(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) **1-0054031**

(51)[2020.01] **H04N 19/426**; H04N 19/52

(13) **B**

(21) 1-2021-03380

(22) 14/11/2019

(86) PCT/JP2019/044730 14/11/2019

(87) WO 2020/100988 22/05/2020

(30) 62/768,772 16/11/2018 US; 62/787,695 02/01/2019 US; 62/792,872 15/01/2019 US; 62/793,080 16/01/2019 US; 62/793,311 16/01/2019 US; 62/815,109 07/03/2019 US

(45) 25/11/2025 452

(43) 25/11/2021 404A

(73) 1. SHARP KABUSHIKI KAISHA (JP)
1, Takumi-cho, Sakai-ku, Sakai City, Osaka 590-8522, Japan
2. FG INNOVATION COMPANY LIMITED (CN)
Flat 2623, 26/F Tuen Mun Central Square, 22 Hoi Wing Road, Tuen Mun, New Territories, Hong Kong, China

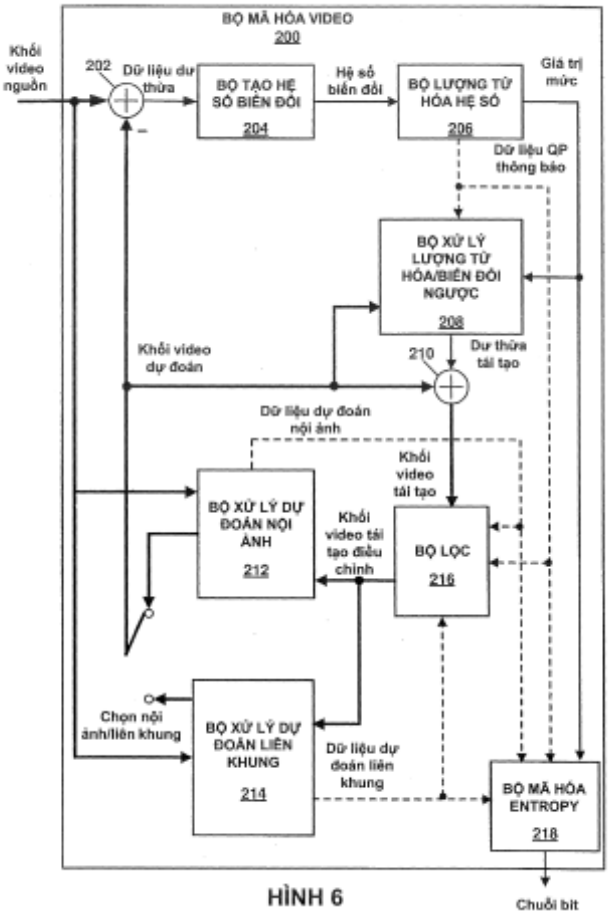
(72) MISRA, Kiran Mukesh (IN); BOSSEN, Frank (NL); SEGALL, Christopher Andrew (US).

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

(54) PHƯƠNG PHÁP VÀ THIẾT BỊ THỰC HIỆN DỰ ĐOÁN VECTƠ CHUYỂN ĐỘNG ĐỂ MÃ HÓA DỮ LIỆU VIDEO

(21) 1-2021-03380

(57) Một phương pháp thực hiện phép dự đoán vectơ chuyển động để mã hóa dữ liệu video được đề xuất. Một vectơ chuyển động chính xác tuyệt đối mv được xác định để tạo một dự đoán cho khối video trong hình ảnh đầu tiên. Một vectơ chuyển động tròn rmv có độ chính xác nhỏ hơn vectơ chuyển động mv chính xác tuyệt đối được lưu trữ. Một ứng viên dự đoán vectơ chuyển động được tạo cho một khối video trong hình ảnh thứ hai từ vectơ chuyển động được lưu trữ.



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 các kỹ thuật lấy dự đoán vectơ chuyển động.

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. 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 dưới dạng tham chiếu 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 nghiên cứu nhu cầu tiềm năng về việc chuẩn hóa công nghệ mã hóa video trong tương lai với khả năng nén vượt trội đáng kể so với 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 dưới dạng tham chiếu qua tài liệu này, 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 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. Là kết quả của nhiều mô tả về mã hóa video, bản thảo về thông

số kỹ thuật mã hóa video được mô tả trong "Mã hóa video đa năng (Bản thảo 1)", Hội nghị lần thứ 10 về ISO/IEC JTC1/SC29/WG11 diễn ra từ ngày 16-20 tháng 4 năm 2018, tại San Diego, California, tài liệu JVET-J1001-v2, được kết hợp dưới dạng tham chiếu qua tài liệu này và được gọi là JVET-J1001. "Mã hóa video đa năng (Bản thảo 2)", Hội nghị lần thứ 11 về ISO/IEC JTC1/SC29/WG11 diễn ra từ ngày 10-18 tháng 7 năm 2018, tại Ljubljana, Slovenia, tài liệu JVET-K1001-v7, được kết hợp dưới dạng tham chiếu qua tài liệu này và được gọi là JVET-K1001, là bản cập nhật cho JVET-J1001. Hơn nữa, "Mã hóa video đa năng (Bản thảo 3)", Hội nghị lần thứ 12 về ISO/IEC JTC1/SC29/WG11 diễn ra từ ngày 3-12 tháng 10 năm 2018, tại Macao, Trung Quốc, tài liệu JVET-L1001-v2, được kết hợp dưới dạng tham chiếu qua tài liệu này và được đề cập là JVET-L1001, là bản cập nhật của JVET-K1001.

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 (ví dụ: nhóm khung hình trong một chuỗi video, một khung hình trong nhóm các khung hình, các vùng trong một khung hình, các khối video trong một vùng và các khối con trong một khối video). Các kỹ thuật mã hóa dự đoán nội ảnh (ví dụ: nội ảnh (theo không gian)) 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 một đơn vị dữ liệu video được mã hóa và một 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 một đơn vị mã hóa tham chiếu (ví dụ: chỉ số chế độ dự đoán nội ảnh, vectơ chuyển động và vectơ khối). 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 đưa vào một chuỗi bit theo chuẩn. Các chuỗi bit theo chuẩn và siêu dữ liệu liên quan có thể được định dạng theo cấu trúc dữ liệu.

Bản chất kỹ thuật của sáng chế

Trong một ví dụ, một phương pháp thực hiện phép dự đoán vectơ chuyển động để mã hóa dữ liệu video bao gồm: xác định vectơ chuyển động (mv) chính xác tuyệt đối để tạo một dự đoán cho khối video trong một ảnh đầu tiên; lưu trữ một vectơ chuyển động tròn (rmv) có độ chính xác nhỏ hơn vectơ chuyển động (mv) chính xác tuyệt đối; và tạo một ứng viên yếu tố dự đoán vectơ chuyển động cho một khối video trong ảnh thứ hai từ vectơ chuyển động được lưu trữ.

Mô tả vắn tắt các hình vẽ

HÌNH 1 là một sơ đồ khái niệm minh họa một 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 một sơ đồ khái niệm minh họa ví dụ về việc mã hóa một khối dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này.

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

HÌNH 3 là một sơ đồ khái niệm minh họa vị trí của các khối video lân cận để đưa vào tập hợp ứng viên cho các yếu tố dự đoán vectơ chuyển động theo một hoặc nhiều kỹ thuật của sáng chế này.

HÌNH 4 là một sơ đồ khái niệm minh họa vị trí các khối video lân cận để đưa vào tập hợp yếu tố dự đoán vectơ chuyển động của ứng viên theo một hoặc nhiều kỹ thuật của sáng chế này.

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

Mô tả chi tiết các phương án thực hiện sáng chế

Nói chung, sáng chế này mô tả nhiều kỹ thuật được dùng để mã hóa dữ liệu video. Cụ thể là, sáng chế này mô tả các kỹ thuật dự đoán vectơ chuyển động trong hoạt động mã hóa video, cũng như các kỹ thuật để thay đổi độ chính xác mà trong đó thông tin về chuyển động dùng để tạo một dự đoán vectơ chuyển động sẽ được lưu trữ. Việc thay đổi độ chính xác của thông tin về chuyển động theo các kỹ thuật được mô tả trong tài liệu này có thể đặc biệt hữu ích, giúp tối ưu hóa hiệu suất mã hóa video và chi phí bộ nhớ của dự đoán vectơ chuyển động. Xin 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, JVET-J1001, JVET-K1001 và JVET-L1001, 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, 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. Do đó, việc tham chiếu đến ITU-T H.264, ITU-T H.265, JVET-J1001, JVET-K1001 và JVET-L1001 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 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.

Trong một ví dụ, thiết bị tái tạo lại dữ liệu video bao gồm một hoặc nhiều bộ xử lý được định cấu hình để xác định vectơ chuyển động chính xác tuyệt đối nhằm tạo một dự đoán cho khối video trong hình ảnh đầu tiên, lưu trữ vectơ chuyển động ở độ chính xác thấp hơn độ chính xác tuyệt đối và tạo ứng viên dự đoán vectơ chuyển động cho một khối video trong hình ảnh thứ hai từ vectơ chuyển động được lưu trữ.

Trong một ví dụ, phương tiện lưu trữ không chuyển tiếp, có thể đọc được bằng máy tính bao gồm các hướng dẫn được lưu trữ trên đó, khi được thực thi, sẽ khiến một hoặc nhiều bộ xử lý của thiết bị xác định vectơ chuyển động chính xác tuyệt đối để tạo một dự đoán cho khối video trong hình ảnh đầu tiên, lưu trữ vectơ chuyển động ở độ chính xác

thấp hơn độ chính xác tuyệt đối và tạo ứng viên yếu tố dự đoán vectơ chuyển động cho một khối video trong hình ảnh thứ hai từ vectơ chuyển động được lưu trữ.

Trong một ví dụ, một thiết bị bao gồm các phương tiện để xác định vectơ chuyển động chính xác tuyệt đối nhằm tạo một dự đoán cho khối video trong hình ảnh đầu tiên, phương tiện để lưu trữ vectơ chuyển động ở độ chính xác thấp hơn độ chính xác tuyệt đối và phương tiện để tạo ứng viên yếu tố dự đoán vectơ chuyển động cho một khối video trong ảnh thứ hai từ vectơ chuyển động được lưu trữ.

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 thường 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. Có thể xác định các vùng theo đơn vị cơ sở (ví dụ: khối video) và các bộ quy tắc xác định vùng (ví dụ: vùng phải là số khối video nguyên được sắp xếp trong một hình chữ nhật). 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 có thể được mã hóa theo dự đoán. 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. Do đó, đối với khối video được định dạng theo định dạng 4:2:0, chiều rộng và chiều cao của mảng mẫu đối với thành phần luma sẽ gấp đôi chiều rộng và chiều cao của mỗi mảng mẫu đối với thành phần sắc độ. Đối với khối video 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 một mảng mẫu đối với mỗi thành phần sắc độ. Ngoài ra, đối với một khối video được định dạng theo định dạng 4:4:4, mảng mẫu của thành phần luma có cùng chiều rộng và chiều cao với một mảng mẫu của mỗi thành phần sắc độ.

Các khối video có thể được sắp xếp trong một hình ảnh và/hoặc một vùng theo một mẫu quét (ví dụ: quét mành). 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, một 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 đặt 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). 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 chroma 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 ảnh hoặc dự đoán liên khung được thực hiện ở cấp CU.

Trong ITU-T H.265, một CU được liên kết với một 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 chroma nhằm mục đích tạo ra các mẫu tham chiếu tương ứng. Nghĩa là, trong ITU-T H.265, CB luma và chroma có thể được chia thành các khối dự đoán (PB) luma và chroma 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 ảnh, trong đó CB có thể tạo thành PB hoặc CB có thể được chia thành bốn PB vuông (tức là các loại PB dự đoán nội ảnh bao gồm MxM hoặc M/2xM/2, trong đó M là

chiều cao và chiều rộng của CB 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 (tức là các loại PB dự đoán liên khung bao gồm MxM, M/2xM/2, M/2xM hoặc MxM/2). 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 (nghĩa là các phân vùng không đối xứng bao gồm M/4xM bên trái, M/4xM bên phải, MxM/4 trên cùng và MxM/4 dưới cùng). 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.

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) trong cùng một đơn vị truy nhập. 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. 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-L1001, có đề xuất rằng các mảnh phải gồm một số nguyên các ô hoàn chỉnh thay vì chỉ gồm một số nguyên các CTU hoàn chỉnh. 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, một mảnh cần phải gồm một số nguyên các ô hoàn chỉnh được gọi là một nhóm ô. Các kỹ thuật được mô tả trong tài liệu này có thể áp dụng cho các 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 nhóm ô. Trong ví dụ minh họa trên HÌNH 1, $Ảnh_4$ được minh họa là gồm hai nhóm ô (tức là Nhóm $ô_1$ và Nhóm $ô_2$). Xin lưu ý rằng, trong một số trường hợp, Nhóm $ô_1$ và Nhóm $ô_2$ có thể được phân loại là mảnh và/hoặc ô.

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-L1001, CTU được phân vùng theo cấu trúc cây đa kiểu hình cộng với cây tứ phân (QTMT). QTMT trong JVET-L1001 tương tự như QTBT trong JEM. Tuy nhiên, trong JVET-L1001, ngoài việc chỉ ra các phép tách nhị phân, cây đa kiểu hình có thể chỉ ra cái gọi là các 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 một lần nữa đến HÌNH 1, HÌNH 1 minh họa ví dụ về một 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 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 một khối video hiện tại. Có thể gọi phần sai khác giữa các giá trị mẫu đưa vào một dự đoán được tạo từ các giá trị mẫu tham chiếu và khối video hiện tại là dữ liệu dư thừa. 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. Xin lưu ý rằng trong ITU-T H.265 và JVET-L1001, một CU được liên kết với một 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.

Có thể thực hiện quá trình lượng tử hóa trên các hệ số biến đổi. 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 cho một 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. 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.

Các HÌNH 2A-2B là những sơ đồ khái niệm minh họa các ví dụ về mã hóa một khối dữ liệu video. Như minh họa trong HÌNH 2A, một khối dữ liệu video hiện tại được mã hóa bằng cách tạo ra phần dư bằngo cách trừ một bộ các giá trị dự đoán khỏi khối dữ liệu video hiện tại, thực hiện phép biến đổi trên phần dư và lượng hóa các hệ số biến đổi để tạo ra các giá trị mức. Như 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 một phép biến đổi ngược và thêm một bộ giá trị dự đoán vào phần dư thu được. Xin lưu ý rằng, trong các ví dụ trên các 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. 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. Hơn nữa, như minh họa trong các HÌNH 2A-2B, quá trình tỷ lệ hóa được thực hiện bằng cách sử dụng một loạt hệ số tỷ lệ.

Như minh họa trong HÌNH 2A, các hệ số biến đổi lượng tử hóa được mã hóa thành một chuỗi bit. 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. 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 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 một giá trị cú pháp thành một chuỗi một hoặc nhiều bit. Các bit này có thể được gọi là "bin" (ngăn). Nhị phân hóa là một quá trình mã hóa không mất dữ liệu và 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 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 đề cập đến 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, chẳng hạn như ITU-T H.265.

Quá trình mã hóa entropy bao gồm mã hóa các giá trị ngăn bằng các thuật toán nén dữ liệu không làm mất dữ liệu. Trong ví dụ về CABAC, đối với một ngăn cụ thể, một mô hình ngữ cảnh có thể được chọn từ một tập hợp mô hình ngữ cảnh có sẵn được liên kết với ngăn. Trong một số ví dụ, một mô hình ngữ cảnh có thể được chọn dựa trên ngăn trước đó và/hoặc các giá trị của các phần tử cú pháp trước đó. Một mô hình ngữ cảnh có thể xác định xác suất của một ngăn có một giá trị cụ thể. Ví dụ: một mô hình ngữ cảnh có thể cho biết xác suất 0,7 của việc mã hóa một ngăn có giá trị là 0. Sau khi chọn một mô hình ngữ cảnh có sẵn, bộ mã hóa entropy CABAC có thể mã hóa theo số học một ngăn dựa trên mô hình ngữ cảnh đã xác định. Có thể cập nhật mô hình ngữ cảnh dựa trên giá trị của một ngăn được mã hóa. Có thể cập nhật mô hình ngữ cảnh dựa trên một biến liên quan được lưu trữ cùng với ngữ cảnh, ví dụ: kích thước khung thích ứng, số lượng ngăn được mã hóa bằng ngữ cảnh. Xin lưu ý rằng có thể triển khai một bộ mã hóa entropy CABAC sao cho một số phần tử cú pháp có thể được mã hóa entropy bằng mã hóa số học mà không cần sử dụng mô hình ngữ cảnh được chỉ định rõ ràng và kỹ thuật mã hóa như vậy có thể được gọi là mã hóa bỏ qua.

Như được mô tả ở trên, dữ liệu dự đoán nội ảnh hoặc dữ liệu dự đoán liên khung cho biết cách tạo một dự đoán cho một khối video hiện tại. Đối với mã hóa dự đoán nội ảnh, chế độ dự đoán nội ảnh có thể chỉ định vị trí của các mẫu tham chiếu trong một hình ảnh được dùng để tạo một dự đoán. Trong ITU-T H.265, có thể xác định các chế độ dự đoán nội ảnh bao gồm một chế độ dự đoán phẳng (tức là ghép nối bề mặt) (predMode: 0), một chế độ dự đoán DC (tức là trung bình tổng thể phẳng) (predMode: 1) và 33 chế độ dự đoán góc (predMode: 2-34). Trong JVET-L1001, có thể xác định các chế độ dự đoán nội ảnh cho luma bao gồm một chế độ dự đoán phẳng (predMode: 0), một chế độ dự đoán DC (predMode: 1) và 65 chế độ dự đoán góc (predMode: 2-66). 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. Ngoài ra, có thể có nhiều cách trong đó, có thể lấy các chế độ dự đoán nội ảnh cho các thành phần sắc độ dựa trên chế độ dự đoán nội ảnh cho thành phần luma. 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, một hoặc nhiều hình ảnh đã được giải mã trước đó, tức là 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 một dự đoán cho một 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 ảnh, và có thể sử dụng dự đoán đơn hướng và các chế độ dự đoán nội ảnh 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 ảnh. Như mô tả ở trên, hình ảnh tham chiếu được xác định thông qua các chỉ số tham chiếu. Trong ITU-T H.265, đối với mảnh P, có một danh sách ảnh tham chiếu duy nhất, RefPicList0 và đối với mảnh B, 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 một dự đoán. Ngoài ra, xin lưu ý rằng trong ITU-T H.265, trong quá trình giải mã, khi bắt đầu giải mã một hình ảnh, (các) danh sách ảnh tham chiếu được tạo từ ảnh đã giải mã trước đó sẽ được lưu trữ trong bộ đệm 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 dựa trên một vectơ chuyển động khác. 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). ITU-T H.265 hỗ trợ hai chế độ để dự đoán vectơ chuyển động: chế độ hợp nhất và Dự đoán vectơ chuyển động nâng cao (AMVP). Trong ITU-T H.265, đối với cả chế độ hợp nhất và AMVP cho PB hiện tại, một tập hợp khối ứng viên sẽ được lấy ra. 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. Một khối ứng viên bao gồm một khối video có thông tin chuyển động được liên kết mà từ đó thông tin chuyển động được dùng để tạo một dự đoán cho một khối video hiện tại có thể được lấy ra. Đối với chế độ hợp nhất trong ITU-T H.265, tất cả thông tin

chuyển động (tức là giá trị dịch chuyển vectơ chuyển động, chỉ số hình ảnh tham chiếu và danh sách hình ảnh tham chiếu) được liên kết với một ứng viên đã chọn sẽ được kế thừa dưới dạng thông tin chuyển động cho PB hiện tại. Có nghĩa là, tại bộ mã hóa video, một khối ứng viên được chọn từ tập hợp ứng viên đã lấy ra và một giá trị chỉ báo được đưa vào trong chuỗi bit cho biết ứng viên được chọn và do đó, cho biết thông tin chuyển động cho PB hiện tại. Đối với AMVP trong ITU-T H.265, thông tin vectơ chuyển động cho ứng viên được chọn được dùng làm bộ dự đoán vectơ chuyển động (MVP) cho vectơ chuyển động của PB hiện tại. Nghĩa là, tại bộ mã hóa video, một khối ứng viên được chọn từ tập hợp ứng viên đã lấy ra và một giá trị chỉ báo cho biết ứng viên được chọn và một giá trị delta (tức là một delta vectơ chuyển động (MVD)) cho biết sự khác biệt giữa bộ dự đoán vectơ chuyển động và vectơ chuyển động cho PB hiện tại được đưa vào chuỗi bit. Hơn nữa, đối với AMVP trong ITU-T H.265, các phần tử cú pháp xác định hình ảnh tham chiếu sẽ được đưa vào chuỗi bit.

