(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) **1-0021721**

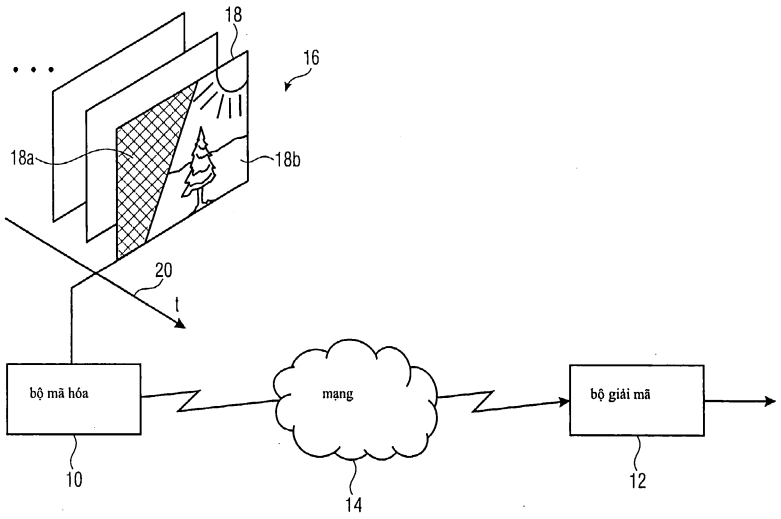
(51)[7] **H04N 7/26**

(13) **B**

(21) 1-2015-00290
(22) 01.07.2013
(86) PCT/EP2013/063853 01.07.2013
(87) WO2014/001573 03.01.2014
(30) 61/666,185 29.06.2012 US
(45) 25.09.2019 378
(43) 25.11.2015 332
(73) GE Video Compression, LLC (US)
8 Southwoods Boulevard, Albany, New York 12211, United States of America
(72) SCHIERL, Thomas (DE), GEORGE, Valeri (DE), HENKEL, Anastasia (DE), MARPE, Detlev (DE), GRUENEBERG, Karsten (DE), SKUPIN, Robert (DE)
(74) Công ty Luật TNHH AMBYS Hà Nội (AMBYS HANOI)

(54) BỘ MÃ HÓA, BỘ GIẢI MÃ, PHƯƠNG PHÁP MÃ HÓA VÀ PHƯƠNG PHÁP GIẢI MÃ DÒNG DỮ LIỆU VIDEO

(57) Sáng chế đề cập đến bộ mã hóa, bộ giải mã, phương pháp mã hóa và phương pháp giải mã dòng dữ liệu video, cụ thể là đề cập đến thông tin định thời gian phục hồi bộ giải mã, thông tin vùng quan tâm (region of interest - ROI) và/hoặc thông tin nhận dạng ô được chuyển bên trong dòng dữ liệu video ở mức cho phép truy cập dễ dàng bởi các thiết bị mạng như MANE hoặc bộ giải mã. Để đạt đến mức này, thông tin của các loại này được chuyển trong dòng dữ liệu video nhờ các gói được đặt rải rác trong các gói của các đơn vị truy cập của dòng dữ liệu video. Theo một phương án, các gói được đặt rải rác thuộc loại gói loại bỏ được, cụ thể, sự loại bỏ các gói được đặt rải rác này duy trì khả năng của bộ giải mã bao hàm hoàn toàn nội dung video được chuyển qua dòng dữ liệu video.



Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến các khái niệm dòng dữ liệu video, cụ thể, các khái niệm này có lợi khi đề cập đến các ứng dụng độ trễ thấp.

Tình trạng kỹ thuật của sáng chế

HEVC [2] cho phép các phương tiện khác nhau tạo tín hiệu cú pháp mức cao cho lớp áp dụng. Các phương tiện này là đoạn đầu đơn vị NAL, các tập hợp tham số và các thông báo Thông tin nâng cao bổ sung (Supplemental Enhancement Information - SEI). Các thông báo SEI không được sử dụng trong quy trình giải mã. Các phương tiện khác của việc tạo tín hiệu cú pháp mức cao bắt nguồn từ các đặc tả giao thức vận chuyển tương ứng như Giao thức vận chuyển MPEG2 [3] hoặc Giao thức vận chuyển thời gian thực [4], và các đặc tả riêng cho tải trọng của chúng, ví dụ các khuyến nghị cho H.264/AVC [5], mã hóa video có thể mở rộng được (scalable video coding - SVC) [6] hoặc HEVC [7]. Các giao thức vận chuyển này có thể đưa vào sự tạo tín hiệu mức cao sử dụng các cấu trúc và cơ chế tương tự như sự tạo tín hiệu mức cao của sự đặc tả bộ mã hóa-bộ giải mã lớp áp dụng tương ứng, ví dụ HEVC [2]. Một ví dụ của sự tạo tín hiệu này là đơn vị NAL thông tin có khả năng mở rộng nội dung tải hữu ích (Payload Content Scalability Information - PACSI) như được mô tả trong [6] cung cấp thông tin bổ sung cho lớp vận chuyển.

Đối với các tập hợp tham số, HEVC bao gồm tập hợp tham số video (Video Parameter Set - VPS), nó biên dịch thông tin dòng quan trọng nhất sẽ được sử dụng bởi lớp áp dụng ở vị trí đơn và trung tâm. Trong các phương pháp ban đầu, thông tin này cần được tập hợp lại từ nhiều tập hợp tham số và đoạn đầu đơn vị NAL.

Trước đơn sáng chế này, trạng thái tiêu chuẩn đối với các hoạt động của vùng đệm hình ảnh đã được mã hóa (Coded Picture Buffer - CPB) của bộ giải mã tham khảo giả thuyết (Hypothetical Reference Decoder - HRD), và tất cả cú pháp liên quan được cung cấp trong tập hợp tham số chuỗi (Sequence Parameter Set - SPS)/thông tin sử dụng video (Video Usability Information - VUI), SEI định thời gian ảnh, SEI giai đoạn

đệm cũng như định nghĩa đơn vị giải mã, mô tả ảnh con và cú pháp của các lát phụ thuộc có mặt trong đoạn đầu lát cũng như tập hợp tham số hình ảnh (Picture Parameter Set -PPS), là như sau.

