đầu vào của quy trình này là:

- mảng mẫu hình ảnh luma tái tạo $recPicture_L$ trước quá trình lọc vòng thích ứng luma,
- mảng mẫu hình ảnh sắc độ tái tạo đã lọc $alfPicture_C$,
- vị trí sắc độ (xC, yC) xác định mẫu của khối mẫu sắc độ hiện tại trên cùng bên trái so với mẫu của hình ảnh hiện tại trên cùng bên trái,
- chiều rộng ccAlfWidth của khối mẫu sắc độ
- chiều cao ccAlfHeight của khối mẫu sắc độ
- hệ số bộ lọc thành phần chéo CcAlfCoeff[j], với j = 0..7

Đầu ra của quy trình này là mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture.

Vị trí luma khối cây mã hóa (xCtb, yCtb) được suy ra như sau:

$$xCtb = (((xC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY$$

$$yCtb = (((yC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY$$

Để suy ra các mẫu sắc độ tái tạo đã lọc ccAlfPicture[xC + x][yC + y], mỗi mẫu sắc độ tái tạo trong khối các mẫu sắc độ hiện tại alfPicturec[xC + x][yC + y] với x = 0..ccAlfWidth – 1, y = 0..ccAlfHeight – 1, được lọc như sau:

- Vị trí luma (xL, yL) tương ứng với mẫu sắc độ hiện tại ở vị trí sắc độ (xC + x, yC + y) được đặt bằng ((xC + x) * SubWidthC, (yC + y) * SubHeightC)
- Vị trí luma (h_{xL+i}, v_{yL+j}) với i = –1..1, j = –1..2 bên trong mảng recPicture$_L$ được suy ra như sau:
- Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và xL – PpsVirtualBoundariesPosX[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n = 0..pps num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic\_width\_in\_luma\_samples – 1, xL + i)$$

- Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và PpsVirtualBoundariesPosX[n]– xL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] – 1, xL + i)$$

- Nếu không sẽ áp dụng điểm sau đây:

$$h_{x+i} = Clip3(0, pic\_width\_in\_luma\_samples – 1, xL + i)$$

– Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và yL – PpsVirtualBoundariesPosY[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic_height_in_luma_samples – 1, yL + j)

– Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và PpsVirtualBoundariesPosY[n] – yL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] – 1, yL + j)

– Nếu không sẽ áp dụng điểm sau đây:

v_{y+j} = Clip3(0, pic_height_in_luma_samples – 1, yL + j)

– Các biến clipLeftPos, clipRightPos, clipTopPos và clipBottomPos được suy ra bằng cách dẫn ra quy trình dẫn xuất vị trí biên ALF như được xác định bên dưới với (xCtb, yCtb) và (xL – xCtb, yL – yCtb) là đầu vào.

– Độ lệch vị trí mẫu hướng dọc yM2, yM2, yM1, yP1, yP2 và yP3 được quy định trong Bảng 22 theo vị trí mẫu luma hướng dọc yL, clipLeftPos và clipRightPos.

– Độ lệch vị trí mẫu hướng ngang xM1, xM2, yP1 và xP2 được quy định trong Bảng 23 theo vị trí mẫu luma hướng ngang xL, clipLeftPos và clipRightPos.

– Biến curr được suy ra như sau:

curr = alfPicture$_C$[xC + x, yC + y]

– Mảng của hệ số bộ lọc thành phần chéo f[j] được suy ra như sau, với j = 0..7:

$$f[j] = CcAlfCoeff[j]$$

Biến centerValue và sum được suy ra như sau:

centerValue = $recPicture_L[h_x, v_y]$

sum = $f[0] * (recPicture_L[h_x, v_{y+yM1}] - centerValue) +$

$f[1] * (recPicture_L[h_{x+xM1}, v_y] - centerValue) +$

$f[2] * (recPicture_L[h_{x+xP1}, v_{y+yM1}] - centerValue) +$

$f[3] * (recPicture_L[h_{x+xP1}, v_y] - centerValue) +$

$f[4] * (recPicture_L[h_{x+xM1}, v_{y+yP1}] - centerValue) +$

$f[5] * (recPicture_L[h_x, v_{y+yP1}] - centerValue) +$

$f[6] * (recPicture_L[h_{x+xP1}, v_{y+yP1}] - centerValue) +$

$f[7] * (recPicture_L[h_x, v_{y+yP2}] - centerValue)$

sum = curr + (sum + 64) >> 7)

Mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture[xC + x][yC + y] được suy ra như sau:

$$ccAlfPicture[xC + x][yC + y] = Clip3(0, (1 << BitDepth_C) - 1, sum)$$

Đối với HÌNH 20B, theo một ví dụ, quy trình lọc tương ứng có thể dựa trên những điểm sau:

Quy trình lọc thành phần chéo cho khối mẫu sắc độ

Giá trị đầu vào của quy trình này là:

– mảng mẫu hình ảnh luma tái tạo $recPicture_L$ trước quá trình lọc vòng thích ứng luma,

- mảng mẫu hình ảnh sắc độ tái tạo đã lọc alfPicture$_C$,
- vị trí sắc độ (xC, yC) xác định mẫu của khối mẫu sắc độ hiện tại trên cùng bên trái so với mẫu của hình ảnh hiện tại trên cùng bên trái,
- chiều rộng ccAlfWidth của khối mẫu sắc độ
- chiều cao ccAlfHeight của khối mẫu sắc độ
- hệ số bộ lọc thành phần chéo CcAlfCoefff[j], with j = 0..5

Đầu ra của quy trình này là mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture. Vị trí luma khối cây mã hóa (xCtb, yCtb) được suy ra như sau:

$$xCtb = (((xC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY$$

$$yCtb = (((yC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY$$

Để suy ra các mẫu sắc độ tái tạo đã lọc ccAlfPicture[xC + x][yC + y], mỗi mẫu sắc độ tái tạo trong khối các mẫu sắc độ hiện tại alfPicture$_C$[xC + x][yC + y] với x = 0..ccAlfWidth – 1, y = 0..ccAlfHeight – 1, được lọc như sau:

- Vị trí luma (xL, yL) tương ứng với mẫu sắc độ hiện tại ở vị trí sắc độ (xC + x, yC + y) được đặt bằng ((xC + x) * SubWidthC, (yC + y) * SubHeightC)
- Vị trí luma (h_{xL+i}, v_{yL+j}) với i = –1..1, j = –1..1 bên trong mảng recPicture$_L$ được suy ra như sau:
- Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và xL – PpsVirtualBoundariesPosX[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{sL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic\_width\_in\_luma\_samples – 1, xL + i)$$

- Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và

PpsVirtualBoundariesPosX[n]- xL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] – 1, xL + i)$$

– Nếu không sẽ áp dụng điểm sau đây:

$$h_{x+i} = Clip3(0, pic\_width\_in\_luma\_samples – 1, xL + i)$$

– Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và yL – PpsVirtualBoundariesPosY[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic\_height\_in\_luma\_samples – 1, yL + j)$$

– Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và PpsVirtualBoundariesPosY[n] – yL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] – 1, yL + j)$$

– Nếu không sẽ áp dụng điểm sau đây:

$$v_{y+j} = Clip3(0, pic\_height\_in\_luma\_samples – 1, yL + j)$$

- Các biến clipLeftPos, clipRightPos, clipTopPos và clipBottomPos được suy ra bằng cách dẫn ra quy trình dẫn xuất vị trí biên ALF như được xác định bên dưới với (xCtb, yCtb) và (xL – xCtb, yL – yCtb) là đầu vào.
- Độ lệch vị trí mẫu hướng dọc yM2, yM2, yM1, yP1, yP2 và yP3 được quy định trong Bảng 22 theo vị trí mẫu luma hướng dọc yL, clipLeftPos và clipRightPos.
- Độ lệch vị trí mẫu hướng ngang xM1, xM2, xP1 và xP2 được quy định trong Bảng 23 theo vị trí mẫu luma hướng ngang xL, clipLeftPos và clipRightPos.
- Biến curr được suy ra như sau:

$$curr = alfPicture_C[xC + x, yC + y]$$

- Mảng của hệ số bộ lọc thành phần chéo f[j] được suy ra như sau, với j = 0..5:
  f[j] = CcAlfCoeff[j]

Biến centerValue và sum được suy ra như sau:

$centerValue = recPicture_L[h_x, v_y]$

$sum = f[0] * (recPicture_L[h_x, v_{y+yM1}] - centerValue) +$

$f[1] * (recPicture_L[h_{x+xM1}, v_y] - centerValue) +$

$f[2] * (recPicture_L[h_{x+xP1}, v_y] - centerValue) +$

$f[3] * (recPicture_L[h_{x+xM1}, v_{y+yP1}] - centerValue) +$

$f[4] * (recPicture_L[h_x, v_{y+yP1}] - centerValue) +$

$f[5] * (recPicture_L[h_{x+xP1}, v_{y+yP1}] - centerValue) +$

$sum = curr + (sum + 64) >> 7)$

Mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture[xC + x][yC + y] được suy ra như sau:

$$ccAlfPicture[xC + x][yC + y] = Clip3(0, (1 << BitDepth_C) - 1, sum)$$

"Mã hóa video đa năng (Bản dự thảo 7)," Hội nghị lần thứ 16 về ISO/IEC JTC1/SC29/WG11 1-11 diễn ra vào tháng 10 năm 2019 tại Geneva, CH, tài liệu JVET-P2001-vE, được kết hợp vào tài liệu này bằng viện dẫn và được gọi là JVET-P2001, là bản cập nhật của JVET-O2001 và biểu thị sự lặp lại hiện tại văn bản nháp về thông số kỹ thuật mã hóa video tương ứng với dự án VVC. JVET-P2001 bao gồm cấu trúc cú pháp phần đầu ảnh chứa thông tin chung cho tất cả các mảnh của hình ảnh được mã hóa liên quan đến ảnh (PH). Hình 28 minh họa phần của cấu trúc cú pháp của phần đầu ảnh được cung cấp trong JVET-P2001 liên quan đến hoạt động lọc.

picture_header_rbsp( ) {	**Bộ mô tả**
...	
if(sps_sao_enabled_flag) {	
pic_sao_enabled_present_flag	u(1)
if(pic_sao_enabled_present_flag) {	
pic_sao_luma_enabled_flag	u(1)
if(ChromaArrayType != 0)	
pic_sao_chroma_enabled_flag	u(1)
}	
}	
if(sps_alf_enabled_flag) {	
pic_alf_enabled_present_flag	u(1)
if(pic_alf_enabled_present_flag) {	
pic_alf_enabled_flag	u(1)
if(pic_alf_enabled_flag) {	
pic_num_alf_aps_ids_luma	u(3)
for(i = 0; i < pic_num_alf_aps_ids_luma; i++)	
pic_alf_aps_id_luma[i]	u(3)
if(ChromaArrayType != 0)	
pic_alf_chroma_idc	u(2)
if(pic_alf_chroma_idc)	
pic_alf_aps_id_chroma	u(3)
}	
}	
}	

if (!pps_dep_quant_enabled_flag)	
pic_dep_quant_enabled_flag	u(1)
if(!pic_dep_quant_enabled_flag)	
sign_data_hiding_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag) {	
pic_deblocking_filter_override_present_flag	u(1)
if(pic_deblocking_filter_override_present_flag) {	
pic_deblocking_filter_override_flag	u(1)
if(pic_deblocking_filter_override_flag) {	
pic_deblocking_filter_disabled_flag	u(1)
if(!pic_deblocking_filter_disabled_flag) {	
pic_beta_offset_div2	se(v)
pic_tc_offset_div2	se(v)
}	
}	
}	
}	
...	
rbsp_trailing_bits()	
}	

Bảng 28

Đối với Bảng 28, JVET-P2001 đề xuất các ngữ nghĩa sau:

PH chứa thông tin chung cho tất cả các mảnh của hình ảnh được mã hóa liên quan đến PH.

...

pic_sao_enabled_present_flag bằng 1 xác định rằng pic_sao_luma_flag and pic_sao_chroma_flag có mặt trong PH. pic_sao_enabled_present_flag bằng 0 xác định rằng pic_sao_luma_flag và pic_sao_chroma_flag không có mặt trong PH. Khi pic_sao_enabled_present_flag không hiện diện sẽ được suy ra là bằng 0.

pic_sao_luma_flag bằng 1 xác định rằng SAO được kích hoạt đối với thành phần luma trong tất cả mảnh liên quan đến PH; pic_sao_luma_flag bằng 0 xác định rằng SAO

đối với thành phần luma có thể được vô hiệu hóa cho một, hoặc nhiều, hoặc tất cả mảnh liên quan đến PH. Khi pic_sao_luma_flag không hiện diện thì sẽ được suy ra là bằng 0.

pic_sao_chroma_flag bằng 1 xác định rằng SAO được kích hoạt đối với thành phần sắc độ trong tất cả mảnh liên quan đến PH; pic_sao_chroma_flag bằng 0 xác định rằng SAO đối với thành phần sắc độ có thể được vô hiệu hóa đối với một, hoặc nhiều, hoặc tất cả mảnh liên quan đến PH. Khi pic_sao_chroma_flag không hiện diện thì sẽ được suy ra là bằng 0.

pic_alf_enabled_present_flag bằng 1 xác định rằng pic_alf_enabled_present_flag, pic_num_alf_aps_ids_luma, pic_alf_aps_id_luma[i], pic_alf_chroma_idc, và pic_alf_aps_id_chroma có mặt trong PH. pic_alf_enabled_present_flag bằng 0 xác định rằng pic_alf_enabled_flag, pic_num_alf_aps_ids_luma, pic_alf_aps_id_luma[i], pic_alf_chroma_idc, và pic_alf_aps_id_chroma không có mặt trong PH. Khi pic_alf_enabled_present_flag không hiện diện thì sẽ được suy ra là bằng 0.

pic_alf_enabled_flag bằng 1 xác định rằng bộ lọc vòng thích ứng được kích hoạt cho tất cả các mảnh liên quan đến PH và có thể được áp dụng cho thành phần màu Y, Cb hoặc Cr trong mảnh. pic_alf_enabled_flag bằng 0 xác định rằng bộ lọc vòng thích ứng bị vô hiệu hóa đối với một, hoặc nhiều, hoặc tất cả các mảnh liên quan đến PH. Khi không hiện diện, pic_alf_enabled_flag sẽ được suy ra là bằng 0.

pic_num_alf_aps_ids_luma xác định số lượng ALF APS mà các mảnh liên quan đến PH tham chiếu đến.

pic_alf_aps_id_luma[i] xác định adaptation_parameter_set_id của ALF APS thứ i mà thành phần luma của các mảnh liên quan đến PH tham chiếu đến.

Giá trị của alf_luma_filter_signal_flag của đơn vị APS NAL có aps_params_type bằng ALF_APS và adaptation_parameter_set_id bằng pic_alf_aps_id_luma[i] sẽ bằng 1.

pic_alf_chroma_idc bằng 0 xác định rằng bộ lọc vòng thích ứng không được áp dụng cho thành phần màu Cb và Cr. pic_alf_chroma_idc bằng 1 biểu thị rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cb. pic_alf_chroma_idc bằng 2 biểu thị rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cr. pic_alf_chroma_idc bằng 3 biểu thị rằng bộ lọc vòng thích ứng được áp dụng cho thành phần màu Cb và Cr. Khi pic_alf_chroma_idc không hiện diện thì sẽ được suy ra là bằng 0.

pic_alf_aps_id_chroma xác định adaptation_parameter_set_id của ALF APS mà thành phần sắc độ của các mảnh liên quan đến PH tham chiếu đến.

Giá trị của alf_chroma_filter_signal_flag của đơn vị APS NAL có aps_params_type bằng ALF_APS và adaptation_parameter_set_id bằng pic_alf_aps_id_chroma sẽ bằng 1.

pic_dep_quant_enabled_flag bằng 0 xác định rằng lượng tử hóa phụ thuộc bị vô hiệu hóa đối với các mảnh liên quan đến PH. pic_dep_quant_enabled_flag bằng 1 xác định rằng lượng tử hóa phụ thuộc được kích hoạt đối với các mảnh liên quan đến PH. Khi không hiện diện, thì giá trị pic_dep_quant_enabled_flag được suy ra là bằng pps_dep_quant_enable_idc – 1.

sign_data_hiding_enabled_flag bằng 0 xác định rằng ẩn bit ký hiệu. bị vô hiệu hóa. sign_data_hiding_enabled_flag bằng 1 xác định rằng ẩn bit ký hiệu được kích hoạt. Khi sign_data_hiding_enabled_flag không hiện diện thì sẽ được suy ra là bằng 0.

pic_deblocking_filter_override_present_flag bằng 1 xác định rằng pic_deblocking_filter_override_flag có mặt trong PH. pic_deblocking_filter_override_present_flag bằng 0 xác định rằng pic_deblocking_filter_override_flag không có mặt trong PH. Khi pic_deblocking_filter_override_present_flag không hiện diện thì sẽ được suy ra là bằng 0.

pic_deblocking_filter_override_flag bằng 1 xác định rằng thông số tách khối có mặt trong PH. pic_deblocking_filter_override_flag bằng 0 xác định rằng thông số tách khối không có mặt trong PH. Khi không hiện diện, giá trị của pic_pic_deblocking_filter_override_flag được suy ra là bằng 0.

pic_deblocking_filter_disabled_flag bằng 1 xác định rằng hoạt động của bộ lọc tách khối không được áp dụng đối với các mảnh liên quan đến PH. pic_deblocking_filter_disabled_flag bằng 0 xác định rằng hoạt động của bộ lọc tách khối được áp dụng cho các mảnh liên quan đến PH. Khi pic_deblocking_filter_disabled_flag không hiện diện thì sẽ được suy ra là bằng pps_deblocking_filter_disabled_flag.

pic_beta_offset_div2 và **pic_tc_offset_div2** xác định độ lệch thông số tách khối cho β và tC (chia cho 2) đối với cách mảnh liên quan đến PH. Giá trị pic_beta_offset_div2 và pic_tc_offset_div2 đều sẽ nằm trong khoảng –6 đến 6, bao gồm cả hai số này. Khi

không hiện diện thì các giá trị pic_beta_offset_div2 và pic_tc_offset_div2 sẽ được suy ra tương ứng bằng pps_beta_offset_div2 và pps_tc_offset_div2.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc triển khai lọc thành phần chéo sử dụng cấu trúc cú pháp phần đầu ảnh có thể dựa trên cú pháp và ngữ nghĩa sau đây:

picture_header_rbsp( ) {	**Bộ mô tả**
...	
if(sps_sao_enabled_flag) {	
pic_sao_enabled_present_flag	u(1)
if(pic_sao_enabled_present_flag) {	
pic_sao_luma_enabled_flag	u(1)
if(ChromaArrayType != 0)	
pic_sao_chroma_enabled_flag	u(1)
}	
}	
if(sps_alf_enabled_flag) {	
pic_alf_enabled_present_flag	u(1)
if(pic_alf_enabled_present_flag) {	
pic_alf_enabled_flag	u(1)
if(pic_alf_enabled_flag) {	
pic_num_alf_aps_ids_luma	u(3)
for(i = 0; i < pic_num_alf_aps_ids_luma; i++)	
pic_alf_aps_id_luma[i]	u(3)
if(ChromaArrayType != 0)	
pic_alf_chroma_idc	u(2)
if(pic_alf_chroma_idc)	
pic_alf_aps_id_chroma	u(3)
}	
pic_cross_component_alf_cb_enabled_flag	u(1)
if(pic_cross_component_alf_cb_enabled_flag) {	
pic_cross_component_alf_cb_aps_id	u(3)
pic_cross_component_cb_filters_signalled_minus1	ue(v)
}	
pic_cross_component_alf_cr_enabled_flag	u(1)

if(pic_cross_component_alf_cr_enabled_flag) {	
pic_cross_component_alf_cr_aps_id	u(3)
pic_cross_component_cr_filters_signalled_minus1	ue(v)
}	
}	
}	
if (!pps_dep_quant_enabled_flag)	
pic_dep_quant_enabled_flag	u(1)
if(!pic_dep_quant_enabled_flag)	
sign_data_hiding_enabled_flag	u(1)
if(deblockmg_filter_override_enabled_flag) {	
pic_deblocking_filter_override_present_flag	u(1)
if(pic_deblocking_filter_override_present_flag) {	
pic_deblocking_filter_override_flag	u(1)
if(pic_deblocking_filter_override_flag) {	
pic_deblocking_filter_disabled_flag	u(1)
if(!pic_deblocking_filter_disabled_flag) {	
pic_beta_offset_div2	se(v)
pic_tc_offset_div2	se(v)
}	
}	
}	
}	
...	
rbsp_trailing_bits()	
}	

Bảng 29

alf_data( ) {	**Bộ mô tả**
alf_luma_filter_signal_flag	u(1)
alf_chroma_fllter_signal_flag	u(1)
alf_cross_component_cb_fllter_signal_flag	u(1)
alf_cross_component_cr_filter_signal_flag	u(1)
if(alf_luma_filter_signal_flag) {	

alf_luma_clip_flag	u(1)
alf_luma_num_filters_signalled_minus1	ue(v)
if(alf_luma_num_filters_signalled_minus1 > 0) {	
for(filtIdx = 0; filtIdx < NumAlfFilters; filtIdx++)	
alf_luma_coeff_delta_idx[filtIdx]	u(v)
}	
for(sfIdx = 0; sfIdx <= alf_luma_num_filters_signalled_minus1; sfIdx++) {	
for(j = 0; j < 12; j++) {	
alf_luma_coeff_abs[sfIdx][j]	uek(v)
if(alf_luma_coeff_abs[sfIdx][j])	
alf_luma_coeff_sign[sfIdx][j]	u(1)
}	
}	
if(alf_luma_clip_flag) {	
for(sfIdx = 0; sfIdx <= alf_luma_num_filters_signalled_minus1; sfIdx++) {	
for(j = 0; j < 12; j++)	
alf_luma_clip_idx[sfIdx][j]	u(2)
}	
}	
}	
if(alf_chroma_filter_signal_flag) {	
alf_chroma_num_alt_filters_minus1	ue(v)
for(altIdx = 0; altIdx <= alf_chroma_num_alt_filters_minus1; altIdx++) {	
alf_chroma_clip_flag[altIdx]	u(1)
for(j = 0; j < 6; j++) {	
alf_chroma_coeff_abs[altIdx] [j]	uek(v)
if(alf_chroma_coeff_abs[altIdx][j] > 0)	
alf_chroma_coeff_sign[altIdx][j]	u(1)
}	
if(alf_chroma_clip_flag[altIdx]) {	
for(j = 0; j < 6; j++)	

alf_chroma_clip_idx[altIdx][j]	u(2)
}	
}	
}	
if (alf_cross_component_cb_filter_signal_flag) {	
alf_cross_component_cb_filters_signalled_minus1	ue(v)
for(k = 0; k < (alf_cross_component_cb_filters_signalled_mmus1+1); k++) {	
for (j = 0; j < 8; j++)	
alf_cross_component_cb_coeff_plus32[k][j]	u(6)
}	
}	
if (alf_cross_component_cr_filter_signal_flag) {	
alf_cross_component_cr_filters_signalled_minus1	ue(v)
for(k = 0; k < (alf_cross_component_cr_filters_signalled_minus1+1); k++) {	
for (j = 0; j < 8; j++)	
alf_cross_component_cr_coeff_plus32[k][j]	u(6)
}	
}	
}	

Bảng 30

slice_header( ) {	**Bộ mô tả**
...	
if (slice_type != I) {	
...	
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
...	
}	
...	
slice_qp_delta	se(v)
...	
if(sps_sao_enabled_flag && !pic_sao_enabled_present_flag) {	

slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(sps_alf_enabled_flag && !pic_alf_enabled_present_flag) {	
slice_alf_enabled_flag	u(1)
if(slice_alf_enabled_flag) {	
slice_num_alf_aps_ids_luma	u(3)
for(i = 0; i < slice_num_alf_aps_ids_luma; i++)	
slice_alf_aps_id_luma[i]	u(3)
if(ChromaArrayType != 0)	
slice_alf_chroma_idc	u(2)
if(slice_alf_chroma_idc)	
slice_alf_aps_id_chroma	u(3)
}	
if(ChromaArrayType != 0)	
slice_cross_component_alf_cb_enabled_flag	u(1)
if(slice_cross_component_alf_cb_enabled_flag) {	
slice_cross_component_alf_cb_aps_id	u(3)
slice_cross_component_cb_filters_signalled_minus1	ue(v)
}	
if(ChromaArrayType != 0)	
slice_cross_component_alf_cr_enabled flag	u(1)
if(slice_cross_component_alf_cr_enabled_flag) {	
slice_cross_component_alf_cr_aps_id	u(3)
slice_cross_component_cr_filters_signalled_minus1	ue(v)
}	
}	
...	
byte_alignment( )	
}	