Trong ITU-T H.265, một tập hợp khối ứng viên có thể được lấy từ các khối lân cận theo không gian và các khối theo thời gian. Hơn nữa, có thể sử dụng thông tin chuyển động được tạo (hoặc mặc định) cho quá trình dự đoán vectơ chuyển động. Trong ITU-T H.265, nơi thông tin chuyển động được dùng để dự đoán vectơ chuyển động của PB hiện tại sẽ bao gồm thông tin chuyển động được liên kết với các khối lân cận theo không gian, thông tin chuyển động được liên kết với khối theo thời gian hoặc thông tin chuyển động được tạo hay không phụ thuộc vào số lượng ứng viên được đưa vào một tập hợp, liệu dự đoán vectơ chuyển động theo thời gian có được bật hay không, tính khả dụng của các khối và/hoặc thông tin chuyển động liên kết với các khối có dư thừa hay không.

Đối với chế độ hợp nhất trong ITU-T H.265, có thể đặt và báo hiệu số lượng ứng viên tối đa có thể được đưa vào một tập hợp khối ứng viên bằng bộ mã hóa video và có thể lên đến năm. Hơn nữa, một bộ mã hóa video có thể vô hiệu hóa việc sử dụng các ứng viên vectơ chuyển động theo thời gian (ví dụ: để giảm lượng tài nguyên bộ nhớ cần thiết để lưu trữ thông tin chuyển động tại một bộ giải mã video) và báo hiệu việc sử dụng các ứng viên vectơ chuyển động theo thời gian được bật hay tắt cho một ảnh. HÌNH 3 minh họa vị trí của các khối lân cận theo không gian và khối theo thời gian có thể được đưa vào một tập hợp khối ứng viên cho chế độ hợp nhất trong ITU-T H.265. Việc lấy tập hợp ứng viên cho chế độ hợp nhất trong ITU-T H.265 bao gồm việc xác định tính khả dụng của A1, B1, B0, A0 và B2. Xin lưu ý rằng một khối được coi là không khả dụng, nếu khối đó được dự đoán liên khung (tức là không có thông tin chuyển động tương ứng) hoặc không được đưa vào trong mảnh (hoặc ô) hiện tại. Sau

khi xác định tính khả dụng của A1, B1, B0, A0 và B2, một tập hợp phép so sánh (được minh họa dưới dạng mũi tên nét đứt trong HÌNH 3) sẽ được thực hiện để loại bỏ các mục thừa khỏi tập hợp ứng viên. Ví dụ: B2 được so sánh với B1 và nếu B1 có thông tin chuyển động liên quan tương đương với B2, thì nó sẽ bị loại khỏi tập hợp ứng viên. Có thể coi việc loại bỏ các mục khỏi một tập hợp ứng viên là một quá trình cắt bớt. Xin lưu ý rằng trong HÌNH 3, để giảm tính phức tạp, việc so sánh toàn bộ các ứng viên sẽ không được thực hiện (ví dụ: A0 không được so sánh với B0) và do đó, có thể có các mục thừa sẽ được đưa vào tập hợp ứng viên.

Tham chiếu một lần nữa đến HÌNH 3, khối nét đứt có nhãn Temp đề cập đến ứng viên tạm thời có thể được đưa vào tập hợp ứng viên. Trong ITU-T H.265 cho chế độ hợp nhất, đối với ứng viên tạm thời, một PU được sắp xếp theo không gian có trong một hình ảnh tham chiếu được xác định và ứng viên tạm thời bao gồm một khối có vị trí ngay bên ngoài phía dưới cùng bên phải của PU được sắp xếp, nếu có, hoặc khối ở vị trí trung tâm của PU sắp xếp. Như đã mô tả ở trên, số lượng ứng viên tối đa có thể được đưa vào một tập hợp khối ứng viên sẽ được thiết đặt. Nếu số lượng ứng viên tối đa được đặt thành N, thì N-1 ứng viên theo không gian và ứng viên theo thời gian được đưa vào tập hợp, trong trường hợp số lượng ứng viên theo không gian có sẵn (sau khi cắt bớt) và ứng viên theo thời gian lớn hơn hoặc bằng N. Trong trường hợp số lượng ứng viên theo không gian có sẵn (sau khi cắt bớt) và ứng viên theo thời gian nhỏ hơn N, thì thông tin chuyển động đã tạo được đưa vào tập hợp để lấp đầy tập hợp.

Đối với AMVP trong ITU-T H.265, tham chiếu đến HÌNH 4, việc lấy tập hợp ứng viên bao gồm thêm một trong số A0 hoặc A1 (tức là một ứng viên bên trái) và một trong B0, B1 hoặc B2 (một ứng viên ở trên) vào tập hợp dựa trên tính khả dụng của chúng. Nghĩa là, ứng viên bên trái có sẵn đầu tiên và ứng viên đầu tiên có sẵn ở trên được thêm vào tập hợp. Khi ứng viên bên trái và ứng viên ở trên có các thành phần vectơ chuyển động dư thừa, thì một ứng viên dư thừa sẽ bị loại khỏi tập hợp. Nếu số lượng ứng viên được đưa vào tập hợp ít hơn hai và dự đoán vectơ chuyển động theo thời gian được bật, thì ứng viên theo thời gian (Temp) sẽ được đưa vào tập hợp. Trong trường hợp số lượng ứng viên theo không gian có sẵn (sau khi cắt bớt) và ứng viên theo thời gian được đưa vào tập hợp ít hơn hai, thì một vectơ chuyển động có giá trị 0 được đưa vào tập hợp để lấp đầy tập hợp.

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
xy	Phép lũy thừa. 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 các phương trình toán học mà không có ý định rút gọn hoặc làm tròn.
$\frac{x}{y}$	Được dùng để biểu thị phép chia trong các phương trình toán học mà không có ý định rút gọn hoặc làm tròn.
x % y	Mô-đun. 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 hàm toán học sau đây:

Log2(x) logarit cơ số 2 của x;

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

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

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

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

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

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x > 0 \\ 0 & ; \quad x = 0 \\ -1 & ; \quad x < 0 \end{cases}$$

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

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 một đối số nhị phân chứa ít bit hơn đối số khác, thì 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 một đối số nhị phân chứa ít bit hơn đối số khác, thì 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 một đối số nhị phân chứa ít bit hơn đối số khác, thì đối số ngắn hơn được mở rộng bằng cách thêm nhiều bit có giá trị cao hơn bằng 0.

x >> y Dịch chuyển số học sang phải của 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 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.

JVET-L1001 bao gồm chế độ hợp nhất dựa trên chế độ hợp nhất được xác định trong ITU-T H.265 và chế độ AMVP dựa trên AMVP được xác định trong ITU-T H.256. Xin lưu ý rằng JVET-L1001 cũng bao gồm các kỹ thuật dự đoán vectơ chuyển động affine. Như đã mô tả ở trên, một vectơ chuyển động có thể bao gồm thành phần dịch chuyển ngang của vectơ chuyển động, thành phần dịch chuyển thẳng đứng của vectơ chuyển động và độ phân giải cho vectơ chuyển động. JVET-L1001 cho phép vị trí vectơ chuyển động luma được lấy ở độ chính xác phân số – mẫu 1/16 và vectơ chuyển động sắc độ được lấy ở độ chính xác phân số – mẫu 1/32. Cụ thể, JVET-L1001 cho phép dự đoán vectơ chuyển động luma, vectơ chuyển động luma mvLX được lấy như sau:

$uLX[0] = (mvpLX[0] + mvdLX[0] + 2^{18}) \% 2^{18}$

$mvLX[0][0][0] = (uLX[0] >= 2^{17}) ? (uLX[0] – 2^{18}) : uLX[0]$

$uLX[1] = (mvpLX[1] + mvdLX[1] + 2^{18}) \% 2^{18}$

$mvLX[0][0][1] = (uLX[1] >= 2^{17}) ? (uLX[1] – 2^{18}) : uLX[1]$

trong đó,

X được thay bằng 0 hoặc 1, theo hướng dự đoán chuyển động tương ứng;

mvpLX là một bộ dự đoán vectơ chuyển động; và

mvdLX là một giá trị delta vectơ chuyển động.

Xin lưu ý rằng dựa trên các phương trình trên, các giá trị thu được của mvLX[0] (cho biết hướng dịch chuyển ngang (trái hoặc phải) và độ lớn) và mvLX[1] (cho biết hướng dịch chuyển dọc (lên hoặc xuống) và độ lớn) như được chỉ định ở trên sẽ luôn nằm trong khoảng từ -2^{17} đến $2^{17} - 1$.

JVET-L1001 cho phép vectơ chuyển động affine, mảng vectơ chuyển động khối con luma được lấy ở độ chính xác phân số – mẫu 1/16 và vectơ mảng vectơ khối chuyển động khối con màu được lấy ở độ chính xác phân số – mẫu 1/32. Cụ thể, JVET-L1001 cho phép bộ dự đoán vectơ chuyển động điểm điều khiển liên kết luma, vectơ chuyển động luma cpMvLX[cpIdx] với cpIdx nằm trong khoảng từ 0 đến NumCpMv – 1, được lấy như sau:

$uLX[cpIdx][0] = (mvpCpLX[cpIdx][0] + mvdCpLX[cpIdx][0] + 2^{18}) \% 2^{18}$

$cpMvLX[cpIdx][0] = (uLX[cpIdx][0] >= 2^{17}) ? (uLX[cpIdx][0] – 2^{18}) :$
$uLX[cpIdx][0]$

$uLX[cpIdx][1] = (mvpCpLX[cpIdx][1] + mvdCpLX[cpIdx][1] + 2^{18}) \% 2^{18}$

$cpMvLX[cpIdx][1] = (uLX[cpIdx][1] >= 2^{17}) ? (uLX[cpIdx][1] - 2^{18}) : uLX[cpIdx][1]$

trong đó,

X được thay bằng 0 hoặc 1, theo hướng dự đoán chuyển động tương ứng;

mvpCpLX là một bộ dự đoán vectơ chuyển động điểm điều khiển; và

mvdCpLX là một giá trị delta vectơ chuyển động điểm điều khiển.

Xin lưu ý rằng dựa trên các phương trình ở trên, các giá trị thu được của cpmvLX[0] và cpmvLX[1] như được chỉ định ở trên sẽ luôn nằm trong khoảng từ -2^{17} đến $2^{17} - 1$.

Như được mô tả ở trên, một ứng viên vectơ chuyển động có thể bao gồm một ứng viên theo thời gian, trong đó một ứng viên theo thời gian có thể bao gồm thông tin chuyển động được liên kết với một khối sắp xếp được đưa vào hình ảnh tham chiếu. Cụ thể, JVET-L1001 cung cấp các yếu tố sau liên quan đến việc lấy ra các vectơ chuyển động sắp xếp:

Giá trị đầu vào cho quy trình này là:

– một biến currCb xác định khối mã hóa hiện tại,

– một biến colCb chỉ định khối mã hóa sắp xếp bên trong hình ảnh sắp xếp được chỉ định bởi ColPic,

– một vị trí luma (xColCb, yColCb) chỉ định mẫu trên cùng bên trái của khối mã hóa luma sắp xếp được chỉ định bởi colCb liên quan đến mẫu luma trên cùng bên trái của hình ảnh sắp xếp do ColPic chỉ định,

– một chỉ mục tham chiếu refIdxLX, với X là 0 hoặc 1,

– cờ cho biết ứng viên hợp nhất theo thời gian của khối con sbFlag.

Giá trị đầu ra của quy trình này là:

– dự đoán vectơ chuyển động mvLXCol với độ chính xác 1/16 phân số – mẫu,

– cờ báo hiệu tính khả dụng availableFlagLXCol.

Biến currPic chỉ định hình ảnh hiện tại.

Các mảng predFlagL0Col[x][y], mvL0Col[x][y] và refIdxL0Col[x][y] được đặt lần lượt bằng PredFlagL0[x][y], MvL0[x][y] và RefIdxL0[x][y], của ảnh sắp xếp được

ColPic chỉ định và các mảng predFlagL1Col[x][y], mvL1Col[x][y] và refIdxL1Col[x][y] được đặt lần lượt bằng PredFlagL1[x][y], MvL1[x][y] và RefIdxL1[x][y], của ảnh sắp xếp được ColPic chỉ định.

Các biến mvLXCol và availableFlagLXCol được lấy như sau:

– Nếu colCb được mã hóa trong chế độ dự đoán nội ảnh, thì cả hai thành phần của mvLXCol được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.

– Nếu không, vectơ chuyển động mvCol, chỉ số tham chiếu refIdxCol và mã định danh danh sách tham chiếu listCol được lấy như sau:

– Nếu sbFlag bằng 0, thì availableFlagLXCol được đặt thành 1 và áp dụng như sau:

– Nếu predFlagL0Col[xColCb][yColCb] bằng 0, thì mvCol, refIdxCol và listCol được đặt lần lượt bằng mvL1Col[xColCb][yColCb], refIdxL1Col[xColCb][yColCb] và L1.

– Nếu không, nếu predFlagL0Col[xColCb][yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 0, thì mvCol, refIdxCol và listCol được đặt lần lượt bằng mvL0Col[xColCb][yColCb], refIdxL0Col[xColCb][yColCb] và L0.

– Nếu không (predFlagL0Col[xColCb][yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 1), các phép gán sau sẽ được thực hiện:

– Nếu NoBackwardPredFlag bằng 1, thì mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb], refIdxLXCol[xColCb][yColCb] và LX.

– Nếu không, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLNCol[xColCb][yColCb], refIdxLNCol[xColCb][yColCb] và LN, trong đó N là giá trị của collocated_from_10_flag.

– Nếu không (sbFlag bằng 1), sẽ áp dụng những điểm sau:

– Nếu PredFlagLXCol[xColCb][yColCb] bằng 1, thì mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb], refIdxLXCol[xColCb][yColCb], và LX, availableFlagLXCol được đặt thành 1.

– Nếu không (PredFlagLXCol[xColCb][yColCb] bằng 0), thì sẽ áp dụng các điểm sau:

– Nếu DiffPicOrderCnt(aPic, currPic) nhỏ hơn hoặc bằng 0 đối với mọi ảnh aPic trong mọi danh sách ảnh tham chiếu của mảnh hiện tại và PredFlagLYCol[xColCb][yColCb] bằng 1, thì mvCol, refIdxCol và listCol được đặt thành mvLYCol[xColCb][yColCb], refIdxLYCol[xColCb][yColCb] và LY tương ứng, với Y bằng !X, trong đó X là giá trị X, quy trình này được gọi cho availableFlagLXCol được đặt thành 1.

– Cả hai thành phần của mvLXCol đều được đặt thành 0 và availableFlagLXCol được đặt bằng 0.

– Khi availableFlagLXCol bằng TRUE, thì mvLXCol và availableFlagLXCol được lấy như sau:

– Nếu LongTermRefPic(currPic, currCb, refIdxLX, LX) không bằng LongTermRefPic(ColPic, colCb, refIdxCol, listCol), thì cả hai thành phần của mvLXCol sẽ được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.

– Nếu không, biến availableFlagLXCol được đặt bằng 1, refPicListCol[refIdxCol] được đặt là hình ảnh có chỉ mục tham chiếu refIdxCol trong danh sách hình ảnh tham chiếu listCol của mảnh chứa khối mã hóa colCb trong hình ảnh sắp xếp được chỉ định bởi ColPic và áp dụng như sau:

colPocDiff = DiffPicOrderCnt(ColPic, refPicListCol[refIdxCol])

currPocDiff= DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])

– Nếu RefPicListX[refIdxLX] là hình ảnh tham chiếu dài hạn hoặc colPocDiff bằng với currPocDiff, thì mvLXCol được lấy như sau:

mvLXCol = mvCol

– Nếu không, mvLXCol được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvCol như sau:

tx = (16384 + (Abs(td) >> 1)) / td

distScaleFactor = Clip3(–4096, 4095, (tb * tx + 32) >> 6)

mvLXCol = Clip3(–32768, 32767, Sign(distScaleFactor * mvCol) *
((Abs(distScaleFactor * mvCol) + 127) >> 8))

trong đó td và tb được lấy như sau:

td = Clip3(–128, 127, colPocDiff)

tb = Clip3(–128, 127, currPocDiff)

trong đó,

Hàm LongTermRefPic(aPic, aPb, refIdx, LX), với X là 0 hoặc 1, có thể được xác định như sau:

– Nếu ảnh có chỉ mục refIdx từ danh sách ảnh tham chiếu LX của mảnh chứa khối dự đoán aPb trong ảnh thì aPic được đánh dấu là "được dùng để tham chiếu dài hạn" tại thời điểm aPic là ảnh hiện tại, LongTermRefPic(aPic, aPb, refIdx, LX) bằng 1.

– Nếu không, LongTermRefPic(aPic, aPb, refIdx, LX) bằng 0.

Và

Hàm DiffPicOrderCnt(picA, picB) được chỉ định như sau:

DiffPicOrderCnt(picA, picB) = PicOrderCnt(picA) - PicOrderCnt(picB)

Do đó, để hỗ trợ dự đoán vectơ chuyển động theo thời gian, thông tin chuyển động từ một ảnh được mã hóa trước đó sẽ được lưu trữ. Thông thường, thông tin như vậy được lưu trữ trong một bộ đệm chuyển động theo thời gian. Nghĩa là, một bộ đệm chuyển động theo thời gian có thể bao gồm các vectơ chuyển động được xác định trong các ảnh được mã hóa trước đó có thể được dùng để dự đoán vectơ chuyển động cho khối hiện tại trong một ảnh hiện tại. Ví dụ: như đã cung cấp ở trên, để lấy mvLXCol, cần có các giá trị MvLX[x][y] 18 bit từ một hình ảnh sắp xếp. Xin lưu ý rằng các vectơ chuyển động thường được lưu trữ vào một bộ đệm chuyển động theo thời gian bằng cách sử dụng một tập hợp hữu hạn các điểm chính xác, tức là, ở cùng độ chính xác với vectơ chuyển động đã lấy. Hơn nữa, có thể lưu trữ một vectơ chuyển động tương ứng với mỗi danh sách ảnh tham chiếu cho một khối mẫu. Có thể lưu trữ một chỉ số tham chiếu tương ứng với hình ảnh được tham chiếu bởi vectơ chuyển động. Có thể lưu trữ một chế độ dự đoán trong bộ đệm theo thời gian (ví dụ: liên khung hoặc không liên khung) cho mỗi khối của một hình ảnh trước đó. Hơn nữa, có thể lưu trữ một chế độ phụ dự đoán liên khung cho khối mẫu (ví dụ: cờ cảnh báo chế độ bù sáng luma).

Do đó, việc cho phép dự đoán vectơ chuyển động theo thời gian thông qua một bộ đệm chuyển động theo thời gian thường có thể tốn nhiều chi phí về bộ nhớ. Theo các kỹ thuật được mô tả trong tài liệu này, độ phân giải của thông tin chuyển động có thể thay đổi để tối ưu hóa các cải tiến hiệu quả mã hóa do dự đoán vectơ chuyển động theo thời gian đồng thời giảm thiểu chi phí bộ nhớ khi triển khai một bộ đệm chuyển động theo thời gian. Xin lưu ý rằng có thể có hai phương pháp giúp giảm thiểu chi phí bộ nhớ khi triển khai một bộ đệm chuyển động theo thời gian. Các phương pháp này có thể sử dụng kết hợp hoặc sử dụng độc lập. Một phương pháp là triển khai riêng bộ đệm chuyển động theo thời gian sử dụng các kỹ thuật nén không mất dữ liệu của thông tin chuyển động. Ví dụ: tham chiếu đến việc lấy mvLXCol ở trên, một tập hợp giá trị MvLX[x][y] 18 bit có thể được lưu trữ trong quá trình triển khai một bộ đệm chuyển động theo thời gian bằng kỹ thuật nén không mất dữ liệu. Một phương pháp khác là cho phép vectơ chuyển động theo thời gian được lấy theo cách yêu cầu độ chính xác thấp hơn so với độ chính xác tuyệt đối của thông tin chuyển động theo thời gian. Ví dụ: theo các kỹ thuật trong tài liệu này, như được mô tả ở phần bên dưới, việc lấy mvLXCol ở trên cho hình ảnh hiện tại có thể được sửa đổi sao cho mvLXCol có thể được lấy từ một giá trị MvLX[x][y] 16 bit từ hình ảnh sắp xếp.