Để cho phép hoạt động CPB độ trễ thấp ở trên mức ảnh con, các hoạt động CPB ảnh con được đề xuất và được tích hợp vào tiêu chuẩn nháp HEVC 7 JCTVC-I1003 [2]. Đặc biệt ở đây, đơn vị giải mã được định nghĩa trong phần 3 của [2] là:

đơn vị giải mã: Đơn vị truy cập hoặc tập con đơn vị truy cập. Nếu SubPicCpbFlag bằng 0, đơn vị mã hóa là đơn vị truy cập. Nếu không, đơn vị giải mã gồm có một hoặc nhiều đơn vị *NAL VCL* trong đơn vị truy cập và các đơn vị *NAL* không phải VCL được kết hợp. Đối với *đơn vị NAL VCL* thứ nhất trong *đơn vị truy cập*, các đơn vị NAL không phải VCL kết hợp và các đơn vị NAL dữ liệu điền đầy, nếu có, theo ngay sau đơn vị NAL VCL thứ nhất và tất cả các đơn vị NAL không phải VCL trong đơn vị truy cập có trước đơn vị NAL VCL thứ nhất. Đối với *đơn vị NAL VCL* không phải là *đơn vị NAL VCL* thứ nhất trong đơn vị truy cập, các đơn vị NAL không phải VCL được kết hợp là *đơn vị* NAL dữ liệu điền đầy, nếu có, theo ngay sau *đơn vị NAL VCL*.

Trong tiêu chuẩn được định ra cho đến thời điểm đó, "Việc định thời gian loại bỏ đơn vị giải mã và giải mã đơn vị giải mã" được mô tả và được thêm vào phụ lục C "Bộ giải mã tham khảo giả thuyết". Để tạo tín hiệu sự định thời gian ảnh con, thông báo SEI giai đoạn đệm và thông báo SEI định thời gian ảnh, cũng như các tham số HRD trong VUI được mở rộng để hỗ trợ các đơn vị giải mã, như các đơn vị ảnh con.

Cú pháp của thông báo SEI giai đoạn đệm của [2] được thể hiện trên Fig.1.

Khi NalHrdBpPresentFlag hoặc VclHrdBpPresentFlag bằng 1, thông báo SEI giai đoạn đệm có thể được kết hợp với đơn vị truy cập bất kỳ trong dòng bit, và thông báo SEI giai đoạn đệm sẽ được kết hợp với từng đơn vị truy cập RAP, và với mỗi đơn vị truy cập được kết hợp với thông báo SEI điểm phục hồi.

Đối với một số ứng dụng, sự có mặt thường xuyên của thông báo SEI giai đoạn đệm có thể được mong muốn.

Giai đoạn đệm được chỉ ra là tập hợp các đơn vị truy cập nằm giữa hai nấc thông

báo SEI giai đoạn đệm theo thứ tự giải mã.

Các ngữ nghĩa học là như sau:

seq_parameter_set_id chỉ ra tập hợp tham số chuỗi chứa các thuộc tính HRD chuỗi. Giá trị của seq_parameter_set_id sẽ bằng với giá trị của seq_parameter_set_id trong tập hợp tham số ảnh được tham khảo bởi ảnh đã được mã hóa đầu tiên được kết hợp với thông báo SEI giai đoạn đệm. Giá trị của seq_parameter_set_id sẽ nằm trong khoảng từ 0 đến 31, bao gồm cả 0 và 31.

rap_cpb_params_present_flag bằng 1 chỉ ra sự có mặt của các phần tử cú pháp initial_alt_cpb_removal_delay[SchedSelIdx] và initial_alt_cpb_removal_delay_offset[SchedSelIdx]. Khi không xuất hiện, giá trị của rap_cpb_params_present_flag được suy ra là bằng 0. Khi ảnh được kết hợp không phải là ảnh CRA hoặc không phải là ảnh BLA, giá trị của rap_cpb_params_present_flag sẽ là bằng 0.

initial_cpb_removal_delay[SchedSelIdx]và **initial_alt_cpb_removal_delay**[SchedSelIdx] chỉ ra các độ trễ loại bỏ CPB ban đầu đối với CPB thứ SchedSelIdx. Các phần tử cú pháp có độ dài tính bằng bit được đưa ra bởi initial_cpb_removal_delay_length_minus1 + 1, và là trong các đơn vị của đồng hồ 90kHz. Các giá trị của các phần tử cú pháp sẽ không bằng 0 và sẽ không vượt quá 90000 * (CpbSize[SchedSelIdx] ÷ BitRate[SchedSelIdx]), sự tương đương thời gian của kích cỡ CPB trong các đơn vị đồng hồ 90kHz.

initial_cpb_removal_delay_offset[SchedSelIdx] và **initial_alt_cpb_removal_delay_offset**[SchedSelIdx] được sử dụng cho CPB thứ SchedSelIdx để chỉ ra thời gian phân phối ban đầu của các đơn vị dữ liệu đã được mã hóa cho CPB. Các phần tử cú pháp có độ dài tính bằng bit được đưa ra bởi initial_cpb_removal_delay_length_minus1 + 1, và là trong các đơn vị của đồng hồ 90 kHz. Các phần tử cú pháp này không được sử dụng bởi các bộ giải mã và có thể được cần chỉ cho bộ lập lịch biểu phân phối (HSS).

Trên toàn bộ chuỗi video đã được mã hóa, tổng của initial_cpb_removal_delay[SchedSelIdx] và initial_cpb_removal_delay_offset[

SchedSelIdx] sẽ là không đổi cho mỗi giá trị của SchedSelIdx, và tổng của initial_alt_cpb_removal_delay[SchedSelIdx] và initial_alt_cpb_removal_delay_offset[SchedSelIdx] sẽ là không đổi cho mỗi giá trị của SchedSelIdx.

Cú pháp thông báo SEI định thời gian ảnh của [2] được thể hiện trên Fig.2.