Bảng 31

coding_tree_unit( ) {	**Bộ mô tả**

xCtb = CtbAddrX << CtbLog2SizeY	
yCtb = CtbAddrY << CtbLog2SizeY	
if(slice_sao_luma_flag | | slice_sao_chroma_flag)	
sao(CtbAddrX, CtbAddrY)	
if(slice_alf_enabled_flag){	
alf_ctb_flag[0][CtbAddrX][CtbAddrY]	ae(v)
if(alf_ctb_flag[0][CtbAddrX][CtbAddrY]) {	
if(slice_num_alf_aps_ids_luma > 0)	
alf_use_aps_flag	ae(v)
if(alf_use_aps_flag) {	
if(slice_num_alf_aps_ids_luma > 1)	
alf_luma_prev_filter_idx	ae(v)
} else	
alf_luma_fixed_filter_idx	ae(v)
}	
if(slice_alf_chroma_idc = = 1 | | slice_alf_chroma_idc = = 3) {	
alf_ctb_flag[1][CtbAddrX][CtbAddrY]	ae(v)
if(alf_ctb_flag[1][CtbAddrX][CtbAddrY] && aps_alf_chroma_num_alt_filters_minus1 > 0)	
alf_ctb_filter_alt_idx[0][CtbAddrX][CtbAddrY]	ae(v)
}	
if(slice_alf_chroma_idc = = 2 | | slice_alf_chroma_idc = = 3) {	
alf_ctb_flag[2][CtbAddrX][CtbAddrY]	ae(v)
if(alf_ctb_flag[2][CtbAddrX][CtbAddrX] && aps_alf_chroma_num_alt_filters_minus1 > 0)	
alf_ctb_filter_alt_idx[1][CtbAddrX][CtbAddrX]	ae(v)
}	
}	
if (slice_cross_component_alf_cb_enabled_flag)	
alf_ctb_cross_component_cb_idc[CtbAddrX][CtbAddrY]	ae(v)
if (slice_cross_component_alf_cr_enabled_flag)	
alf_ctb_cross_component_cr_idc[CtbAddrX][CtbAddrY]	ae(v)
if(slice_type = = I && qtbtt_dual_tree_intra_flag)	
dual_tree_implicit_qt_split (xCtb, yCtb, CtbSizeY, 0)	

khác	
coding_tree(xCtb, yCtb, CtbSizeY, CtbSizeY, 1, 1, 0, 0, 0, 0, 0, SINGLE_TREE, MODE_TYPE_ALL)	
}	

Bảng 32

Đối với Bảng 29-32, theo một ví dụ, ngữ nghĩa học có thể dựa trên ngữ nghĩa học được cung cấp ở trên và như sau:

pic_alf_enabled_present_flag bằng 1 xác định pic_alf_enabled_flag, pic_num_alf_aps_ids_luma, pic_alf_aps_id_luma[i], pic_alf_chroma_idc, pic_alf_aps_id_chroma, pic_cross_component_alf_cb_enabled_flag, pic_cross_component_alf_cb_aps_id, pic_cross_component_cb_filters_signalled_minus1, pic_cross_component_alf_cr_enabled_flag, pic_cross_component_alf_cr_aps_id, và pic_cross_component_cr_filters_signalled_minus 1 có mặt trong PH. pic_alf_enabled_present_flag bằng 0 xác định rằng pic_alf_enabled_flag, pic_num_alf_aps_ids_luma, pic_alf_aps_id_luma[i], pic_alf_chroma_idc, pic_alf_aps_id_chroma, pic_cross_component_alf_cb_enabled_flag, pic_cross_component_alf_cb_aps_id, pic_cross_component_cb_filters_signalled_minus1, pic_cross_component_alf_cr_enabled flag, pic_cross_component_alf_cr_aps_id, và pic_cross_component_cr_filters_signalled_minus1 không có mặt trong PH. Khi pic_alf_enabled_present_flag không hiện diện thì sẽ được suy ra là bằng 0.

pic_cross_component_alf_cb_enabled_flag bằng 1 xác định rằng bộ lọc Cb thành phần chéo được kích hoạt cho tất cả các mảnh liên quan đến PH và có thể được áp dụng cho thành phần màu Cb trong mảnh. pic_cross_component_alf_cb_enabled_flag bằng 0 xác định rằng bộ lọc Cb thành phần chéo bị vô hiệu hóa đối với một, hoặc nhiều, hoặc tất cả các mảnh liên quan đến PH. Khi không hiện diện, pic_cross_component_alf_cb_enabled_flag sẽ được suy ra là bằng 0.

pic_cross_component_alf_cb_aps_id xác định adaptation_parameter_set_id của ALF APS mà thành phần sắc độ của các mảnh liên quan đến PH tham chiếu đến.

Giá trị của alf_cross_component_cb_filter_signal_flag của đơn vị APS NAL có aps_params_type bằng ALF_APS và adaptation_parameter_set_id bằng pic_cross_component_alf_cb_aps_id sẽ bằng 1.

pic_cross_component_cb_filters_signalled_minus1 cộng 1 xác định số lượng bộ lọc Cb thành phần chéo. Giá trị của pic_cross_component_cb_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến 3.

Khi pic_cross_component_alf_cb_enabled_flag bằng 1, thì yêu cầu về tương thích chuỗi bit là pic_cross_component_cb_filters_signalled_minus1 phải nhỏ hơn hoặc bằng giá trị của alf_cross_component_cb_filters_signalled_minus1 trong ALF APS được tham chiếu do pic_cross_component_alf_cb_aps_id tham chiếu đến.

pic_cross_component_alf_cr_enabled_flag bằng 1 xác định rằng bộ lọc Cr thành phần chéo được kích hoạt cho tất cả các mảnh liên quan đến PH và có thể được áp dụng cho thành phần màu Cr trong mảnh. pic_cross_component_alf_cr_enabled_flag bằng 0 xác định rằng bộ lọc Cr thành phần chéo bị vô hiệu hóa đối với một, hoặc nhiều, hoặc tất cả các mảnh liên quan đến PH. Khi không hiện diện, pic_cross_component_alf_cr_enabled_flag sẽ được suy ra là bằng 0.

pic_cross_component_alf_cr_aps_id xác định adaptation_parameter_set_id của ALF APS mà thành phần màu Cr của các mảnh liên quan đến PH tham chiếu đến.

Giá trị của alf_cross_component_cr_filter_signal_flag của đơn vị APS NAL có aps_params_type bằng ALF_APS và adaptation_parameter_set_id bằng pic_cross_component_alf_cr_aps_id sẽ bằng 1.

pic_cross_component_cr_filters_signalled_minus1 cộng 1 xác định số lượng bộ lọc Cr thành phần chéo. Giá trị của pic_cross_component_cr_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến 3.

Khi pic_cross_component_alf_cr_enabled_flag bằng 1, thì yêu cầu về tương thích chuỗi bit là ppic_cross_component_cr_filters_signalled_minus1 phải nhỏ hơn hoặc bằng giá trị của alf_cross_component_cr_filters_signalled_minus1 trong ALF APS được tham chiếu do pic_cross_component_alf_cr_aps_id chiếu đến.

alf_luma_filter_signal_flag bằng 1 xác định rằng tập hợp bộ lọc luma được báo hiệu. alf_luma_filter_signal_flag bằng 0 xác định rằng tập hợp bộ lọc luma không được báo hiệu.

alf_chroma_filter_signal_flag bằng 1 xác định rằng bộ lọc sắc độ được báo hiệu. alf_chroma_filter_signal_flag bằng 0 xác định rằng bộ lọc sắc độ không được báo hiệu. Khi ChromaArrayType bằng 0 thì alf_chroma_filter_signal_flag sẽ bằng 0.

Không phải tất cả giá trị của alf_luma_filter_signal_flag, alf_chroma_filter_signal_flag, alf_cross_component_cb_filter_signal_flag và alf_cross_component_cr_filter_signal_flag đều bằng 0.

Biến NumAlfFilters xác định số lượng các bộ lọc vòng thích ứng khác nhau được đặt bằng 25.

alf_cross_component_cb_filter_signal_flag bằng 1 xác định rằng bộ lọc Cb thành phần chéo được báo hiệu. alf_cross_component_cb_filter_signal_flag bằng 0 xác định rằng bộ lọc Cb thành phần chéo không được báo hiệu. Khi ChromaArrayType bằng 0 thì alf_cross_component_cb_filter_signal_flag sẽ bằng 0.

alf_cross_component_cb_filters_signalled_minus1 cộng 1 xác định số lượng bộ lọc Cb thành phần chéo được báo hiệu trong ALF APS hiện tại. Giá trị của alf_cross_component_cb_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến 3..

alf_cross_component_cb_coeff_plus32[k][j] trừ 32 xác định giá trị của hệ số thứ j của tập hợp bộ lọc Cb thành phần chéo thứ k được báo hiệu. Khi alf_cross_component_cb_coeff_plus32[k][j] không hiện diện thì sẽ được suy ra là bằng 32.

Hệ số bộ lọc Cb thành phần chéo thứ k được báo hiệu $CcAlfApsCoeff_{Cb}$[adaptation_parameter_set_id][k] với phần tử $CcAlfApsCoeff_{Cb}$[adaptation_parameter_set_id][k][j], với j = 0..7 được suy ra từ:

$CcAlfApsCoeff_{Cb}$[adaptation_parameter_set_id][k][j] =
alf_cross_component_cb_coeff_plus32[k][j] – 32

alf_cross_component_cr_filter_signal_flag bằng 1 xác định rằng bộ lọc Cr thành phần chéo được báo hiệu. alf_cross_component_cr_filter_signal_flag bằng 0 xác định rằng bộ lọc Cr thành phần chéo không được báo hiệu. Khi ChromaArrayType bằng 0 thì alf_cross_component_cr_filter_signal_flag sẽ bằng 0.

alf_cross_component_cr_filters_signalled_minus1 cộng 1 xác định số lượng bộ lọc Cr thành phần chéo được báo hiệu trong ALF APS hiện tại. Giá trị của alf_cross_component_cr_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến 3.

alf_cross_component_cr_coeff_plus32[k][j] trừ 32 xác định giá trị của hệ số thứ j của tập hợp bộ lọc Cr thành phần chéo thứ k được báo hiệu. Khi alf_cross_component_cr_coeff_abs[k][j] không hiện diện thì sẽ được suy ra là bằng 32.

Hệ số bộ lọc Cr thành phần chéo thứ k được báo hiệu $CcAlfApsCoeff_{Cr}$[adaptation_parameter_set_id][k] với phần tử $CcAlfApsCoeff_{Cr}$[adaptation_parameter_set_id][k][j], với j = 0..7 được suy ra từ:

$CcAlfApsCoeff_{Cr}$[adaptation_parameter_set_id][k][j] = alf_cross_component_cr_coeff_plus32[k][j] – 32

slice_cross_component_alf_cb_enabled_flag bằng 0 xác định rằng bộ lọc Cb thành phần chéo không được áp dụng cho thành phần màu Cb. slice_ cross_component_alf_cb_enabled_flag bằng 1 biểu thị rằng bộ lọc Cb thành phần chéo được áp dụng cho thành phần màu Cb. Khi slice_cross_component_alf_cb_enabled_flag không hiện diện thì giá trị này được suy ra là bằng pic_cross_component_alf_cb_enabled_flag.

slice_cross_component_alf_cb_aps_id xác định adaptation_parameter_set_id mà thành phần màu Cb của mảnh đề cập đến. TemporalId của đơn vị APS NAL có apsparamstype bằng với ALF_APS và adaptation_parameter_set_id bằng slice_cross_component_alf_cb_aps_id phải nhỏ hơn hoặc bằng TemporalId của đơn vị NAL mảnh được mã hóa. Khi slice_cross_component_alf_cb_enabled_flag bằng 1 và slicc_cross_component_alf_cb_aps_id không có mặt, giá trị của slice_cross_component_alf_cb_aps_id được suy ra là bằng giá trị của pic_cross_component_alf_cb_aps_id.

Giá trị của alf_cross_component_cb_fliter_signal_flag của đơn vị APS NAL có aps_params_type bằng ALF_APS và adaptation_parameter_set_id bằng slice_cross_componcnt_alf_cb_aps_id sẽ bằng 1.

slice_cross_component_cb_fllters_signalled_minus1 cộng 1 xác định số lượng bộ lọc Cb thành phần chéo. Giá trị của slice_cross_component_cb_filters_signalled_minus1 nằm trong khoảng từ 0 đến 3. Khi slice_cross_component_alf_cb_enabled_flag bằng 1 và slice_cross_component_cb_filters_signalled_minus1 không có mặt, giá trị của slice_cross_component_cb_filters_signalled_minus1 được suy ra là bằng giá trị của pic_cross_component_cb_filters_signalled_minus1.

Khi slice_cross_component_alf_cb_enabled_flag bằng 1, thì yêu cầu về tương thích chuỗi bit là slice_cross_component_cb_filters_signalled_minus1 phải nhỏ hơn hoặc bằng giá trị của alf_cross_component_cb_filters_signalled_minus1 trong ALF APS do slice_cross_component_alf_cb_aps_id của mảnh hiện tại tham chiếu đến.

slice_cross_component_alf_cr_enabled_flag bằng 0 xác định rằng bộ lọc Cr thành phần chéo không được áp dụng cho thành phần màu Cr. slice_cross_component_alf_cb_enabled_flag bằng 1 biểu thị rằng bộ lọc vòng thích ứng thành phần chéo được áp dụng cho thành phần màu Cr. Khi slice_cross_component_alf_cr_enabled_flag không hiện diện thì sẽ được suy ra là bằng pic_cross_component_alf_cr_enabled_flag.

slice_cross_component_alf_cr_aps_id xác định adaptation_parameter_set_id mà thành phần màu Cr của mảnh đề cập đến. TemporalId của đơn vị APS NAL có aps_params_type bằng với ALF_APS và adaptation_parameter_set_id bằng với slice_cross_component_alf_cr_aps_id phải nhỏ hơn hoặc bằng với TemporalId của đơn vị NAL mảnh được mã hóa. Khi slice_cross_component_alf_cr_enabled_flag bằng 1 và slice_cross_component_alf_cr_aps_id không hiện diện thì giá trị của slice_cross_component_alf_cr_aps_id được suy ra là bằng giá trị của pic_cross_component_alf_cr_aps_id.

Giá trị của alf_cross_component_cr_filter_signal_flag của đơn vị APS NAL có aps_params_type bằng ALF_APS và adaptation_parameter_set_id bằng slice_cross_component_alf_cr_aps_id sẽ bằng 1.

slice_cross_component_cr_filters_signalled_minus1 cộng 1 xác định số lượng bộ lọc Cr thành phần chéo. Giá trị của slice_cross_component_cr_filters_signalled_minus1 sẽ nằm trong khoảng từ 0 đến 3. Khi slice_cross_component_alf_cr_enabled_flag bằng 1 và slice_cross_component_cr_filters_signalled_minus1 không hiệ diện, giá trị của slice_cross_component_cr_filters_signalled_minus1 được suy ra là bằng giá trị của pic_cross_component_cr_filters_signalled_minus1.

Khi slice_cross_component_alf_cr_enabled_flag bằng 1, thì yêu cầu về tương thích chuỗi bit là slice_cross_component_cr_filters_signalled_minus1 sẽ phải nhỏ hơn hoặc bằng giá trị của alf_cross_component_cr_filters_signalled_minus1 trong ALF APS được tham chiếu do slice_cross_component_alf_cr_aps_id của mảnh hiện tại tham chiếu đến.

alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] bằng 0 biểu thị rằng bộ lọc Cb thành phần chéo không được áp dụng cho khối của mẫu thành phần màu Cb tại vị trí luma (xCtb, yCtb). alf_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] khác 0 biểu thị rằng bộ lọc Cb thành phần chéo alf_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]-th được áp dụng cho khối của mẫu thành phần màu Cb tại vị trí luma (xCtb, yCtb). Khi alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] không hiện diện thì sẽ được suy ra là bằng 0.

alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] bằng 0 biểu thị rằng bộ lọc Cr thành phần chéo không được áp dụng cho khối của mẫu thành phần màu Cr tại vị trí luma (xCtb, yCtb). alf_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] khác 0 biểu thị rằng bộ lọc Cr thành phần chéo alf_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]-th được áp dụng cho khối của mẫu thành phần màu Cr tại vị trí luma (xCtb, yCtb). Khi alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] không hiện diện thì sẽ được suy ra là bằng 0.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, ví dụ như đối với cú pháp và ngữ nghĩa học được cung cấp ở trên đối với Bảng 29-32, quy trình lọc vòng thích ứng có thể được thực hiện dựa trên những điểm sau:

Đầu vào của quy trình này là mảng mẫu hình ảnh tái tạo trước bộ lọc vòng thích ứng $recPicture_L$ và, khi ChromaArrayType không bằng 0, các mảng $recPicture_{Cb}$ và $recPicture_{Cr}$.

Đầu ra của quy trình này là mảng mẫu hình ảnh tái tạo đã sửa đổi sau bộ lọc vòng thích ứng $alfPicture_L$ và, khi ChromaArrayType khác 0, các mảng $ccAlfPicture_{Cb}$ và $ccAlfPicture_{Cr}$.

Giá trị mẫu trong mảng mẫu hình ảnh tái tạo đã sửa đổi sau bộ lọc vòng thích ứng $alfPicture_L$ và, khi ChromaArrayType khác 0, các mảng $alfPicture_{Cb}$ và $alfPicture_{Cr}$ ban đầu được đặt bằng giá trị mẫu trong mảng mẫu hình ảnh tái tạo trước bộ lọc vòng thích ứng $recPicture_L$ và, khi ChromaArrayType khác 0, các mảng $recPicture_{Cb}$ và $recPicture_{Cr}$, tương ứng.

Áp dụng các bước theo thứ tự sau đây:

Đối với mọi đơn vị cây mã hóa có vị trí khối cây mã hóa luma (rx, ry), trong đó rx = 0..PicWidthInCtbsY – 1 và ry = 0..PicHeightInCtbsY – 1, các điểm sau sẽ được áp dụng:

- Khi alf_ctb_flag[0][rx][ry] bằng 1, quy trình lọc khối cây mã hóa cho mẫu luma như đã xác định được dẫn ra với $recPicture_L$, $alfPicture_L$ và vị trí khối cây mã hóa luma (xCtb, yCtb) được đặt bằng (rx << CtbLog2SizeY, ry << CtbLog2SizeY) làm đầu vào, và đầu ra là $alfPicture_L$ hình ảnh được lọc đã sửa đổi.
- Khi ChromaArrayType khác 0 và alf_ctb_flag[1][rx][ry] bằng 1, quy trình lọc khối cây mã hóa đối với mẫu sắc độ như được xác định được dẫn ra với recPicture được đặt là $recPicture_{Cb}$, alfPicture được đặt bằng $alfPicture_{Cb}$, vị trí khối cây mã hóa sắc độ (xCtbC, yCtbC) được đặt bằng ((rx << CtbLog2SizeY) / SubWidthC, (ry << CtbLog2SizeY) / SubHeightC), và chỉ số bộ lọc sắc độ thay thế altIdx được đặt bằng alf_ctb_filter_alt_idx[0][rx][ry] là đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $alfPicture_{Cb}$.
- Khi ChromaArrayType khác 0 và alf_ctb_flag[2][rx][ry] bằng 1, quy trình lọc khối cây mã hóa đối với mẫu sắc độ như được xác định được dẫn ra với recPicture được đặt là $recPicture_{Cr}$, alfPicture được đặt bằng $alfPicture_{Cr}$, vị trí khối cây mã hóa sắc độ (xCtbC, yCtbC) được đặt bằng ((rx << CtbLog2SizeY) / SubWidthC, (ry << CtbLog2SizeY) / SubHeightC), và chỉ số bộ lọc sắc độ thay thế altIdx được đặt bằng alf_ctb_filter_alt_idx[0][rx][ry] là đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $alfPicture_{Cr}$.
- Khi ChromaArrayType khác 0, các giá trị mẫu trong mảng $ccAlfPicture_{Cb}$ và $ccAlfPicture_{Cr}$ được đặt bằng tương ứng với các giá trị mẫu trong mảng $alfPicture_{Cb}$ và $alfPicture_{Cr}$.
- Đối với mọi đơn vị cây mã hóa có vị trí khối cây mã hóa luma (rx, ry), trong đó rx = 0..PicWidthInCtbsY – 1 và ry = 0..PicHeightInCtbsY – 1, các điểm sau sẽ được áp dụng:
- Khi ChromaArrayType khác 0 và alf_ctb_cross_component_cb_idc[rx][ry] khác 0, quy trình lọc thành phần chéo của mẫu sắc độ như được mô tả bên dưới được dẫn ra với $recPicture_L$ được đặt bằng $recPicture_L$, $alfPicture_C$ được đặt bằng $alfPicture_{Cb}$, vị trí khối cây mã hóa sắc độ (xCtbC, yCtbC) được đặt bằng ((rx << CtbLog2SizeY) / SubWidthC, (ry << CtbLog2SizeY)/

SubHeightC)), vị trí khối cây mã hóa luma (xCtb, yCtb) được đặt bằng (rx << CtbLog2SizeY, ry << CtbLog2SizeY), ccAlfWidth được đặt bằng (1 << << CtbLog2SizeY) / SubWidthC, ccAlfHeight được đặt bằng (1<< << CtbLog2SizeY) / SubHeightC, và hệ số bộ lọc thành phần chéo CcAlfCoeff[j] được đặt bằng $CcAlfCoeff_{Cb}$[slice_cross_component_alf_cb_aps_id][alf_ctb_cross_component_cb_idc[rx][ry] – 1][j], với j = 0..7, làm đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $ccAlfPicture_{Cb}$.

– Khi ChromaArrayType khác 0 và alf_ctb_cross_component_cr_idc[rx][ry] khác 0, quy trình lọc thành phần chéo đối với khối mẫu sắc độ như xác định bên dưới được dẫn ra với $recPicture_L$ được đặt bằng $recPicture_L$, $alfPicture_C$ được đặt bằng $alfPicture_{Cr}$, vị trí khối cây mã hóa sắc độ (xCtbC, yCtbC) được đặt bằng ((rx << CtbLog2SizeY) / SubWidthC, (ry << CtbLog2SizeY)/ SubHeightC)), vị trí khối cây mã hóa luma (xCtb, yCtb) được đặt bằng (rx << CtbLog2SizeY, ry << CtbLog2SizeY), ccAlfWidth được đặt bằng (1 << << CtbLog2SizeY) / SubWidthC, ccAlfHeight được đặt bằng (1<< << CtbLog2SizeY) / SubHeightC, và hệ số bộ lọc thành phần chéo CcAlfCoeff[j] được đặt bằng $CcAlfCoeff_{Cr}$[slice_cross_component_alf_cr_aps_id][alf_ctb_cross_component_cr_idc[rx][ry] – 1][j], với j = 0..7, làm đầu vào, và đầu ra là hình ảnh được lọc đã sửa đổi $ccAlfPicture_{Cr}$.

Quy trình lọc thành phần chéo cho khối mẫu sắc độ

Giá trị đầu vào của quy trình này là:

– mảng mẫu hình ảnh luma tái tạo $recPicture_L$ trước quá trình lọc vòng thích ứng luma,

– mảng mẫu hình ảnh sắc độ tái tạo đã lọc $alfPicture_C$,

– vị trí sắc độ (xCtbC, yCtbC) xác định mẫu của khối cây mã hóa sắc độ hiện tại trên cùng bên trái so với mẫu của hình ảnh hiện tại trên cùng bên trái,

– vị trí luma (xCtb, yCtb) xác định mẫu của khối cây mã hóa luma hiện tại trên cùng bên trái so với mẫu của hình ảnh hiện tại trên cùng bên trái

- chiều rộng ccAlfWidth của khối mẫu sắc độ
- chiều cao ccAlfHeight của khối mẫu sắc độ
- hệ số bộ lọc thành phần chéo CcAlfCoeff[j], với j = 0..7

Đầu ra của quy trình này là mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture.