HÌNH 5 là sơ đồ khối minh họa một ví dụ về hệ thống có thể được định cấu hình để mã hóa (tức là mã hóa và/hoặc giải mã) dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này. Hệ thống 100 đại diện cho một ví dụ về hệ thống có thể thực hiện mã hóa video bằng kỹ thuật dự đoán vectơ chuyển động được mô tả theo một hoặc nhiều ví dụ của sáng chế này. Như minh họa trong HÌNH 5, hệ thống 100 bao gồm thiết bị nguồn 102, phương tiện truyền thông 110 và thiết bị đích 120. Trong ví dụ minh họa trên HÌNH 5, thiết bị nguồn 102 có thể bao gồm bất kỳ thiết bị nào được định cấu hình để mã hóa dữ liệu video và truyền dữ liệu video đã mã hóa tới phương tiện 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 cho 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 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). (Các) 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 một 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 định cấu hình để nhận dữ liệu video và tạo chuỗi bit tương thích đại diện cho dữ liệu video đó. Chuỗi bit tương thích có thể đề cập đến chuỗi bit mà từ đó bộ giải mã video có thể nhận 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 định cấu hình để nhận chuỗi bit video tương thích và truyền và/hoặc lưu trữ chuỗi bit video tương thích tới một phương tiện 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 một giao diện hệ thống máy tính 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, 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 một 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 bất kỳ thiết bị nào được định cấu hình để nhận chuỗi bit video tương thích từ một phương tiện 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 nhận 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 định cấu hình để nhận 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 bị 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). Xin lưu ý rằng, mặc dù trong ví dụ minh

họa trên HÌNH 8, bộ giải mã video 124 được mô tả là xuất dữ liệu ra màn hình 126, nhưng có thể định 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 định 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. Xin lưu ý rằng, mặc dù bộ mã hóa video làm ví dụ 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 kiếntrúc 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. Trong một ví dụ, có thể định 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 vìư vậy, có thể được gọi là bộ mã hóa video lai. Trong ví dụ minh họa trên HÌNH 6, bộ mã hóa video 200 nhận 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. Trong một số ví dụ, có thể định cấu hình bộ mã hóa video 200 để thực hiện phân vùng phụ bổ sung của 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. Trong ví dụ minh họa trên 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 minh họa trong HÌNH 6, bộ mã hóa video 200 nhận các khối video nguồn và xuất ra một chuỗi bit. Bộ mã hóa video 200 có thể tạo ra dữ liệu dư thừa bằng cách lấy một khối video nguồn trừ đi một khối video dự đoán. Bộ cộng 202 đại diện cho một thành phần được định cấu hình để thực hiện phép tính 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 một 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 thừa hoặc các khối con của chúng (ví dụ: có thể áp dụng bốn phép biến đổi 8x8 cho một mảng 16x16 giá trị dư) để tạo ra một tập hợp hệ

số biến đổi dư. Có thể định 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. Bộ tạo hệ số biến đổi 204 có thể xuất hệ số biến đổi thành bộ lượng tử hóa hệ số 206. Có thể định cấu hình Bộ lượng tử hóa hệ số 206 để thực hiện lượng tử hóa các 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 định cấu hình thêm cho Bộ lượng tử hóa hệ số 206 để xác định các tham 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 các giá trị QP delta) có thể được bộ giải mã video sử dụng để tạo lại 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.

Như minh họa trong HÌNH 6, hệ số biến đổi lượng tử hóa được xuất ra bộ xử lý lượng tử hóa/biến đổi ngược 208. Có thể định 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 thừa. Như minh họa trong HÌNH 6, tại bộ cộng 210, dữ liệu tái tạo dư thừa 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 định 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 đã mô tả ở trên, có thể mã hóa khối video bằng chế độ dự đoán nội ảnh. Có thể được định cấu hình Bộ xử lý dự đoán nội ảnh 212 để chọn chế độ dự đoán nội ảnh cho một khối video hiện tại. Có thể định cấu hình Bộ xử lý dự đoán nội ảnh 212 để đánh giá một khung hình và/hoặc một vùng của khung hình và xác định một chế độ dự đoán nội ảnh sẽ sử dụng để mã hóa một khối hiện tại. Như 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) tới bộ mã hóa entropy 218 và bộ tạo hệ số biến đổi 204. 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. Có thể định cấu hình Bộ xử lý dự đoán liên khung 214 để thực hiện mã hóa dự đoán liên khung cho một khối video hiện tại. Có thể định cấu hình Bộ xử lý dự đoán liên khung 214 để nhận các khối video nguồn và tính toán thông tin 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. Ví dụ: bộ xử lý dự đoán liên khung 214 có thể xác định vị trí khối video dự đoán trong bộ đệm khung hình (không được minh họa trong HÌNH 6). Xin lưu ý rằng, có thể định 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 các giá trị pixel số nguyên phụ để sử dụng trong ước tính chuyển động. 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 định 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, ví dụ: tổng chênh lệch tuyệt đối (SAD), tổng chênh lệch bình phương (SSD) hoặc các số liệu chênh lệch khác. Bộ xử lý dự đoán liên khung 214 có thể xuất dữ liệu dự đoán chuyển động cho một vectơ chuyển động đã tính toán tới bộ mã hóa entropy 218.

Như đã mô tả ở trên, có thể xác định và chỉ định thông tin chuyển động theo kỹ thuật dự đoán véc tơ chuyển động. Có thể định cấu hình Bộ xử lý dự đoán liên khung 214 để thực hiện các kỹ thuật dự đoán vectơ chuyển động, bao gồm cả các kỹ thuật được mô tả ở trên. Hơn nữa, có thể định cấu hình bộ xử lý dự đoán liên khung 214 để thực hiện phép dự đoán vectơ chuyển động theo các kỹ thuật được mô tả ở trên. Cụ thể, có thể định cấu hình bộ xử lý dự đoán liên khung 214 để thực hiện phép dự đoán vectơ chuyển động theo thời gian. Như đã mô tả ở trên, việc cho phép dự đoán vectơ chuyển động theo thời gian thông qua một bộ đệm chuyển động theo thời gian thường có thể tốn nhiều chi phí về bộ nhớ.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, có thể định cấu hình bộ xử lý dự đoán liên khung 214 để lưu trữ các giá trị đại diện cho một vectơ chuyển động bằng cách sử dụng số lượng bit giảm đi so với số lượng bit của một vectơ chuyển động đã lấy (tức là vectơ chuyển động được dùng để tạo một khối dự đoán). Xin lưu ý rằng ngoài và/hoặc thay thế cho các cách xác định vectơ chuyển động được đề xuất ở trên, có thể có một số cách để xác định vectơ chuyển động (ví dụ: tọa độ cực, v.v.). Các kỹ thuật được mô tả trong tài liệu này có thể áp dụng chung bất kể vectơ chuyển động được xác định như thế nào.

Như được mô tả ở trên, việc cho phép dự đoán vectơ chuyển động theo thời gian thông qua một bộ đệm chuyển động theo thời gian thường có thể tốn nhiều chi phí về bộ nhớ. Trong một ví dụ, một bộ đệm chuyển động theo thời gian có thể chứa các trường sau cho mỗi khối (ví dụ: PU trong ITU-T H.265) hoặc khối con:

- Cờ hiệu 1 bit cho biết liệu khối có được mã hóa bằng cách sử dụng dự đoán liên khung, ví dụ như trường isInter hay không;

- Cờ hiệu 2 bit cho biết hướng dự đoán liên khung, ví dụ, trường interDir (ví dụ: danh sách 0, danh sách 1 hoặc dự đoán hai hướng);

- Mã định danh thành viên 16 bit không dấu cho tập hợp ô, ví dụ trường tileSetIdx;

- Giá trị chỉ mục mảnh 16 bit không dấu, ví dụ trường sliceIdx:

- Đối với mỗi thành phần vectơ chuyển động sẽ là một số nguyên 18 bit có dấu. Ví dụ: trường mv[MAX_RPL_COUNT][2], trong đó [2] đại diện cho hướng dịch chuyển ngang và dọc và MAX_RPL_COUNT là 2 và tương ứng với list0 và list1.

- một mảng số nguyên 4 bit không dấu cho biết chỉ mục ảnh tham chiếu, ví dụ referenceIdx[MAX_RPL_COUNT].

Xin lưu ý rằng giá trị 4 bit không dấu có thể chỉ ra chỉ số ảnh tham chiếu, vì số ảnh tối đa trong danh sách ảnh tham chiếu (RPL) trong JVET-L1001 là 16. Ngoài ra, xin lưu ý rằng trong một số trường hợp, MAX_RPL_COUNT có thể tăng lên, ví dụ như nếu danh sách hình ảnh tham chiếu bổ sung được hỗ trợ.

Như đã mô tả ở trên, trong JVET-L1001, mvLX[0] và mvLX[1] được lấy dưới dạng giá trị 18 bit. Hơn nữa, trong JVET-L1001, cpmvLX[0] và cpmvLX[1] là mỗi giá trị 18 bit. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, mvLX[0], mvLX[1], cpmvLX[0] và cpmvLX[1] có thể được lưu trữ vào bộ đệm chuyển động theo thời gian dưới dạng giá trị N-bit, trong đó N nhỏ hơn 18 trở lên, trong đó N nhỏ hơn số bit giá trị thu được của một thành phần vectơ chuyển động. Trong một ví dụ, một giá trị đại diện cho thành phần vectơ chuyển động có thể được cắt bớt xuống một khoảng sao cho có thể biểu diễn bằng N-bit trước khi được lưu trữ trong bộ đệm chuyển động theo thời gian. Ví dụ: Có thể lưu trữ MV_x dưới dạng Clip3(-2^^15, 2^^15 – 1, MV_x) và có thể lưu trữ MV_y dưới dạng Clip3(-2^^15, 2^^15 – 1, MV_y). Liên quan đến việc lấy vectơ chuyển động sắp xếp trong JVET-L1001 được trình

bày ở trên, trong trường hợp một giá trị đại diện cho thành phần vectơ chuyển động bị cắt bớt, thì trong một ví dụ, giá trị được lưu trữ của mvCol có thể như sau:

mvCol = Clip3(- (1<< (N-1)), (1 << (N-1)) - 1, mvCol)

Trong một ví dụ khác, chỉ có thể lưu trữ một tập con gồm N MSB-bit của thành phần vectơ chuyển động thu được vào một bộ đệm chuyển động theo thời gian. Ví dụ: đối với MV_x và MV_y, có thể lưu trữ các giá trị sau vào bộ đệm chuyển động theo thời gian:

$MVDD_x = MV_x/4$; và

$MVDD_y = MV_y/4$

Trong ví dụ này, trong quá trình đọc từ bộ đệm chuyển động theo thời gian, các giá trị được tái tạo của MV_x và MV_y, MV_{xR} và MV_{yR} có thể được lấy như sau:

$MV_{xR} = 4* MVDD_x$: và

$MV_{yR} = 4* MVDD_y$

Liên quan đến việc lấy vectơ chuyển động sắp xếp trong JVET-L1001 được trình bày ở trên, trong trường hợp chỉ một tập con gồm N MSB-bit của thành phần vectơ chuyển động thu được được lưu trữ vào bộ đệm chuyển động theo thời gian, thì trong một ví dụ, giá trị được lưu trữ của mvCol có thể như sau:

mvCol = (mvCol >> 4)

và giá trị thu được của mvCol có thể như sau:

mvCol = (mvCol << 4)

Xin lưu ý rằng trong một số trường hợp, MV_{xR} và/hoặc MV_{yR} có thể bằng MV_x và MV_y và trong các trường hợp khác thì MV_{xR} và/hoặc MV_{yR} không bằng MV_x và MV_y. Do đó, trong một số ví dụ, giá trị delta có thể được cho có điều kiện đối với MV_{xR} và/hoặc MV_{yR}, sao cho MV_x và/hoặc MV_y có thể suy ra được trong trường hợp MV_{xR} và/hoặc MV_{yR} không bằng MV_x và MV_y. Ví dụ: trong một ví dụ, một bộ đệm chuyển động theo thời gian có thể bao gồm một cờ hiệu cho mỗi mục nhập MV_{xR} và MV_{yR} cho biết liệu giá trị delta tương ứng có trong bộ đệm chuyển động theo thời gian hay không. Trong trường hợp này, trong một ví dụ liên quan đến việc lấy vectơ chuyển động sắp xếp trong JVET-L1001 được trình bày ở trên, giá trị được lưu trữ của mvCol có thể như sau:

mvCol = (mvCol >> 4)

và giá trị thu được của mvCol có thể như sau:

mvCol = (mvCol << 4) + giá trị delta,

trong đó giá trị delta có điều kiện dựa trên giá trị của cờ hiệu và được suy ra là 0 khi không có.

Trong một ví dụ khác, chỉ có thể lưu trữ một tập con N của các bit LSB của thành phần vectơ chuyển động thu được. Ví dụ: một thành phần vectơ chuyển động thu được là X bit và có thể được lưu trữ dưới dạng N-bit, trong đó (N-X) MSB-bit không được lưu trữ. Trong một ví dụ, (N-X) MSB-bit có thể thu được bằng cách sử dụng các vectơ chuyển động lân cận theo không gian (ví dụ: bằng cách lấy trung bình các vectơ chuyển động lân cận theo không gian hoặc ví dụ, sử dụng (các) vectơ chuyển động tại (các) vị trí xác định trước theo mức độ ưu tiên xác định trước để lấy (N-X) MSB bit). Ví dụ: trong một ví dụ, một bộ đệm chuyển động theo thời gian có thể bao gồm một giá trị delta để lấy ra (N-X) MSB bit từ các vectơ chuyển động lân cận theo không gian. Trong trường hợp này, trong

một ví dụ liên quan đến việc lấy vectơ chuyển động sắp xếp trong JVET-L1001 được trình bày ở trên, giá trị được lưu trữ của mvCol có thể như sau:

mvCol = (mvCol & LSBMask), trong đó LSBMask = 0x000000FF

và giá trị thu được của mvCol có thể như sau:

mvCol = mvCol | (mvAdjacent & MSBMask), trong đó MSBMask = 0xFFFFFF00 và mvAdjacent được lấy từ các vectơ chuyển động lân cận theo không gian, ví dụ, một vectơ ở bên trái khối hiện tại

Hơn nữa, trong một ví dụ khác, có thể kết hợp các kỹ thuật trên sao cho, đối với thành phần vectơ chuyển động A-bit, một tập con các bit không bao gồm các bit B MSB và không bao gồm C LSB sẽ được lưu trữ (tức là, các bit ở giữa được lưu trữ).

Trong một ví dụ khác, có thể sử dụng độ phân giải cao hơn để lưu trữ các thành phần vectơ chuyển động nhỏ. Ví dụ: một thành phần vectơ chuyển động có thể được xác định là một thành phần vectơ chuyển động nhỏ nếu thành phần vectơ chuyển động thu được nằm trong một khoảng cụ thể (ví dụ: $-2^{\wedge 15}$ đến $(2^{\wedge 15}-1)$, bao hàm). Trong một ví dụ, khoảng này có thể là NULL (tức là cờ hiệu vectơ chuyển động nhỏ không được báo hiệu và được suy ra là 0). Trong một ví dụ, một cờ hiệu có thể được lưu trữ vào/đọc từ bộ đệm chuyển động theo thời gian cho biết liệu một vectơ chuyển động có phải là vectơ chuyển động nhỏ hay không. Trong trường hợp này, trong một ví dụ, khi thành phần vectơ chuyển động là thành phần vectơ chuyển động nhỏ, thì cờ hiệu vectơ chuyển động nhỏ được đặt thành 1 và giá trị vectơ chuyển động nhỏ được lưu trữ (ví dụ: ở độ phân giải 1/16-pel) và khi thành phần vectơ chuyển động không phải là thành phần vectơ chuyển động nhỏ, thì cờ hiệu vectơ chuyển động nhỏ được đặt thành 0 và thành phần vectơ chuyển động được chia tỷ lệ (ví dụ: chia tỷ lệ bằng cách chia cho 4 (dịch chuyển sang phải 2 có/không có giá trị bù làm tròn 1)) và được lưu trữ (ví dụ: ở độ phân giải 1/4-pel). Hơn nữa, trong trường hợp này, khi thành phần vectơ chuyển động nhỏ được đọc từ bộ đệm chuyển động theo thời gian, khi cờ hiệu vectơ chuyển động nhỏ là 1 (có thể được suy ra dựa trên khoảng giá trị được đọc từ bộ đệm chuyển động), thì giá trị thành phần của vectơ chuyển động được đọc từ bộ đệm chuyển động theo thời gian

ở độ phân giải được liên kết với vectơ chuyển động nhỏ và khi cờ hiệu vectơ chuyển động nhỏ bằng 0, giá trị vectơ chuyển động được đọc từ bộ đệm chuyển động theo thời gian ở độ phân giải xác định khác (ví dụ: tại 1/4-pel) và được chia tỷ lệ (ví dụ: nhân với 4 – dịch chuyển sang trái 2) thành độ phân giải vectơ chuyển động nhỏ (ví dụ: độ phân giải 1/16-pel). Trong trường hợp này, trong một ví dụ liên quan đến việc lấy vectơ chuyển động sắp xếp trong JVET-L1OOl được trình bày ở trên, giá trị được lưu trữ của mvCol có thể như sau:

mvCol = (smallMotionVectorFlag) ? mvCol : (mvCol >> 2)

và giá trị thu được của mvCol có thể như sau:

mvCol = (smallMotionVectorFlag) ? mvCol : (mvCol << 2)

Trong một ví dụ khác, nếu một giá trị được lưu trữ gần biên bao quanh của N-bit, (ví dụ: 8 bit cho giá trị LSB và LSB>229 hoặc giá trị LSB<25), thì một phép trừ/cộng tự động (ví dụ: của 128) có thể áp dụng cho giá trị được lưu trữ. Trong một ví dụ, phép trừ/cộng có thể được dự đoán thông qua giá trị trung bình của các ứng viên theo không gian của ảnh hiện tại. Trong một ví dụ, các vectơ chuyển động có thể được cắt thành 8 bit, và nếu giá trị là +/-128, thì nó không được đưa vào làm ứng viên. Trong một ví dụ, có thể báo hiệu rồi áp dụng hệ số dịch chuyển trước khi cắt. Trong ví dụ này, nếu kết quả ở điểm rìa cắt, trong một số trường hợp nó có thể không được đưa vào làm ứng viên vectơ chuyển động. Ví dụ: nếu một vectơ chuyển động đã lấy là 0x0F, thì 4 LSB bit là 0xF. Nếu trung bình của các ứng viên theo không gian là 0x10, thì việc chỉ ghép nối 0x10 với 0x0F sẽ cho 0x1F, đây có thể là giá trị không mong muốn. Do đó, trong một ví dụ, khi tiếp cận biên bao quanh, có thể thực hiện thêm một phép toán di chuyển các bit LSB ra khỏi biên đó (ví dụ: thêm 0x7). Thao tác này sau đó có thể bị đảo ngược sau khi ghép nối. Vì vậy, đối với trường hợp trên, trong một ví dụ, 0x0F + 0x07 = 0x16 có thể được dùng và 0x6 có thể được lưu trữ. Trong trường hợp này, trung bình theo không gian sẽ trở thành 0x10 + 0x07 = 0x17. Quá trình ghép nối sẽ cho 0x16 và cuối cùng việc loại bỏ phép cộng chứng tỏ: 0x16 - 0x07 = 0x0F. Trong một ví dụ, việc biên bao quanh có được tiếp cận hay không có thể được xác định bằng cách so sánh với các giá trị ngưỡng (ví dụ: 229 và 25).