Cú pháp thông báo SEI định thời gian ảnh là phụ thuộc vào nội dung của tập hợp tham số chuỗi mà tập hợp tham số này có hiệu lực đối với ảnh đã được mã hóa được kết hợp với thông báo SEI định thời gian ảnh. Tuy nhiên, trừ khi thông báo SEI định thời gian ảnh của đơn vị truy cập IDR hoặc BLA được đứng trước bởi thông báo SEI giai đoạn đệm trong cùng đơn vị truy cập, việc kích hoạt tập hợp tham số chuỗi kết hợp (và, đối với các ảnh IDR hoặc BLA mà chúng không phải là ảnh thứ nhất trong dòng bit thì xác định rằng ảnh đã được mã hóa là ảnh IDR hoặc ảnh BLA) không xảy ra cho đến khi giải mã đơn vị NAL lát được mã hóa thứ nhất của ảnh đã được mã hóa. Vì đơn vị NAL lát đã được mã hóa của ảnh đã được mã hóa theo sau thông báo SEI định thời gian ảnh theo thứ tự đơn vị NAL, có thể có các trường hợp mà trong đó nó là cần thiết cho bộ giải mã lưu trữ RBSP chứa thông báo SEI định thời gian ảnh cho đến khi xác định các tham số của tham số chuỗi mà các tham số của chuỗi tham số này sẽ có hiệu lực đối với ảnh đã được mã hóa, và tiếp đó thực hiện phân tích cú pháp thông báo SEI định thời gian ảnh.

Sự có mặt của thông báo SEI định thời gian ảnh trong dòng bit được chỉ ra như sau.

– Nếu CpbDpbDelaysPresentFlag = 1, một thông báo SEI định thời gian ảnh sẽ được xuất hiện trong mỗi đơn vị truy cập của chuỗi video đã được mã hóa.

– Mặt khác (CpbDpbDelaysPresentFlag = 0), không có thông báo SEI định thời gian ảnh nào sẽ được xuất hiện trong bất kỳ đơn vị truy cập nào của chuỗi video đã được mã hóa.

Các ngữ nghĩa học được định nghĩa như sau:

cpb_removal_delay chỉ ra bao nhiêu nhịp đồng hồ để đợi sau khi loại bỏ khỏi CPB của đơn vị truy cập được kết hợp với thông báo SEI giai đoạn đệm gần đây nhất

trong đơn vị truy cập có trước trước khi loại bỏ khỏi bộ đệm dữ liệu đơn vị truy cập được kết hợp với thông báo SEI định thời gian ảnh. Giá trị này cũng được sử dụng để tính toán thời gian đến khả thi sớm nhất của dữ liệu đơn vị truy cập trong CPB đối với HSS. Phần tử cú pháp có mã chiều dài cố định mà chiều dài của nó tính bằng bit được nêu ra bởi cpb_removal_delay_length_minus1 + 1. cpb_removal_delay là phần còn lại của bộ đếm modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$.

Giá trị của cpb_removal_delay_length_minus1 xác định độ dài (tính bằng bit) của phần tử cú pháp cpb_removal_delay là giá trị của cpb_removal_delay_length_minus1 đã được mã hóa trong tập hợp tham số chuỗi là có hiệu lực đối với ảnh được mã hóa đầu tiên được kết hợp với thông báo SEI định thời gian ảnh, mặc dù cpb_removal_delay chỉ ra số lượng nhịp đồng hồ đối với thời gian loại bỏ đơn vị truy cập đứng trước chứa thông báo SEI giai đoạn đệm, mà nó có thể là đơn vị truy cập của chuỗi video đã được mã hóa khác.

dpb_output_delay được sử dụng để tính toán thời gian ra DPB của ảnh. Nếu chỉ ra bao nhiêu nhịp đồng hồ để đợi sau khi loại bỏ đơn vị giải mã cuối cùng trong đơn vị truy cập từ CPB trước khi ảnh được giải mã được xuất ra từ DPB.

Ảnh không được loại bỏ khỏi DPB ở thời gian xuất ra của nó khi nó vẫn được đánh dấu là "được sử dụng để tham chiếu ngắn hạn" hoặc "được sử dụng để tham chiếu dài hạn".

Duy nhất một dpb_output_delay được chỉ ra cho ảnh đã được giải mã.

Chiều dài của phần tử cú pháp dpb_output_delay theo bit bởi dpb_output_delay_length_minus1 + 1. Khi sps_max_dec_pic_buffering[max_temporal_layers_minus1] = 0, dpb_output_delay sẽ bằng 0.

Thời gian xuất ra suy ra từ dpb_output_delay của ảnh bất kỳ mà ảnh này được xuất ra từ bộ giải mã phù hợp với việc định thời gian xuất ra sẽ đứng trước thời gian xuất ra được suy ra từ dpb_output_delay của tất cả các ảnh trong chuỗi video đã được mã hóa liên tiếp theo thứ tự giải mã.

Thứ tự xuất ảnh được thiết lập bởi các giá trị của phần tử cú pháp này sẽ là thứ tự

giống như được thiết lập bởi các giá trị của PicOrderCntVal.

Đối với các ảnh không được xuất ra bởi quy trình "va đụng" bởi vì chúng đứng trước, theo thứ tự giải mã, ảnh IDR hoặc BLA với no_output_of_prior_pics_flag = 1 hoặc được suy ra là bằng 1, thời gian xuất ra nhận được từ dpb_output_delay sẽ tăng dần với giá trị tăng dần của PicOrderCntVal tương ứng với tất cả các ảnh nằm trong cùng một chuỗi video đã được mã hóa.

num_decoding_units_minus1 cộng 1 chỉ ra số lượng đơn vị giải mã trong đơn vị truy cập mà thông báo SEI định thời gian ảnh được kết hợp cùng. Giá trị của num_decoding_units_minus1 sẽ nằm trong khoảng từ 0 đến PicWidthInCtbs * PicHeightInCtbs – 1.

num_nalus_in_du_minus1[i **]** cộng 1 chỉ ra số lượng đơn vị NAL trong đơn vị giải mã thứ i của đơn vị truy cập mà thông báo SEI định thời gian ảnh được kết hợp cùng. Giá trị của num_nalus_in_du_minus1[i] sẽ nằm trong khoảng từ 0 đến PicWidthInCtbs * PicHeightInCtbs – 1.