Để suy ra các mẫu sắc độ tái tạo đã lọc ccAlfPicture[xCtbC + x][yCtbC + y], mỗi mẫu sắc độ tái tạo trong khối các mẫu sắc độ hiện tại alfPicturec[xCtbC + x][yCtbC + y] với x = 0..ccAlfWidth – 1, y = 0..ccAlfHeight – 1, được lọc như sau:

- Vị trí luma (xL, yL) tương ứng với mẫu sắc độ hiện tại ở vị trí sắc độ (xCtbC + x, yCtbC + y) được đặt bằng ((xCtbC + x) * SubWidthC, (yCtbC + y) * SubHeightC)
- Vị trí luma (h_{xL+i}, v_{yL+j}) với i = –1..1, j = –1..2 bên trong mảng recPicture$_L$ được suy ra như sau:
  - Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và xL – PpsVirtualBoundariesPosX[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic\_width\_in\_luma\_samples - 1, xL + i)$$

  - Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosX[n] % CtbSizeY khác 0, và PpsVirtualBoundariesPosX[n]– xL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_ver_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$h_{x+i} = Clip3(0, PpsVinualBoundariesPosX[n] - 1, xL + i)$$

  - Nếu không sẽ áp dụng điểm sau đây:

$$h_{x+i} = Clip3(0, pic\_width\_in\_luma\_samples - 1, xL + i)$$

- Nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và yL – PpsVirtualBoundariesPosY[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic\_height\_in\_luma\_samples - 1, yL + j)$$

- Mặt khác, nếu pps_loop_filter_across_virtual_boundaries_disabled_flag bằng 1, và PpsVirtualBoundariesPosY[n] % CtbSizeY khác 0, và PpsVirtualBoundariesPosY[n] – yL lớn hơn 0 và nhỏ hơn 4 cho mọi n = 0..pps_num_hor_virtual_boundaries – 1, điểm sau sẽ được áp dụng:

$$v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] - 1, yL + j)$$

- Nếu không sẽ áp dụng điểm sau đây:

$$v_{y+j} = Clip3(0, pic\_height\_in\_luma\_samples - 1, yL + j)$$

- Các biến clipLeftPos, clipRightPos, clipTopPos và clipBottomPos được suy ra bằng cách dẫn ra quy trình dẫn xuất vị trí biên ALF như được xác định bên dưới với (xCtb, yCtb) và (xL – xCtb, yL – yCtb) là đầu vào.
- Độ lệch vị trí mẫu hướng dọc yM1, yP1 và yP2 được quy định trong Bảng 33 theo vị trí mẫu luma hướng dọc yL, clipLeftPos và clipRightPos.
- Độ lệch vị trí mẫu hướng ngang xM1 và yP1 được quy định trong Bảng 34 theo vị trí mẫu luma hướng ngang xL, clipLeftPos và clipRightPos.
- Biến curr được suy ra như sau:

$$curr = alfPicture_C[xCtbC + x, yCtbC + y]$$

– Mảng của hệ số bộ lọc thành phần chéo f[j] được suy ra như sau, với j = 0..7:

$$f[j] = CcAlfCoeff[j]$$

– Biến sum được suy ra như sau:

$$sum = f[0] * recPicture_L[h_x, v_{y + yM1}] +$$
$$f[1] * recPicture_L[h_{x+xM1}, v_y] +$$
$$f[2] * recPicture_L[h_x, v_y] +$$
$$f[3] * recPicture_L[h_{x+xP1}, v_y] +$$
$$f[4] * recPicture_L[h_{x+xM1}, v_{y+yP1}] +$$
$$f[5] * recPicture_L[h_x, v_{y+yP1}] +$$
$$f[6] * recPicture_L[h_{x+xP1}, v_{y+yP1}] +$$
$$f[7] * recPicture_L[h_x, v_{y+yP2}]$$

$$deltaBitDepth = BitDepth_Y - BitDepth_C$$

$$scaledSum = (sum + (1 << (6 + deltaBitDepth))) >> (7 + deltaBitDepth)$$

$$scaledSum = Clip3(-(1 << (BitDepth_C - 1)), (1 << (BitDepth_C - 1)) - 1, scaledSum)$$

$$sum = curr + scaledSum$$

– Mảng mẫu hình ảnh sắc độ tái tạo được lọc đã sửa đổi ccAlfPicture[xCtbC + x][yCtbC + y] được suy ra như sau:

$$ccAlfPicture[xCtbC + x][yCtbC + y] = Clip3(0, (1 << BitDepth_C) - 1, sum)$$

Điều kiện	yM1	yP1	yP2
yL = = clipTopPos + 1	–1	1	1
yL = = clipTopPos	0	0	1
yL = = clipBottomPos – 1	0	0	1
yL = = clipBottomPos – 2	–1	1	1

Nếu không	–1	1	2

Bảng 33

Điều kiện	xM1	yP1
xL = = clipLeftPos	0	0
xL = = clipRightPos – 1	0	0
xL = = clipRightPos – 2	–1	1
Nếu không	–1	1

Bảng 34

Cần lưu ý rằng việc thực hiện lọc thành phần chéo dựa trên cú pháp và ngữ nghĩa học trong Bảng 26-32 cung cấp bộ lọc 8 vòi. Đơn xin cấp bằng sáng chế tạm thời Hoa Kỳ số 62/913,065, nộp ngày 9 tháng 10 năm 2019, mỗi đơn đều được kết hợp bằng viện dẫn, đề xuất triển khai ví dụ bộ lọc 6 vòi tương ứng.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, quy trình dẫn xuất vị trí biên ALF có thể được dựa trên điểm sau, trong đó đầu vào bao gồm độ lệch đường biên ảo vbOffset xác định khoảng cách trong các mẫu luma giữa đường biên dưới cùng của khối cây mã hóa hiện tại và đường biên ảo.

Dẫn xuất vị trí biên ALF

Giá trị đầu vào của quy trình này là:

- vị trí luma (xCtb, yCtb) xác định mẫu của khối cây mã hóa luma hiện tại trên cùng bên trái so với mẫu của hình ảnh hiện tại trên cùng bên trái,
- vị trí luma (x, y) xác định mẫu hiện tại so với mẫu trên cùng bên trái của khối cây mã hóa luma hiện tại.

Giá trị đầu ra của quy trình này là:

- vị trí đường biên dọc bên trái clipLeftPos,
- vị trí đường biên dọc bên phải clipRightPos,
- vị trí đường biên ngang bên trên clipTopPos,
- vị trí đường biên ngang bên dưới clipBottomPos,
- chỉ báo đường biên trên cùng bên trái clipTopLeftFlag,

– chỉ báo đường biên trên cùng bên phải clipBotRightFlag.

Các biến clipLeftPos, clipRightPos, clipTopPos và clipBottomPos được đặt bằng –128.

Các biến clipTopLeftFlag và clipBotRightFlag đều được đặt bằng 0.

Biến clipTopPos được sửa đổi như sau:

– Nếu y – (CtbSizeY – 4) lớn hơn hoặc bằng 0, biến clipTopPos được đặt bằng yCtb + CtbSizeY – 4.

– Nếu không, nếu VirtualBoundariesDisabledFlag bằng 1, và yCtb + y – VirtualBoundariesPosY[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với bất kỳ n = 0..VirtualBoundariesNumHor – 1, điểm sau được áp dụng:

clipTopPos = VirtualBoundariesPosY[n]

– Nếu không, nếu y nhỏ hơn 3 và một hoặc nhiều điều kiện sau đúng, biến clipTopPos được đặt bằng yCtb:

– Đường biên trên của khối cây mã hóa hiện tại là đường biên trên cùng của ô, và loop_filter_across_tiles_enabled_flag bằng 0.

– Đường biên trên của khối cây mã hóa hiện tại là đường biên trên cùng của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.

– Nếu đường biên trên cùng của khối cây mã hóa hiện tại là đường biên trên cùng của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Biến clipBottomPos được sửa đổi như sau:

– Nếu VirtualBoundariesDisabledFlag bằng 1, VirtualBoundariesPosY[n] khác pic_height_in_luma_samples – 1 hoặc 0, và VirtualBoundariesPosY[n] – yCtb – y lớn hơn 0 và nhỏ hơn 5 đối với bất kỳ n = 0..VirtualBoundariesNumHor – 1, điểm sau được áp dụng:

clipBottomPos = VirtualBoundariesPosY[n]

- Nếu không, nếu CtbSizeY – 4 – y lớn hơn 0 và nhỏ hơn 5, biến clipBottomPos được đặt bằng yCtb + CtbSizeY – 4.
- Nếu không, nếu CtbSizeY – y nhỏ hơn 5, và một hoặc nhiều điều kiện sau đúng, biến clipBottomPos được đặt bằng yCtb + CtbSizeY:
- Đường biên dưới cùng của khối cây mã hóa hiện tại là đường biên dưới cùng của ô, và loop_filter_across_tiles_enabled_flag bằng 0.
- Đường biên dưới cùng của khối cây mã hóa hiện tại là đường biên dưới cùng của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.
- Đường biên dưới cùng của khối cây mã hóa hiện tại là đường biên dưới cùng của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Biến clipLeftPos được sửa đổi như sau:

- Nếu VirtualBoundariesDisabledFlag bằng 1, và xCtb + x - VirtualBoundariesPosX[n] lớn hơn hoặc bằng 0 và nhỏ hơn 3 đối với bất kỳ n = 0..VirtualBoundariesNumVer – 1, điểm sau được áp dụng:

$$clipLeftPos = VirtualBoundariesPosX[n]$$

- Nếu không, nếu x nhỏ hơn 3, và một hoặc nhiều điều kiện đúng, biến clipLeftPos được đặt bằng xCtb:
- Đường biên trái của khối cây mã hóa hiện tại là đường biên trái của ô, và loop_filter_across_tiles_enabled_flag bằng 0.
- Nếu đường biên trái của khối cây mã hóa hiện tại là đường biên trái của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.
- Nếu đường biên trái của khối cây mã hóa hiện tại là đường biên trái của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Biến clipRightPos được sửa đổi như sau:

- Nếu VirtualBoundariesDisabledFlag bằng 1, và VirtualBoundariesPosX[n] – xCtb – x lớn hơn 0 và nhỏ hơn 5 đối với bất kỳ n = 0..VirtualBoundariesNumVer – 1, điểm sau được áp dụng:

clipRightPos = VirtualBoundariesPosX[n]

- Nếu không, nếu CtbSizeY – x nhỏ hơn 5, và một hoặc nhiều điều kiện sau đúng, biến clipRightPos được đặt bằng xCtb + CtbSizeY.
- Đường biên phải của khối cây mã hóa hiện tại là đường biên phải của ô, và loop_filter_across_tiless_enabled_flag bằng 0.
- Đường biên phải của khối cây mã hóa hiện tại là đường biên phải của mảnh, và loop_filter_across_slices_enabled_flag bằng 0.
- Đường biên phải của khối cây mã hóa hiện tại là đường biên phải của hình ảnh phụ, và loop_filter_across_subpic_enabled_flag[SubPicIdx] bằng 0.

Biến clipTopLeftFlag và clipBotRightFlag được sửa đổi như sau:

- Nếu khối cây mã hóa bao phủ vị trí luma (xCtb, yCtb) và khối cây mã hóa bao phủ vị trí luma (xCtb – CtbSizeY, yCtb – CtbSizeY) thuộc các mảnh khác nhau, và loop_filter_across_slices_enabled_flag bằng 0, clipTopLeftFlag được đặt bằng 1.
- Nếu khối cây mã hóa bao phủ vị trí luma (xCtb, yCtb) và khối cây mã hóa bao phủ vị trí luma (xCtb + CtbSizeY, yCtb + CtbSizeY) thuộc các mảnh khác nhau, và loop_filter_across_slices_enabled_flag bằng 0, clipBotRightFlag được đặt bằng 1.

Như được mô tả ở trên, phần tử cú pháp có thể được mã hóa entropy theo CABAC hoặc tương tự. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, nhị phân hóa **alf_cross_component_cb_idc[][]** và/hoặc **alf_cross_component_cr_idc[][]** có thể được cung cấp trong Bảng 35.

Phần tử cú pháp	Nhị phân hóa

	Quy trình	Thông số đầu vào
alf_ctb_cross_component_cb_idc[ ][ ]	TR	cMax = (slice_cross_component_cb_filters_signalled_minus1 + 1), cRiceParam = 0
alf_ctb_cross_component_cr_idc[ ][ ]	TR	cMax = (slice_cross_component_cr_filters_signalled_minus1 + 1), cRiceParam = 0

Bảng 35

Ngoài ra, trong một ví dụ, theo các kỹ thuật trong tài liệu này, đối với **alf_cross_component_cb_idc[][]** và/hoặc **alf_cross_component_cr_idc[][]** biến pStateIdx0 và pStateIdx1 tương ứng với các chỉ số trạng thái xác suất để mã hóa CABAC có thể được khởi tạo như sau:

Bảng 36 và Bảng 37 chứa các giá trị của biến 6 bit initValue được sử dụng để khởi tạo các biến ngữ cảnh được gán cho các phần tử cú pháp alf_cross_component_cb_flag và alf_cross_component_cr_flag. Trong Bảng 36 và Bảng 37, các giá trị của biến ctxIdx được ánh xạ đến initValue và shiftIdx. Cần lưu ý rằng trong Bảng 36 và Bảng 37, giá trị shiftIdx được sử dụng để suy ra giá trị shifr cho quy trình chuyển tiếp trạng thái. Ngoài ra, giá trị EP biểu thị xác suất ngang nhau và tương ứng với giá trị 35 trong JVET-P2001.

Biến khởi tạo	**ctxIdx of alf_ctb_cross_component_cb_idc**								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
initValue	EP	EP	EP	EP	EP	EP	EP	EP	EP
shiftIdx	0	0	0	0	0	0	0	0	0

Bảng 36

Biến khởi tạo	**ctxIdx of alf_ctb_cross_component_cr_idc**								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
initValue	EP	EP	EP	EP	EP	EP	EP	EP	EP
shiftIdx	0	0	0	0	0	0	0	0	0

Bảng 37

Từ mục nhập trong bảng 6 bit initValue, hai biến 3 bit slopeIdx và offsetIdx được suy ra như đã đề xuất ở trên.

Các biến m và n, được sử dụng khi khởi tạo các biến ngữ cảnh, được suy ra từ slopeIdx và offsetIdx như đã cung cấp ở trên.

Hai giá trị được gán cho pStateIdx0 và pStateIdx1 để khởi tạo được suy ra từ $SliceQp_Y$, như đã cung cấp ở trên.

Hai giá trị được gán cho pStateIdx0 và pStateIdx1 để khởi tạo được suy ra như đã cung cấp ở trên.

ctxIdx mà cần khởi tạo cho mỗi kiểu trong số ba kiểu khởi tạo, được xác định bởi biến initType, được liệt kê trong Bảng 38. Đối với loại mảnh P và B, việc suy ra initType phụ thuộc vào giá trị của phần tử cú pháp cabac_init_flag. Biến initType được suy ra như sau:

if(slice_type = = I)

initType = 0

else if(slice_type = = P)

initType = cabac_init_flag ? 2 : 1

khác

initType = cabac_init_flag ? 1 : 2

Phần tử cú pháp	initType		
	0	1	2
alf_cross_component_cb_flag[ ][ ]	0..2	3..5	6..8
alf_cross_component_cr_flag[ ][ ]	0..2	3..5	6..8

Bảng 38

Từ Bảng 38, biến ctxIdxOffset được đặt bằng với initType. Biến ctxIdx được đặt bằng tổng ctxInc và ctxIdxOffset.

Theo một ví dụ, gán ctxInc được xác định như sau:

Phần tử cú pháp	binIdx					
	0	1	2	3	4	>= 5
alf_ctb_cross_component_cb_idc[ ][ ]	0..2 (Bảng 38)	bypass	bypass	bypass	bypass	bypass

alf_ctb_cross_component_cr_idc [ ][ ]	0..2 (Bảng 38)	bypass	bypass	bypass	bypass	bypass

Bảng 39

– Đối với phần tử cú pháp, alf_ctb_cross_component_cb_idc[x0 >> CtbLog2SizeY][y0 >> CtbLog2SizeY], và alf_ctb_cross_component_cr_idc[x0 >> CtbLog2SizeY][y0 >> CtbLog2SizeY]:

ctxInc = (condL && availableL) + (condA && availableA) + ctxSetIdx * 3

Phần tử cú pháp	condL	condA	ctxSetIdx
alf_ctb_cross_component_cb_idc[x0 >> CtbLog2SizeY][y0 >> CtbLog2SizeY]	alf_ctb_cross_component_cb_idc[xNbL >> CtbLog2SizeY][yNbL >> CtbLog2SizeY]	alf_ctb_cross_component_cb_idc[xNbA >> CtbLog2SizeY][yNbA >> CtbLog2SizeY]	0
alf_ctb_cross_component_cr_idc[x0 >> CtbLog2SizeY][y0 >> CtbLog2SizeY]	alf_ctb_cross_component_cr_idc[xNbL >> CtbLog2SizeY][yNbL >> CtbLog2SizeY]	alf_ctb_cross_component_cr_idc[xNbA >> CtbLog2SizeY][yNbA >> CtbLog2SizeY]	0

Bảng 40

Theo cách này, bộ mã hóa video đại diện cho ví dụ về thiết bị được tạo cấu hình để thu dữ liệu mẫu đã tái tạo cho thành phần dữ liệu video hiện tại, thu dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác, dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác, và áp dụng bộ lọc cho dữ liệu mẫu đã tái tạo đối với thành phần dữ liệu video hiện tại dựa trên bộ lọc thành phần chéo đã dẫn xuất và dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác.

HÌNH 17 là sơ đồ khối minh họa ví dụ về bộ giải mã video mà có thể được tạo cấu hình để giải mã dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này. Theo một ví dụ, có thể tạo cấu hình bộ giải mã video 500 để tái tạo dữ liệu video dựa trên một hoặc nhiều kỹ thuật được mô tả trong tài liệu này. Nghĩa là, bộ giải mã video 500 có thể hoạt động theo cách đối ứng với bộ mã hóa video 200 được mô tả ở trên. Có thể tạo cấu hình bộ giải mã video 500 để

thực hiện giải mã dự đoán nội ảnh và giải mã dự đoán liên khung và do đó, có thể được gọi là bộ giải mã lai. Theo ví dụ được minh họa trong HÌNH 18, bộ giải mã video 500 bao gồm bộ giải mã entropy 502, bộ lượng tử hóa ngược 504, bộ xử lý biến đổi ngược 506, bộ xử lý dự đoán nội ảnh 508, bộ xử lý dự đoán liên khung 510, bộ cộng 512, bộ lọc 514 và bộ đệm tham chiếu 516. Có thể tạo cấu hình bộ giải mã video 500 để giải mã dữ liệu video theo cách nhất quán với hệ thống mã hóa video, hệ thống này có thể thực hiện một hoặc nhiều phương án chuẩn mã hóa video. Lưu ý rằng, mặc dù bộ giải mã video điển hình 500 được minh họa là có khối chức năng riêng biệt, nhưng hình ảnh minh họa đó là nhằm mục đích mô tả và không giới hạn bộ giải mã video 500 và/hoặc các thành phần phụ của chúng đối với cấu trúc phần cứng hoặc phần mềm cụ thể. Có thể hiện thực hóa chức năng của bộ giải mã video 500 bằng cách triển khai kết hợp phần cứng, phần sụn và/hoặc phần mềm bất kỳ.

Như được minh họa trong HÌNH 17, bộ giải mã entropy 502 thu chuỗi bit được mã hóa entropy. Có thể tạo cấu hình bộ giải mã entropy 502 để giải mã phần tử cú pháp được lượng tử hóa và hệ số được lượng tử hóa từ chuỗi bit theo quy trình đối ứng với quy trình mã hóa entropy. Có thể tạo cấu hình bộ giải mã entropy 502 để thực hiện giải mã entropy theo kỹ thuật mã hóa entropy bất kỳ được mô tả ở trên. Bộ giải mã entropy 502 có thể phân tách chuỗi bit được mã hóa theo cách phù hợp với chuẩn mã hóa video. Có thể tạo cấu hình bộ giải mã video 500 để phân tách chuỗi bit đã mã hóa, trong đó chuỗi bit mã hóa này được tạo ra dựa trên các kỹ thuật mô tả ở trên.

Tham chiếu lại đến HÌNH 17, bộ lượng tử hóa ngược 504 thu hệ số biến đổi lượng tử hóa (nghĩa là giá trị mức) và dữ liệu thông số lượng tử hóa từ bộ giải mã entropy 502. Dữ liệu thông số lượng tử hóa có thể bao gồm bất kỳ và tất cả các tổ hợp giá trị QP delta và/hoặc giá trị kích thước nhóm lượng tử hóa và dữ liệu tương tự được mô tả ở trên. Có thể tạo cấu hình bộ giải mã video 500 và/hoặc bộ lượng tử hóa ngược 504 để xác định các giá trị QP được sử dụng cho lượng tử hóa ngược dựa trên giá trị được bộ mã hóa video báo hiệu và/hoặc thông qua đặc tính video và/hoặc thông số mã hóa. Nghĩa là, bộ lượng tử hóa ngược 504 có thể hoạt động theo cách đối ứng với bộ lượng tử hóa hệ số 206 được mô tả ở trên. Ví dụ: bộ lượng tử hóa ngược 504 có thể được tạo cấu hình để suy ra các giá trị đã xác định trước, kích thước nhóm lượng tử hóa cho phép và những thứ tương tự, theo các kỹ thuật được mô tả ở trên. Có thể tạo cấu hình bộ lượng tử hóa ngược 504 để áp dụng lượng tử hóa ngược. Có thể tạo cấu hình bộ xử lý biến đổi ngược 506 để thực hiện biến đổi ngược nhằm tạo ra dữ liệu tái tạo dư. Các kỹ thuật tương ứng do bộ lượng tử hóa ngược 504 và bộ xử lý biến đổi ngược 506 thực

hiện có thể tương tự như các kỹ thuật do bộ xử lý biến đổi/lượng tử hóa ngược 208 được mô tả ở trên thực hiện. Có thể tạo cấu hình bộ xử lý biến đổi ngược 506 để áp dụng DCT ngược, DST ngược, biến đổi số nguyên ngược, Biến đổi thứ cấp không tách rời (NSST) hoặc quy trình biến đổi ngược tương tự về mặt khái niệm đối với hệ số biến đổi để tạo ra các khối dư trong miền pixel. Hơn nữa, như được mô tả ở trên, việc thực hiện một phép biến đổi cụ thể (hoặc kiểu biến đổi cụ thể) có thể phụ thuộc vào chế độ dự đoán nội ảnh. Như được minh họa trong HÌNH 17, dữ liệu tái tạo dư có thể được cung cấp cho bộ cộng 512. Bộ cộng 512 có thể thêm dữ liệu tái tạo dư vào khối video dự đoán và tạo dữ liệu video tái tạo. Có thể xác định một khối video dự đoán theo kỹ thuật video dự đoán (tức là dự đoán nội ảnh và dự đoán liên khung).

Có thể tạo cấu hình bộ xử lý dự đoán nội ảnh 508 để thu phần tử cú pháp dự đoán nội ảnh và truy xuất khối video dự đoán từ bộ đệm tham chiếu 516. Bộ đệm tham chiếu 516 có thể bao gồm thiết bị nhớ được tạo cấu hình để lưu trữ một hoặc nhiều khung hình dữ liệu video. Các phần tử cú pháp dự đoán nội ảnh có thể xác định chế độ dự đoán nội ảnh, chẳng hạn như chế độ dự đoán nội ảnh được mô tả ở trên. Theo một ví dụ, bộ xử lý dự đoán nội ảnh 508 có thể tái tạo khối video thông qua một hoặc nhiều kỹ thuật mã hóa dự đoán nội ảnh được mô tả trong tài liệu này. Bộ xử lý dự đoán liên khung 510 có thể thu phần tử cú pháp dự đoán liên khung và tạo vectơ chuyển động để xác định khối dự đoán trong một hoặc nhiều hệ quy chiếu được lưu trữ trong bộ đệm tham chiếu 516. Bộ xử lý dự đoán liên khung 510 có thể tạo ra các khối bù chuyển động, có thể thực hiện nội suy dựa trên bộ lọc nội suy. Mã định danh cho bộ lọc nội suy sẽ được dùng để ước tính chuyển động với độ chính xác điểm ảnh phụ có thể được đưa vào các phần tử cú pháp. Bộ xử lý dự đoán liên khung 510 có thể sử dụng bộ lọc nội suy để tính toán giá trị nội suy cho pixel số nguyên phụ của khối tham chiếu.

Có thể tạo cấu hình bộ lọc 514 để thực hiện lọc dữ liệu video đã tái tạo. Ví dụ: có thể tạo cấu hình bộ lọc 514 để thực hiện tách khối và/hoặc lọc SAO, như được mô tả ở trên đối với bộ lọc 216. Theo ví dụ, bộ lọc 514 có thể bao gồm bộ lọc thành phần chéo 600 được mô tả bên dưới. Ngoài ra, lưu ý rằng theo một số ví dụ, có thể tạo cấu hình bộ lọc 514 để thực hiện chức năng lọc riêng tùy ý (ví dụ: cải tiến hình ảnh). Như được minh họa trong HÌNH 17, có thể xuất khối video tái tạo bằng bộ giải mã video 500.