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, có thể định cấu hình bộ xử lý dự đoán liên khung 214 để lưu trữ giá trị đại diện cho thành phần vectơ chuyển động bằng cách sử dụng khoảng cắt tương ứng với độ chính xác của phép tính vectơ chuyển động. Như được trình bày ở trên, trong JVET-L1001, tỷ lệ của các bộ dự đoán vectơ chuyển động theo thời gian/không gian/chuyển động thu được dựa trên khoảng cách số thứ tự hình ảnh (POC), trong đó khoảng cách POC tương ứng với khoảng cách giữa các ảnh theo thời gian. Có thể chia tỷ lệ một vectơ chuyển động tham chiếu thành ảnh mục tiêu dựa trên tỷ lệ giữa khoảng cách POC giữa ảnh dự đoán và ảnh tham chiếu của vectơ chuyển động tham chiếu và khoảng cách POC giữa ảnh dự đoán hiện tại và ảnh tham chiếu. Có thể dùng vectơ chuyển động được chia tỷ lệ làm một dự đoán. Phép chia tỷ lệ vectơ chuyển động theo thời gian/không gian/thu được có thể tiếp theo sau thao tác cắt (ví dụ: Clip3(-2^15, 2^15-1, scaledMv), trong đó scaledMv là thành phần vectơ chuyển động có hướng dịch chuyển vectơ chuyển động được chia tỷ lệ) cho mỗi hướng dịch chuyển vectơ chuyển động chia tỷ lệ. Trong trường hợp này, khi sử dụng vectơ chuyển động 1/16-pel trong quá trình giải mã, điều mong muốn là các giới hạn cắt không bị hạn chế một cách không cần thiết, vì điều này có thể dẫn đến cắt xén các vectơ chuyển động lớn. Trong một ví dụ, theo các kỹ thuật trong tài liệu này, có thể định cấu hình bộ xử lý dự đoán liên khung 214 để cho phép độ sâu bit lớn hơn (ví dụ: 18 bit) của hướng dịch chuyển vectơ chuyển động được chia tỷ lệ mà giới hạn cắt cần được tăng lên (ví dụ: Clip3(-2^17, 2^17-1, scaledMv). Trong trường hợp này, trong một ví dụ liên quan đến việc lấy vectơ chuyển động sắp xếp trong JVET-L1001 được trình bày ở trên, mvCol và mvLXCol có thể được lấy như sau:

mvCol = Clip3(– 131072, 131071, Sign(distScaleFactor * mvCol) *
((Abs(distScaleFactor * mvCol) + 127) >> 8))

mvLXCol = Clip3(– 131072, 131071, Sign(distScaleFactor * mvCol) *
((Abs(distScaleFactor * mvCol) +127) >> 8))

Hơn nữa, trong trường hợp này, các ứng viên dự đoán vectơ chuyển động khác có thể được cắt theo cách tương tự.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, có thể định cấu hình bộ xử lý dự đoán liên khung 214 để sử dụng một tập con các bit để so sánh quy trình cắt xén. Như đã mô tả ở trên, có thể dùng quy trình cắt xén có thể để loại bỏ các mục nhập thừa khỏi tập hợp ứng viên vectơ chuyển động. Trong một ví dụ, theo kỹ thuật trong tài liệu này, quy

trình cắt xén có thể sử dụng một tập con các bit để xác định xem ứng viên chuyển động có dư thừa hay không và do đó, xác định xem liệu ứng viên chuyển động có được thêm vào danh sách ứng viên hay không. Hơn nữa, trong một ví dụ, theo kỹ thuật trong tài liệu này, quy trình cắt xén có thể được dùng để xác định xem thông tin chuyển động cho một khối có được lưu trữ vào bộ đệm chuyển động theo thời gian hay không. Trong một ví dụ, quy trình cắt xén có thể chỉ sử dụng một tập con các bit của thành phần vectơ chuyển động cho mục đích so sánh. Ví dụ: chỉ x bit LSB của thành phần vectơ chuyển động được dùng để so sánh hoặc chỉ x bit MSB của thành phần vectơ chuyển động được dùng để so sánh.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, có thể định cấu hình bộ xử lý dự đoán liên khung 214 để sử dụng chuẩn hóa nhằm giảm độ sâu bit của vectơ chuyển động được lưu trữ trong bộ đệm chuyển động theo thời gian. Ví dụ: một kiếntrúc minh họa của bộ đệm chuyển động theo thời gian có thể chứa 82 bit (hoặc 90 bit) sau đây trên mỗi khối 16x16:

4 thành phần MV 16 bit (hoặc 18 bit) mỗi thành phần

2 giá trị delta POC mỗi giá trị 8 bit

2 LTRP cờ hiệu (hình ảnh tham chiếu dài hạn) mỗi 1 bit

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, để giảm băng thông bộ nhớ, có thể cần giảm yêu cầu lưu trữ xuống 64 bit. Trong một ví dụ, có thể giảm các yêu cầu về lưu trữ bằng cách (1) lưu trữ vectơ chuyển động chuẩn hóa (ví dụ: chuẩn hóa đối với POC delta, sau đó sẽ không cần phải lưu trữ); (2) sử dụng cách biểu diễn kiểu dấu chấm động cho vectơ chuyển động chuẩn hóa (có thể độ chính xác 1/16 không thực sự hữu ích đối với các vectơ chuyển động lớn); và/hoặc cắt xén ở phạm vi giảm (nghĩa là các MV chuẩn hóa không được lớn bằng). Trong một ví dụ, việc chuẩn hóa đối với POC delta có thể bao gồm việc xác định giá trị chênh lệch POC ban đầu thể hiện sự khác biệt về giá trị POC của hình ảnh hiện tại và hình ảnh tham chiếu hiện tại. Để giảm lưu trữ, có thể điều chỉnh giá trị chênh lệch POC thành một giá trị không đổi (ví dụ: 1, 2, 4) và vectơ chuyển động tương ứng có thể được chia tỷ lệ thích hợp xét đến tỷ lệ của chênh lệch POC ban đầu và chênh lệch POC mới. Việc chia tỷ lệ vectơ chuyển động có thể tiếp theo sau một hoạt động cắt xén (nghĩa là trong quá trình lưu trữ). Việc lựa chọn giá trị chênh lệch POC không đổi không những giúp giảm dung lượng lưu trữ mà còn đơn giản hóa hoạt động chia tỷ lệ theo thời gian thường bao gồm việc lấy tỷ lệ của hai giá trị chênh lệch POC, một giá trị tương ứng với sự chênh lệch giữa POC của hình ảnh hiện tại và POC của hình ảnh tham chiếu, trong khi giá trị kia tương ứng với chênh

lệch giữa POC của hình ảnh theo thời gian và POC hình ảnh tham chiếu của hình ảnh theo thời gian. Trong quá trình lưu trữ, giá trị thứ hai có thể được đặt về một đại lượng không đổi, do đó làm giảm độ phức tạp tính toán của tỷ lệ chênh lệch của hai POC. Ví dụ: nếu mẫu số là 1 thì phép chia có thể được bỏ qua, hoặc nếu mẫu số là lũy thừa của 2 thì phép chia sẽ trở thành phép toán dịch chuyển bit. Phép chuẩn hóa còn có thể dựa trên việc liệu vectơ chuyển động có tham chiếu đến hình ảnh tham chiếu dài hạn hay không. Ví dụ: vectơ chuyển động tham chiếu đến hình ảnh tham chiếu dài hạn có thể không được chuẩn hóa. Trong trường hợp này, trong một ví dụ liên quan đến việc lấy vectơ chuyển động sắp xếp trong JVET-L1001 được trình bày ở trên, mvCol có thể được suy ra như sau:

Khi Hình ảnh tương ứng với refPicListCol[refIdxCol] không phải là hình ảnh tham chiếu dài hạn, thì các bước tuần tự sau sẽ được thực hiện:

colPocDiff = DiffPicOrderCnt[ColPic, refPicListCol[refIdxCol])

newColPocDiff = CONSTANT

tx = (16384 + (Abs(td) >> 1)) / td

distScaleFactor = Clip3(–4096, 4095, (tb * tx + 32) >> 6)

mvCol = Clip3(– 131072, 131071, Sign(distScaleFactor * mvCol) *
((Abs(distScaleFactor * mvCol) + 127) >> 8))

trong đó td và tb được lấy như sau:

td = Clip3(–128, 127, colPocDiff)

tb = Clip3(–128, 127, newColPocDiff)

- Khi availableFlagLXCol bằng TRUE, thì mvLXCol và availableFlagLXCol được lấy như sau:
  - Nếu LongTermRefPic(currPic, currCb, refIdxLX, LX) không bằng LongTermRefPic(ColPic, colCb, refIdxCol, listCol), thì cả hai thành phần của mvLXCol sẽ được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.
  - Nếu không, biến availableFlagLXCol được đặt bằng 1, refPicListCol[refIdxCol] được đặt là hình ảnh có chỉ mục tham chiếu refIdxCol trong danh sách hình ảnh tham chiếu listCol của mảnh chứa khối mã hóa colCb trong hình ảnh sắp xếp được chỉ định bởi ColPic và áp dụng như sau:

colPocDiff = newColPocDiff

currPocDiff = DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])

– Nếu RefPicListX[refIdxLX] là hình ảnh tham chiếu dài hạn hoặc colPocDiff bằng với currPocDiff, thì mvLXCol được lấy như sau:

mvLXCol = mvCol

– Nếu không, mvLXCol được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvCol như sau:

tx = (16384 + (Abs(td) >> 1)) / td

distScaleFactor = Clip3(– 4096, 4095, (tb * tx + 32) >> 6)

mvLXCol = Clip3(– 131072, 131071, Sign(distScaleFactor * mvCol) *
((Abs(distScaleFactor * mvCol) + 127) >> 8))

trong đó td và tb được lấy như sau:

td = Clip3(–128, 127, colPocDiff)

tb = Clip3(–128, 127, currPocDiff)

Như đã mô tả ở trên, trong một ví dụ, có thể giảm các yêu cầu lưu trữ của bộ đệm chuyển động theo thời gian bằng cách lưu trữ vectơ chuyển động chuẩn hóa; sử dụng phép biểu diễn kiểu dấu chấm động cho vectơ chuyển động chuẩn hóa; và/hoặc cắt xén giảm xuống một khoảng. Hơn nữa, trong một ví dụ, các yêu cầu lưu trữ của bộ đệm chuyển động theo thời gian có thể được giảm bớt bằng cách chuyển đổi chuyển vị vectơ chuyển động thành phép biểu diễn Phần định trị-Số mũ trước khi lưu trữ. Trong một ví dụ, nếu khoảng biểu diễn lớn hơn một giá trị ngưỡng (ví dụ: 18 bit), thì không cần cắt dịch chuyển vectơ chuyển động. Trong một ví dụ, phần định trị có thể nhận 6 bit và nằm trong khoảng từ -32 đến 31 (bao hàm) và số mũ có thể nhận 4 bit và nằm trong khoảng từ 0 đến 15 (bao hàm). Hơn nữa, trong một ví dụ,

các vectơ chuyển động tham chiếu đến hình ảnh dài hạn cũng có thể được chia tỷ lệ và/hoặc chuyển đổi thành dạng biểu diễn dấu chấm động trước khi lưu trữ.

Trong một ví dụ, phép biểu diễn Phần định trị-Số mũ của phần trị số có thể được tính như sau:

```
int convertNumberToMantissaExponentForm(int number)
{
  int mantissa = number;
  int exponent = 0;
  if (number < MV_MANTISSA_LOWER_LIMIT || number >
MV_MANTISSA_UPPER_LIMIT)
  {
     exponent = 1;
       while (number < -(1 << (exponent + MV_MANTISSA_BITCOUNT - 1)) ||
         number > ((1<< (exponent + MV_MANTISSA_BITCOUNT - 1)) - 1))
     {
        exponent++;
     }
     mantissa = (number >> (exponent - 1)) ^ MV_MANTISSA_LIMIT;
}
int mantissaExponentForm = exponent | (mantissa <<
MV_EXPONENT_BITCOUNT);
trả về mantissaExponentForm;}
```

Trong đó,

```
static const int MV_EXPONENT_BITCOUNT = 4;
static const int MV_MANTISSA_BITCOUNT = 6;
static const int MV_MANTISSA_LOWER_LIMIT = -(1<<
(MV_MANTISSA_BITCOUNT - 1));
```

static const int MV_MANTISSA_UPPER_LIMIT = ((1 << (MV_MANTISSA_BITCOUNT - 1)) - 1);

static const int MV_MANTISSA_LIMIT = (1 << (MV_MANTISSA_BITCOUNT - 1)); và

static const int MV_EXPONENT_MASK = ((1 << MV_EXPONENT_BITCOUNT) - 1);

Hơn nữa, việc chuyển đổi thành một số từ phép biểu diễn Phần định trị-Số mũ có thể được thực hiện như sau:

```
int convertMantissaExponentFormToNumber(int mantissaExponentForm)
{
   int exponent = mantissaExponentForm & MV_EXPONENT_MASK;
   int mantissa = mantissaExponentForm >> MV_EXPONENT_BITCOUNT;
   if (exponent == 0)
   {
    trả về phần định trị;
   }
   khác
   }
    trả về ((mantissa ^ MV_MANTISSA_LIMIT) << (exponent - 1));
   }
}
```

Do đó, trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc lấy vectơ chuyển động sắp xếp có thể như sau:

Giá trị đầu vào cho quy trình này là:

- một biến currCb xác định khối mã hóa hiện tại,
- một biến colCb chỉ định khối mã hóa sắp xếp bên trong hình ảnh sắp xếp được chỉ định bởi ColPic,

- một vị trí luma (xColCb, yColCb) chỉ định mẫu trên cùng bên trái của khối mã hóa luma sắp xếp được chỉ định bởi colCb liên quan đến mẫu luma trên cùng bên trái của hình ảnh sắp xếp do ColPic chỉ định,
- một chỉ mục tham chiếu refIdxLX, với X là 0 hoặc 1,
- cờ cho biết ứng viên hợp nhất theo thời gian của khối con sbFlag.

Giá trị đầu ra của quy trình này là:

- dự đoán vectơ chuyển động mvLXCol với độ chính xác 1/16 phân số – mẫu,
- cờ báo hiệu tính khả dụng availableFlagLXCol.

Biến currPic chỉ định hình ảnh hiện tại.

Các mảng predFlagL0Col[x][y], mvL0Col[x][y] và refIdxL0Col[x][y] được đặt lần lượt bằng PredFlagL0[x][y], MvL0[x][y] và RefIdxL0[x][y], của ảnh sắp xếp được ColPic chỉ định và các mảng predFlagL1Col[x][y], mvL1Col[x][y] và refIdxL1Col[x][y] được đặt lần lượt bằng PredFlagL1[x][y], MvL1[x][y] và RefIdxL1[x][y], của ảnh sắp xếp được ColPic chỉ định.

Các biến mvLXCol và availableFlagLXCol được lấy như sau:

- Nếu colCb được mã hóa trong chế độ dự đoán nội ảnh, thì cả hai thành phần của mvLXCol được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.
- Nếu không, vectơ chuyển động mvCol, chỉ số tham chiếu refIdxCol và mã định danh danh sách tham chiếu listCol được lấy như sau:
  - Nếu sbFlag bằng 0, thì availableFlagLXCol được đặt thành 1 và áp dụng như sau:
    - Nếu predFlagL0Col[xColCb][yColCb] bằng 0, mvCol, refIdxCol và listCol được đặt bằng mvL1Col[xColCb][yColCb], ref;dxL1Col[xColCb][yColCb] và L1 tương ứng.
    - Nếu không, nếu predFlagL0Col[xColCb] [yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 0, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvL0Col[xColCb][yColCb], refIdxL0Col[xColCb] E yColCb] và L0.
    - Nếu không (predFlagL0Col[xColCb] [yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 1), các phép gán sau sẽ được thực hiện:

- Nếu NoBackwardPredFlag bằng 1, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb], refIdxLXCol[xColCb][yColCb] và LX.
- Nếu không, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLNCol[xColCb][yColCb], refIdxLNCol[xColCb][yColCb] và LN, trong đó N là giá trị của collocated_from_10_flag.

– Nếu không (sbFlag bằng 1), sẽ áp dụng những điểm sau:

- Nếu PredFlagLXCol[xColCb] [yColCb] bằng 1, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb], refIdxLXCol[xColCb][yColCb] và LX, availableFlagLXCol được đặt thành 1.
- Nếu không (PredFlagLXCol[xColCb][yColCb] bằng 0), thì sẽ áp dụng các điểm sau:
  - Nếu DiffPicOrderCnt(aPic, currPic) nhỏ hơn hoặc bằng 0 đối với mọi ảnh aPic trong mọi danh sách ảnh tham chiếu của mảnh hiện tại và PredFlagLYCol[xColCb][yColCb] bằng 1, thì mvCol, refIdxCol và listCol được đặt thành mvLYCol[xColCb][yColCb], refIdxLYCol[xColCb][yColCb] và LY tương ứng, với Y bằng !X trong đó X là giá trị X, quy trình này được gọi cho availableFlagLXCol được đặt thành 1.
  - Cả hai thành phần của mvLXCol đều được đặt thành 0 và availableFlagLXCol được đặt bằng 0.

– Khi availableFlagLXCol bằng TRUE, thì mvLXCol và availableFlagLXCol được lấy như sau:

- Nếu LongTermRefPic(currPic, currCb, refIdxLX, LX) không bằng LongTermRefPic(ColPic, colCb, refIdxCol, listCol), thì cả hai thành phần của mvLXCol sẽ được đặt bằng 0 và availableFlagLXCol sẽ được đặt bằng 0.
- Nếu không, biến availableFlagLXCol được đặt bằng 1, refPicListCol[refIdxCol] được đặt là hình ảnh có chỉ mục tham chiếu refIdxCol trong danh sách hình ảnh tham chiếu listCol của mảnh chứa khối mã hóa colCb trong hình ảnh sắp xếp được chỉ định bởi ColPic và áp dụng như sau:

colPocDiff= DiffPicOrderCnt(ColPic, refPicListCol[refIdxCol])

currPocDiff = DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])

– mvColScaled được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvCol như sau:

txScaled = (16384 + (Abs(tdScaled) >> 1)) / 4

distScaleFactor = Clip3(–4096, 4095, (4 * txScaled + 32) >> 6)

mvColScaled = Clip3(–32768, 32767, Sign(distScaleFactor * mvCol) * ((Abs(distScaleFactor * mvCol) + 127) >> 8))

trong đó tdScaled được lấy như sau:

tdScaled = Clip3(–128, 127, colPocDiff)

– Quá trình suy diễn để chuyển đổi phép dịch chuyển vectơ chuyển động thành phép biểu diễn phần định trị và số mũ như được nêu trong Mệnh đề A (bên dưới) được gọi ra với mvColScaled làm giá trị đầu vào và đầu ra được gán cho mvColScaledME.

– Quá trình suy diễn cho phép dịch chuyển vectơ chuyển động từ phép biểu diễn phần định trị và số mũ như được nêu trong Mệnh đề B (bên dưới) được gọi ra với mvColScaledME làm giá trị đầu vào và đầu ra được gán cho mvColScaledQ.

– Nếu RefPicListX[refIdxLX] là hình ảnh tham chiếu dài hạn hoặc colPocDiff bằng với currPocDiff, thì mvLXCol được lấy như sau:

mvLXCol = mvColScaledQ

– Nếu không, mvLXCol được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvColScaledQ như sau:

tx = 4096

distScaleFactor = Clip3(–4096, 4095, (tb * tx + 32) >> 6)

mvLXCol = Clip3(–32768, 32767, Sign(distScaleFactor * mvColScaledQ) * ((Abs(distScaleFactor * mvColScaledQ) + 127) >> 8))

trong đó tb được lấy như sau:

tb = Clip3(–128, 127, currPocDiff)

Mệnh đề A Quá trình suy diễn để chuyển đổi phép dịch chuyển vectơ chuyển động thành phép biểu diễn phần định trị và số mũ

Giá trị đầu vào cho quy trình này là:

– một biến mvColScaled xác định vectơ chuyển động theo thời gian được chia tỷ lệ,

Giá trị đầu ra của quy trình này là:

– một biến mvColScaledME xác định phép biểu diễn phần định trị cộng số mũ của vectơ chuyển động theo thời gian được chia tỷ lệ

Các biến phần định trị và số mũ được lấy như sau:

```
exponent = 0
if (mvColScaled < -32 || mvColScaled > 31) {
  exponent = 1
  while (mvColScaled <-(1 <<(exponent + 5)) || mvColScaled >((1 <<(exponent + 5)) - 1)) {
    exponent++
  }
  mantissa = (number >> (exponent - 1)) ^ 32
}
```

Biến mvColScaledME được lấy như sau:

```
mvColScaledME = exponent | (mantissa << 4)
```

Mệnh đề B Quá trình suy diễn cho phép dịch chuyển vectơ chuyển động từ phép biểu diễn phần định trị và số mũ

Giá trị đầu vào cho quy trình này là:

– một biến mvColScaledME chỉ định vectơ chuyển động theo thời gian được chia tỷ lệ trong phần định trị cộng với phép biểu diễn số mũ,

Giá trị đầu ra của quy trình này là:

– một biến mvColScaledQ chỉ định vectơ chuyển động theo thời gian được chia tỷ lệ được lấy từ phần định trị của nó cộng với phép biểu diễn số mũ

Các biến phần định trị và số mũ được lấy như sau:

số mũ = mvColScaledME & 15

phần định trị = mvColScaledME >> 4

Biến mvColScaledQ được lấy như sau:

if (exponent == 0)

mvColScaledQ = phần định trị

khác

mvColScaledQ = ((phần định trị ^ 32) << (số mũ - 1));

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, phần định trị và số mũ đối với các giá trị xác định trước (ví dụ: -32 và 15, tương ứng) để biểu thị sự dịch chuyển vectơ chuyển động theo thời gian không có sẵn cho dự đoán chuyển động. Ví dụ: khi sử dụng chế độ dự đoán nội ảnh thì tất cả bốn giá trị hướng dịch chuyển vectơ chuyển động được gán một phần định trị là -32 và số mũ là 15. Tương tự, khi chỉ một trong hai vectơ chuyển động hợp lệ (ví dụ: inter_pred_idc[][] là PRED_L0 hoặc PRED_L1), thì vectơ chuyển động không có thông tin chuyển động hợp lệ sẽ được gán một phần định trị và số mũ là -32 và 15 cho cả hai hướng dịch chuyển. Cũng có thể áp dụng phương pháp để chỉ báo tính khả dụng này cho thông tin chuyển động theo thời gian tương ứng với tham chiếu hình ảnh hiện tại.