Đơn vị giải mã thứ nhất của đơn vị truy cập gồm có num_nalus_in_du_minus1[0] + 1 thứ nhất kế tiếp các đơn vị NAL theo thứ tự giải mã trong đơn vị truy cập. Đơn vị giải mã thứ i (với i lớn hơn 0) của đơn vị truy cập gồm có các đơn vị NAL nối tiếp num_nalus_in_du_minus1[i] + 1 theo ngay sau đơn vị NAL cuối cùng trong đơn vị giải mã trước của đơn vị truy cập, theo thứ tự giải mã. Sẽ có ít nhất một đơn vị NAL VCL trong mỗi đơn vị giải mã. Tất cả các đơn vị NAL không phải VCL được kết hợp với đơn vị NAL VCL sẽ nằm trong cùng một đơn vị giải mã.

du_cpb_removal_delay[i **]** chỉ ra bao nhiêu nhịp đồng hồ ảnh con chờ đợi sau khi loại bỏ khỏi CPB của đơn vị giải mã thứ nhất trong đơn vị truy cập được kết hợp với thông báo SEI khoảng thời gian đệm gần đây nhất trong đơn vị truy cập đứng trước trước khi loại bỏ khỏi CPB đơn vị giải mã thứ i trong đơn vị truy cập được kết hợp với thông báo SEI định thời gian ảnh. Giá trị này cũng được sử dụng để tính toán thời gian đến khả thi sớm nhất của dữ liệu đơn vị giải mã trong CPB đối với HSS. Phần tử cú pháp có mã chiều dài cố định mà chiều dài theo bit được cho bởi cpb_removal_delay_length_minus1 + 1. du_cpb_removal_delay[i] là phần còn lại

của bộ đếm modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$.

Giá trị của cpb_removal_delay_length_minus1 xác định độ dài (theo bit) của phần tử cú pháp du_cpb_removal_delay[i] là giá trị của cpb_removal_delay_length_minus1 đã được mã hóa trong tập hợp tham số chuỗi là có hiệu lực đối với ảnh được mã hóa được kết hợp với thông báo SEI định thời gian ảnh, mặc dù du_cpb_removal_delay[i] chỉ ra số lượng nhịp đồng hồ ảnh con liên quan đến thời gian loại bỏ đơn vị giải mã thứ nhất trong đơn vị truy cập đứng trước chứa thông báo SEI giai đoạn đệm, mà có thể là đơn vị truy cập của chuỗi video đã được mã hóa khác nhau.

Một số thông tin nằm trong cú pháp VUI của [2]. Cú pháp các tham số VUI của [2] được thể hiện trên Fig.3. Cú pháp các tham số HRD của [2] được thể hiện trên Fig.4. Các ngữ nghĩa học được định nghĩa như sau:

sub_pic_cpb_params_present_flag bằng 1 chỉ ra rằng các tham số độ trễ loại bỏ CPB mức ảnh con xuất hiện và CPB có thể hoạt động ở mức đơn vị truy cập hoặc mức ảnh con. sub_pic_cpb_params_present_flag bằng 0 chỉ ra rằng các tham số độ trễ loại bỏ CPB mức ảnh con không được xuất hiện và CPB hoạt động ở mức đơn vị truy cập. Khi sub_pic_cpb_params_present_flag không xuất hiện, giá trị của nó được suy ra là bằng 0.

num_units_in_sub_tick là số lượng đơn vị thời gian của đồng hồ hoạt động ở tần số time_scale Hz tương ứng với một số gia (được gọi là nhịp đồng hồ ảnh con) của bộ đếm nhịp đồng hồ ảnh con. num_units_in_sub_tick sẽ là lớn hơn 0. Nhịp đồng hồ ảnh con là khoảng thời gian nhỏ nhất có thể được hiện diện trong dữ liệu đã được mã hóa khi sub_pic_cpb_params_present_flag bằng 1.

tiles_fixed_structure_flag bằng 1 chỉ ra rằng mỗi tập hợp tham số ảnh có hiệu lực trong chuỗi video đã được mã hóa có cùng giá trị các phần tử cú pháp num_tile_columns_minus1, num_tile_rows_minus1, uniform_spacing_flag, column_width[i], row_height[i] và loop_filter_across_tiles_enabled_flag, khi xuất hiện. tiles_fixed_structure_flag bằng 0 chỉ ra rằng các phần tử cú pháp ô trong các tập hợp tham số ảnh khác nhau có thể hoặc không thể có cùng giá trị. Khi phần tử cú pháp tiles_fixed_structure_flag không xuất hiện, nó được suy ra là bằng 0.

Sự tạo tín hiệu tiles_fixed_structure_flag bằng 1 là sự bảo đảm cho bộ giải mã rằng mỗi ảnh trong chuỗi video đã được mã hóa có cùng số ô được phân bố theo cùng cách thức mà cách thức này có thể là hữu dụng cho sự phân phối tải làm việc trong trường hợp giải mã đa luồng.

Dữ liệu điền đầy của [2] được tạo tín hiệu sử dụng cú pháp RBSP dữ liệu bộ lọc được thể hiện trên Fig.5.

Bộ giải mã tham khảo giả thuyết của [2] được sử dụng để kiểm tra dòng bit và sự phù hợp bộ giải mã được xác định như sau:

Hai loại dòng bit trải qua kiểm tra sự phù hợp HRD cho Khuyến nghị | Tiêu chuẩn quốc tế này. Loại dòng bit thứ nhất, được gọi là dòng bit loại I, là dòng đơn vị NAL chứa duy nhất các đơn vị NAL VCL và các đơn vị NAL dữ liệu điền đầy cho tất cả các đơn vị truy cập trong dòng bit. Loại dòng bit thứ hai, được gọi là dòng bit loại II, chứa, ngoài các đơn vị NAL VCL và các đơn vị NAL dữ liệu điền đầy cho tất cả các đơn vị truy cập trong dòng bit, ít nhất một trong số sau đây:

– các đơn vị NAL không phải VCL bổ sung khác với các đơn vị NAL dữ liệu điền đầy,

– tất cả các phần tử cú pháp leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes, và trailing_zero_8bits tạo nên dòng byte từ dòng đơn vị NAL.

Fig.6 thể hiện các loại điểm tương hợp dòng bit được kiểm tra bởi HRD của [2].

Hai loại tập hợp tham số HRD (các tham số NAL HRD và các tham số VCL HRD) được sử dụng. Các tập hợp tham số HRD được tạo tín hiệu nhờ thông tin có khả năng sử dụng video, thông tin này là một phần của cấu trúc cú pháp tập hợp tham số chuỗi.