Như được mô tả ở trên, HÌNH 7 minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để mã hóa dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này. HÌNH 18 minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để mã hóa dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này. Nghĩa là, bộ lọc thành phần

chéo 600 có thể tính theo cách đối ứng với bộ lọc thành phần chéo 300. Như được minh họa trong HÌNH 18, bộ lọc thành phần 600 bao gồm thiết bị xác định bộ lọc 602 và bộ sửa đổi mẫu 604. Bộ sửa đổi mẫu 604 có thể hoạt động theo cách tương tự như bộ sửa đổi mẫu 304. Nghĩa là, bộ sửa đổi mẫu 604 có thể thực hiện lọc theo bộ lọc được dẫn xuất, bao gồm một hoặc nhiều bộ lọc mô tả trong tài liệu này. Như được minh họa trong HÌNH 18, bộ sửa đổi mẫu 604 có thể xuất khối tái tạo đã sửa đổi đến bộ đệm hình ảnh tham chiếu (nghĩa là dưới dạng bộ lọc trong vòng) và xuất khối tái tạo đã sửa đổi đến đầu ra (ví dụ: màn hình). Thiết bị xác định bộ lọc 602 có thể thu thông tin thông số mã hóa (ví dụ: dự đoán nội ảnh) có sẵn tại thời điểm khối hiện tại được giải mã và có sẵn dữ liệu khối video, như được minh họa trong HÌNH 18 tại bộ giải mã video, có thể bao gồm: Khối được tái tạo thành phần chéo và Khối được tái tạo thành phần hiện tại. Tuy nhiên, như được minh họa trong HÌNH 18, thiết bị xác định bộ lọc 602 có thể thu dữ liệu bộ lọc. Nghĩa là, dữ liệu bộ lọc xác định bộ lọc đã dẫn xuất có thể được báo hiệu đến thiết bị xác định bộ lọc 602. Ví dụ về việc báo hiệu này được mô tả ở trên. Do đó, thiết bị xác định bộ lọc 302 có thể dẫn xuất bộ lọc được sử dụng trên khối tái tạo sắc độ dựa trên dữ liệu video, thông số mã hóa và/hoặc dữ liệu bộ lọc.

Như được mô tả ở trên, các kỹ thuật lọc thành phần chéo được mô tả trong tài liệu này có thể được áp dụng chung cho từng thành phần của dữ liệu video. Do đó, một hoặc nhiều tổ hợp thành phần của dữ liệu video có thể được sử dụng để giảm lỗi tái tạo cho một hoặc nhiều thành phần dữ liệu video khác. HÌNH 19A-19C là các sơ đồ khối minh họa ví dụ về bộ lọc thành phần chéo mà có thể được tạo cấu hình để giảm lỗi tái tạo theo một hoặc nhiều kỹ thuật của sáng chế này. Nghĩa là, HÌNH 19A-19C minh họa các ví dụ về bộ lọc vòng mà có thể có trong bộ lọc 514. Theo HÌNH 19A-19C, các phần tử số thông thường được mô tả ở trên. Bộ lọc Thành phần chéo Cr 702 là ví dụ về bộ lọc được tạo cấu hình để lọc thành phần Cr dựa trên thành phần luma, thành phần Cb, dữ liệu bộ lọc và thông số mã hóa. Bộ lọc Thành phần chéo luma 704 là ví dụ về bộ lọc được tạo cấu hình để lọc thành phần luma dựa trên thành phần luma, thành phần Cb, thành phần Cr, dữ liệu bộ lọc và thông số mã hóa. Do đó, bộ giải mã video 500 biểu diễn ví dụ về cấu hình thiết bị để thu dữ liệu mẫu đã tái tạo cho thành phần dữ liệu video hiện tại, thu dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác, dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác, và áp dụng bộ lọc cho dữ liệu mẫu đã tái tạo đối với thành phần dữ liệu video hiện tại dựa trên bộ lọc thành phần chéo đã dẫn xuất và dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác.

Trong một hoặc nhiều ví dụ thực hiện sáng chế, các chức năng được mô tả có thể được triển khai trong phần cứng, phần mềm, chương trình cơ sở hoặc bất kỳ phương án kết hợp nào giữa chúng. Nếu được triển khai trong phần mềm, các chức năng có thể được lưu trữ hoặc truyền dưới dạng một hoặc nhiều lệnh hoặc mã trên phương tiện có thể đọc được bằng máy tính và được thực thi bởi bộ xử lý dựa trên phần cứng. Phương tiện có thể đọc được bằng máy tính có thể bao gồm phương tiện lưu trữ có thể đọc được bằng máy tính, 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 giúp chuyển chương trình máy tính từ nơi này sang nơi khác, ví dụ, theo một giao thức truyền thông. Theo cách này, môi trường máy tính đọc được thường có thể tương ứng với (1) môi trường lưu trữ hữu hình mà có thể đọc được bằng máy tính, không chuyển tiếp 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 nhập 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 kỹ thuật được mô tả trong sáng chế này. Một sản phẩm chương trình máy tính có thể bao gồm một phương tiện có thể đọc được bằng máy tính.

Ví dụ và không giới hạn, phương tiện lưu trữ có thể đọc được bằng máy tính như vậy có thể bao gồm RAM, ROM, EEPROM, CD-ROM hoặc bộ nhớ đĩa quang khác, bộ lưu trữ đĩa từ hoặc các thiết bị lưu trữ từ tính khác, bộ nhớ flash hoặc bất kỳ phương tiện nào khác có thể dùng để lưu trữ mã chương trình mong muốn dưới dạng lệnh hoặc cấu trúc dữ liệu và máy tính có thể truy nhập được. Ngoài ra, bất kỳ kết nối nào cũng được gọi là phương tiện có thể đọc được bằng máy tính. Ví dụ: nếu lệnh được truyền từ một trang web, máy chủ hoặc nguồn từ xa khác bằng cáp đồng trục, cáp quang, cáp xoắn đôi, đường dây thuê bao số (DSL) hoặc các công nghệ không dây như hồng ngoại, vô tuyến và vi sóng, thì cáp đồng trục, cáp quang, cáp xoắn đôi, DSL, hoặc các công nghệ không dây như hồng ngoại, vô tuyến và vi ba được bao hàm trong định nghĩa về phương tiện. Tuy nhiên, cần hiểu rằng phương tiện lưu trữ có thể đọc được bằng máy tính và phương tiện lưu trữ dữ liệu không bao gồm các kết nối, sóng mang, tín hiệu hoặc các phương tiện chuyển tiếp khác, mà thay vào đó là hướng đến phương tiện lưu trữ hữu hình, không chuyển tiếp. Trong bản mô tả này, đĩa disk và đĩa disc bao gồm đĩa (disc) compact (CD), đĩa (disc) laze, đĩa (disc) quang, đĩa (disc) đa năng số (DVD), đĩa (disk) mềm và đĩa (disc) Blu-ray, trong đó các đĩa disk thường sao chép dữ liệu từ tính, trong khi các đĩa disc sao chép dữ liệu quang học với laze. Việc sử dụng kết hợp các phương tiện trên cũng thuộc phạm vi của phương tiện có thể đọc được bằng máy tính.

Các lệnh có thể do một hoặc nhiều bộ xử lý thực thi, chẳng hạn như một hoặc nhiều bộ xử lý tín hiệu số (DSP), bộ vi xử lý đa năng, mạch tích hợp cho ứng dụng cụ thể (ASIC), mảng logic khả lập bằng trường (FPGA) hoặc mạch logic tích hợp hoặc rời rạc tương đương khác. Do đó, thuật ngữ "bộ xử lý", như được sử dụng trong tài liệu này, có thể đề cập đến bất kỳ cấu trúc nào đã nêu ở trên hoặc bất kỳ cấu trúc nào khác phù hợp để thực hiện các kỹ thuật được mô tả trong bản mô tả này. Ngoài ra, theo một số khía cạnh, chức năng được mô tả trong tài liệu này có thể được trang bị trong các mô-đun phần cứng và/hoặc phần mềm chuyên dụng được đinh cấu hình để mã hóa và giải mã, hoặc được tích hợp trong một codec kết hợp. Ngoài ra, có thể thực hiện đầy đủ các kỹ thuật trong một hoặc nhiều mạch hoặc phần tử logic.

Có thể thực hiện đầy đủ các kỹ thuật của sáng chế này trong nhiều loại thiết bị hoặc dụng cụ, bao gồm thiết bị cầm tay không dây, mạch tích hợp (IC) hoặc một bộ IC (ví dụ: một bộ chip). Các thành phần, mô-đun hoặc thiết bị khác nhau được mô tả trong sáng chế này để nhấn mạnh các khía cạnh chức năng của thiết bị được định cấu hình để thực hiện các kỹ thuật được đề xuất, nhưng không nhất thiết phải thực hiện bởi các thiết bị phần cứng khác nhau. Thay vào đó, như đã mô tả ở trên, các thiết bị khác nhau có thể được sử dụng kết hợp trong một thiết bị phần cứng codec hoặc được cung cấp bởi một tập hợp thiết bị phần cứng tương thích với nhau, bao gồm một hoặc nhiều bộ xử lý như mô tả ở trên, kết hợp với phần mềm và/hoặc chương trình cơ sở phù hợp.

Ngoài ra, mỗi khối chức năng hoặc nhiều tính năng của thiết bị trạm gốc và thiết bị đầu cuối được sử dụng trong mỗi phương án đã nêu trên có thể được thực hiện hoặc triển khai bởi hệ mạch, thường là một mạch tích hợp hoặc nhiều mạch tích hợp. Hệ mạch được thiết kế để thực hiện các chức năng được mô tả trong đặc tả kỹ thuật này có thể bao gồm bộ xử lý đa dụng, bộ xử lý tín hiệu số (DSP), mạch tích hợp chuyên dụng hoặc áp dụng chung (ASIC), hệ mạch mảng phần tử logic có thể lập trình (FPGA), hoặc các thiết bị lôgic có thể lập trình khác, các cổng rời rạc hoặc lôgic bóng bán dẫn, bộ phận phần cứng rời rạc, hoặc kết hợp của chúng. Bộ xử lý đa dụng có thể là bộ vi xử lý, hoặc, bộ xử lý có thể là bộ xử lý thông thường, bộ điều khiển, bộ vi điều khiển hoặc máy trạng thái. Bộ xử lý đa dụng hoặc mỗi mạch được mô tả trên đây có thể được tạo cấu hình bằng mạch số hoặc có thể được tạo cấu hình bằng mạch tương tự. Ngoài ra, khi công nghệ chế tạo mạch tích hợp thay thế các mạch tích hợp có mặt ở thời điểm hiện tại nhờ sự tiến bộ của công nghệ bán dẫn, mạch tích hợp theo công nghệ này cũng có thể được sử dụng.

Nhiều ví dụ thực hiện sáng chế khác nhau đã được mô tả. Đây là các ví dụ khác nằm trong phạm vi của các yêu cầu bảo hộ sau đây.

<Bản chất kỹ thuật của sáng chế>

Theo một ví dụ, phương pháp giảm lỗi tái tạo trong dữ liệu video, phương pháp này bao gồm: bước thu dữ liệu mẫu được tái tạo cho thành phần dữ liệu video hiện tại; bước thu dữ liệu mẫu được tái tạo cho một hoặc nhiều thành phần dữ liệu video khác; bước dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác; và bước áp dụng bộ lọc cho dữ liệu mẫu đã tái tạo cho thành phần dữ liệu video hiện tại dựa trên bộ lọc thành phần chéo đã dẫn xuất và dữ liệu mẫu đã tái tạo cho một hoặc nhiều thành phần dữ liệu video khác.

Theo một ví dụ, phương pháp này còn bao gồm thông tin báo hiệu liên quan đến bộ lọc thành phần chéo được dẫn xuất.

Theo một ví dụ, phương pháp này, trong đó bước dẫn xuất bộ lọc thành phần chéo bao gồm bước phân tách tín hiệu để xác định thông số bộ lọc thành phần chéo.

Theo một ví dụ, phương pháp này, trong đó bước dẫn xuất bộ lọc thành phần chéo dựa trên dữ liệu liên quan đến một hoặc nhiều thành phần dữ liệu video khác bao gồm bước dẫn xuất bộ lọc thành phần chéo dựa trên lỗi tái tạo đã biết.

Theo một ví dụ, phương pháp này, bộ lọc thành phần chéo được xác định theo hệ số bộ lọc.

Theo một ví dụ, thiết bị để mã hóa dữ liệu video, thiết bị bao gồm một hoặc nhiều bộ xử lý được tạo cấu hình để thực hiện bất kỳ và tất cả các phương án kết hợp của các bước.

Theo một ví dụ, thiết bị, trong đó thiết bị bao gồm bộ mã hóa video.

Theo một ví dụ, thiết bị, trong đó thiết bị bao gồm bộ giải mã video.

Theo một ví dụ, hệ thống bao gồm: bộ mã hóa video; và thiết bị bao gồm bộ giải mã video.

Theo một ví dụ, thiết bị để mã hóa dữ liệu video, thiết bị này bao gồm các phương tiện để thực hiện bước bất kỳ và tất cả các kết hợp của các bước.

Theo một ví dụ, phương tiện lưu trữ có thể đọc được bằng máy tính, không chuyển tiếp bao gồm các lệnh được lưu trữ trên đó, khi được thực thi, sẽ khiến một hoặc nhiều bộ xử lý của thiết bị mã hóa dữ liệu video thực hiện bất kỳ và tất cả phương án kết hợp theo các bước.

Theo một ví dụ, phương pháp lọc dữ liệu video được tái tạo, phương pháp này bao gồm: bước phân tích cú pháp phần tử cú pháp đầu tiên được sử dụng để thiết lập hệ số lọc thành phần chéo; bước nhập mảng mẫu của hình ảnh luma đã tái tạo, bước suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại; bước suy ra mảng hệ số lọc bằng cách sử dụng hệ số lọc thành phần chéo; bước suy ra biến bằng cách sử dụng mảng hệ số lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma; và bước suy ra biến tỷ lệ bằng cách sử dụng biến, trong đó biến được sửa đổi bằng cách tính tổng của một mẫu khối sắc độ hiện tại, được xác định bằng vị trí đã xác định trước, và biến tỷ lệ.

Theo một ví dụ, phương pháp, trong đó phần tử cú pháp là phần tử cú pháp để xác định mã nhận dạng tập hợp thông số thích ứng và phần tử cú pháp có trong phần đầu mảnh.

Theo một ví dụ, phương pháp còn bao gồm thêm: bước giải mã phần tử cú pháp thứ hai được sử dụng để thiết lập hệ số bộ lọc thành phần chéo, trong đó phần tử cú pháp thứ hai này có trong cấu trúc cú pháp đơn vị cây mã hóa.

Theo một ví dụ, phương pháp, trong đó phần tử cú pháp thứ nhất được giải mã theo một giá trị của chỉ báo được kích hoạt bộ lọc vòng thích ứng thành phần chéo.

Theo một ví dụ, phương pháp, trong đó phần tử cú pháp thứ hai xác định liệu bộ lọc thành phần chéo có được áp dụng cho một khối hay không.

Theo một ví dụ, phương pháp, trong đó phần tử cú pháp thứ hai xác định liệu bộ lọc thành phần chéo có được áp dụng cho một khối hay không.

Theo một ví dụ, phương pháp, trong đó phần tử cú pháp thứ hai được mã hóa entropy bằng cách sử dụng quy trình nhị phân hóa Truncated Rice.

Theo một ví dụ, thiết bị mã hóa dữ liệu video, thiết bị này bao gồm một hoặc nhiều bộ xử lý được tạo cấu hình để: mã hóa phần tử cú pháp đầu tiên được sử dụng để thiết lập hệ số lọc thành phần chéo; nhập mảng mẫu của hình ảnh luma đã tái tạo; suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại; suy ra mảng hệ số lọc bằng cách sử dụng hệ số lọc thành phần chéo; suy ra biến bằng cách sử dụng mảng hệ số

lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma; và suy ra biến tỷ lệ bằng cách sử dụng biến đó, trong đó biến được sửa đổi bằng cách tính tổng của một mẫu khối sắc độ hiện tại, được xác định bằng vị trí đã xác định trước, và biến tỷ lệ.

Theo một ví dụ, thiết bị giải mã dữ liệu video, thiết bị này bao gồm một hoặc nhiều bộ xử lý được tạo cấu hình để: giải mã phần tử cú pháp đầu tiên được sử dụng để thiết lập hệ số lọc thành phần chéo; nhập mảng mẫu của hình ảnh luma đã tái tạo; suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại; suy ra mảng hệ số lọc bằng cách sử dụng hệ số lọc thành phần chéo; suy ra biến bằng cách sử dụng mảng hệ số lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma; và suy ra biến tỷ lệ bằng cách sử dụng biến đó, trong đó biến được sửa đổi bằng cách tính tổng của một mẫu khối sắc độ hiện tại, được xác định bằng vị trí đã xác định trước, và biến tỷ lệ.

<Tham chiếu chéo>

Đơn xin cấp bằng sáng chế không tạm thời này yêu cầu quyền ưu tiên theo 35 U.S.C. § 119 về Đơn xin cấp bằng sáng chế tạm thời số 62/913,065 ngày 9 tháng 10 năm 2019. số 62/950,000 ngày 18 tháng 12, nội dung của đơn được hợp nhất đầy đủ vào tài liệu này bằng viện dẫn.

YÊU CẦU BẢO HỘ

1. Phương pháp lọc dữ liệu video được tái tạo, phương pháp này bao gồm:

bước phân tách phần tử cú pháp thứ nhất , từ phần đầu mảnh, theo giá trị của chỉ báo được kích hoạt bộ lọc vòng thích ứng thành phần chéo;

bước phân tách phần tử cú pháp thứ hai, từ phần tử cú pháp đơn vị cây mã hóa, theo giá trị của chỉ báo được kích hoạt bộ lọc vòng thích ứng thành phần chéo;

bước phân tách phần tử cú pháp thứ ba từ các phần tử cú pháp phần đầu hình ảnh;

bước suy ra giá trị phần tử cú pháp thứ nhất bằng với giá trị phần tử cú pháp thứ ba trong trường hợp chỉ báo được kích hoạt bộ lọc vòng thích ứng thành phần chéo bằng 1 và phần tử cú pháp thứ nhất vắng mặt;

bước thiết lập các hệ số bộ lọc thành phần chéo nhờ sử dụng phần tử cú pháp thứ nhất và phần tử cú pháp thứ hai;

bước nhập mảng mẫu của hình ảnh luma đã tái tạo trước quy trình lọc vòng thích ứng luma;

bước nhập mảng mẫu hình ảnh sắc độ đã được tái tạo được lọc theo quy trình lọc vòng thích ứng sắc độ;

bước suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại;

bước suy ra mảng hệ số lọc bằng cách sử dụng hệ số lọc thành phần chéo;

bước suy ra biến bằng cách sử dụng mảng hệ số lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma;

bước suy ra biến tỷ lệ bằng cách sử dụng biến này;

bước suy ra biến được sửa đổi bằng cách tính tổng của mảng mẫu hình ảnh sắc độ được tái tạo và biến tỷ lệ; và

bước suy ra mảng mẫu hình ảnh sắc độ được tái tạo được lọc được sửa đổi nhờ sử dụng biến được sửa đổi.

2. Thiết bị để giải mã dữ liệu video, thiết bị này bao gồm một hoặc nhiều bộ xử lý được tạo cấu hình để:

giải mã phần tử cú pháp thứ nhất, từ phần đầu mảnh, theo giá trị của chỉ báo được kích hoạt bộ lọc vòng thích ứng thành phần chéo;

phân tách phần tử cú pháp thứ hai, từ phần tử cú pháp đơn vị cây mã hóa, theo giá trị của chỉ báo được kích hoạt bộ lọc vòng thích ứng thành phần chéo;

phân tách phần tử cú pháp thứ ba từ các phần tử cú pháp phần đầu hình ảnh;

suy ra giá trị phần tử cú pháp thứ nhất bằng với giá trị phần tử cú pháp thứ ba trong trường hợp chỉ báo được kích hoạt bộ lọc vòng thích ứng thành phần chéo bằng 1 và phần tử cú pháp thứ nhất vắng mặt;

thiết lập các hệ số bộ lọc thành phần chéo nhờ sử dụng phần tử cú pháp thứ nhất và phần tử cú pháp thứ hai;

nhập mảng mẫu của hình ảnh luma đã tái tạo trước quy trình lọc vòng thích ứng luma;

nhập mảng mẫu hình ảnh sắc độ đã được tái tạo được lọc theo quy trình lọc vòng thích ứng sắc độ;

suy ra vị trí luma bằng cách sử dụng vị trí tương ứng với mẫu sắc độ hiện tại;

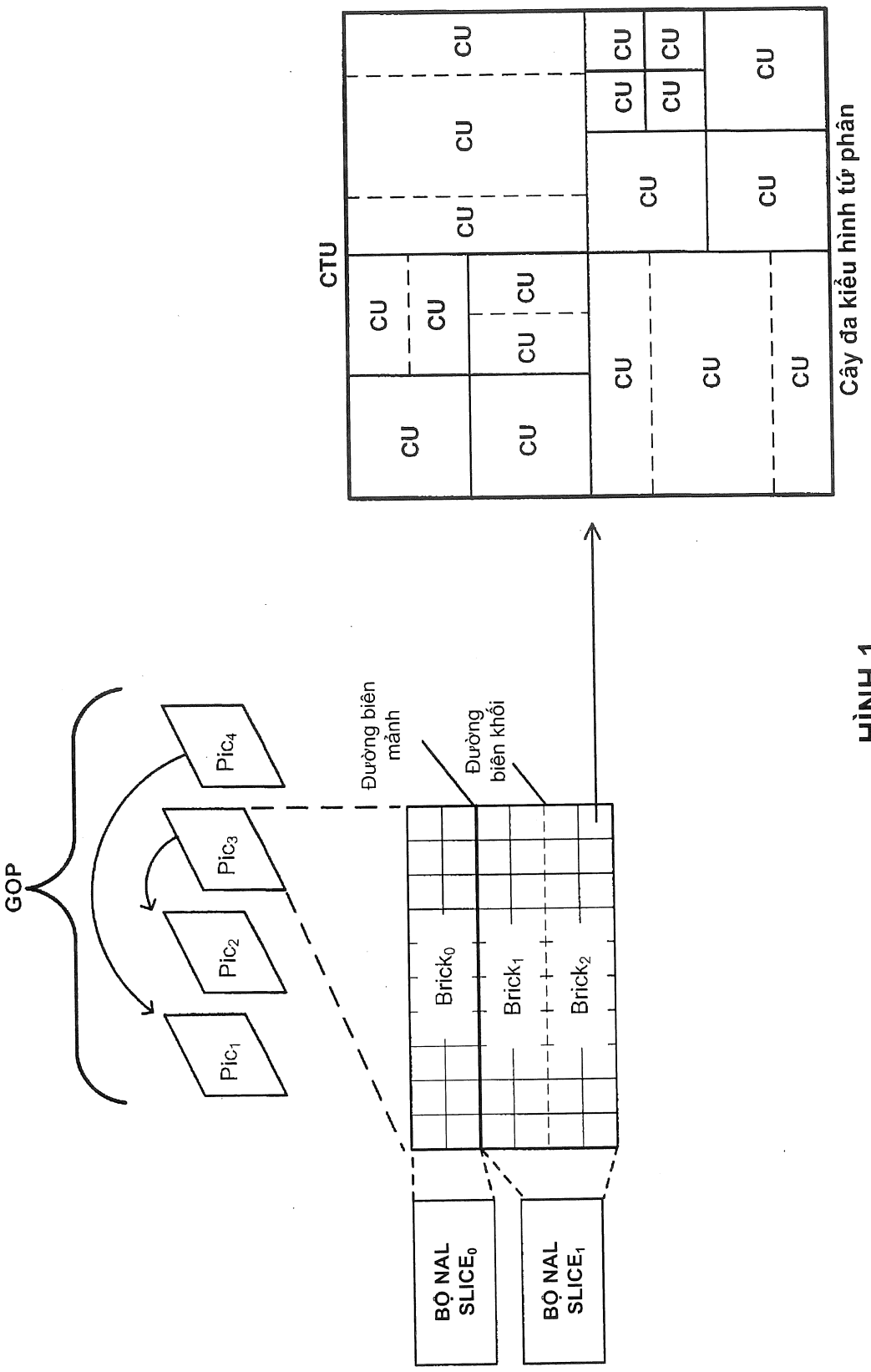
suy ra mảng hệ số lọc bằng cách sử dụng hệ số bộ lọc thành phần chéo;

suy ra biến bằng cách sử dụng mảng hệ số bộ lọc và mảng mẫu hình ảnh luma tái tạo được xác định bởi các vị trí luma;