Trong một ví dụ, việc tính toán phép biểu diễn Phần định trị-Số mũ của số có thể bao gồm một quy trình hai bước. Bước đầu tiên xác định khoảng lượng tử hóa, và bước thứ hai thêm một nửa khoảng lượng tử hóa rồi lượng tử hóa. Biểu diễn Phần định trị-Số mũ có thể được tính như sau:

int convertNumberToMantissaExponentForm(số nguyên)

{

int mantissa = 0;

int exponent = 0;

```
   if (number < MV_MANTISSA_LOWER_LIMIT | | number >
MV_MANTISSA_UPPER_LIMIT)
   {
    // xác định khoảng lượng tử hóa
    exponent++;
    while (number < -(1 << (exponent + MV_MANTISSA_BITCOUNT - 1)) | |
            number > ((1<< (exponent + MV_MANTISSA_BITCOUNT - 1)) - 1))
    {
       exponent ++;
    }
    // tính toán phần định trị và số mũ cho giá trị đầu vào với phần bù làm tròn
    (lượng tử hóa)
    int numberWithOffset = number + ((1 << (exponent - 1)) >> 1);
    exponent = 1;
    while (numberWithOffset < -(1 << (exponent +
    MV_MANTISSA__BITCOUNT - 1))
||
            numberWithOffset > ((1 << (exponent +
MV_MANTISSA_BITCOUNT - 1)) - 1))
    {
       exponent ++;
    }
    mantissa = (numberWithOffset >> (exponent - 1)) ^ MV_MANTISSA_LIMIT;
 }
 khác
 {
    phần định trị = số;
```

}

int mantissaExponentForm = exponent | (mantissa <<
MV_EXPONENT_BITCOUNT);

trả về mantissaExponentForm;}

Trong đó,

static const int MV_EXPONENT_BITCOUNT = 4;

static const int MV_MANTISSA_BITCOUNT = 6;

static const int MV_MANTISSA_LOWER_LIMIT = -(1 <<
(MV_MANTISSA_BITCOUNT - 1));

static const int MV_MANTISSA_UPPER_LIMIT = ((1 <<
(MV_MANTISSA_BITCOUNT - 1)) - 1);

static const int MV_MANTISSA_LIMIT = (1 << (MV_MANTISSA_BITCOUNT - 1)); và

static const int MV_EXPONENT_MASK = ((1<< MV_EXPONENT_BITCOUNT) - 1);

Hơn nữa, trong một ví dụ về tính toán biểu diễn Phần định trị-Số mũ, tính toán phần định trị và số mũ cho giá trị đầu vào với phần bù làm tròn (lượng tử hóa) có thể như sau:

int numberWithOffset = number + (((1 << (exponent - 1)) – ((number < 0) ? 0 : 1)) >> 1);

Hơn nữa, trong một ví dụ về tính toán biểu diễn Phần định trị-Số mũ, tính toán phần định trị và số mũ cho giá trị đầu vào với phần bù làm tròn (lượng tử hóa) có thể như sau:

int numberWithOffset = number + (((1<< (exponent - 1)) – ((number < 0) ? 1 : 0)) >> 1);

Do đó, trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc lấy vectơ chuyển động sắp xếp có thể như sau:

Viện dẫn mệnh đề A, quy trình lưu trữ thông tin chuyển động luma theo thời gian:

Mệnh đề A sẽ được gọi ra như một phần của quá trình giải mã cho hình ảnh hiện tại. Mệnh đề A sẽ được gọi ra sau khi tất cả thông tin chuyển động (vectơ chuyển động, chỉ số tham chiếu, cờ hiệu sử dụng dự đoán, gắn cờ hiệu hình ảnh tham chiếu dài hạn) cho hình ảnh hiện tại đã được giải mã và xác định giá trị của nó. Mệnh đề A (và các mệnh đề được gọi ra trong Mệnh đề A) sẽ là mệnh đề cuối cùng truy nhập thông tin chuyển động trong quá trình giải mã của hình ảnh hiện tại.

Hàm LongTermRefPic(aPic, aCb, refIdx, LX), với X là 0 hoặc 1, có thể được xác định như sau:

- Nếu ảnh có chỉ mục refIdx từ danh sách ảnh tham chiếu LX của mảnh chứa khối mã hóa aCb trong ảnh thì aPic được đánh dấu là "được dùng để tham chiếu dài hạn" tại thời điểm aPic là ảnh hiện tại, LongTermRefPic(aPic, aCb, refIdx, LX) bằng 1.
- Nếu không, LongTermRefPic(aPic, aCb, refIdx, LX) sẽ bằng 0.

Mệnh đề A. Quy trình lưu trữ thông tin chuyển động luma theo thời gian

Giá trị đầu vào cho quy trình này là:

- Mảng danh sách dự đoán sử dụng cờ hiệu PredFlagL0 và PredFlagL1
- Mảng vectơ chuyển động luma MvL0 và MvL1
- Mảng các chỉ số tham chiếu RefIdxL0 và RefIdxL1

Giá trị đầu ra của quy trình này là:

- Mảng MvL0Mantissa, MvL0Exponent, MvL1Mantissa và MvL1MExponent của vectơ chuyển động luma được lưu trữ trong phép biểu diễn phần định trị và số mũ
- Mảng LtrpFlagL0 và LtrpFlagL1 của cờ hiệu biểu thị việc sử dụng Hình ảnh tham chiếu dài hạn

Biến currPic chỉ định hình ảnh hiện tại

Các điểm sau được áp dụng cho xL từ 0 đến (pic_width_in_luma_samples >> 3):

- Các điểm sau áp dụng cho yL từ 0 đến (pic_height_in_luma_samples >> 3):
  - Các biến x và y lần lượt được gán giá trị (xL << 3) và (yL << 3)
  - Biến currCb chỉ định khối mã hóa tại vị trí luma (x, y)
  - Khi PredFlagL0[x][y] bằng 0, hãy gán -32 cho MvL0Mantissa[x][y][0] và MvL0Mantissa[x][y][1], và 15 cho MvL0Exponent[x][y][0] và MvL0Exponent[x][y][1], Nếu không thì quá trình lấy để chia tỷ lệ và chuyển đổi độ dịch chuyển vectơ chuyển động thành biểu diễn phần định trị và số mũ như được chỉ định trong mệnh đề B được gọi ra với MvL0[x][y] làm giá trị đầu vào, và giá trị đầu ra được gán cho MvL0Mantissa[x][y] và MvL0Exponent[x][y]

- Gán giá trị đầu ra LongTermRefPic(currPic, currCb, RefIdxL0[x][y], L0) cho LtrpFlagL0 [x] [y]
- Khi PredFlagL1[x][y] bằng 0, hãy gán -32 cho MvL1Mantissa[x][y][0] và MvL1Mantissa[x][y][1] và 15 cho MvL1Exponent[x][y][0] và MvL1Exponent[x][y][1] Nếu không thì quá trình lấy để chia tỷ lệ và chuyển đổi độ dịch chuyển vectơ chuyển động thành biểu diễn phần định trị và số mũ như được chỉ định trong mệnh đề B được gọi ra với MvL1[x][y] làm giá trị đầu vào, và giá trị đầu ra được gán cho MvL1Mantissa[x][y] và MvL1Exponent[x][y]
- Gán giá trị đầu ra LongTermRefPic(currPic, currCb, RefIdxL1[x][y], L1) cho LtrpFlagL1[x][y]

Quá trình suy diễn đối với các vectơ chuyển động sắp xếp

Giá trị đầu vào cho quy trình này là:

- một biến currCb xác định khối mã hóa hiện tại,
- một biến colCb chỉ định khối mã hóa sắp xếp bên trong hình ảnh sắp xếp được chỉ định bởi ColPic,
- một vị trí luma (xColCb, yColCb) chỉ định mẫu trên cùng bên trái của khối mã hóa luma sắp xếp được chỉ định bởi colCb liên quan đến mẫu luma trên cùng bên trái của hình ảnh sắp xếp do ColPic chỉ định,
- một chỉ mục tham chiếu refIdxLX, với X là 0 hoặc 1,
- cờ cho biết ứng viên hợp nhất theo thời gian của khối con sbFlag.

Giá trị đầu ra của quy trình này là:

- dự đoán vectơ chuyển động mvLXCol với độ chính xác 1/16 phân số – mẫu,
- cờ báo hiệu tính khả dụng availableFlagLXCol.

Biến currPic chỉ định hình ảnh hiện tại.

Mảng mvL0MantissaCol[x][y], mvL0ExponentCol[x][y], ltrpL0FlagCol[x][y], mvL1MantissaCol[x][y], mvL1ExponentCol[x] [y] và ItrpL1FlagCol[x] [y] được đặt lần lượt bằng MvL0Mantissa[x] [y], MvL0Exponent[x][y] LtrpL0Flagt x][y], MvL1Manitssa[x][y], MvL1Exponent[x][y] và LtrpL1Flag[x][y] của ảnh sắp xếp được ColPic chỉ định.

Khi mvL0MantissaCol[x][y][0], mvL0ExponentCol[x][y][0] lần lượt bằng -32 và 15, hãy đặt predFlagL0Col[x][y] bằng 0, Nếu không, đặt predFlagL0Col[x][y] bằng 1.

Khi mvL1MantissaCol[x][y][0], mvL1ExponentCol[x][y][0] lần lượt bằng -32 và 15, hãy đặt predFlagL1Col[x][y] bằng 0, Nếu không, đặt predFlagL1Col[x][y] bằng 1.

Các biến mvLXCol và availableFlagLXCol được lấy như sau:

- Nếu colCb được mã hóa trong chế độ dự đoán nội ảnh, thì cả hai thành phần của mvLXCol được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.
- Nếu không, vectơ chuyển động mvCol, chỉ số tham chiếu refIdxCol và mã định danh danh sách tham chiếu listCol được lấy như sau:
  - Nếu sbFlag bằng 0, thì availableFlagLXCol được đặt thành 1 và áp dụng như sau:
    - Nếu predFlagL0Col[xColCb][yColCb] bằng 0, mvCol và listCol được đặt lần lượt bằng mvL1Col[xColCb][yColCb] và L1.
    - Nếu không, nếu predFlagL0Col[xColCb][yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 0, mvCol và listCol được đặt lần lượt bằng mvL0Col[xColCb][yColCb] và L0.
    - Nếu không (predFlagL0Col[xColCb] [yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 1), các phép gán sau sẽ được thực hiện:
      - Nếu NoBackwardPredFlag bằng 1, mvCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb] và LX.
      - Nếu không, mvCol, và listCol được đặt lần lượt bằng mvLNCol[xColCb][yColCb] và LN, với N là giá trị của collocated_from_10_flag.
  - Nếu không (sbFlag bằng 1), sẽ áp dụng những điểm sau:
    - Nếu PredFlagLXCol[xColCb] [yColCb] bằng 1, mvCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb] và LX, availableFlagLXCol được đặt thành 1.
    - Nếu không (PredFlagLXCol[xColCb][yColCb] bằng 0), thì sẽ áp dụng các điểm sau:

- Nếu DiffPicOrderCnt(aPic, currPic) nhỏ hơn hoặc bằng 0 đối với mọi ảnh aPic trong mọi danh sách ảnh tham chiếu của mảnh hiện tại và PredFlagLYCol[xColCb][yColCb] bằng 1, thì mvCol và listCol được đặt lần lượt bằng mvLYCol[xColCb] [yColCb] và LY, với Y bằng !X trong đó X là giá trị X, quy trình này được gọi cho availableFlagLXCol được đặt thành 1.
- Cả hai thành phần của mvLXCol đều được đặt thành 0 và availableFlagLXCol được đặt bằng 0.

- Khi availableFlagLXCol bằng TRUE, thì mvLXCol và availableFlagLXCol được lấy như sau:
  - Nếu LongTermRefPic(currPic, currCb, refIdxLX, LX) không bằng ltrpLXFlagCol[xColCb][yColCb], thì cả hai thành phần của mvLXCol sẽ được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.
  - Nếu không, biến availableFlagLXCol được đặt bằng 1, refPicListCol[refIdxCol] được đặt là hình ảnh có chỉ mục tham chiếu refIdxCol trong danh sách hình ảnh tham chiếu listCol của mảnh chứa khối mã hóa colCb trong hình ảnh sắp xếp được chỉ định bởi ColPic và áp dụng như sau:

    colPocDiff = DiffPicOrderCnt(ColPic, refPicListCol[refIdxCol])

    currPocDiff = DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])

    - Quá trình suy diễn đối với phép dịch chuyển vectơ chuyển động từ phép biểu diễn phần định trị và số mũ như được nêu trong mệnh đề C được gọi ra với mvL0MantissaCol[xColCb][yColCb], mvL0ExponentCol[xColCb][yColCb] làm giá trị đầu vào và giá trị đầu ra được gán cho mvColScaledQ.
    - Nếu RefPicListXt refIdxLX] là hình ảnh tham chiếu dài hạn hoặc colPocDiff bằng với currPocDiff, thì mvLXCol được lấy như sau:

      mvLXCol = mvColScaledQ

    - Nếu không, mvLXCol được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvColScaledQ như sau:

      tx = 4096

distScaleFactor = Clip3(–4096, 4095, (tb * tx + 32) >> 6)

mvLXCol = Clip3(–32768, 32767, Sign(distScaleFactor * mvColScaledQ) * ((Abs(distScaleFactor * mvColScaledQ) + 127) >> 8))

trong đó td được lấy như sau:

tb = Clip3(–128, 127, currPocDiff)

Mệnh đề B. Quá trình suy diễn để tỷ lệ hóa và chuyển đổi phép dịch chuyển vectơ chuyển động thành phép biểu diễn phần định trị và số mũ

Giá trị đầu vào cho quy trình này là:

– một mảng mvCol chỉ định vector chuyển động đầu vào,

Giá trị đầu ra của quy trình này là:

– Các mảng mvColScaledMantissa, mvColScaledExponent xác định phép biểu diễn phần định trị và số mũ của vectơ chuyển động theo thời gian được chia tỷ lệ

Biến mvColScaled được lấy như sau:

txScaled = (16384 + (Abs(tdScaled) >> 1)) / tdScaled

distColScaleFactor = (4 * txScaled + 32) >> 6

mvColScaled = Clip3(–32768, 32767, Sign(distColScaleFactor * mvCol) * ((Abs(distColScaleFactor * mvCol) + 127) >> 8))

trong đó tdScaled được lấy như sau:

tdScaled = Clip3(–128, 127, colPocDiff)

Mảng mvColScaledMantissa, mvColScaledExponent được lấy như sau:

for(dir = 0; dir < 2; dir++) {

mantissa[dir] = mvColScaled[dir]

exponent[dir] = 0

if (mvColScaled[dir] < -32 || mvColScaled[dir] > 31) {

exponentoffset = 1

while (mvColScaled[dir] < -(1 << (exponentOffset[dir] + 5)) ||

```
                    mvColScaled[dir] > ((1 << (exponentOffset [dir] + 5)) -1)) {
            exponentOffset++
          }
          offset = (1 << (exponentOffset - 1)) >> 1
          exponent[dir] = 1
          while ((mvColScaled[dir] + offset) < -(1 << (exponent; dir] + 5)) ||
                    (mvColScaled; dir] + offset) > ((1 << (exponent; dir] + 5)) - 1)){
            exponent[dir]++
            offset = (1 << (exponent[dir] + 4))
          }
          mantissa[dir] = ((mvColScaled[dir] + offset) >> (exponent[dir] – 1)) ^ 32
        }
        mvColScaledMantissa[dir] = mantissa[dir]
        mvColScaledExponent[dir] = exponent[dir]
      }
```

Ngoài ra, trong một ví dụ, mảng mvColScaledMantissa, mvColScaledExponent có thể được lấy như sau:

```
for(dir = 0; dir < 2; dir++) {
  mantissa[dir] = mvColScaled[dir]
  exponent[dir] = 0
  if (mvColScaled[dir] < -32 || mvColScaled[dir] > 31) {
    exponentOffset = 1
    while (mvColScaled[dir] < -(1 << (exponentoffset + 5)) ||
              mvColScaled[dir] > ((1 << (exponentOffset + 5)) - 1)){
      exponentOffset++
    }
```

```
        offset = ((1 << (exponentOffset - 1)) - ((mvColScaled[dir] < 0) ? 0 : 1)) >> 1
 exponent[dir] = 1
        while ((mvColScaled[dir] + offset) < -(1 << (exponent[dir] + 5)) ||
                        (mvColScaled[dir] + offset) > ((1 << (exponent[dir] + 5)) - 1)) {
           exponent[dir]++
           offset = (1 << (exponent[dir] + 4))
        }
        mantissa[dir] = ((mvColScaled[dir] + offset) >> (exponent[dir] – 1)) ^ 32
     }
     mvColScaledMantissa[dir] = mantissa[dir]
     mvColScaledExponent[dir] = exponent[dir]
}
```

Ngoài ra, trong một ví dụ, mảng mvColScaledMantissa, mvColScaledExponent có thể được lấy như sau:

```
for(dir = 0; dir < 2; dir++) {
  mantissa[dir] = mvColScaled[dir]
  exponent[dir] = 0
  if (mvColScaled[dir] < -32 || mvColScaled[dir] > 31) {
    exponentOffset = 1
    while (mvColScaled[dir] < -(1 << (exponentOffset + 5)) ||
                    mvColScaled[dir] > ((1 << (exponentOffset + 5))-l)){exponentOffset++
    }
    offset = offset = ((1 << (exponentOffset - 1)) - ((mvColScaled[dir] < 0) ? 1 : 0)) >> 1
exponent[dir] = 1
    while ((mvColScaled[dir] + offset) < -(1 << (exponent[dir] + 5)) ||
            (mvColScaled[dir] + offset) > ((1 << (exponent[dir] + 5)) - 1)) {
```

```
        exponent[dir]++
        offset = (1 << (exponent[dir] + 4))
      }
     mantissa[dir] = ((mvColScaled[dir] + offset) >> (exponent[dir] – 1)) ^ 32
    }
   mvColScaledMantissa[dir] = mantissa[dir]
   mvColScaledExponent[dir] = exponent[dir]
}
```

Mệnh đề C. Quá trình suy diễn cho phép dịch chuyển vectơ chuyển động từ phép biểu diễn phần định trị và số mũ

Giá trị đầu vào cho quy trình này là:

– Các mảng mvColScaledMantissa, mvColScaledExponent xác định vectơ chuyển động theo thời gian được chia tỷ lệ trong phép biểu diễn định trị và số mũ,

Giá trị đầu ra của quy trình này là:

– một mảng mvColScaledQ xác định vectơ chuyển động theo thời gian đã chia tỷ lệ được lấy từ phép biểu diễn phần định trị và số mũ

Mảng mvColScaledQ được lấy như sau:

```
for(dir = 0; dir < 2; dir++) {
    exponent[dir] = mvColScaledExponent[dir]
    mantissa[dir] = mvColScaledMantissa[dir]
    if (exponent[dir] = = 0)
      mvColScaledQ[dir] = mantissa[dir]
    khác
      mvColScaledQ[dir] = ((mantissa[dir] ^ 32) << (exponent[dir] - 1))
  }
```

trong đó,

dir đại diện cho hướng dịch chuyển, ví dụ: 0 đại diện cho chiều ngang còn 1 đại diện cho chiều dọc

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, phạm vi cắt cho giá trị vectơ chuyển động được sửa đổi thành 18 bit. Trong một ví dụ, Clip3(-32768, 32767, ...) được sửa đổi thành Clip3(-131072, 131071, ...).

Trong một ví dụ, việc chuyển đổi từ biểu diễn phần định trị thành số nguyên thu được có thể dựa trên việc số mũ có phải là một số được xác định trước hay không. Ví dụ: khi số mũ bằng 0 thì số nguyên thu được bằng phần định trị.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc chuyển đổi từ biểu diễn phần định trị thành số nguyên có thể dẫn đến sai lệch. Có thể là do biểu diễn phần định trị-số mũ không hoàn toàn đối xứng (ví dụ: khi phần định trị là một số nguyên có dấu 6 bit). Điều mong muốn là không có sai lệch dựa trên dấu. Trong một ví dụ, để tránh sai lệch dấu, phần định trị có thể bao gồm một số nguyên không dấu 5 bit và một dấu 1 bit. Đối với các số mũ khác không, việc tái tạo sẽ là (MV_MANTISSA_LIMIT + phần định trị) * (1 - 2 * dấu) << (số mũ - 1) thay vì (phần định trị ^ MV_MANTISSA_LIMIT) << (số mũ - 1). Mặt khác, việc khôi phục từ phần định trị có dấu 6 bit có thể như sau nếu muốn đối xứng: ((phần định trị ^ MV_MANTISSA_LIMIT) + (phần định trị < 0 ? 1 : 0) << (số mũ - 1). Trong một ví dụ, MV_MANTISSA_LIMIT có thể là 32.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, phần định trị phải là dấu đầu tiên được khai triển trước khi áp dụng phép toán XOR. Ví dụ: phần định trị có dấu 6 bit trước tiên phải là dấu được khai triển thành giá trị có dấu 18 bit trước khi áp dụng phép toán XOR.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, phép toán XOR giữa phần định trị và một số được thay thế bằng phần định trị cộng với trị số đối với giá trị dương (tức là lớn hơn hoặc bằng 0) của phần định trị và phần định trị trừ đi trị số đối với giá trị âm của phần định trị.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc lấy vectơ chuyển động sắp xếp có thể như sau:

Viện dẫn mệnh đề A, quy trình lưu trữ thông tin chuyển động luma theo thời gian:

Mệnh đề A sẽ được gọi ra như một phần của quá trình giải mã cho hình ảnh hiện tại. Mệnh đề A sẽ được gọi ra sau khi tất cả thông tin chuyển động (vectơ chuyển động, chỉ số tham chiếu, cờ hiệu sử dụng dự đoán, gắn cờ hiệu hình ảnh tham chiếu dài hạn)

cho hình ảnh hiện tại đã được giải mã và xác định giá trị của nó. Mệnh đề A (và các mệnh đề được gọi ra trong Mệnh đề A) sẽ là mệnh đề cuối cùng truy nhập thông tin chuyển động trong quá trình giải mã của hình ảnh hiện tại.