Tất cả các tập hợp tham số chuỗi và các tập hợp tham số ảnh được đề cập đến trong các đơn vị NAL VCL, và các thông báo SEI định thời gian ảnh và khoảng thời gian đệm tương ứng sẽ được truyền đến HRD, đúng lúc, hoặc trong dòng bit, hoặc bằng phương tiện khác.

Mô tả đối với "sự có mặt" của các đơn vị NAL không phải VCL cũng được thỏa

mãn khi các đơn vị NAL đó (hoặc chỉ một số chúng) được truyền đến các bộ giải mã (hoặc đến HRD) bởi các phương tiện khác không được chỉ ra bởi Khuyến nghị | Tiêu chuẩn quốc tế. Cho mục đích đếm bit, chỉ các bit thích hợp là thực sự có mặt trong dòng bit được đếm.

Ví dụ, sự đồng bộ hóa đơn vị NAL không phải VCL, được truyền bởi phương tiện không có mặt trong dòng bit, với các đơn vị NAL có mặt trong dòng bit, có thể đạt được bằng cách chỉ ra hai điểm trong dòng bit, mà giữa hai điểm này đơn vị không phải NAL VCL được có mặt trong dòng bit, có bộ mã hóa quyết định truyền nó trong dòng bit.

Khi nội dung của đơn vị NAL không phải VCL được truyền để áp dụng bởi phương tiện nào đó không có mặt trong dòng bit, sự thể hiện nội dung đơn vị NAL không phải VCL không được yêu cầu để sử dụng cùng cú pháp được chỉ ra trong phụ lục này.

Lưu ý rằng khi thông tin HRD được chứa trong dòng bit, có thể thay đổi sự phù hợp của dòng bit đối với các yêu cầu của mệnh đề con này chỉ dựa trên thông tin được chứa trong dòng bit. Khi thông tin HRD không xuất hiện trong dòng bit, như là trường hợp cho tất cả các dòng bit loại I "đứng một mình", sự phù hợp có thể chỉ được thay đổi khi dữ liệu HRD được cung cấp bởi một số phương tiện khác không được chỉ ra trong Khuyến nghị | Tiêu chuẩn quốc tế này.

HRD chứa bộ nhớ đệm ảnh đã được mã hóa (coded picture buffer - CPB), quy trình giải mã tức thời, bộ nhớ đệm ảnh đã được giải mã (decoded picture buffer - DPB), và việc cắt đầu ra như được thể hiện trên Fig.7.

Kích thước CPB (số lượng bit) là CpbSize[SchedSelIdx]. Kích thước DPB (số lượng đệm lưu giữ ảnh) cho lớp thời gian X là sps_max_dec_pic_buffering[X] cho từng X nằm trong khoảng từ 0 đến sps_max_temporal_layers_minus1.

Biến SubPicCpbPreferredFlag hoặc được chỉ ra bởi các phương tiện bên ngoài, hoặc khi không được chỉ ra bởi các phương tiện bên ngoài, được thiết lập bằng 0.

Biến SubPicCpbFlag được dẫn ra như sau:

SubPicCpbFlag = SubPicCpbPreferredFlag &&

sub_pic_cpb_params_present_flag

Nếu SubPicCpbFlag là bằng 0, CPB hoạt động ở mức đơn vị truy cập và mỗi đơn vị giải mã là đơn vị truy cập. Nói cách khác, CPB hoạt động ở mức ảnh con và mỗi đơn vị giải mã là tập con của đơn vị truy cập.

HRD hoạt động như sau. Dữ liệu được kết hợp với các đơn vị giải mã đi vào trong CPB theo sự lập biểu đến chi tiết được phân phối bởi HSS. Dữ liệu được kết hợp với mỗi đơn vị giải mã được loại bỏ và được giải mã ngay lập tức bởi quy trình giải mã tức thời tại các thời gian loại bỏ CPB. Mỗi ảnh được giải mã được đặt trong DPB. Ảnh đã giải mã được bỏ khỏi DPB tại cuối thời gian xuất ra DPB hoặc thời gian mà nó không còn được cần cho sự tham chiếu dự báo liên khung nữa.

HRD được khởi tạo như được chỉ ra bởi SEI giai đoạn đệm. Sự định thời gian loại bỏ các đơn vị giải mã khỏi CPB và sự định thời gian xuất ra các ảnh đã được giải mã khỏi DPB được chỉ ra trong thông báo SEI định thời gian ảnh. Tất cả thông tin định thời gian đề cập đến đơn vị giải mã riêng sẽ đến trước thời gian loại CPB của đơn vị giải mã.

HRD được sử dụng để kiểm tra sự tương thích của các dòng bit và các bộ giải mã.

Trong khi sự phù hợp được đảm bảo dưới giả định rằng tất cả các tần số khung và các đồng hồ được sử dụng để tạo ra dòng bit phù hợp một cách chính xác với các giá trị được tạo tín hiệu trong dòng bit, trong hệ thống thực mà từng giá trị trong các giá trị được truyền tín hiệu trong dòng bit này có thể biến đổi từ giá trị đã được tạo tín hiệu hoặc giá trị đã được chỉ ra.

Tất cả phép tính số học được thực hiện với các giá trị thực, do đó không có lỗi làm tròn có thể lan truyền. Ví dụ, số lượng bit trong CPB ngay trước hoặc sau khi loại bỏ đơn vị giải mã không cần thiết là một số nguyên.

Biến t_c được suy ra như sau và được gọi là nhịp đồng hồ:

$$t_c = \text{num\_units\_in\_tick} \div \text{time\_scale}$$

Biến t_{c_sub} được suy ra như sau và được gọi là nhịp đồng hồ ảnh con:

$$t_{c\_sub} = num\_units\_in\_sub\_tick \div time\_scale$$

Những thứ sau đây được chỉ ra để thể hiện các sự ràng buộc:

– Cho đơn vị truy cập n là đơn vị truy cập thứ n theo thứ tự giải mã với đơn vị truy cập thứ nhất là đơn vị truy cập 0.

– Cho ảnh n là ảnh đã được mã hóa hoặc ảnh đã được giải mã của đơn vị truy cập n.

– Để đơn vị giải mã m là đơn vị giải mã thứ m trong thứ tự giải mã với đơn vị giải mã thứ nhất là đơn vị giải mã 0.