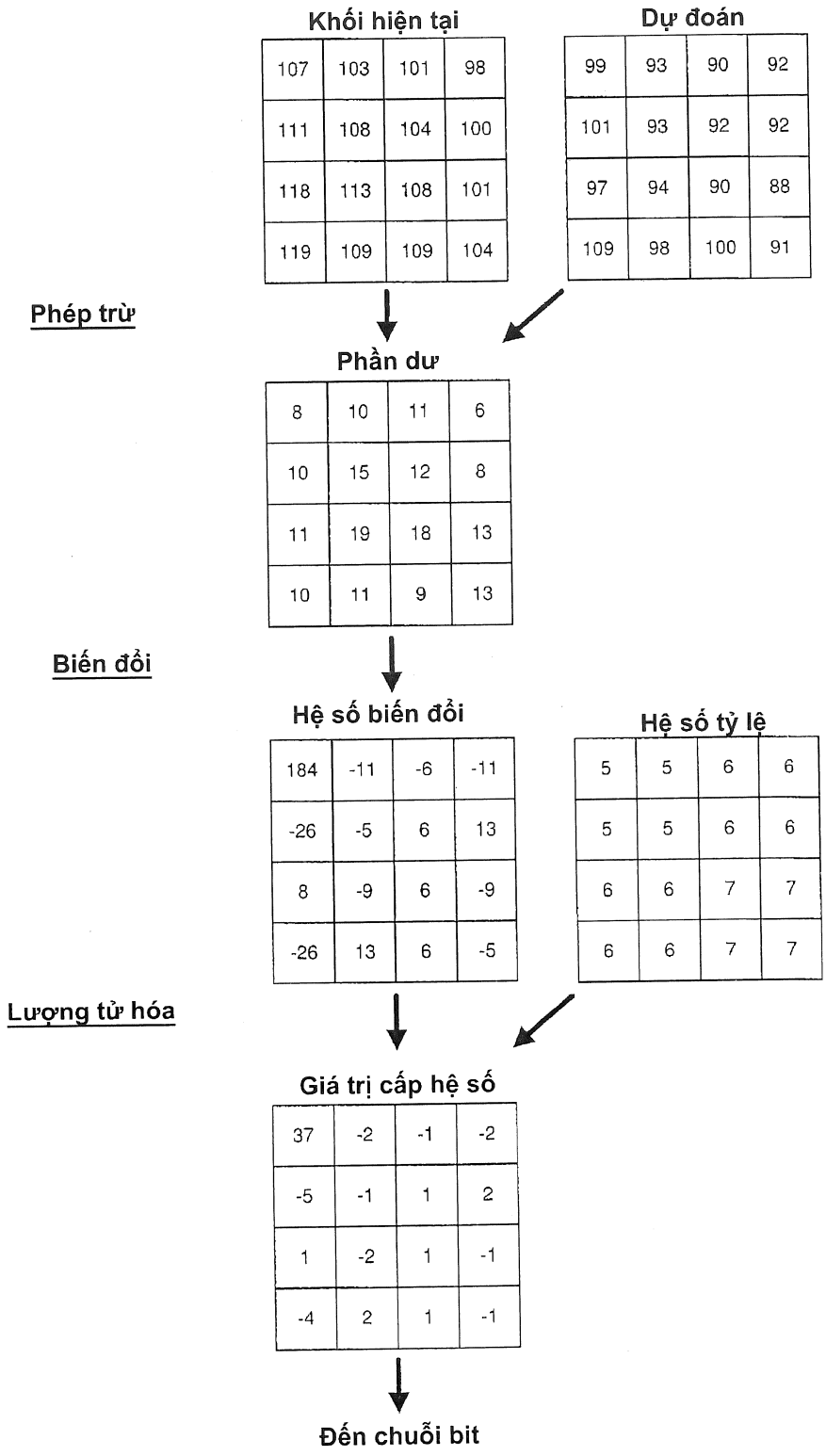
suy ra biến tỷ lệ bằng cách sử dụng biến này;

suy ra biến được sửa đổi bằng cách tính tổng của mảng mẫu hình ảnh sắc độ được tái tạo và biến tỷ lệ;

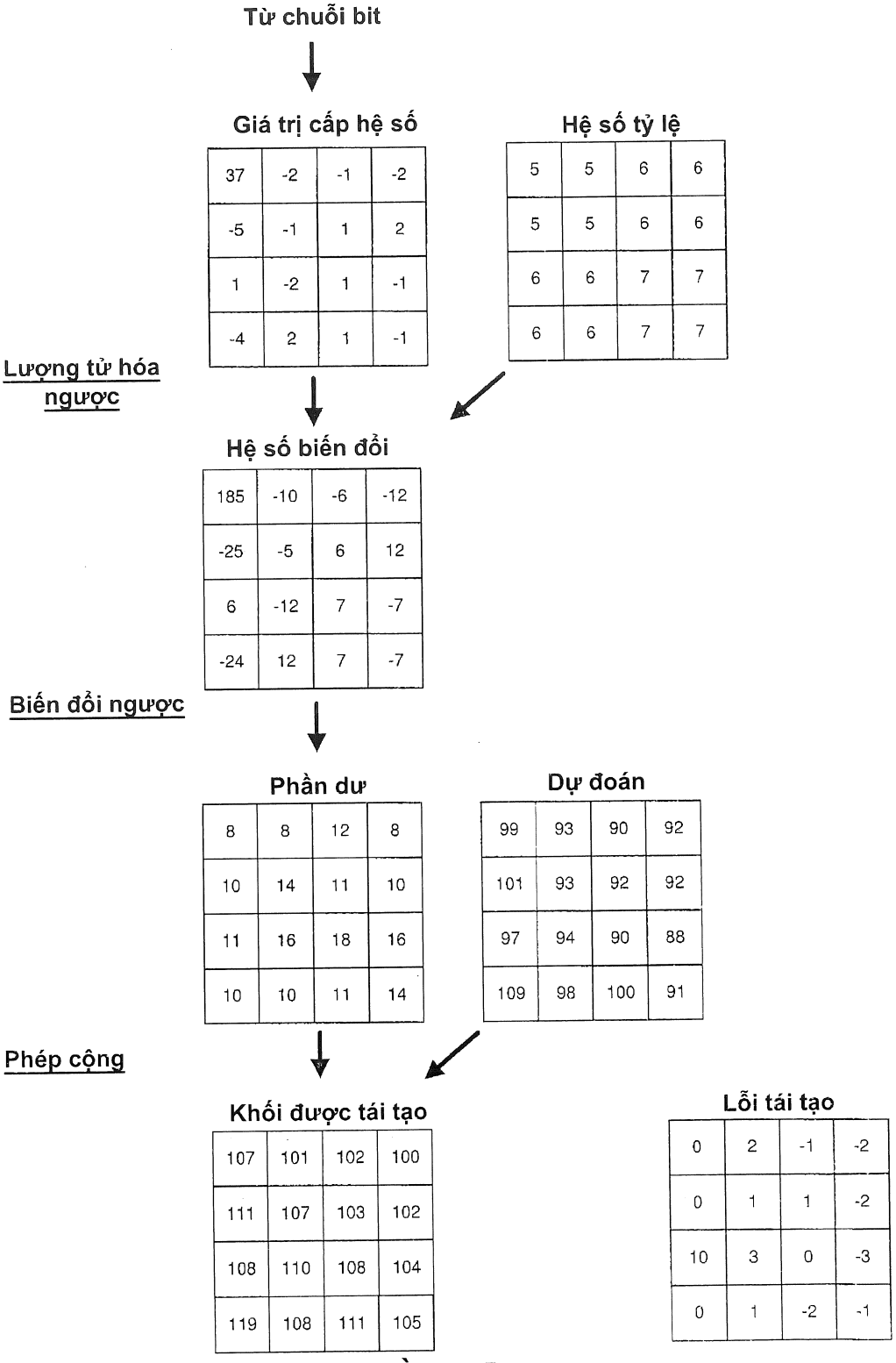
suy ra mảng mẫu hình ảnh sắc độ được tái tạo được lọc được sửa đổi nhờ sử dụng biến được sửa đổi.


GOP
Pic1
Pic2
Pic3
Pic4
Đường biên mảnh
Đường biên khối
Brick0
Brick1
Brick2
BỘ NAL SLICE0
BỘ NAL SLICE1
CTU
CU
Cây đa kiểu hình tứ phân

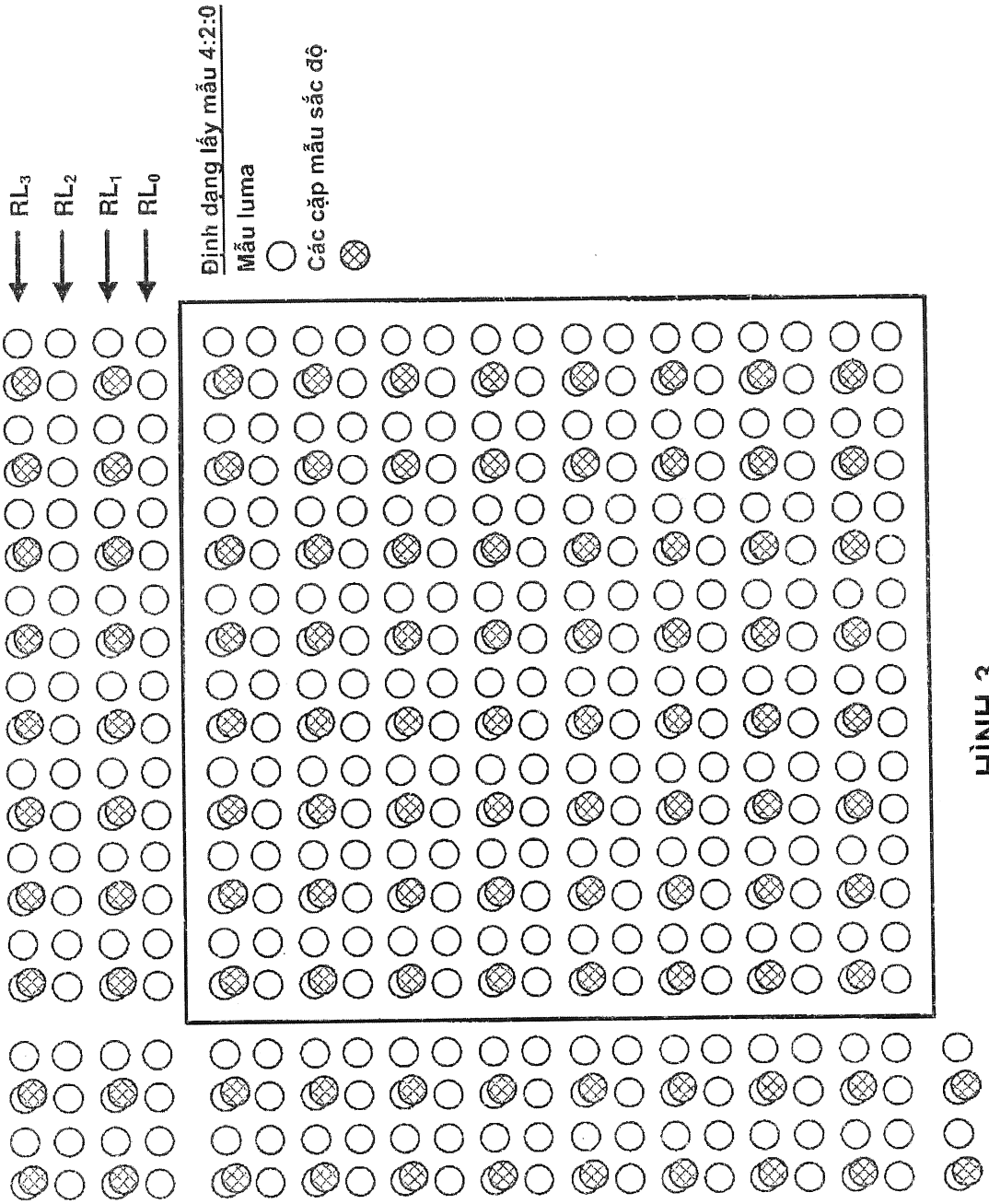
HÌNH 1


Khối hiện tại
107 103 101 98
111 108 104 100
118 113 108 101
119 109 109 104
Dự đoán
99 93 90 92
101 93 92 92
97 94 90 88
109 98 100 91
Phép trừ
Phần dư
8 10 11 6
10 15 12 8
11 19 18 13
10 11 9 13
Biến đổi
Hệ số biến đổi
184 -11 -6 -11
-26 -5 6 13
8 -9 6 -9
-26 13 6 -5
Hệ số tỷ lệ
5 5 6 6
5 5 6 6
6 6 7 7
6 6 7 7
Lượng tử hóa
Giá trị cấp hệ số
37 -2 -1 -2
-5 -1 1 2
1 -2 1 -1
-4 2 1 -1
Đến chuỗi bit

HÌNH 2A


Từ chuỗi bit
Giá trị cấp hệ số
37 -2 -1 -2
-5 -1 1 2
1 -2 1 -1
-4 2 1 -1
Hệ số tỷ lệ
5 5 6 6
5 5 6 6
6 6 7 7
6 6 7 7
Lượng tử hóa ngược
Hệ số biến đổi
185 -10 -6 -12
-25 -5 6 12
6 -12 7 -7
-24 12 7 -7
Biến đổi ngược
Phần dư
8 8 12 8
10 14 11 10
11 16 18 16
10 10 11 14
Dự đoán
99 93 90 92
101 93 92 92
97 94 90 88
109 98 100 91
Phép cộng
Khối được tái tạo
107 101 102 100
111 107 103 102
108 110 108 104
119 108 111 105
Lỗi tái tạo
0 2 -1 -2
0 1 1 -2
10 3 0 -3
0 1 -2 -1

HÌNH 2B


RL3
RL2
RL1
RL0
Định dạng lấy mẫu 4:2:0
Mẫu luma
Các cặp mẫu sắc độ

HÌNH 3

Loại vị trí chroma 0

HÌNH 4A

Loại vị trí chroma 1

HÌNH 4B

Loại vị trí chroma 4

HÌNH 4C

Loại vị trí chroma 2

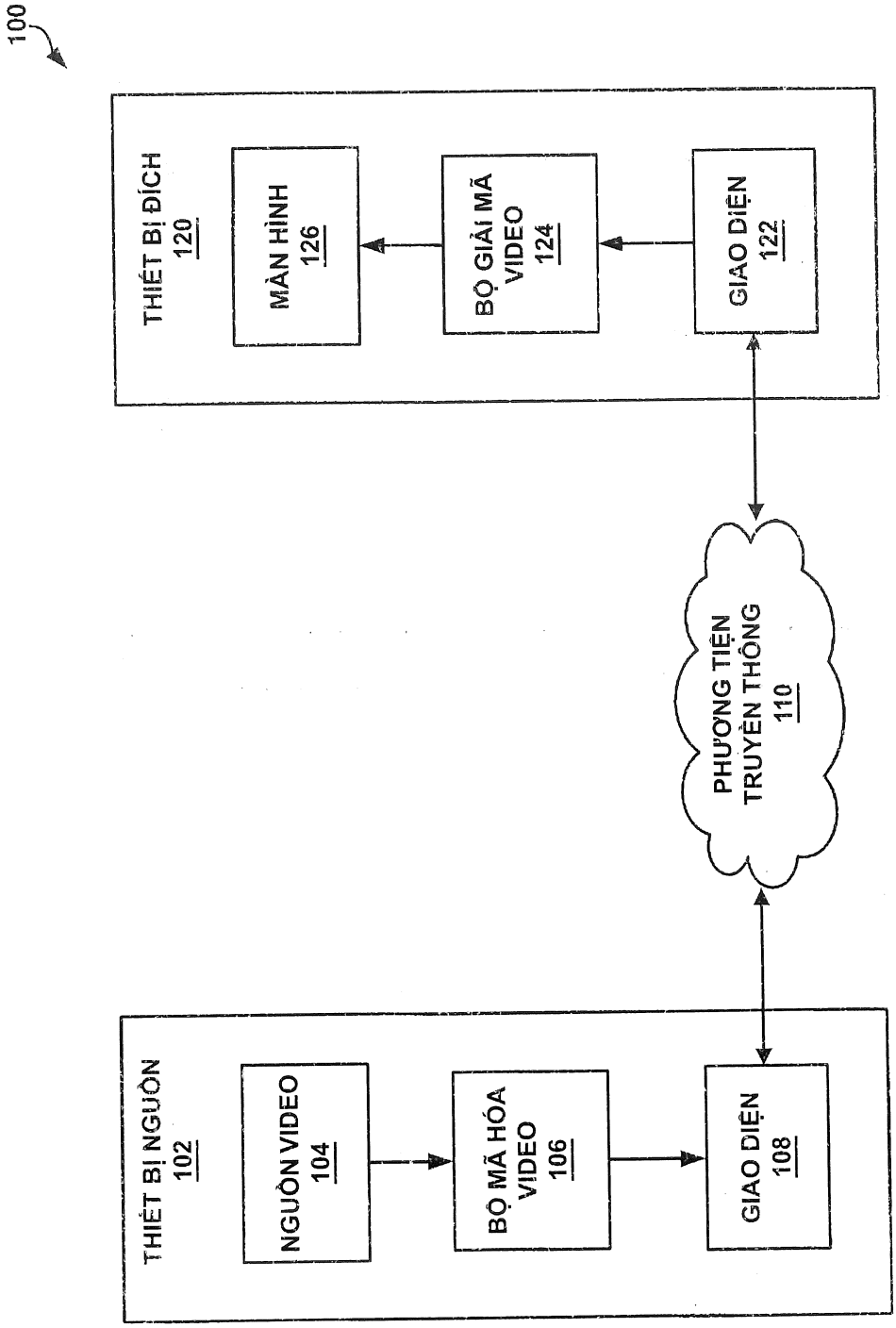
HÌNH 4D

Loại vị trí chroma 3

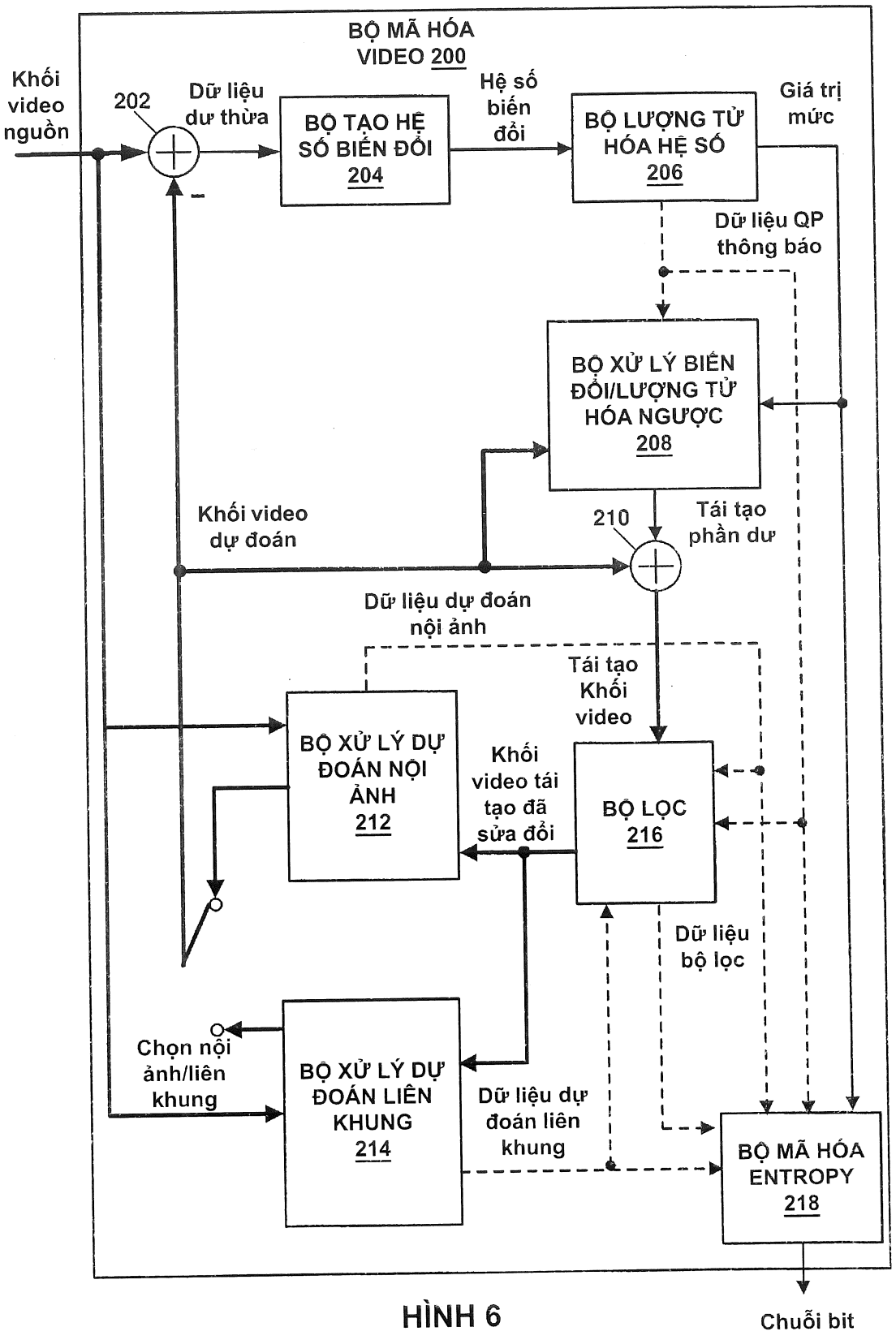
HÌNH 4E

Loại vị trí chroma 5

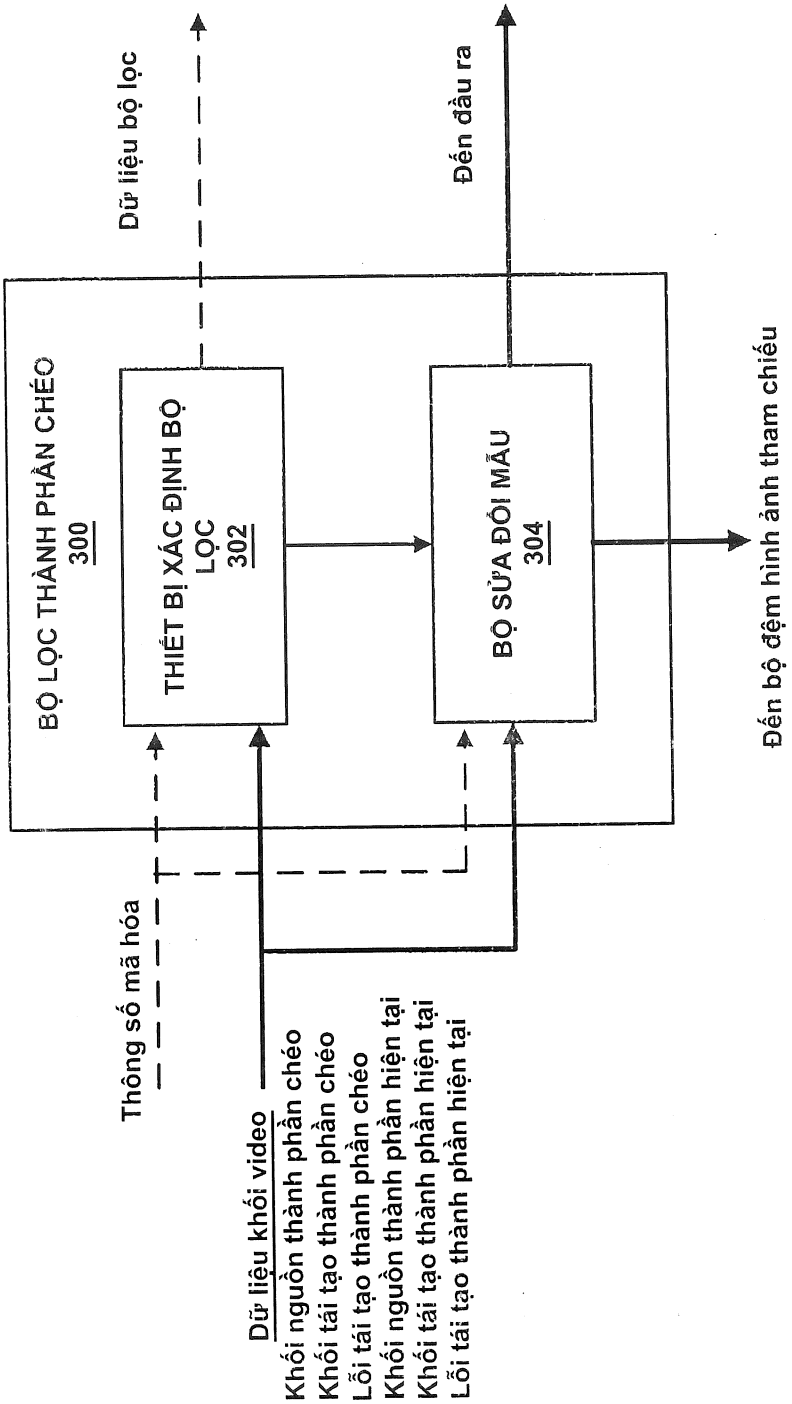
HÌNH 4F


100
THIẾT BỊ NGUỒN
102
NGUỒN VIDEO
104
BỘ MÃ HÓA VIDEO
106
GIAO DIỆN
108
PHƯƠNG TIỆN TRUYỀN THÔNG
110
THIẾT BỊ ĐÍCH
120
GIAO DIỆN
122
BỘ GIẢI MÃ VIDEO
124
MÀN HÌNH
126