Mệnh đề A. Quy trình lưu trữ thông tin chuyển động luma theo thời gian

Giá trị đầu vào cho quy trình này là:

– Mảng danh sách dự đoán sử dụng cờ hiệu PredFlagL0 và PredFlagL1

– Mảng vectơ chuyển động luma MvL0 và MvL1

– Mảng các chỉ số tham chiếu RefIdxL0 và RefIdxL1

Giá trị đầu ra của quy trình này là:

– Mảng MvL0Mantissa, MvL0Exponent, MvL1Mantissa và MvL1MExponent của vectơ chuyển động luma được lưu trữ trong phép biểu diễn phần định trị và số mũ

Biến currPic chỉ định hình ảnh hiện tại

Các điểm sau được áp dụng cho xL từ 0 đến (pic_width_in_luma_samples >> 3):

– Các điểm dưới đây áp dụng cho yL từ 0 đến (pic_height_in_luma_samples >> 3):

– Các biến x và y lần lượt được gán giá trị (xL << 3) và (yL << 3)

– Biến currCb chỉ định khối mã hóa tại vị trí luma (x, y)

– Quá trình suy diễn để chuyển đổi phép dịch chuyển vectơ chuyển động thành phép biểu diễn phần định trị và số mũ như được nêu trong mệnh đề B được gọi ra với MvL0[x][y] làm giá trị đầu vào, và giá trị đầu ra được gán cho MvL0Mantissa[x][y] và MvL0Exponent[x][y]

– Quá trình suy diễn để chuyển đổi phép dịch chuyển vectơ chuyển động thành phép biểu diễn phần định trị và số mũ như được nêu trong mệnh đề B được gọi ra với MvL1[x][y] làm giá trị đầu vào, và giá trị đầu ra được gán cho MvL1MantissaE x] [y] và MvL1Exponent[x][y]

Quá trình suy diễn đối với các vectơ chuyển động sắp xếp

Giá trị đầu vào cho quy trình này là:

– một biến currCb xác định khối mã hóa hiện tại,

- một biến colCb chỉ định khối mã hóa sắp xếp bên trong hình ảnh sắp xếp được chỉ định bởi ColPic,
- một vị trí luma (xColCb, yColCb) chỉ định mẫu trên cùng bên trái của khối mã hóa luma sắp xếp được chỉ định bởi colCb liên quan đến mẫu luma trên cùng bên trái của hình ảnh sắp xếp do ColPic chỉ định,
- một chỉ mục tham chiếu refIdxLX, với X là 0 hoặc 1,
- cờ cho biết ứng viên hợp nhất theo thời gian của khối con sbFlag.

Giá trị đầu ra của quy trình này là:

- dự đoán vectơ chuyển động mvLXCol với độ chính xác 1/16 phân số – mẫu,
- cờ báo hiệu tính khả dụng availableFlagLXCol.

Biến currPic chỉ định hình ảnh hiện tại.

Các mảng predFlagL0Col[x][y] và refIdxL0Col[x][y] được đặt lần lượt bằng PredFlagL0[x][y] và RefIdxL0[x][y], của ảnh sắp xếp được ColPic chỉ định và các mảng predFlagL1Col[x][y] và refIdxL1Col[x][y] được đặt lần lượt bằng PredFlagL1[x][y] và RefIdxL1[x] [y], của ảnh sắp xếp được ColPic chỉ định.

Các mảng mvL0MantissaCol[x][y], mvL0ExponentCol[x][y], mvL1MantissaCol[x][y] và mvL1ExponentCol[x] [y] được đặt lần lượt bằng MvL0Mantissa[x][y], MvL0Exponent[x][y], MvL1Manitssat x][y] và MvL1Exponent[x][y] của ảnh sắp xếp được ColPic chỉ định.

Các biến mvLXCol và availableFlagLXCol được lấy như sau:

- Nếu colCb được mã hóa trong chế độ dự đoán nội ảnh, thì cả hai thành phần của mvLXCol được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.
- Nếu không, vectơ chuyển động mvCol, chỉ số tham chiếu refIdxCol và mã định danh danh sách tham chiếu listCol được lấy như sau:
  - Nếu sbFlag bằng 0, thì availableFlagLXCol được đặt thành 1 và áp dụng như sau:
    - Nếu predFlagL0Col[xColCb][yColCb] bằng 0, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvL1Col[xColCb][yColCb], refIdxL1Col[xColCb][yColCb] và L1.

- Nếu không, nếu predFlagL0Col[xColCb][yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 0, thì mvCol, refIdxCol và listCol được đặt lần lượt bằng mvL0Col[xColCb][yColCb], refIdxL0Col[xColCb][yColCb] và L0.
- Nếu không (predFlagL0Col[xColCb] [yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 1), các phép gán sau sẽ được thực hiện:
  - Nếu NoBackwardPredFlag bằng 1, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCb] [yColCb], refIdxLXCol[xColCb] [yColCb] và LX.
  - Nếu không, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLNCol[xColCb][yColCb], refIdxLNCol[xColCb][yColCb] và LN, với N là giá trị của collocated_from_10_flag.

- Nếu không (sbFlag bằng 1), sẽ áp dụng những điểm sau:
  - Nếu PredFlagLXCol[xColCb] [yColCb] bằng 1, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb], refIdxLXCol[xColCb][yColCb] và LX, availableFlagLXCol được đặt thành 1.
  - Nếu không (PredFlagLXCol[xColCb][yColCb] bằng 0), thì sẽ áp dụng các điểm sau:
    - Nếu DiffPicOrderCnt(aPic, currPic) nhỏ hơn hoặc bằng 0 đối với mọi ảnh aPic trong mọi danh sách ảnh tham chiếu của mảnh hiện tại và PredFlagLYCol[xColCb][yColCb] bằng 1, mvCol, refIdxCol và listCol được đặt lần lượt thành mvLYCol[xColCb ItyColCb], refIdxLYCol[xColCb][yColCb] và LY, với Y bằng !X trong đó X là giá trị X, quy trình này được gọi cho availableFlagLXCol được đặt thành 1.
    - Cả hai thành phần của mvLXCol đều được đặt thành 0 và availableFlagLXCol được đặt bằng 0.

- Khi availableFlagLXCol bằng TRUE, thì mvLXCol và availableFlagLXCol được lấy như sau:

– Nếu LongTermRefPic(currPic, currCb, refIdxLX, LX) không bằng LongTermRefPicC ColPic, colCb, refIdxCol, listCol), thì cả hai thành phần của mvLXCol sẽ được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.

– Nếu không, biến availableFlagLXCol được đặt bằng 1, refPicListCol[refIdxCol] được đặt là hình ảnh có chỉ mục tham chiếu refIdxCol trong danh sách hình ảnh tham chiếu listCol của mảnh chứa khối mã hóa colCb trong hình ảnh sắp xếp được chỉ định bởi ColPic và áp dụng như sau:

colPocDiff = DiffPicOrderCnt(ColPic, refPicListCol[refIdxCol])

currPocDiff = DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])

– Quá trình suy diễn đối với phép dịch chuyển vectơ chuyển động từ phép biểu diễn phần định trị và số mũ như được nêu trong mệnh đề C được gọi ra với mvL0MantissaCol[xColCb][yColCb], mvL0ExponentCol[xColCb][yColCb] làm giá trị đầu vào và giá trị đầu ra được gán cho mvColQ.

– Nếu RefPicListX[refIdxLX] là hình ảnh tham chiếu dài hạn hoặc colPocDiff bằng với currPocDiff, thì mvLXCol được lấy như sau:

mvLXCol = mvColQ

– Nếu không, mvLXCol được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvColQ như sau:

tx = (16384 + (Abs(td) >> 1)) / td

distScaleFactor = Clip3(–4096, 4095, (tb * tx + 32) >> 6)

mvLXCol = Clip3(–32768, 32767, Sign(distScaleFactor * mvColQ) * ((Abs(distScaleFactor * mvColQ) + 127) >> 8))

trong đó td và tb được lấy như sau:

td = Clip3(–128, 127, colPocDiff)

tb = Clip3(–128, 127, currPocDiff)

Mệnh đề B. Quá trình suy diễn để chuyển đổi phép dịch chuyển vectơ chuyển động thành phép biểu diễn phần định trị và số mũ

Giá trị đầu vào cho quy trình này là:

– một mảng mvCol xác định vectơ chuyển động đầu vào, Giá trị đầu ra của quá trình này là:

– Các mảng mvColMantissa, mvColExponent xác định phép biểu diễn phần định trị và số mũ của vectơ chuyển động theo thời gian

Mảng mvColMantissa, mvColExponent được lấy như sau:

```
for(dir = 0,- dir < 2; dir++) {
  mantissa[dir] = mvCol [dir]
  exponent[dir] = 0
  if (mvCol[dir] < -32 | | mvCol[dir] > 31) {
    exponentOffset = 1
    while (mvCol[dir] < -(1 << (exponentOffset + 5)) | |
                mvCol[dir] > ((1 << (exponentOffset + 5)) - 1)){
      exponentOffset++
    }
    offset = (1 << (exponentOffset - 1)) >> 1
    exponent[dir] = 1
    while ((mvCol[dir] + offset) < -(1 << (exponent[dir] + 5)) | |
                (mvCol[dir] + offset) > ((1 << (exponent[dir] + 5)) - 1)){
      exponent[dir]++
    }
    mantissa[dir] = ((mvCol [dir] + offset) >> (exponent[dir] – 1)) ^ 32
  }
  mvColMantissa[dir] = mantissa[dir]
```

mvColExponent[dir] = exponent[dir]

}

Mệnh đề C. Quá trình suy diễn cho phép dịch chuyển vectơ chuyển động từ phép biểu diễn phần định trị và số mũ

Giá trị đầu vào cho quy trình này là:

– Các mảng mvColMantissa, mvColExponent xác định vectơ chuyển động theo thời gian trong phép biểu diễn định trị và số mũ,

Giá trị đầu ra của quy trình này là:

– một mảng mvColQ xác định vectơ chuyển động theo thời gian thu được từ phép biểu diễn phần định trị và số mũ

Mảng mvColQ được lấy như sau:

for(dir = 0; dir < 2; dir++) {

exponent[dir] = mvColExponent[dir]

mantissa[dir] = mvColMantissa[dir]

if (exponent[dir] = = 0)

mvColQ[dir] = mantissa[dir]

khác

mvColQ[dir] = ((mantissa[dir] ^ 32) << (exponent[dir] - 1))

}

Trong một ví dụ, tỷ lệ của vectơ chuyển động bị bỏ qua, trong trường hợp đó mvColScaled bằng mvCol trong Mệnh đề B.

Trong một ví dụ, theo các kỹ thuật trong tài liệu này, việc lấy vectơ chuyển động sắp xếp có thể như sau:

Giá trị đầu vào cho quy trình này là:

– một biến currCb xác định khối mã hóa hiện tại,

– một biến colCb chỉ định khối mã hóa sắp xếp bên trong hình ảnh sắp xếp được chỉ định bởi ColPic,

- một vị trí luma (xColCb, yColCb) chỉ định mẫu trên cùng bên trái của khối mã hóa luma sắp xếp được chỉ định bởi colCb liên quan đến mẫu luma trên cùng bên trái của hình ảnh sắp xếp do ColPic chỉ định,
- một chỉ mục tham chiếu refIdxLX, với X là 0 hoặc 1,
- cờ cho biết ứng viên hợp nhất theo thời gian của khối con sbFlag.

Giá trị đầu ra của quy trình này là:

- dự đoán vectơ chuyển động mvLXCol với độ chính xác 1/16 phân số – mẫu,
- cờ báo hiệu tính khả dụng availableFlagLXCoL

Biến currPic chỉ định hình ảnh hiện tại.

Các mảng predFlagL0Col[x][y], mvL0Col[x][y] và ref;dxL0Col[x][y] được đặt lần lượt bằng PredFlagL0[x][y], MvL0[x][y] và RefIdxL0[x][y] của hình ảnh sắp xếp được xác định bằng ColPic và các mảng predFlagL1Col[x][y], mvL1Col[x][y] và ref;dxL1Col[x][y] được đặt lần lượt bằng PredFlagL1[x][y], MvL1[x][y] và RefIdxL1[x][y], của ảnh sắp xếp được ColPic chỉ định.

Các biến mvLXCol và availableFlagLXCol được lấy như sau:

- Nếu colCb được mã hóa trong chế độ dự đoán liên khung, hoặc ảnh tham chiếu của nó là ColPic, thì cả hai thành phần của mvLXCol được đặt bằng 0 và availableFlagLXCol được đặt bằng 0.
- Nếu không, vectơ chuyển động mvCol, chỉ số tham chiếu refIdxCol và mã định danh danh sách tham chiếu listCol được lấy như sau:
  - Nếu sbFlag bằng 0, thì availableFlagLXCol được đặt thành 1 và áp dụng như sau:
    - Nếu predFlagL0Col[xColCb][yColCb] bằng 0, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvL1Col[xColCb][yColCb], refIdxL1Col[xColCb][yColCb] và L1.
    - Nếu không, nếu predFlagL0Col[xColCb][yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 0, thì mvCol, refIdxCol và listCol được đặt lần lượt bằng mvL0Col[xColCb][yColCb], refIdxL0Col[xColCb][yColCb] và L0.

- Nếu không (predFlagL0Col[xColCb] [yColCb] bằng 1 và predFlagL1Col[xColCb][yColCb] bằng 1), các phép gán sau sẽ được thực hiện:
  - Nếu NoBackwardPredFlag bằng 1, mvCol, thì refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCbltyColCb], refIdxLXCol[xColCb][yColCb] và LX.
  - Nếu không, mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLNCol[xColCb][yColCb], refIdxLNCol[xColCb][yColCb] và LN, với N là giá trị của collocated_from_10_flag.
- Nếu không (sbFlag bằng 1), sẽ áp dụng những điểm sau:
  - Nếu PredFlagLXCol[xColCb][yColCb] bằng 1, thì mvCol, refIdxCol và listCol được đặt lần lượt bằng mvLXCol[xColCb][yColCb], refIdxLXCol[xColCb][yColCb], và LX, availableFlagLXCol được đặt thành 1.
  - Nếu không (PredFlagLXCol[xColCb][yColCb] bằng 0), thì sẽ áp dụng các điểm sau:
    - Nếu DiffPicOrderCnt(aPic, currPic) nhỏ hơn hoặc bằng 0 đối với mọi ảnh aPic trong mọi danh sách ảnh tham chiếu của nhóm ô hiện tại và PredFlagLYCol[xColCb][yColCb] bằng 1, thì mvCol, refIdxCol và listCol được đặt lần lượt thành mvLYCol[xColCb][yColCb], refIdxLYCol[xColCb] [yColCb] và LY, với Y bằng !X trong đó X là giá trị X, quy trình này được gọi cho availableFlagLXCol được đặt thành 1.
    - Cả hai thành phần của mvLXCol đều được đặt thành 0 và availableFlagLXCol được đặt bằng 0.
- Khi availableFlagLXCol bằng TRUE, thì mvLXCol và availableFlagLXCol được lấy như sau:
  - Nếu LongTermRefPic(currPic, currCb, refIdxLX, LX) không bằng LongTermRefPic(ColPic, colCb, refIdxCol, listCol), thì cả hai thành phần của mvLXCol sẽ được đặt bằng 0 và availableFlagLXCol sẽ được đặt bằng 0.
  - Nếu không, biến availableFlagLXCol được đặt bằng 1, refPicListCol[refIdxCol] sẽ được đặt là hình ảnh có chỉ mục tham chiếu refIdxCol trong danh sách hình

ảnh tham chiếu listCol của nhóm ô chứa khối mã hóa colCb trong hình ảnh sắp xếp được chỉ định bởi ColPic và áp dụng như sau:

colPocDiff = DiffPicOrderCnt(ColPic, refPicListCol[refIdxCol])

currPocDiff = DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])

– Quá trình nén bộ đệm chuyển động theo thời gian cho các vectơ chuyển động sắp xếp như được xác định bên dưới được gọi ra với mvCol làm giá trị đầu vào và giá trị đầu ra được gán cho mvCol.

– Nếu RefPicListXt refIdxLX] là hình ảnh tham chiếu dài hạn hoặc colPocDiff bằng với currPocDiff, thì mvLXCol được lấy như sau:

mvLXCol = mvCol

– Nếu không, mvLXCol được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvCol như sau:

tx = (16384 + (Abs(td) >> 1)) / td

distScaleFactor = Clip3(–4096, 4095, (tb * tx + 32) >> 6)

mvLXCol = Clip3(–32768, 32767, Sign(distScaleFactor * mvCol) *
((Abs(distScaleFactor * mvCol) + 127) >> 8))

trong đó td và tb được lấy như sau:

td = Clip3(–128, 127, colPocDiff)

tb = Clip3(–128, 127, currPocDiff)

Quá trình nén bộ đệm chuyển động theo thời gian cho các vectơ chuyển động sắp xếp

Giá trị đầu vào cho quy trình này là:

– Một vectơ chuyển động mv,

Giá trị đầu ra của quy trình này là:

– Một vectơ chuyển động tròn rmv

Đối với mỗi thành phần vectơ chuyển động compIdx, rmv[compIdx] được suy ra từ mv[compIdx] như sau:

s = mv[compIdx] >> 17

f = Floor(Log2((mv[compIdx] ^ s) | 31)) – 4

mask = (-1 << f) >> 1

round = (1 << f) >> 2

rmv[compIdx] = (mv[compIdx] + round) & mask

Xin lưu ý rằng quá trình này cho phép lưu trữ các vectơ chuyển động sắp xếp bằng cách sử dụng phép biểu diễn giảm bit. Mỗi thành phần vectơ chuyển động 18 bit có dấu có thể được biểu diễn ở định dạng phần định trị cộng với số mũ với phần định trị 6 bit có dấu và số mũ 4 bit.