Trong [2], cú pháp đoạn đầu lát được cho phép đối với cái được gọi là các lát phụ thuộc.

Fig.8 thể hiện cú pháp đoạn đầu lát của [2].

Các ngữ nghĩa học đoạn đầu lát được định nghĩa như sau:

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

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

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

$$SliceCtbAddrRS = (slice\_address >> SliceGranularity)$$

Biến SliceCbAddrZS, chỉ ra địa chỉ của khối mã hóa thứ nhất trong lát trong tính hạt khối mã hóa nhỏ nhất theo thứ tự quét z, được suy ra như sau.

$$SliceCbAddrZS = slice\_address << ((log2\_diff\_max\_min\_coding\_block\_size - SliceGranularity) << 1)$$

Việc giải mã lát bắt đầu với đơn vị mã hóa lớn nhất có thể nằm ở tọa độ bắt đầu của lát.

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

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

num_entry_point_offsets chỉ ra số lượng phần tử cú pháp entry_point_offset[i] trong đoạn đầu lát. Khi tiles_or_entropy_coding_sync_idc là bằng 1, giá trị của num_entry_point_offsets sẽ nằm trong khoảng từ 0 đến (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) – 1, bao gồm cả hai giá trị đầu mút. Khi tiles_or_entropy_coding_sync_idc là bằng 2, giá trị của num_entry_point_offsets sẽ nằm trong khoảng từ 0 đến PicHeightInCtbs – 1. Khi không có mặt, giá trị của num_entry_point_offsets được suy ra là bằng 0.

offset_len_minus1 cộng 1 chỉ ra chiều dài tính bằng bit, của các phần tử cú pháp entry_point_offset[i].

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

Khi tiles_or_entropy_coding_sync_idc = 1 và num_entry_point_offsets là lớn hơn 0, mỗi tập con sẽ chứa tất cả các bit đã được mã hóa của chính xác một ô, và số

lượng tập con (cụ thể, giá trị của num_entry_point_offsets + 1) sẽ là bằng với hoặc nhỏ hơn so với số lượng ô trong lát.

Khi tiles_or_entropy_coding_sync_idc bằng 1, mỗi lát phải bao gồm tập con của một ô (trong đó trường hợp tạo tín hiệu các điểm vào là không cần thiết) hoặc số nguyên của các ô hoàn thiện.

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

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

Fig.9 thể hiện cú pháp RBSP tập hợp tham số ảnh của [2], các ngữ nghĩa RBSP tập hợp tham số ảnh của [2] được định nghĩa như sau:

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

tiles_or_entropy_coding_sync_idc bằng 0 chỉ ra rằng sẽ có duy nhất một ô trong mỗi ảnh đề cập đến tập hợp tham số ảnh, sẽ không có quy trình đồng bộ hóa riêng nào

cho các biến ngữ cảnh được dẫn ra trước khi giải mã khối cây mã hóa thứ nhất của một hàng các khối cây mã hóa trong mỗi ảnh đề cập đến tập hợp tham số ảnh, và các giá trị của cabac_independent_flag và dependent_slice_flag cho các ảnh đã được mã hóa đề cập đến tập hợp tham số ảnh cả hai sẽ không bằng 1.

Khi cabac_independent_flag và depedent_slice_flag cả hai đều bằng 1 cho lát, lát là lát entrôpi.]

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

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

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

Khi dependent_slice_enabled_flag sẽ bằng 0, tiles_or_entropy_coding_sync_idc sẽ không bằng 3.

Yêu cầu phù hợp dòng bit là giá trị của tiles_or_entropy_coding_sync_idc sẽ là

giống nhau đối với tất cả các tập hợp tham số ảnh được kích hoạt trong chuỗi video đã được mã hóa.

Đối với mỗi lát đề cập đến tập hợp tham số ảnh, khi tiles_or_entropy_coding_sync_idc là bằng 2 và khối mã hóa thứ nhất trong lát không phải là khối mã hóa thứ nhất trong khối cây mã hóa thứ nhất của hàng các khối cây mã hóa, khối mã hóa cuối cùng trong lát sẽ thuộc về cùng hàng các khối cây mã hóa giống như khối cây mã hóa thứ nhất trong lát.

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

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

Khi num_tile_columns_minus1 bằng 0, num_tile_rows_minus1 sẽ không bằng 0.

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

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

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

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

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

Vectơ CtbAddrTStoRS[ctbAddrTS] chỉ ra sự biến đổi từ địa chỉ CTB theo thứ tự quét ô thành địa chỉ CTB theo thứ tự quét mành với chỉ số ctbAddrTS nằm trong khoảng từ 0 đến (picHeightInCtbs * picWidthInCtbs) − 1, bao gồm cả hai giá trị đầu

mút.

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

Các giá trị của colWidth, CtbAddrRStoTS, CtbAddrTStoRS và TileId được dẫn xuất bằng cách dẫn ra quy trình biến đổi quét ô và quét mành CTB như được chỉ ra trong mệnh đề con 6.5.1 với PicHeightInCtbs và PicWidthInCtbs làm thành các đầu vào và đầu ra được gán cho colWidth, CtbAddrRStoTS và TileId.

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

Mảng MinCbAddrZS[x][y], chỉ ra sự biến đổi từ vị trí (x, y) trong các đơn vị của các CB nhỏ nhất thành địa chỉ CB nhỏ nhất theo thứ tự quét z với x nằm trong khoảng từ 0 đến picWidthInMinCbs − 1, bao gồm cả hai giá trị đầu mút, và y nằm trong khoảng từ 0 đến picHeightInMinCbs − 1, bao gồm cả hai giá trị đầu mút, được suy ra bằng cách dẫn ra quy trình khởi tạo mảng thứ tự quét Z như được chỉ ra trong mệnh đề con 6.5.2 với Log2MinCbSize, Log2CtbSize, PicHeightInCtbs, PicWidthInCtbs, và vectơ CtbAddrRStoTS làm thành các đầu vào và đầu ra được gán cho MinCbAddrZS.

loop_filter_across_tiles_enabled_flag bằng 1 chỉ ra rằng các hoạt động lọc trong vòng được thực hiện qua các biên ô. loop_filter_across_tiles_enabled_flag bằng 0 chỉ ra rằng các hoạt động lọc trong vòng không được thực hiện qua các biên ô. Các hoạt động lọc trong vòng bao gồm bộ lọc phân rã khối, độ lệch tương thích mẫu, và các hoạt động bộ lọc vòng tương thích. Khi không có mặt, giá trị của loop_filter_across_tiles_enabled_flag được suy ra là bằng 1.