HÌNH 5


BỘ MÃ HÓA VIDEO 200
Khối video nguồn
202
Dữ liệu dư thừa
BỘ TẠO HỆ SỐ BIẾN ĐỔI 204
Hệ số biến đổi
BỘ LƯỢNG TỬ HÓA HỆ SỐ 206
Giá trị mức
Dữ liệu QP thông báo
BỘ XỬ LÝ BIẾN ĐỔI/LƯỢNG TỬ HÓA NGƯỢC 208
Khối video dự đoán
210
Tái tạo phần dư
Dữ liệu dự đoán nội ảnh
Tái tạo Khối video
BỘ XỬ LÝ DỰ ĐOÁN NỘI ẢNH 212
Khối video tái tạo đã sửa đổi
BỘ LỌC 216
Dữ liệu bộ lọc
Chọn nội ảnh/liên khung
BỘ XỬ LÝ DỰ ĐOÁN LIÊN KHUNG 214
Dữ liệu dự đoán liên khung
BỘ MÃ HÓA ENTROPY 218
Chuỗi bit

HÌNH 6


BỘ LỌC THÀNH PHẦN CHÉO
300
THIẾT BỊ XÁC ĐỊNH BỘ LỌC
302
BỘ SỬA ĐỔI MẪU
304
Dữ liệu bộ lọc
Đến đầu ra
Đến bộ đệm hình ảnh tham chiếu
Thông số mã hóa
Dữ liệu khối video
Khối nguồn thành phần chéo
Khối tái tạo thành phần chéo
Lỗi tái tạo thành phần chéo
Khối nguồn thành phần hiện tại
Khối tái tạo thành phần hiện tại
Lỗi tái tạo thành phần hiện tại

HÌNH 7

Khối nguồn LUMA

107	101	102	100	107	101	102	50
111	107	103	102	111	107	51	48
108	110	108	104	108	51	45	45
119	108	111	105	50	49	48	45
107	101	102	52	53	50	50	42
111	107	51	52	48	45	46	40
108	50	51	48	46	45	41	42
53	50	51	48	41	41	40	42

Khối tái tạo LUMA

107	99	103	98	107	99	103	52
111	106	102	104	111	106	50	50
98	107	108	107	98	48	45	48
119	107	113	106	50	48	50	46
107	99	103	54	53	48	51	44
111	106	50	54	48	44	45	42
98	47	51	51	36	42	41	45
53	49	53	49	41	40	42	43

Lỗi tái tạo LUMA

0	2	-1	-2	0	2	-1	-2
0	1	1	-2	0	1	1	-2
10	3	0	-3	10	3	0	-3
0	1	-2	-1	0	1	-2	-1
0	2	-1	-2	0	2	-1	-2
0	1	1	-2	0	1	1	-2
10	3	0	-3	10	3	0	-3
0	1	-2	-1	0	1	-2	-1

Khối nguồn Chroma

52	53	54	20
51	52	21	25
52	20	20	18
22	20	25	17

Khối tái tạo Chroma

36	36	36	36
36	36	36	36
36	36	36	36
36	36	36	36

Lỗi tái tạo sắc độ

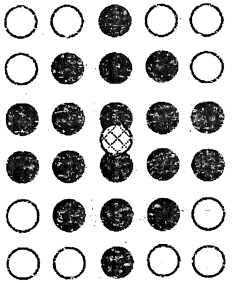
16	17	18	-16
15	16	-15	-11
16	-16	-16	-18
-14	-16	-11	-19

HÌNH 8

Mẫu giá đỡ

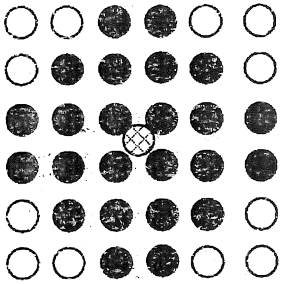
Mẫu cần lọc

Loại vị trí chroma 0



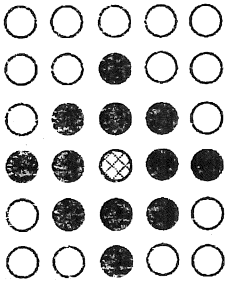
HÌNH 9A

Loại vị trí chroma 1



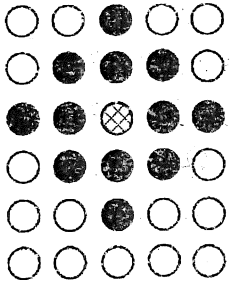
HÌNH 9B

Loại vị trí chroma 4



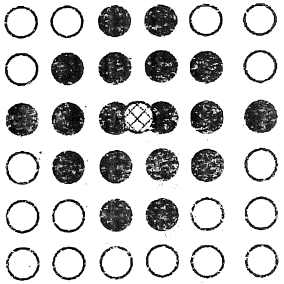
HÌNH 9C

Loại vị trí chroma 2



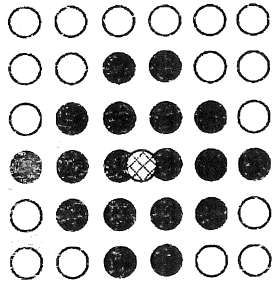
HÌNH 9D

Loại vị trí chroma 3



HÌNH 9E

Loại vị trí chroma 5



HÌNH 9F

Lỗi tái tạo sắc độ

36	36	36	36
36	36	36	36
36	36	36	36
36	36	36	36

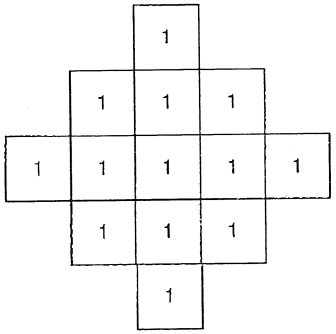
Khối nguồn Chroma

52	53	54	20
51	52	21	25
52	20	20	18
22	20	25	17

Lỗi tái tạo sắc độ trước lọc

16	17	18	-16
15	16	-15	-11
16	-16	-16	-18
-14	-16	-11	-19

Giá đỡ bộ lọc



Khối tái tạo LUMA

107	99	103	98	107	99	103	52
111	106	102	104	111	106	50	50
98	107	108	107	98	48	45	48
119	107	113	106	50	48	50	46
107	99	103	54	53	48	51	44
111	106	50	54	48	44	45	42
98	47	51	51	36	42	41	45
53	49	53	49	41	40	42	43

Trung bình của các mẫu giá đỡ luma (ánh sáng) tại mỗi vị trí sắc độ

104	104	103	76
108	105	84	62
105	85	61	46
78	62	49	43

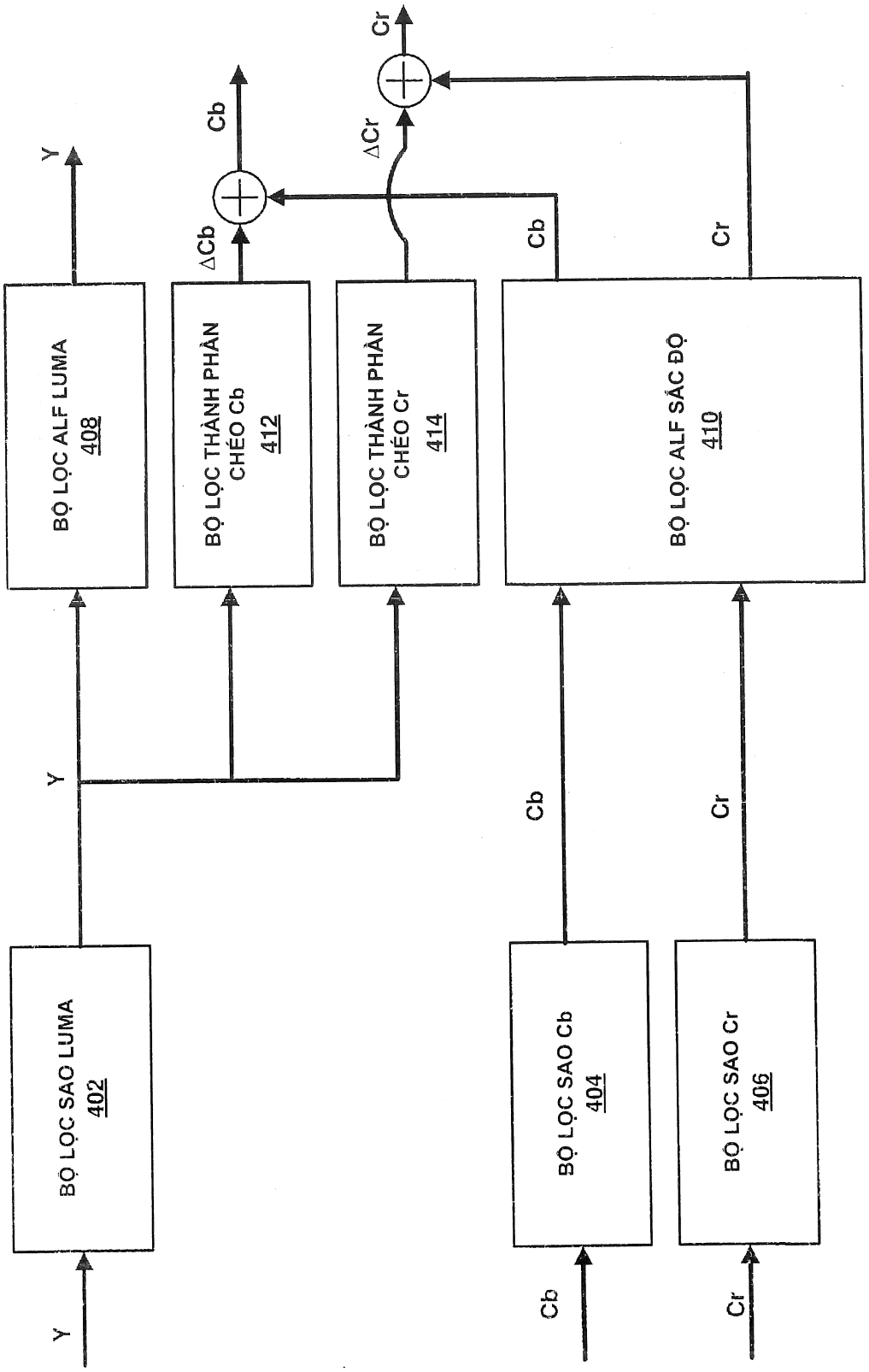
Khối tái tạo sắc độ đã sửa đổi

46	46	46	28
47	47	28	30
47	27	30	31
28	30	31	32

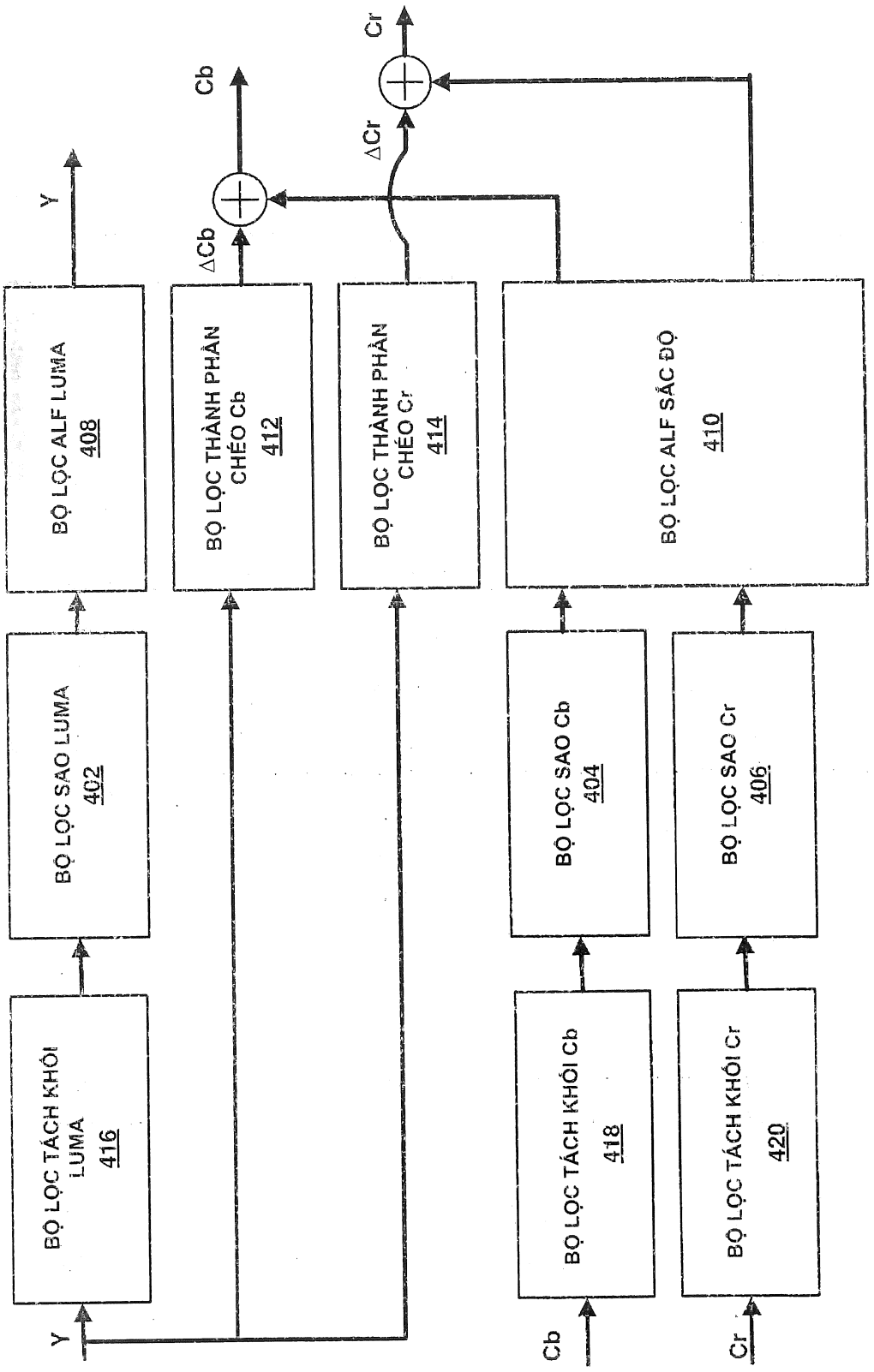
Lỗi tái tạo sắc độ sau lọc

6	7	8	-8
4	5	-7	-5
5	-7	-10	-13
-6	-10	-6	-15

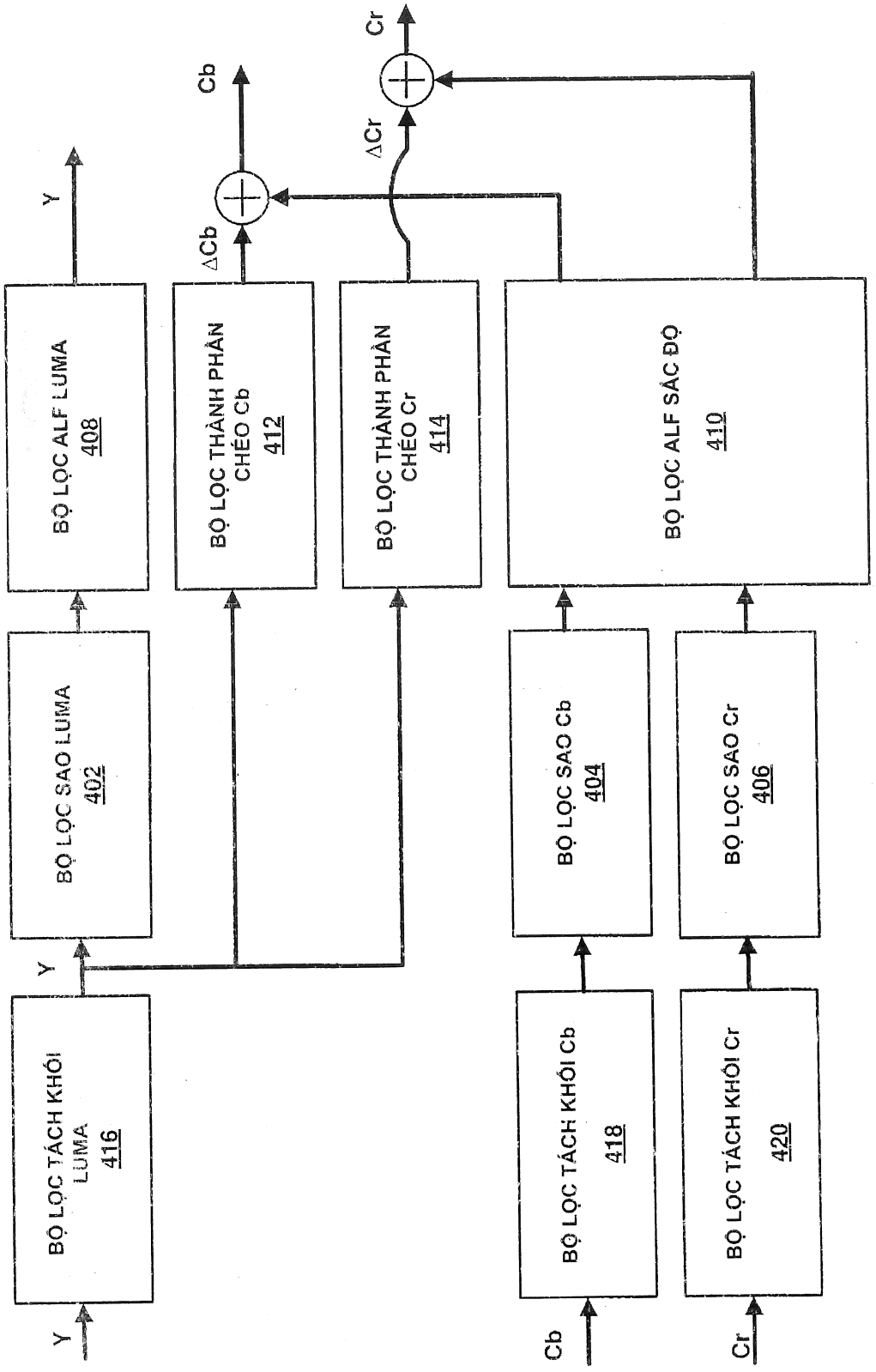
HÌNH 10


Y
Cb
Cr
BỘ LỌC SAO LUMA
402
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cr
406
BỘ LỌC ALF LUMA
408
BỘ LỌC THÀNH PHẦN CHÉO Cb
412
BỘ LỌC THÀNH PHẦN CHÉO Cr
414
BỘ LỌC ALF SẮC ĐỘ
410
ΔCb
ΔCr

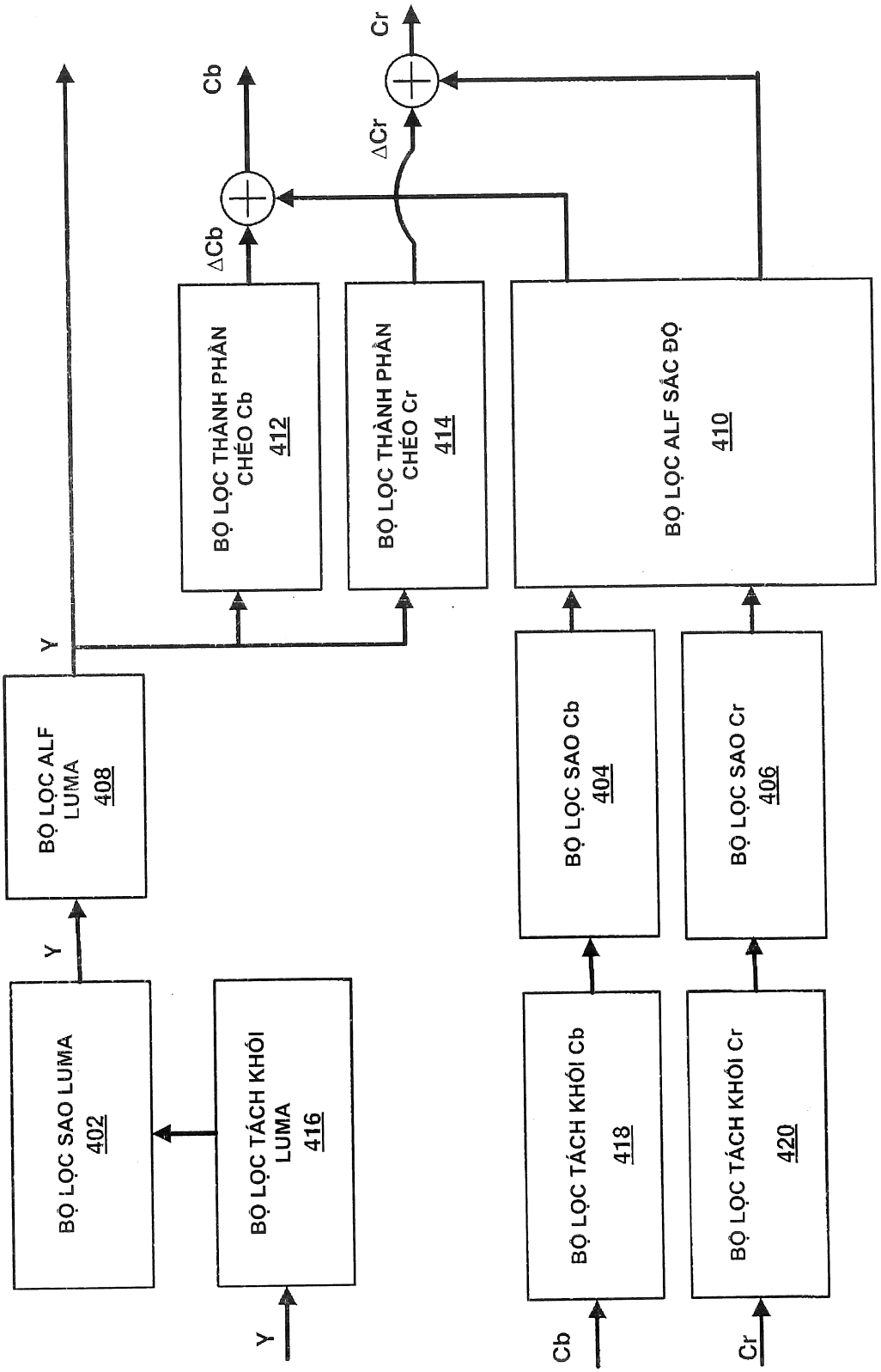
HÌNH 11A


Y
BỘ LỌC ALF LUMA
408
BỘ LỌC SAO LUMA
402
BỘ LỌC TÁCH KHỐI LUMA
416
Y
Cb
ΔCb
BỘ LỌC THÀNH PHẦN CHÉO Cb
412
Cr
ΔCr
BỘ LỌC THÀNH PHẦN CHÉO Cr
414
BỘ LỌC ALF SẮC ĐỘ
410
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cr
406
Cb
BỘ LỌC TÁCH KHỐI Cb
418
Cr
BỘ LỌC TÁCH KHỐI Cr
420