Trong một ví dụ khác, quá trình nén bộ đệm chuyển động theo thời gian đối với các vectơ chuyển động sắp xếp có thuộc tính đối xứng xung quanh giá trị 0. Nghĩa là, nếu –mv[compIdx] làm giá trị đầu vào thay vì mv[compIdx], thì đầu ra sẽ là –rmv[compIdx] thay vì rmv[compIdx]. Do đó, quá trình nén bộ đệm chuyển động theo thời gian cho các vectơ chuyển động sắp xếp có thể như sau:

Quá trình nén bộ đệm chuyển động theo thời gian cho các vectơ chuyển động sắp xếp

Giá trị đầu vào cho quy trình này là:

– Một vectơ chuyển động mv,

Giá trị đầu ra của quy trình này là:

– Một vectơ chuyển động tròn rmv

Đối với mỗi thành phần vectơ chuyển động compIdx, rmv[compIdx] được suy ra từ mv[compIdx] như sau:

s = mv[compIdx] >> 17

f = Floor(Log2((mv[compIdx] ^ s) | 31)) – 4

mask = (-1 << f) >> 1

round = ((1 << f + s) >> 2

rmv[compIdx] = (mv[compIdx] + round) & mask

Trong một ví dụ khác, quá trình nén bộ đệm chuyển động theo thời gian cho các vectơ chuyển động sắp xếp có thể như sau:

Giá trị đầu vào cho quy trình này là:

– Một vectơ chuyển động mv,

Giá trị đầu ra của quy trình này là:

– Một vectơ chuyển động tròn rmv

Đối với mỗi thành phần vectơ chuyển động compIdx, rmv[compIdx] được suy ra từ mv[compIdx] như sau:

s = mv[compIdx] >> 17

f = Floor(Log2((mv[compIdx] ^ s) | 31)) – 4

mask = (-1 << f) >> 1

round = ((1 << f) - 1 - s) >> 2

rmv[compIdx] = (mv[compIdx] + round) & mask

Trong một ví dụ khác, quá trình nén bộ đệm chuyển động theo thời gian cho các vectơ chuyển động sắp xếp có thể như sau:

Giá trị đầu vào cho quy trình này là:

– Một vectơ chuyển động mv,

Giá trị đầu ra của quy trình này là:

– Một vectơ chuyển động tròn rmv

Đối với mỗi thành phần vectơ chuyển động compIdx, rmv[compIdx] được suy ra từ mv[compIdx] như sau:

s = mv[compIdx] >> 17

f = Floor(Log2((mv[compIdx] ^ s) | 31)) – 4

mask = (-1<< f) >> 1

round = ((1 << f) + s) >> 2

rmv[compIdx] = Clip3(-(1<<17), (1<<17)-1, (mv[compIdx] + round) & mask)

Trong một ví dụ khác, quá trình nén bộ đệm chuyển động theo thời gian cho các vectơ chuyển động sắp xếp có thể như sau:

Giá trị đầu vào cho quy trình này là:

– Một vectơ chuyển động mv,

Giá trị đầu ra của quy trình này là:

– Một vectơ chuyển động tròn rmv

Đối với mỗi thành phần vectơ chuyển động compIdx, rmv[compIdx] được suy ra từ mv[compIdx] như sau:

s = mv[compIdx] >> 17

f = Floor(Log2((mv[compIdx] ^ s) | 31)) – 4

mask = (-1 << f) >> 1

round = ((1 << f) + s) >> 2

rmv[compIdx] = Clip3(-(64<<11), (63<<11), (mv[compIdx] + round) & mask)

Xin lưu ý rằng trong trường hợp ở trên khi rmv[compIdx] = Clip3(-(64<<11), (63<<11), (mv[compIdx] + round) & mask), thì tập hợp phép tính sau:

s = mv[compIdx] >> 17

f = Floor(Log2((mv[compIdx] ^ s) | 31)) – 4

mask = (-1 << f) >> 1

round = ((1 << f) + s) >> 2

rmv[compIdx] = Clip3(-(64<<11), (63<<11), (mv[compIdx] + round) & mask)

tương đương với:

cmv[compIdx] = Clip3(-(64<<11), (63<<11), mv[compIdx])

s = mv[compIdx] >> 17

f = Floor(Log2((cmv[compIdx] ^ s) | 31)) – 4

mask = (-1 << f) >> 1

round = ((1 << f) + s) >> 2

rmv[compIdx] = (cmv[compIdx] + round) & mask

Như vậy, trong trường hợp rmv[compIdx] = Clip3(-(64<<11), (63<<11), (mv[compIdx] + giá trị làm tròn) & mặt nạ), phép toán cắt xén có thể được thực hiện sớm hơn trong quá trình suy diễn, nếu muốn (ví dụ: để cải thiện thông lượng tính toán).

Hơn nữa, xin lưu ý rằng trong một ví dụ, mvLXCol có thể được cắt bớt trong quy trình lấy vectơ chuyển động sắp xếp như sau:

- Nếu RefPicListX[refIdxLX] là hình ảnh tham chiếu dài hạn hoặc colPocDiff bằng với currPocDiff, thì mvLXCol được lấy như sau:

  mvLXCol = Clip3(-(1<<17), (1<<17)-1, mvCol)

- Nếu không, mvLXCol được lấy dưới dạng biến thể theo tỷ lệ của vectơ chuyển động mvCol như sau:

  tx = (16384+ (Abs(td) >> 1)) / td

  distScaleFactor = Clip3(–4096, 4095, (tb * tx + 32) >> 6)

  mvLXCol = Clip3(–(1<<17), (1<<17)-1, Sign(distScaleFactor * mvCol) *
  ((Abs(distScaleFactor * mvCol) + 127) >> 8))

  trong đó td và tb được lấy như sau:

  td = Clip3(–128, 127, colPocDiff)

  tb = Clip3(–128, 127, currPocDiff)

Tham chiếu một lần nữa đến HÌNH 6, như minh họa trong HÌNH 6, bộ xử lý dự đoán liên khung 214 có thể nhận khối video được tái tạo thông qua bộ lọc 216, có thể là một phần của quy trình lọc trong vòng. Có thể định cấu hình Bộ lọc 216 để thực hiện quá trình lọc tách khối và/hoặc lọc Bù độ lệch tương thích mẫu (SAO). Tách khối đề cập đến quá trình làm phẳng biên của các khối video được tái tạo (ví dụ: làm cho người xem ít cảm nhận được biên hơn). Lọc SAO là một ánh xạ biên độ phi tuyến tính có thể được dùng để cải thiện việc tái tạo bằng cách thêm phần bù vào dữ liệu video tái tạo. Bộ mã hóa entropy 218 nhận các hệ số biến đổi lượng tử hóa và dữ liệu cú pháp dự đoán (tức là dữ liệu dự đoán nội ảnh, dữ liệu dự đoán chuyển động, dữ liệu QP, v.v.). Có thể định 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 định cấu hình Bộ mã hóa entropy 218 để xuất chuỗi bit tương thích, tức là chuỗi bit mà từ đó bộ giải mã video có thể nhận và tái tạo dữ liệu video. Theo cách này, bộ mã hóa video

200 đại diện cho thiết bị minh họa được định cấu hình để xác định vectơ chuyển động chính xác tuyệt đối để tạo một dự đoán cho khối video trong hình ảnh đầu tiên, lưu trữ vectơ chuyển động ở độ chính xác thấp hơn và tạo ứng viên dự đoán vectơ chuyển động cho một khối video trong hình ảnh thứ hai từ vectơ chuyển động được lưu trữ.

HÌNH 7 là một sơ đồ khối minh họa ví dụ về bộ giải mã video có thể được định cấu hình để giải mã dữ liệu video theo một hoặc nhiều kỹ thuật của sáng chế này. Trong một ví dụ, có thể định cấu hình bộ giải mã video 300 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 300 có thể hoạt động theo cách tương hỗ với bộ mã hóa video 200 được mô tả ở trên. Có thể định cấu hình Bộ giải mã video 300 để thực hiện giải mã dự đoán nội ảnh và dự đoán liên khung và như vậy, có thể được gọi là bộ giải mã lai. Trong ví dụ minh họa trên HÌNH 7 bộ giải mã video 300 bao gồm bộ giải mã entropy 302, bộ lượng tử hóa ngược 304, bộ xử lý biến đổi ngược 306, bộ xử lý dự đoán nội ảnh 308, bộ xử lý dự đoán liên khung 310, bộ cộng 312, bộ lọc 314 và bộ đệm tham chiếu 316. Có thể định cấu hình Bộ giải mã video 300 để giải mã dữ liệu video theo cách nhất quán với hệ thống mã hóa video, hệ thống này có thể triển khai một hoặc nhiều khía cạnh của chuẩn mã hóa video. Xin lưu ý rằng, mặc dù bộ giải mã video ví dụ 300 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 giải mã video 300 và/hoặc các thành phần phụ của chúng đối với một kiếntrúc phần cứng hoặc phần mềm cụ thể. Có thể hiện thực hóa Các chức năng của bộ giải mã video 300 bằng cách triển khai kết hợp phần cứng, chương trình cơ sở và/hoặc phần mềm bất kỳ.

Như minh họa trong HÌNH 7, bộ giải mã entropy 302 nhận một chuỗi bit được mã hóa entropy. Có thể định cấu hình Bộ giải mã entropy 302 để giải mã các phần tử cú pháp lượng tử hóa và các hệ số lượng tử hóa từ chuỗi bit theo một quy trình đối ứng với quy trình mã hóa entropy. Có thể định cấu hình Bộ giải mã entropy 302 để thực hiện giải mã entropy theo bất kỳ kỹ thuật mã hóa entropy nào được mô tả ở trên. Bộ giải mã entropy 302 có thể phân tích cú pháp chuỗi bit được mã hóa theo cách phù hợp với chuẩn mã hóa video. Có thể định cấu hình Bộ giải mã video 300 để phân tích cú pháp một chuỗi bit đã mã hóa trong đó chuỗi bit mã hóa được tạo dựa trên các kỹ thuật được mô tả ở trên. Bộ lượng tử hóa ngược 304 nhận các hệ số biến đổi lượng tử hóa (tức là giá trị mức) và dữ liệu thông số lượng tử hóa từ bộ giải mã entropy 302. 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 định cấu hình Bộ giải mã video 300 và/hoặc bộ lượng tử hóa ngược 304 để xác định các giá trị QP được sử dụng cho lượng tử hóa ngược dựa trên các giá trị được bộ mã hóa video báo hiệu và/hoặc thông qua các thuộc tính video và/hoặc các thông số mã hóa. Có nghĩa là, bộ lượng tử hóa ngược 304 có thể hoạt động theo cách tương hỗ với bộ lượng tử hóa hệ số 206 được mô tả ở trên. Có thể định cấu hình Bộ lượng tử hóa ngược 304 để áp dụng lượng tử hóa ngược. Có thể định cấu hình Bộ xử lý biến đổi ngược 306 để thực hiện biến đổi ngược nhằm tạo ra dữ liệu tái tạo dư. Các kỹ thuật tương ứng được thực hiện bởi bộ lượng tử hóa ngược 304 và bộ xử lý biến đổi ngược 306 có thể tương tự như các kỹ thuật được thực hiện bởi bộ xử lý lượng tử hóa ngược 208 được mô tả ở trên. Có thể định cấu hình Bộ xử lý biến đổi ngược 306 để áp dụng DCT ngược, DST ngược, biến đổi số nguyên nghịch đảo, 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 minh họa trong HÌNH 7, dữ liệu tái tạo dư có thể được cung cấp cho bộ cộng 312. Bộ cộng 312 có thể thêm dữ liệu được tái tạo dư vào khối video dự đoán và tạo dữ liệu video đã tái tạo.

Như được mô tả ở trên, 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 định cấu hình Bộ xử lý dự đoán nội ảnh 308 để nhận các phần tử cú pháp dự đoán nội ảnh và truy xuất khối video dự đoán từ bộ đệm tham chiếu 316. Bộ đệm tham chiếu 316 có thể bao gồm một thiết bị nhớ được định 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. Trong một ví dụ, bộ xử lý dự đoán nội ảnh 308 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 310 có thể nhận các 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 316. Bộ xử lý dự đoán liên khung 310 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 310 có thể sử dụng bộ lọc nội suy để tính toán các giá trị nội suy cho các pixel số nguyên phụ của khối tham chiếu.

Như được mô tả ở trên, bộ giải mã video 300 có thể phân tích cú pháp một chuỗi bit đã mã hóa, trong đó chuỗi bit mã hóa được tạo dựa trên các kỹ thuật được mô tả ở trên và như đã mô tả ở trên, bộ mã hóa video 200 có thể tạo ra chuỗi bit theo các kỹ thuật dự đoán vectơ chuyển động được mô tả ở trên. Hơn nữa, có thể định cấu hình bộ giải mã video 300 để thực hiện phép dự đoán vectơ chuyển động theo các kỹ thuật được mô tả ở trên. Theo cách này, bộ giải mã video 300 đại diện cho thiết bị minh họa được định cấu hình để xác định vectơ chuyển động chính xác tuyệt đối để tạo một dự đoán cho khối video trong hình ảnh đầu tiên, lưu trữ vectơ chuyển động ở độ chính xác thấp hơn và tạo ứng viên dự đoán vectơ chuyển động cho một khối video trong hình ảnh thứ hai từ vectơ chuyển động được lưu trữ.

Tham chiếu một lần nữa đến HÌNH 7, có thể định cấu hình bộ lọc 314 để thực hiện lọc dữ liệu video đã tái tạo. Ví dụ: có thể định cấu hình bộ lọc 314 để thực hiện tách khối và/hoặc lọc SAO, như được mô tả ở trên đối với bộ lọc 216. Hơn nữa, xin lưu ý rằng trong một số ví dụ, có thể định cấu hình bộ lọc 314 để thực hiện chức năng lọc riêng tùy ý (ví dụ: các cải tiến hình ảnh). Như được minh họa trên HÌNH 7 có thể xuất một khối video tái tạo bằng bộ giải mã video 300.

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, phương tiện có thể đọc được bằng máy tính thường có thể tương ứng với (1) phương tiện 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 định 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.

<Tham chiếu chéo>

Yêu cầu quyền ưu tiên của Đơn xin cấp bằng sáng chế không tạm thời này theo 35 U.S.C. § 119 về Đơn xin cấp bằng sáng chế tạm thời số 62/768,772 ngày 16 tháng 11 năm 2018, Số 62/787,695 ngày 2 tháng 1 năm 2019, Số 62/792,872 ngày 15 tháng 1 năm 2019, Số 62/793,080 ngày 16 tháng 1 năm 2019, Số 62/793,311 ngày 16 tháng 1 năm 2019, Số 62/815,109 ngày 7 tháng 3 năm 2019, được kết hợp toàn bộ vào đơn này bằng viện dẫn.

YÊU CẦU BẢO HỘ

1. Phương pháp thực hiện phép dự đoán vectơ chuyển động để mã hóa dữ liệu video, phương pháp bao gồm:

xác định vectơ chuyển động (mv);

xác định một vectơ chuyển động tròn tương đương với giá trị rmv theo các phương trình sau:

s = mv >> 17;

f = Floor(Log2((mv ^ s) | 31)) – 4;

mask = (-1 << f) >> 1;

round = (1 << f) >> 2;

rmv = (mv + round) & mask; và

tạo một ứng viên dự đoán vectơ chuyển động dựa trên vectơ chuyển động tròn.

2. Phương pháp theo điểm 1, trong đó vectơ chuyển động (mv) thu được là vectơ chuyển động tương ứng với khối mã hóa luma được sắp xếp bên trong một hình ảnh được sắp xếp cụ thể.

3. Phương pháp theo điểm 1, trong đó ứng viên dự đoán vectơ chuyển động được tạo dựa trên vectơ chuyển động tròn bằng cách cắt bớt giá trị trong phạm vi từ -131072 đến 131071.

4. Thiết bị thực hiện dự đoán vectơ chuyển động để mã hóa dữ liệu video bao gồm một hoặc nhiều bộ xử lý được định cấu hình để:

xác định vectơ chuyển động (mv);

xác định vectơ chuyển động tròn tương đương với giá trị rmv theo các phương trình sau:

s = mv >> 17;

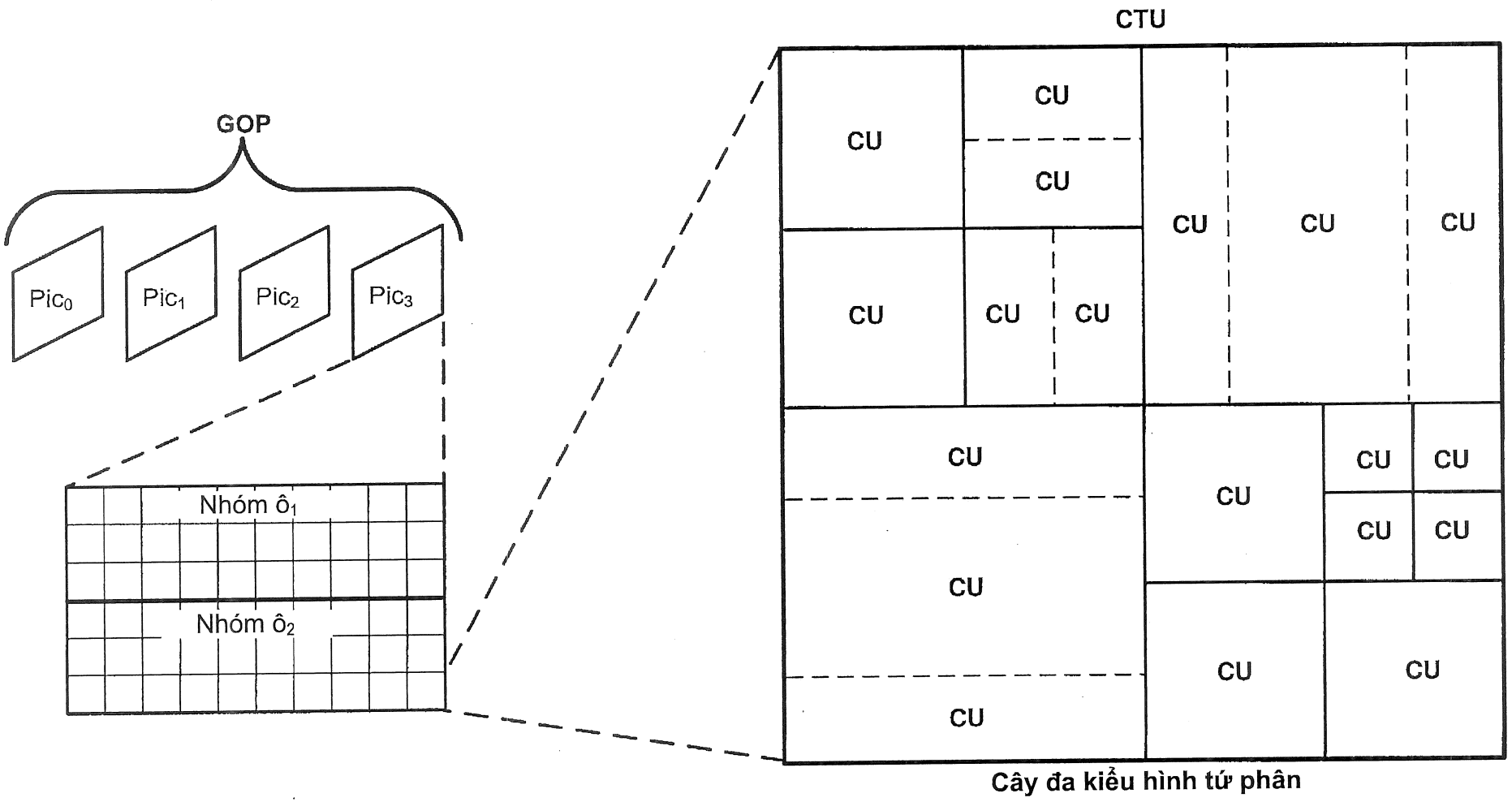
f = Floor(Log2((mv ^ s) | 31)) – 4;

mask = (-1 << f) >> 1;

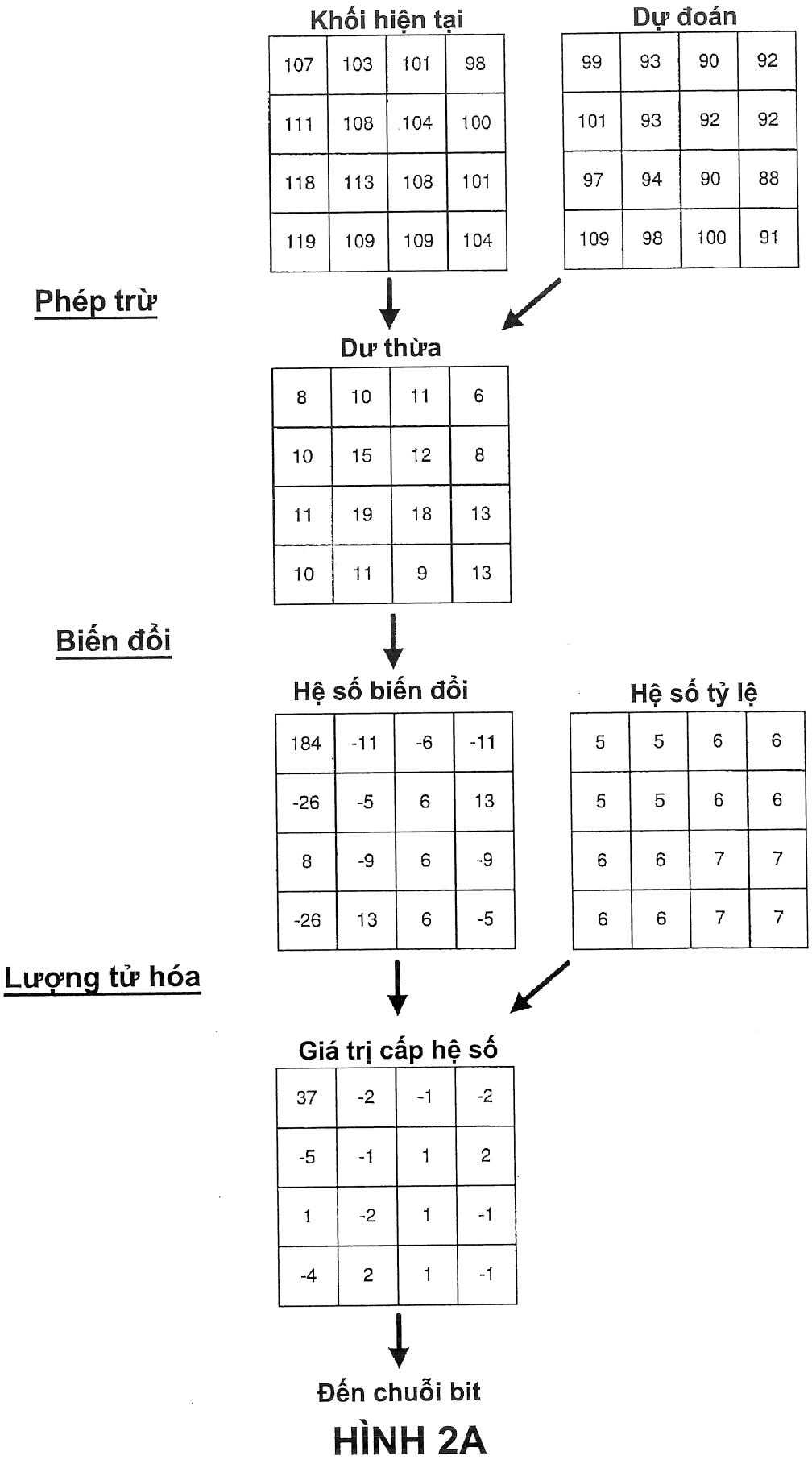
round = (1 << f) >> 2;