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

Quy trình suy ra cho sự khả dụng của khối mã hóa với địa chỉ khối mã hóa nhỏ nhất được mô tả như sau:

Các đầu vào cho quy trình này là

- địa chỉ khối mã hóa nhỏ nhất minCbAddrZS theo thứ tự quét z
- địa chỉ khối mã hóa nhỏ nhất hiện thời minCbAddrZS theo thứ tự quét z

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

Lưu ý rằng nghĩa của sự khả dụng được xác định khi quy trình này được dẫn ra.

Lưu ý rằng khối mã hóa bất kỳ, bất kể kích thước, được kết hợp với địa chỉ khối mã hóa nhỏ nhất, là địa chỉ của khối mã hóa với kích thước khối mã hóa nhỏ nhất theo thứ tự quét z.

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

– minCbAddrZS là nhỏ hơn 0.

– minCbAddrZS là lớn hơn currMinCBAddrZS.

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

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

– Nói cách khác, cbAvailable được thiết lập là TRUE.

Quy trình phân tích cú pháp CABAC cho dữ liệu lát [2] là như sau:

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

Các đầu vào cho quy trình này là yêu cầu giá trị của phần tử cú pháp và các giá

trị của các phần tử cú pháp đã được phân tích trước.

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

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

Địa chỉ khối mã hóa nhỏ nhất của khối cây mã hóa chứa khối lân cận không gian T (Fig.10a), ctbMinCbAddrT, được dẫn ra sử dụng vị trí (x0, y0) của mẫu luma đỉnh bên trái của khối cây mã hóa hiện thời như sau.

x = x0 + 2 << Log2CtbSize – 1

y = y0 – 1

ctbMinCbAddrT = MinCbAddrZS[x >> Log2MinCbSize][y >> Log2Min CbSize]

Biến availableFlagT thu được bằng cách dẫn ra quy trình dẫn xuất có thể dùng khối mã hóa với ctbMinCbAddrT làm đầu vào.

Khi bắt đầu phân tích cú pháp cây mã hóa, các bước được đặt thứ tự tiếp theo áp dụng.

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

– Nếu CtbAddrRS bằng slice_address, dependent_slice_flag bằng 1 và entropy_coding_reset_flag = 0, áp dụng phần sau đây.

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

– Quy trình giải mã cho các quyết định nhị phân trước khi kết thúc được dẫn ra, sau đó là quy trình khởi tạo để giải mã số học.

– Mặt khác nếu tiles_or_entropy_coding_sync_idc bằng 2, và CtbAddrRS % PicWidthInCtbs bằng 0, áp dụng phần sau đây.

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

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

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

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

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

Sự phân tích cú pháp các phần tử cú pháp tiến hành như sau.

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

Sự nhị phân hóa cho phần tử cú pháp và chuỗi các bin được phân tích cú pháp xác định dòng quy trình giải mã.

Đối với mỗi hệ số nhị phân của quy trình nhị phân hóa phần tử cú pháp, hệ số nhị phân được ghi chỉ số bởi biến binIdx, chỉ số ngữ cảnh ctxIdx được suy ra.

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

Chuỗi thu được ($b_{0..}b_{binIdx}$) của các hệ số nhị phân được phân tích cú pháp được so sánh với tập hợp các ký tự nhị phân được đưa ra bởi quy trình nhị phân hóa sau khi giải mã mỗi hệ số nhị phân. Khi chuỗi ăn khớp ký tự nhị phân trong tập hợp đã cho, giá trị tương ứng được gán cho phần tử cú pháp.

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

Trong khung thiết kế được mô tả đến giờ vấn đề sau đây bị xuất hiện.

Sự định thời gian của các đơn vị giải mã cần được biết trước khi mã hóa và gửi dữ liệu trong kịch bản độ trễ thấp, ở đó các đơn vị NAL sẽ được gửi ra ngoài bởi bộ mã hóa, trong khi bộ mã hóa vẫn đang mã hóa các phần của ảnh, tức là, các đơn vị giải mã ảnh con khác. Tức là, bởi vì thứ tự đơn vị NAL trong đơn vị truy cập chỉ cho phép các thông báo SEI đứng trước VCL (các đơn vị NAL mã hóa video) trong đơn vị truy cập, nhưng trong kịch bản độ trễ thấp như vậy, các đơn vị NAL không phải VCL cần ở sẵn trên dây, tức là, được gửi ra ngoài, nếu bộ mã hóa bắt đầu mã hóa các đơn vị giải mã. Fig.10b minh họa cấu trúc của đơn vị truy cập như được định nghĩa trong [2]. [2] không chỉ ra đầu chuỗi hoặc dòng, do đó sự có mặt của chúng trong đơn vị truy cập là tạm thời.

Hơn nữa, số lượng các đơn vị NAL được kết hợp với ảnh con cũng cần được biết sớm hơn trong kịch bản độ trễ thấp, như thông báo SEI định thời gian ảnh chứa thông tin này và phải được biên soạn và được gửi đi trước khi bộ mã hóa bắt đầu mã hóa ảnh thực sự. Tác giả bất tắc dĩ chèn các đơn vị NAL dữ liệu điền đầy, với khả năng không có dữ liệu điền đầy để tương thích với số lượng đơn vị NAL, như được tạo tín hiệu trên mỗi đơn vị giải mã trong SEI định thời gian ảnh, cần phương tiện để tạo tín hiệu thông tin này trên mức ảnh con. Thông tin về mức ảnh con giữ sự định thời gian ảnh con, nó hiện tại được cố định tại đơn vị truy cập bởi các tham số được đưa ra trong thông báo SEI định thời gian.

Hơn nữa các thiếu sót của bản mô tả phác thảo [2] bao gồm nhiều sự tạo tín hiệu mức ảnh con, sự tạo tín hiệu được yêu cầu cho các ứng dụng cụ thể, như tạo tín hiệu ROI hoặc tạo tín hiệu các kích thước ô.