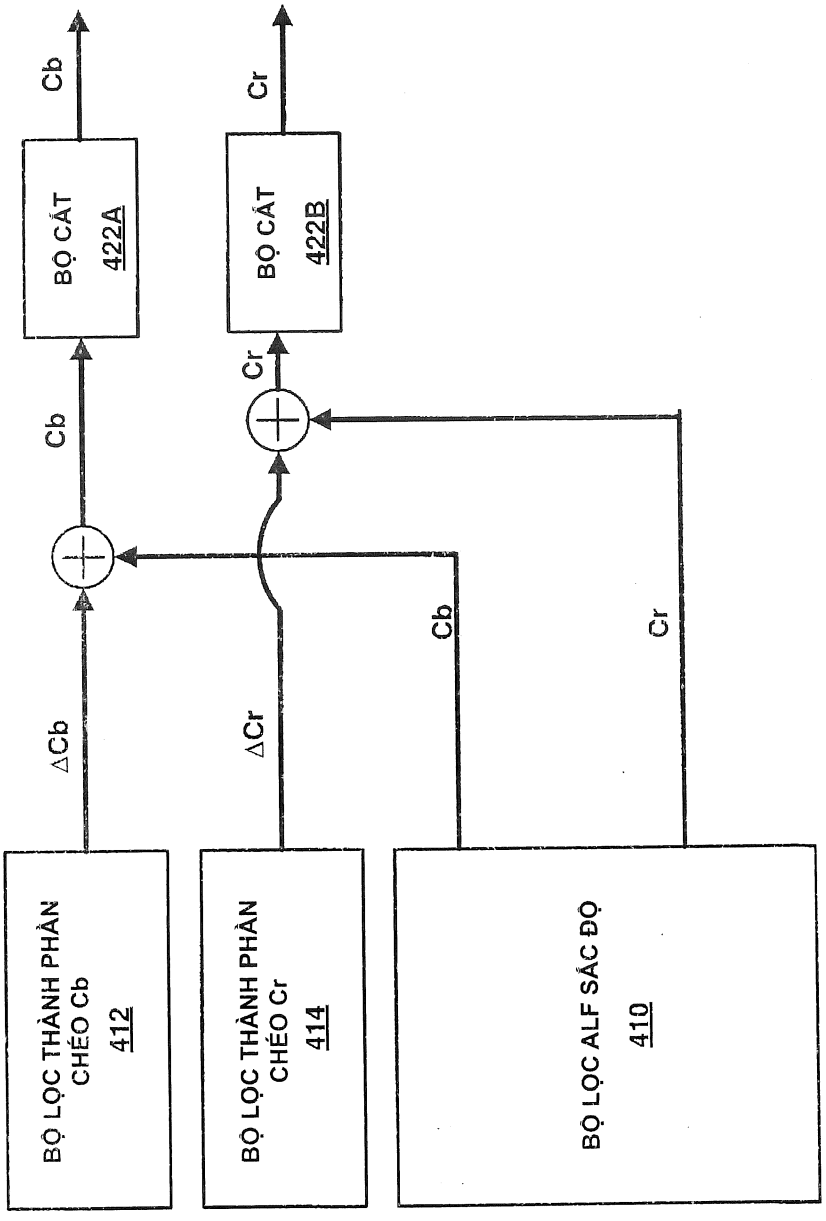
HÌNH 11B


BỘ LỌC TÁCH KHỐI LUMA 416
BỘ LỌC SAO LUMA 402
BỘ LỌC ALF LUMA 408
BỘ LỌC TÁCH KHỐI Cb 418
BỘ LỌC SAO Cb 404
BỘ LỌC THÀNH PHẦN CHÉO Cb 412
BỘ LỌC TÁCH KHỐI Cr 420
BỘ LỌC SAO Cr 406
BỘ LỌC THÀNH PHẦN CHÉO Cr 414
BỘ LỌC ALF SÁC ĐỘ 410
Y
Cb
Cr
ΔCb
ΔCr

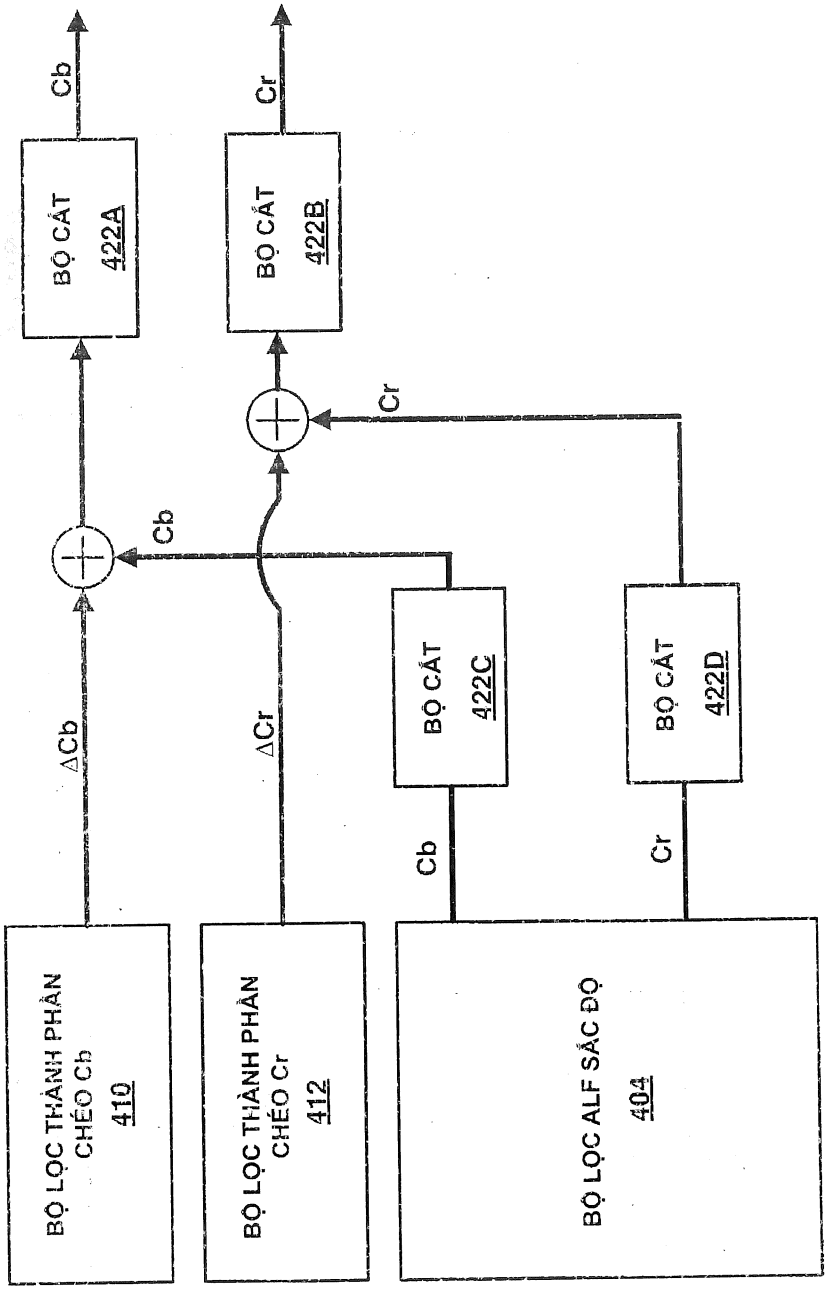
HÌNH 11C


BỘ LỌC SAO LUMA
402
BỘ LỌC TÁCH KHỐI LUMA
416
Y
Y
BỘ LỌC ALF LUMA
408
Y
BỘ LỌC THÀNH PHẦN CHÉO Cb
412
BỘ LỌC THÀNH PHẦN CHÉO Cr
414
ΔCb
ΔCr
Cb
Cr
BỘ LỌC ALF SẮC ĐỘ
410
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cr
406
BỘ LỌC TÁCH KHỐI Cb
418
BỘ LỌC TÁCH KHỐI Cr
420
Cb
Cr

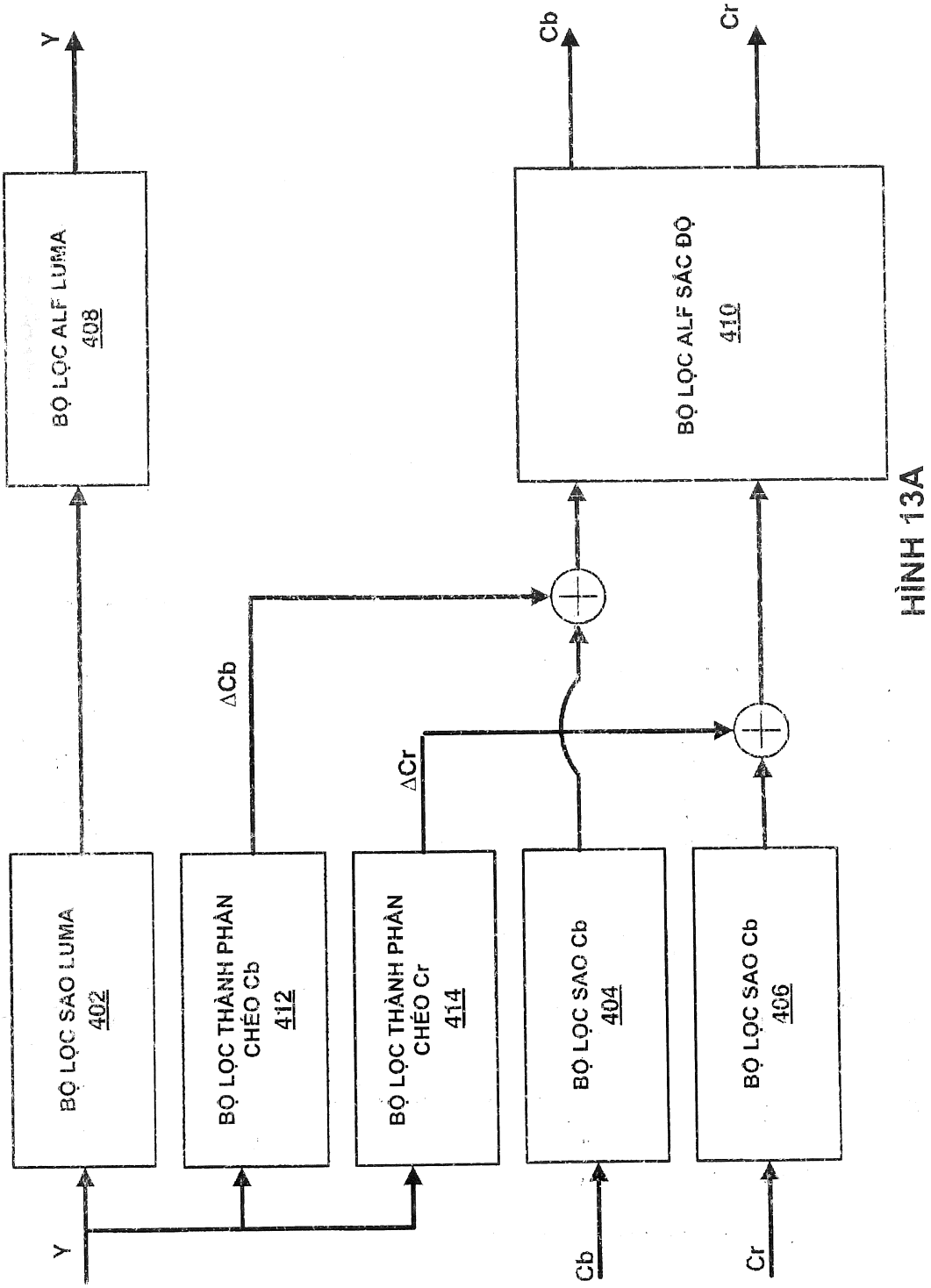
HÌNH 11D


BỘ LỌC THÀNH PHẦN CHÉO Cb
412
BỘ LỌC THÀNH PHẦN CHÉO Cr
414
BỘ LỌC ALF SÁC ĐỘ
410
ΔCb
ΔCr
Cb
Cr
Cb
Cr
BỘ CẮT
422A
BỘ CẮT
422B
Cb
Cr

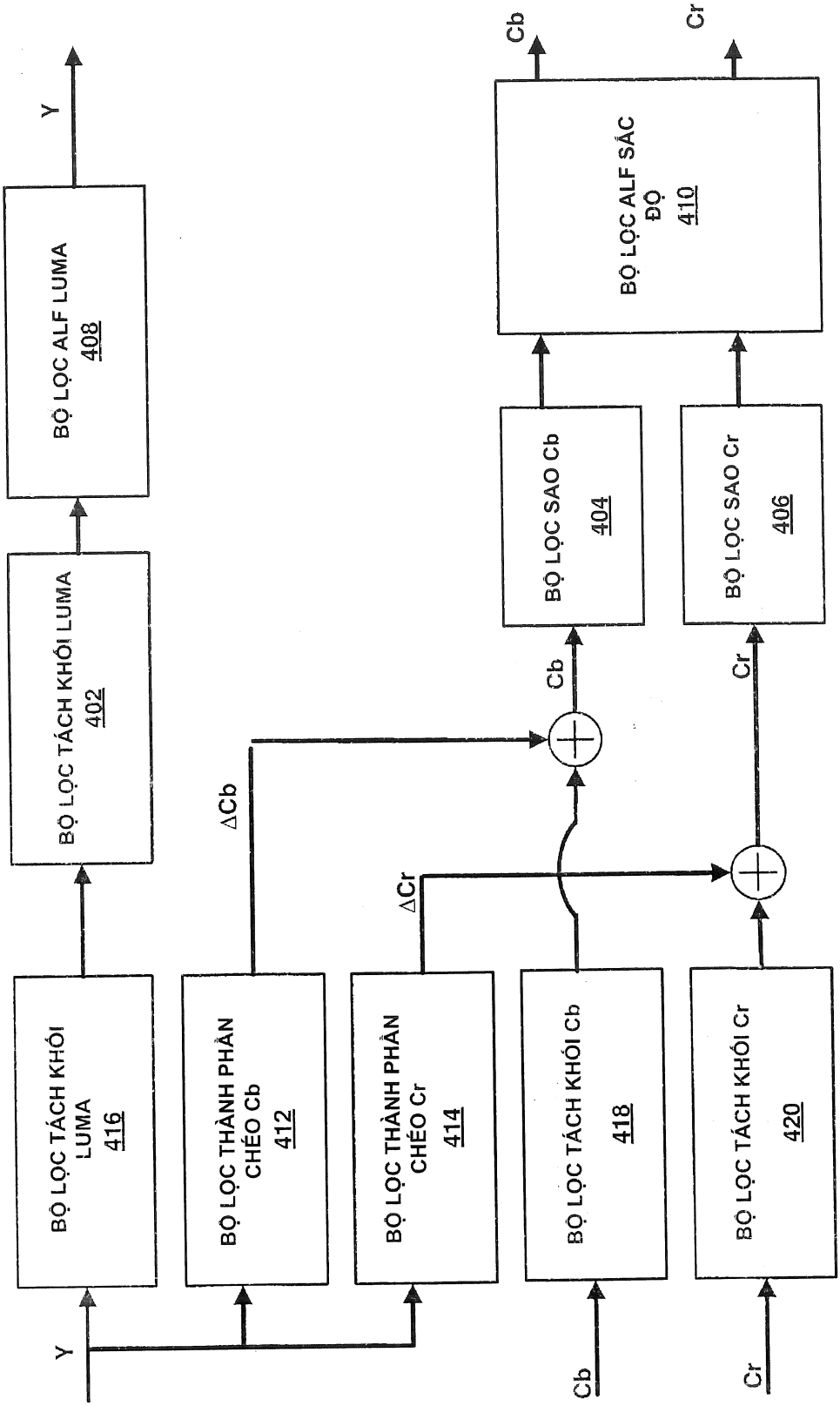
HÌNH 12A


BỘ LỌC THÀNH PHẦN CHÉO Cb
410
BỘ LỌC THÀNH PHẦN CHÉO Cr
412
BỘ LỌC ALF SẮC ĐỘ
404
ΔCb
ΔCr
Cb
Cr
BỘ CẮT
422C
BỘ CẮT
422D
Cb
Cr
BỘ CẮT
422A
BỘ CẮT
422B
Cb
Cr

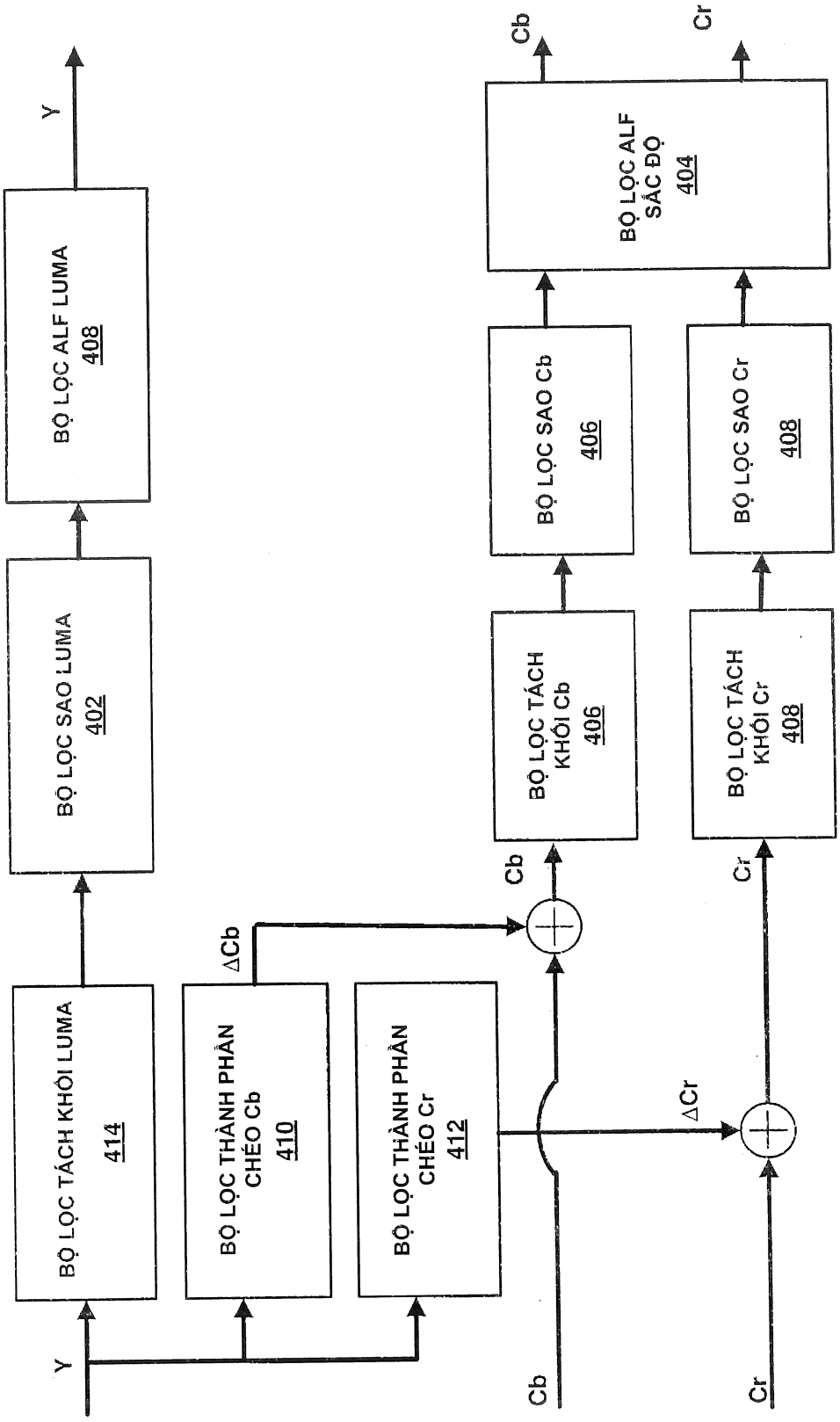
HÌNH 12B


Y
Cb
Cr
BỘ LỌC ALF LUMA
408
BỘ LỌC ALF SẮC ĐỘ
410
ΔCb
ΔCr
BỘ LỌC SAO LUMA
402
BỘ LỌC THÀNH PHẦN CHÉO Cb
412
BỘ LỌC THÀNH PHẦN CHÉO Cr
414
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cb
406
Y
Cb
Cr

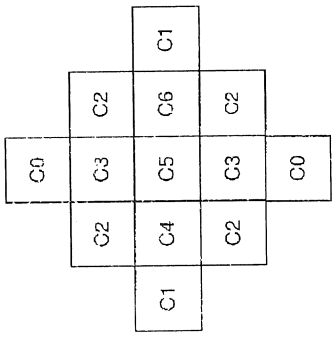
HÌNH 13A


Y
BỘ LỌC ALF LUMA
408
BỘ LỌC TÁCH KHỐI LUMA
402
BỘ LỌC TÁCH KHỐI LUMA
416
BỘ LỌC THÀNH PHẦN CHÉO Cb
412
BỘ LỌC THÀNH PHẦN CHÉO Cr
414
BỘ LỌC TÁCH KHỐI Cb
418
BỘ LỌC TÁCH KHỐI Cr
420
ΔCb
ΔCr
Cb
Cr
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cr
406
BỘ LỌC ALF SÁC ĐỘ
410
Cb
Cr

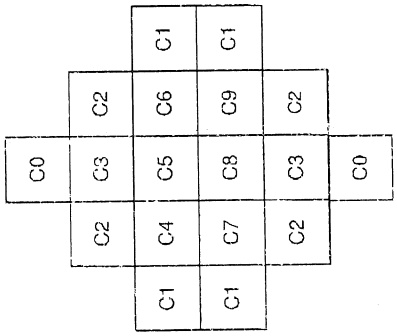
HÌNH 13B


Y
BỘ LỌC TÁCH KHỐI LUMA
414
BỘ LỌC THÀNH PHẦN CHÉO Cb
410
BỘ LỌC THÀNH PHẦN CHÉO Cr
412
ΔCb
ΔCr
Cb
Cr
BỘ LỌC SAO LUMA
402
BỘ LỌC ALF LUMA
408
Y
BỘ LỌC TÁCH KHỐI Cb
406
BỘ LỌC TÁCH KHỐI Cr
408
BỘ LỌC SAO Cb
406
BỘ LỌC SAO Cr
408
BỘ LỌC ALF SẮC ĐỘ
404
Cb
Cr