rmv = (mv + round) & mask; và

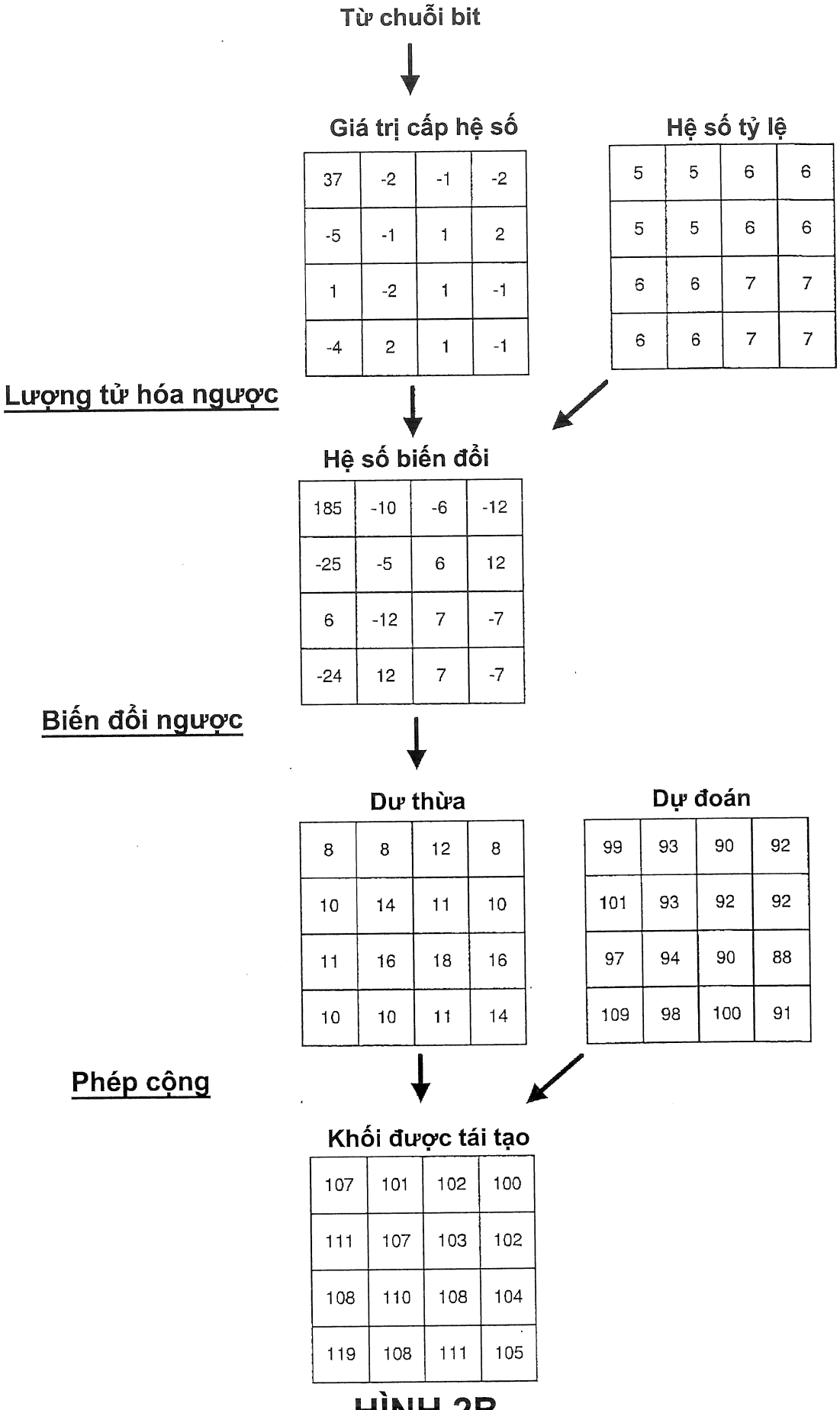
tạo một ứng viên dự đoán vectơ chuyển động dựa trên vectơ chuyển động tròn.


GOP
Pic0
Pic1
Pic2
Pic3
Nhóm ô1
Nhóm ô2
CTU
CU
Cây đa kiểu hình tứ phân

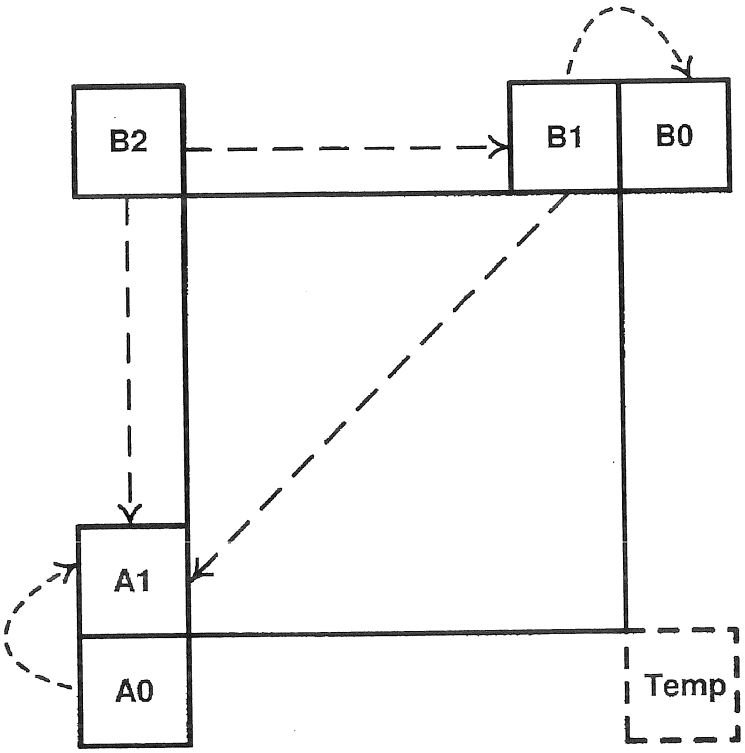
HÌNH 1


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
Dư thừa
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

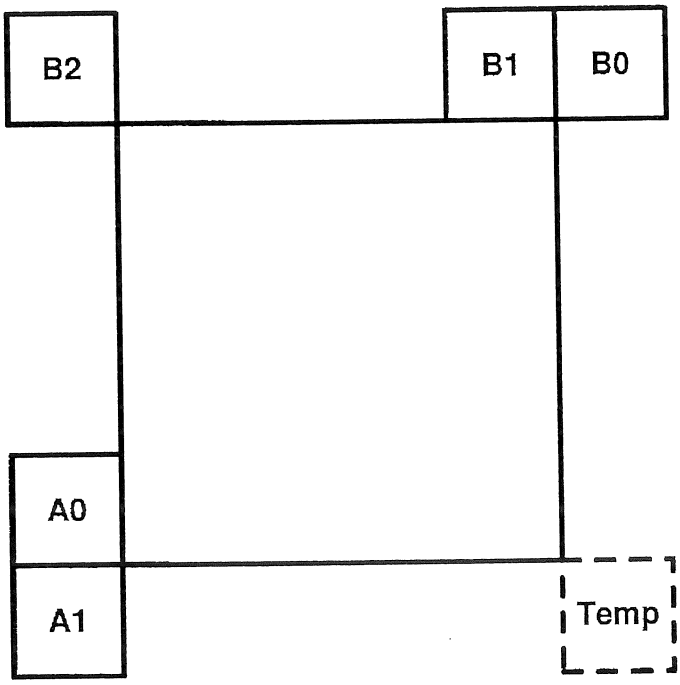
HÌNH 2A


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
Dư thừa
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

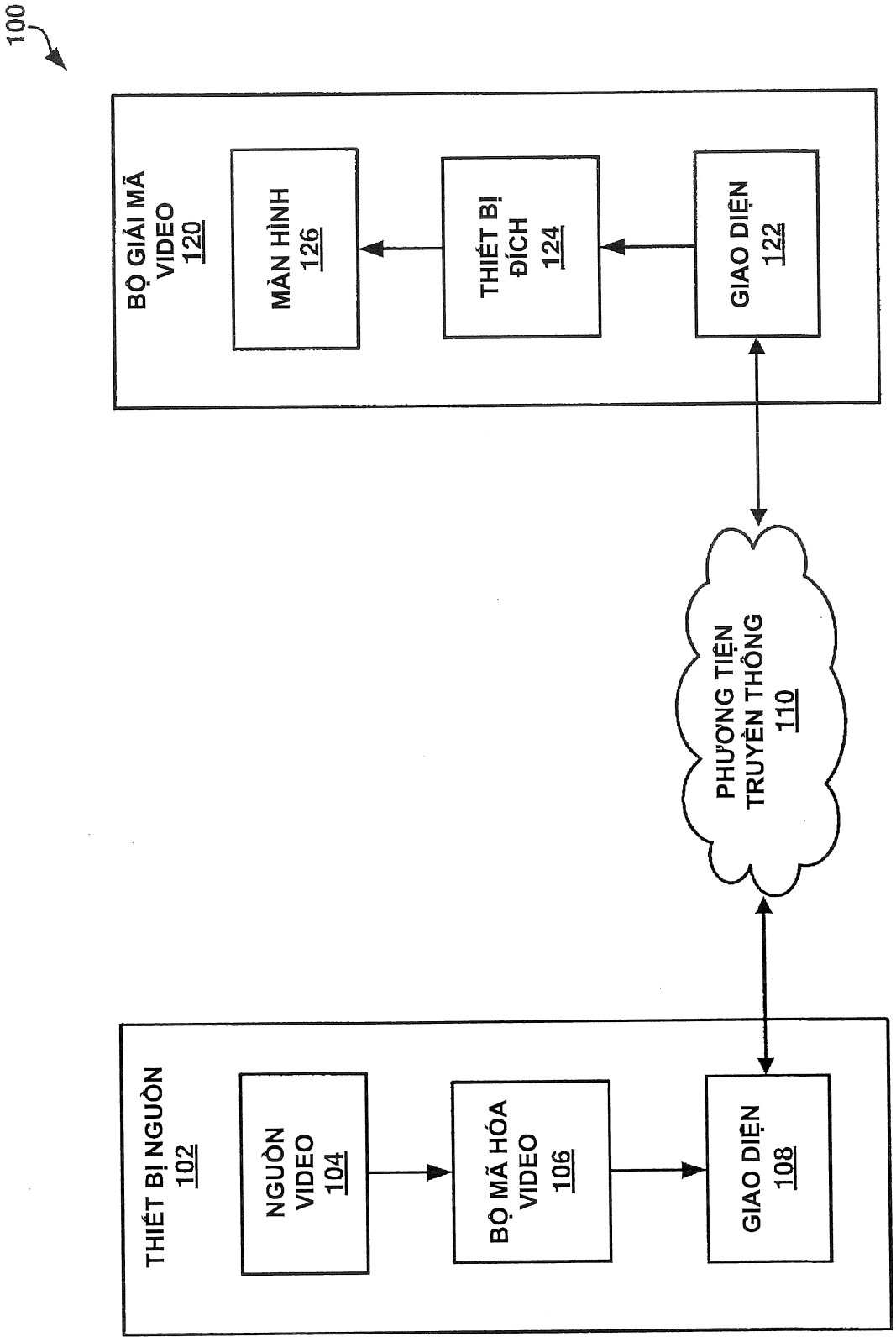
HÌNH 2B



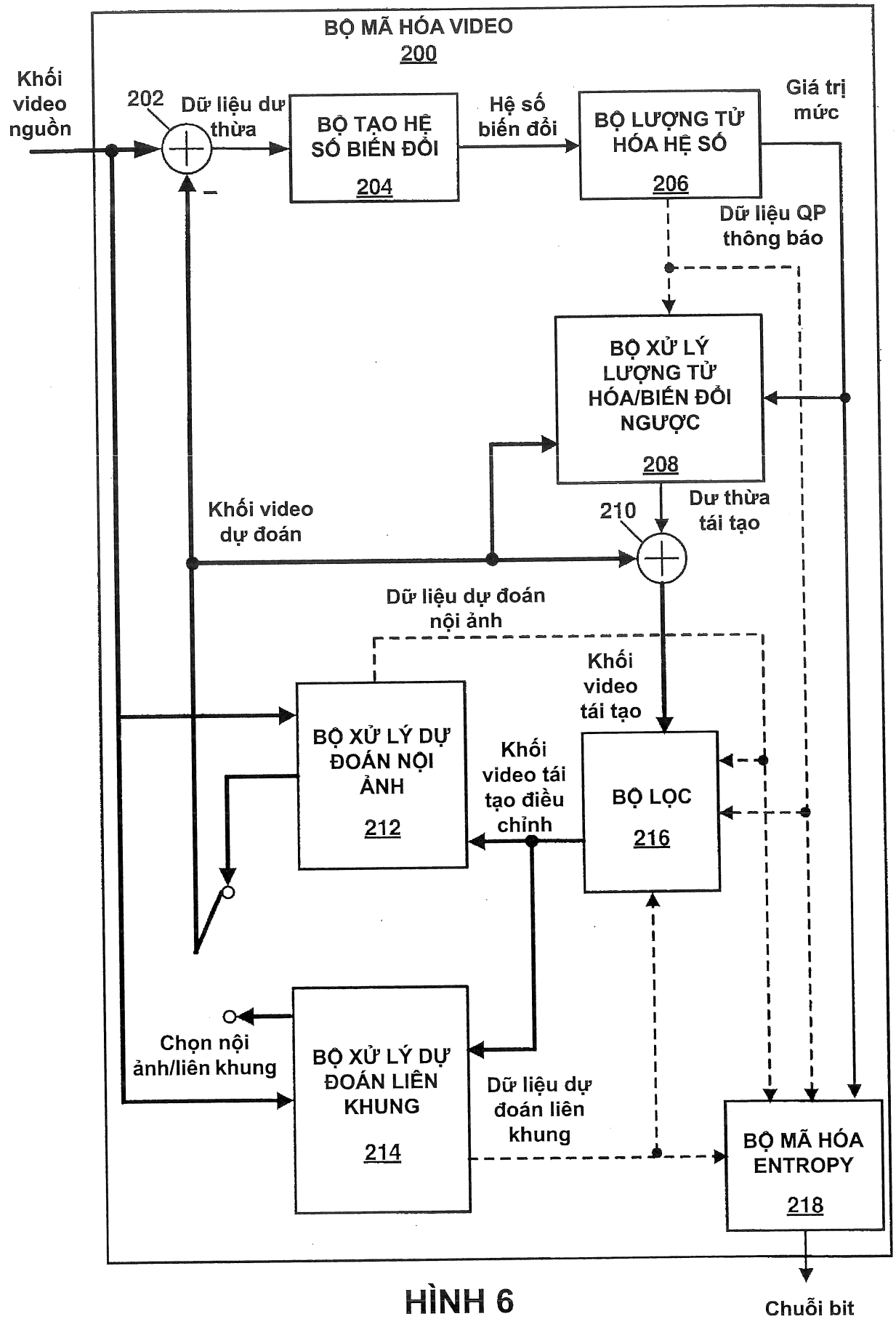
HÌNH 3



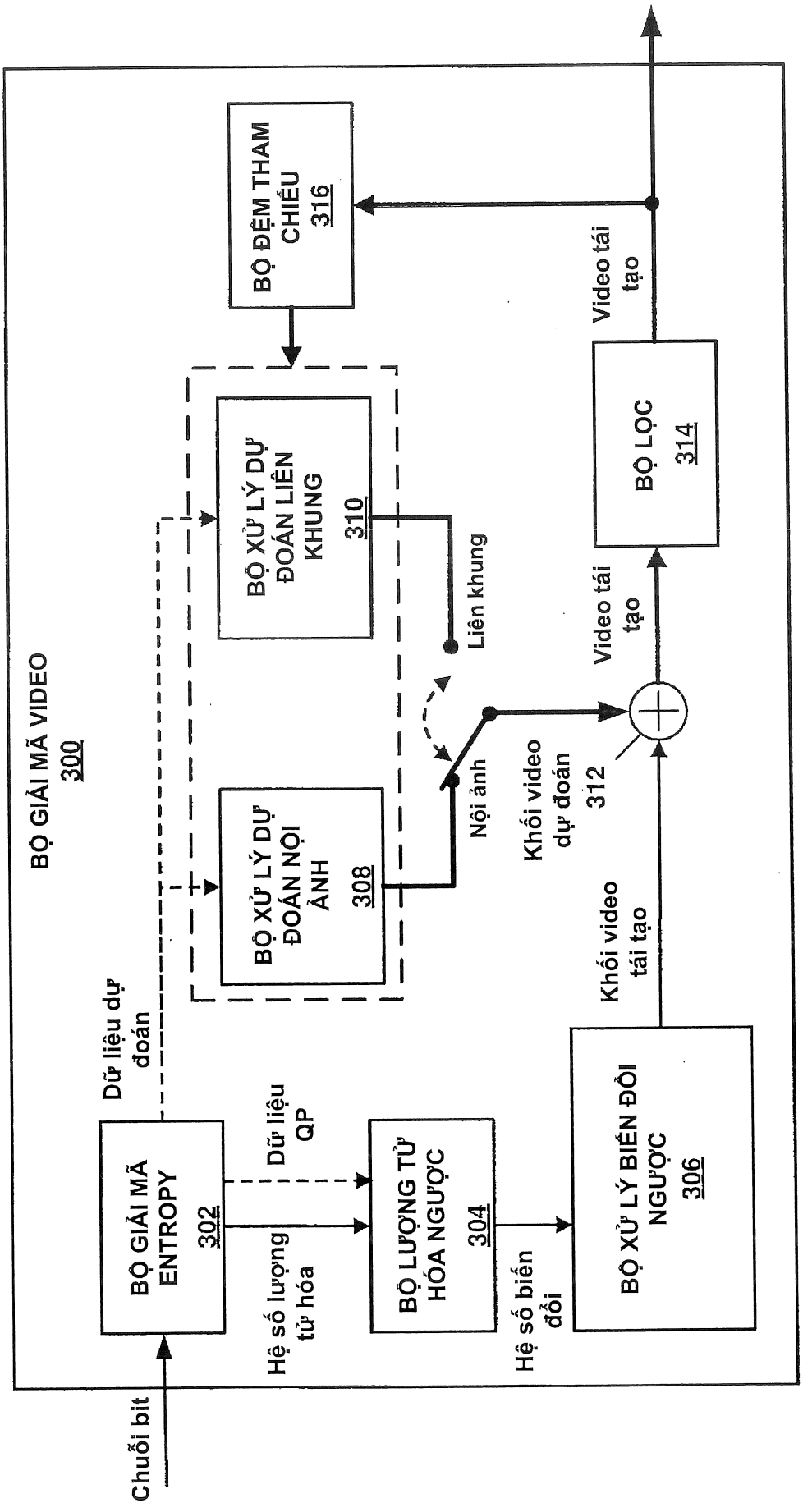
HÌNH 4


100
THIẾT BỊ NGUỒN
102
NGUỒN VIDEO
104
BỘ MÃ HÓA VIDEO
106
GIAO DIỆN
108
PHƯƠNG TIỆN TRUYỀN THÔNG
110
BỘ GIẢI MÃ VIDEO
120
MÀN HÌNH
126
THIẾT BỊ ĐÍCH
124
GIAO DIỆN
122

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 LƯỢNG TỬ HÓA/BIẾN ĐỔI NGƯỢC
208
Khối video dự đoán
210
Dư thừa tái tạo
Dữ liệu dự đoán nội ảnh
Khối video tái tạo
BỘ XỬ LÝ DỰ ĐOÁN NỘI ẢNH
212
Khối video tái tạo điều chỉnh
BỘ LỌC
216
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 GIẢI MÃ VIDEO
300
Chuỗi bit
BỘ GIẢI MÃ ENTROPY
302
Dữ liệu dự đoán
Dữ liệu QP
Hệ số lượng tử hóa
BỘ LƯỢNG TỬ HÓA NGƯỢC
304
Hệ số biến đổi
BỘ XỬ LÝ BIẾN ĐỔI NGƯỢC
306
Khối video tái tạo
BỘ XỬ LÝ DỰ ĐOÁN NỘI ẢNH
308
BỘ XỬ LÝ DỰ ĐOÁN LIÊN KHUNG
310
Liên khung
Nội ảnh
Khối video dự đoán
312
Video tái tạo
BỘ LỌC
314
Video tái tạo
BỘ ĐỆM THAM CHIẾU
316

HÌNH 7