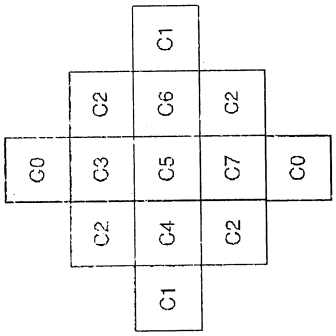
HÌNH 13C



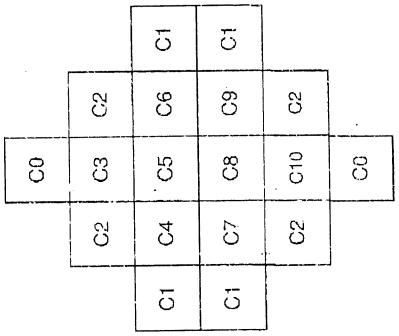
HÌNH 14C



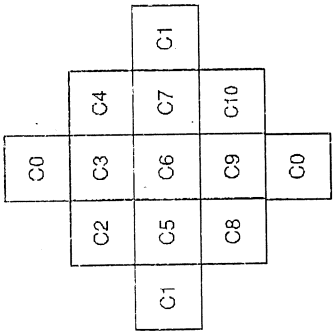
HÌNH 14F



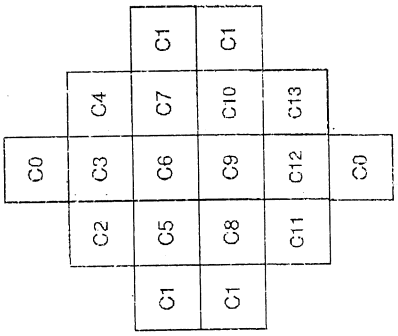
HÌNH 14B



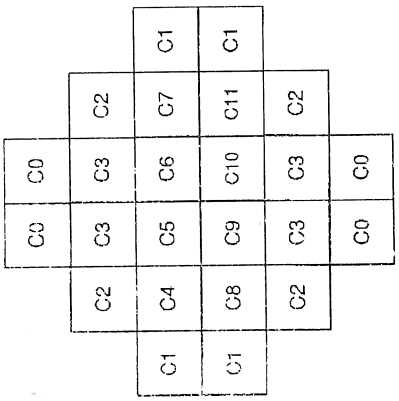
HÌNH 14E



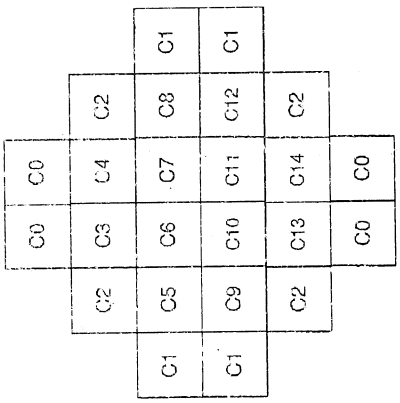
HÌNH 14A



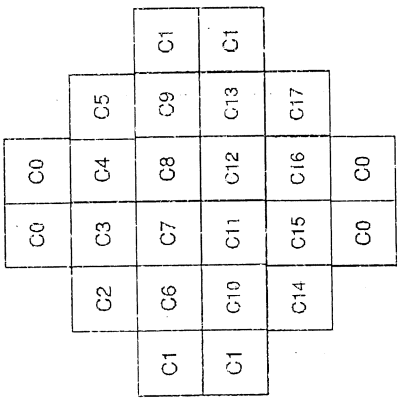
HÌNH 14D



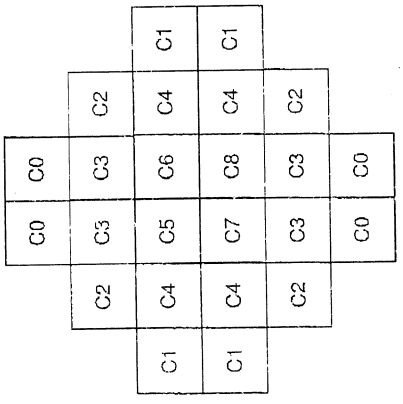
HÌNH 15C



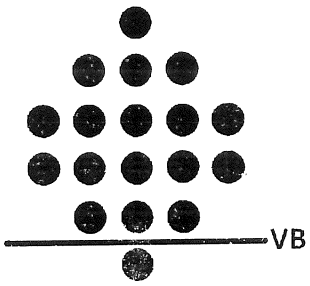
HÌNH 15B

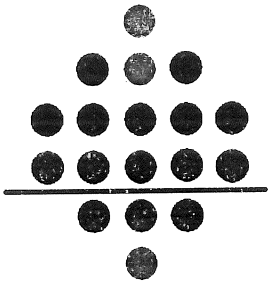


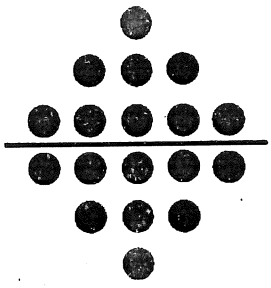
HÌNH 15A



HÌNH 15D

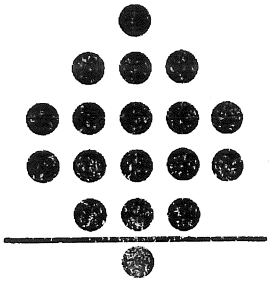


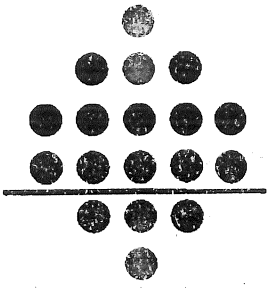


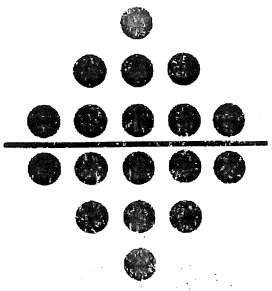


Mẫu được xác định trước
bên trên đường biên dòng

HÌNH 16A

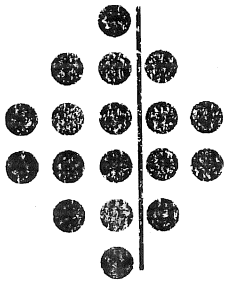


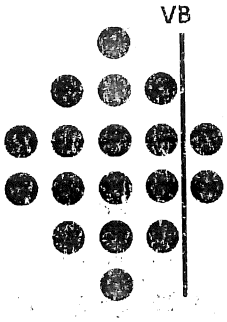




Mẫu được xác định trước
bên dưới đường biên
dòng

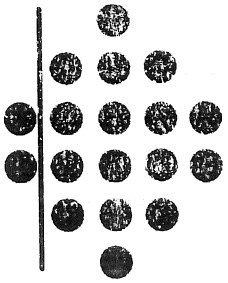
HÌNH 16B

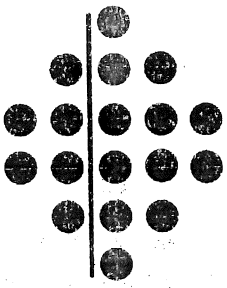




Mẫu được xác định
trước ở bên trái đường
biên dòng

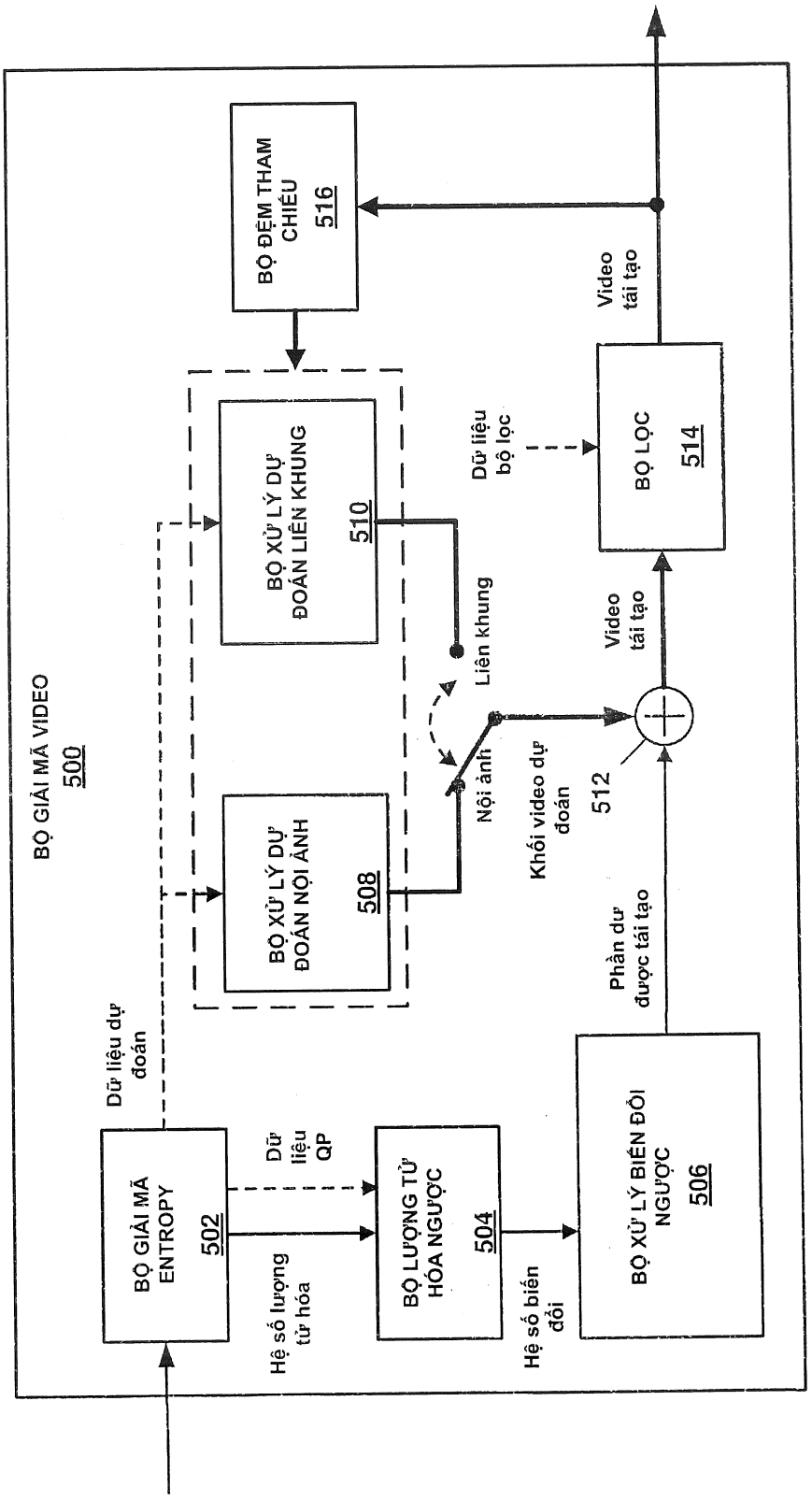
HÌNH 16C



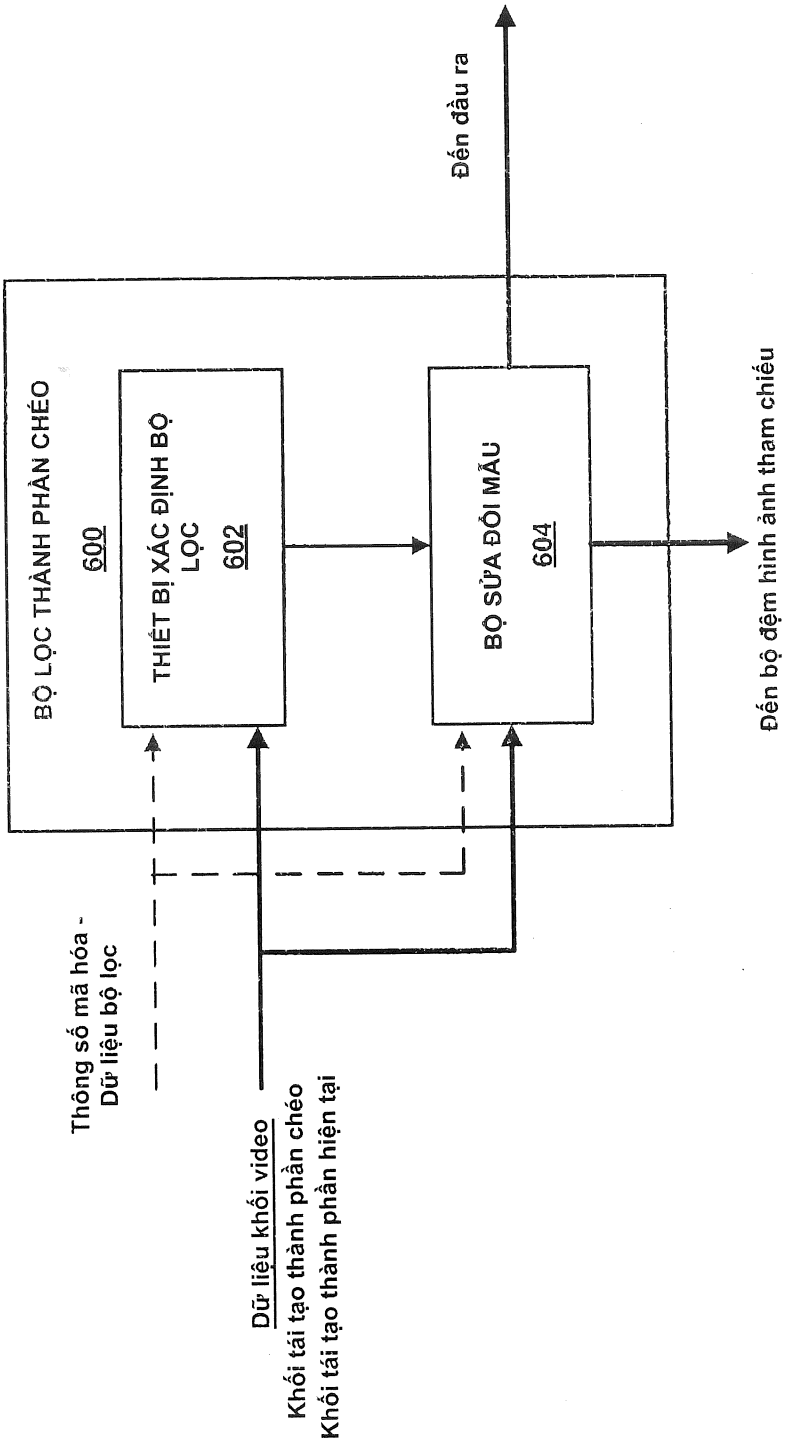


Mẫu được xác định
trước ở bên phải đường
biên dòng

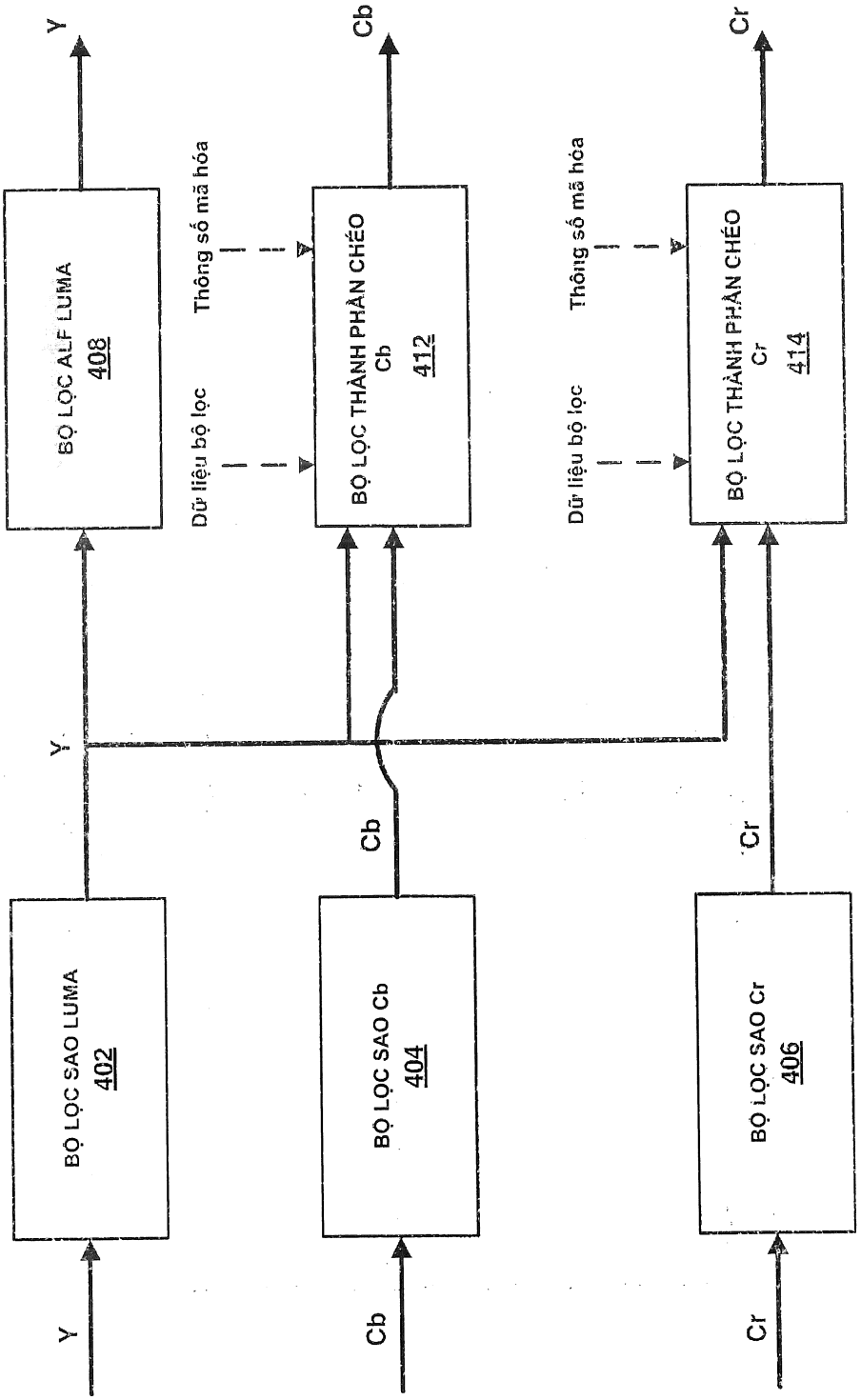
HÌNH 16D


BỘ GIẢI MÃ VIDEO
500
BỘ GIẢI MÃ ENTROPY
502
BỘ LƯỢNG TỬ HÓA NGƯỢC
504
BỘ XỬ LÝ BIẾN ĐỔI NGƯỢC
506
BỘ XỬ LÝ DỰ ĐOÁN NỘI ẢNH
508
BỘ XỬ LÝ DỰ ĐOÁN LIÊN KHUNG
510
512
BỘ LỌC
514
BỘ ĐỆM THAM CHIẾU
516
Dữ liệu dự đoán
Dữ liệu QP
Hệ số lượng tử hóa
Hệ số biến đổi
Phần dư được tái tạo
Khối video dự đoán
Nội ảnh
Liên khung
Video tái tạo
Dữ liệu bộ lọc
Video tái tạo

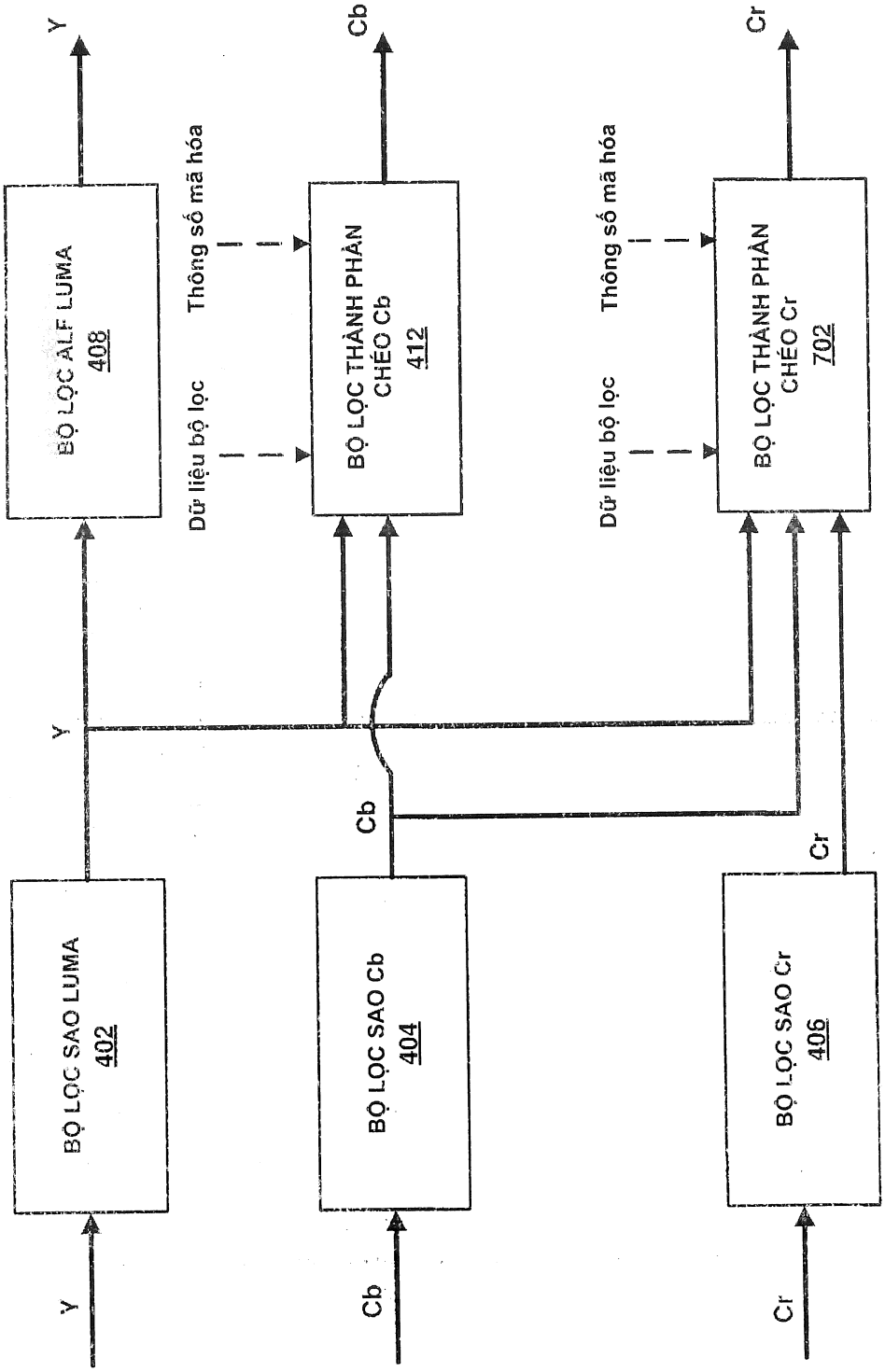
HÌNH 17


Đến đầu ra
BỘ LỌC THÀNH PHẦN CHÉO
600
THIẾT BỊ XÁC ĐỊNH BỘ LỌC
602
BỘ SỬA ĐỔI MẪU
604
Đến bộ đệm hình ảnh tham chiếu
Thông số mã hóa -
Dữ liệu bộ lọc
Dữ liệu khối video
Khối tái tạo thành phần chéo
Khối tái tạo thành phần hiện tại

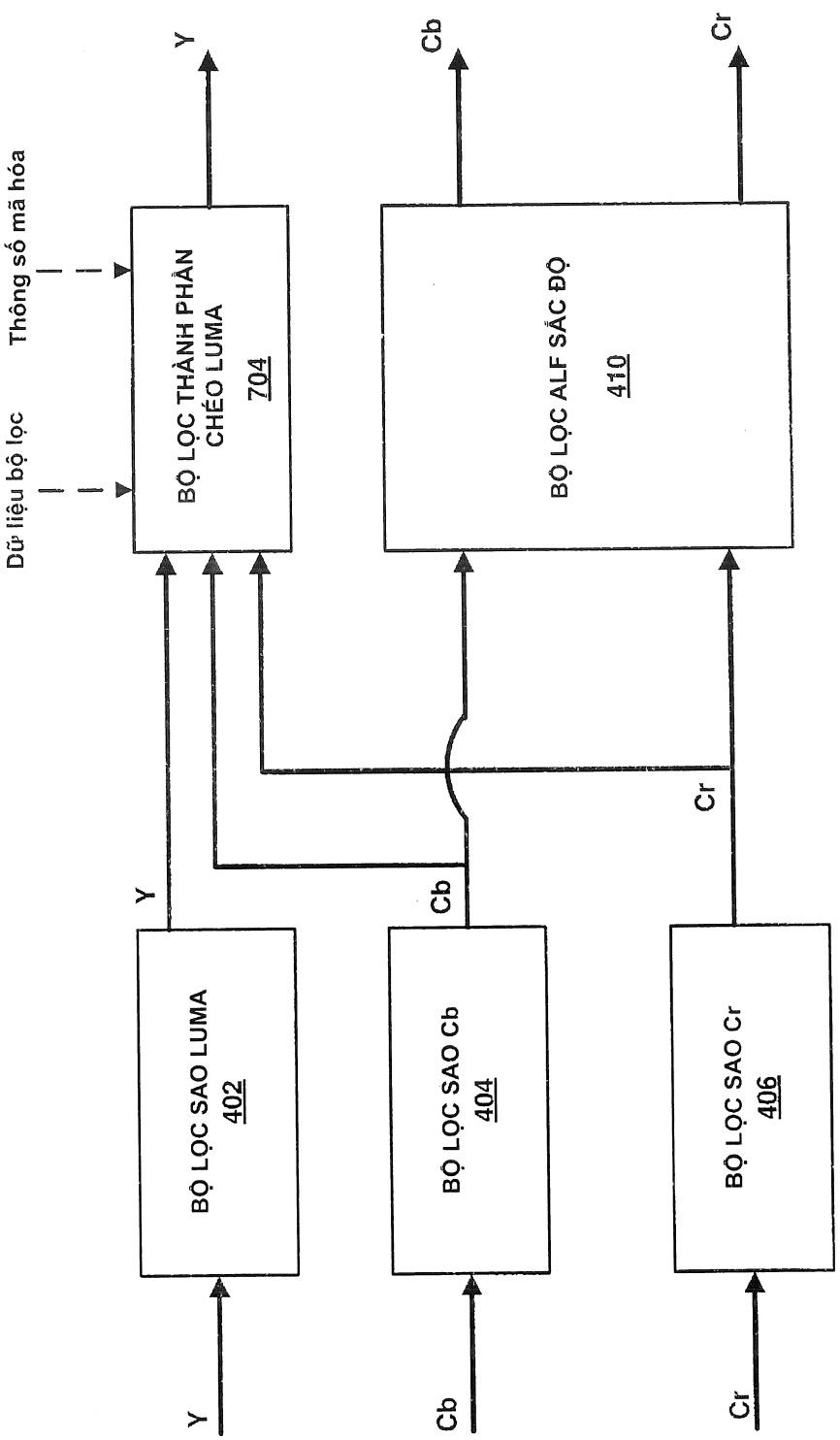
HÌNH 18


Y
Cb
Cr
BỘ LỌC ALF LUMA
408
Thông số mã hóa
Dữ liệu bộ lọc
BỘ LỌC THÀNH PHẦN CHÉO
Cb
412
Thông số mã hóa
Dữ liệu bộ lọc
BỘ LỌC THÀNH PHẦN CHÉO
Cr
414
Y
Cb
Cr
BỘ LỌC SAO LUMA
402
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cr
406
Y
Cb
Cr

HÌNH 19A


Y
Cb
Cr
BỘ LỌC ALF LUMA
408
Thông số mã hóa
Dữ liệu bộ lọc
BỘ LỌC THÀNH PHẦN CHÉO Cb
412
Thông số mã hóa
Dữ liệu bộ lọc
BỘ LỌC THÀNH PHẦN CHÉO Cr
702
Y
Cb
Cr
BỘ LỌC SAO LUMA
402
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cr
406
Y
Cb
Cr

HÌNH 19B

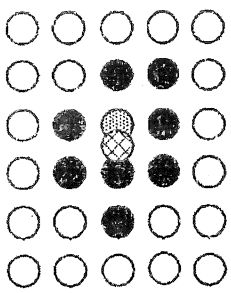

Thông số mã hóa
Dữ liệu bộ lọc
BỘ LỌC THÀNH PHẦN CHÉO LUMA
704
BỘ LỌC ALF SẮC ĐỘ
410
BỘ LỌC SAO LUMA
402
BỘ LỌC SAO Cb
404
BỘ LỌC SAO Cr
406
Y
Cb
Cr

HÌNH 19C

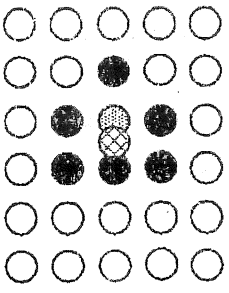
Mẫu giá đỡ,
hệ số được báo hiệu

Mẫu giá đỡ,
hệ số được suy ra

Mẫu cần lọc



HÌNH 20A



HÌNH 20B