Các vấn đề đã nêu trên là không riêng gì đối với tiêu chuẩn HEVC. Đúng hơn, vấn đề này cũng xảy ra kết hợp với các bộ mã hóa bộ giải mã video khác. Fig.11 thể hiện, tổng quát hơn, ngữ cảnh truyền dẫn video trong đó cặp bộ mã hóa 10 và bộ giải mã 12 được kết nối thông qua mạng 14 để truyền video 16 từ bộ mã hóa 10 đến bộ giải mã 12 ở độ trễ đầu đến cuối ngắn. Vấn đề như được chỉ ra bên trên là sau đây. Bộ mã hóa 10 mã hóa chuỗi các khung 18 của video 16 theo thứ tự giải mã đã cho mà thứ tự này về cơ bản, không nhất thiết, đi theo thứ tự tái tạo 20 các khung 18, và trong mỗi

khung 18 di chuyển qua vùng khung của các khung 18 theo một số cách thức được định, ví dụ, theo cách quét mành mà có hoặc không có sự chia thành ô của các khung 18. Thứ tự giải mã điều khiển độ khả dụng của thông tin cho các kỹ thuật mã hóa được sử dụng bởi bộ mã hóa 10 ví dụ như, mã hóa dự báo và/hoặc entrôpi, tức là, tính khả dụng của thông tin đề cập đến các phần lân cận không gian và/hoặc thời gian của video 16 có thể làm thành cơ sở cho sự dự báo hoặc lựa chọn ngữ cảnh. Dù là bộ mã hóa 10 có thể sử dụng phép xử lý song song để mã hóa các khung 18 của video 16, bộ mã hóa 10 nhất thiết cần thời gian nào đó để mã hóa một khung 18 nhất định, như khung hiện thời. Fig.11, ví dụ, minh họa thời điểm ở đó bộ mã hóa 10 đã kết thúc mã hóa phần 18a của khung hiện thời 18, trong khi phần 18b khác của khung hiện thời 18 vẫn chưa được mã hóa. Vì bộ mã hóa 10 vẫn chưa mã hóa phần 18b, bộ mã hóa 10 không thể dự báo tốc độ bit khả dụng để mã hóa khung hiện thời 18 được phân bố không gian trên khung hiện thời 18 như thế nào để đạt được điều kiện tốt nhất, ví dụ, về tốc độ/độ méo. Do đó, bộ mã hóa 10 chỉ có hai lựa chọn: hoặc bộ mã hóa 10 ước lượng sự phân bố gần tối ưu của tốc độ bit khả dụng cho khung hiện thời 18 trên các lát mà khung hiện thời 18 được chia nhỏ không gian trước thành các lát này, do đó chấp nhận rằng sự ước lượng có thể là sai, hoặc bộ mã 10 hoàn thành mã hóa khung hiện thời 18 trước khi truyền các gói chứa các lát từ bộ mã hóa 10 đến bộ giải mã 12. Trong trường hợp bất kỳ, để có thể tận dụng sự truyền dẫn bất kỳ các gói lát của khung 18 đã được mã hóa hiện tại trước khi hoàn thành mã hóa nó, mạng 14 được thông báo tốc độ bit được gắn với mỗi gói lát này dưới dạng các thời gian phục hồi bộ nhớ đệm cho ảnh đã được mã hóa. Tuy nhiên, như chỉ ra bên trên, mặc dù bộ mã hóa 10, phù hợp với phiên bản hiện tại của HEVC, có thể thay đổi tốc độ bit được phân bố trên các khung 18 bằng cách định ra các thời gian phục hồi bộ nhớ đệm bộ giải mã cho các vùng ảnh con riêng biệt, bộ mã hóa 10 cần truyền hoặc gửi đi thông tin này thông qua mạng 14 đến bộ giải mã 12 tại lúc bắt đầu mỗi đơn vị truy cập tập hợp tất cả dữ liệu đề cập đến khung hiện thời 18, do đó thúc giục bộ mã hóa 10 lựa chọn trong số hai phương án vừa nêu ra, một phương án dẫn đến độ trễ thấp nhưng tốc độ/độ méo lại kém hơn, phương án kia dẫn đến tốc độ/độ méo tối ưu, tuy nhiên độ trễ đầu đến cuối bị tăng lên.

Do đó, cho đến khi không có bộ mã hóa-bộ giải mã video nào cho phép đạt được độ trễ thấp để bộ mã hóa được phép bắt đầu truyền dẫn các gói đề cập đến các phần

18a của khung hiện thời trước khi mã hóa phần 18b còn lại của khung hiện thời, bộ giải mã có thể khai thác sự truyền dẫn trung gian này của các gói đề cập đến các phần 18a mở đầu bởi mạng 16, bộ giải mã tuân theo sự định thời gian phục hồi bộ nhớ đệm giải mã được chuyển trong dòng dữ liệu video được gửi từ bộ mã hóa 12 đến bộ giải mã 14. Các ứng dụng, ví dụ, tận dụng độ trễ thấp này bao gồm các ứng dụng công nghiệp như, ví dụ, giám sát công việc hoặc chế tạo cho các mục đích tự động hóa hoặc kiểm tra hoặc tương tự. Cho đến bây giờ, cũng không có giải pháp thỏa đáng nào để thông báo phía giải mã về sự kết hợp của gói với các ô mà khung hiện thời được cấu trúc thành các ô này, và quan tâm đến các vùng (vùng quan tâm) của khung hiện thời do đó các thiết bị mạng trung gian trong mạng 16 được phép lấy thông tin này từ dòng dữ liệu mà không phải kiểm tra kỹ phía bên trong của các gói, cụ thể cú pháp các lát.

Tài liệu tham khảo

[1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. Circuits Syst. Video Technol., vol. 13, N7, July 2003.

[2] JCT-VC, "High-Efficiency Video Coding (HEVC) text specification Working Draft 7", JCTVC-I1003, May 2012.

[3] ISO/IEC 13818-1: MPEG-2 Systems specification.

[4] IETF RFC 3550 – Real-time Transport Protocol.

[5] Y.-K. Wang et al. ,"RTP Payload Format for H.264 Video", IETF RFC 6184, http://tools.ietf.org/html/

[6] S. Wenger et al., „RTP Payload Format for Scalable Video Coding", IETF RFC 6190, http://tools.ietf.org/html/rfc6190

[7] T. Schierl et al., "RTP Payload Format for High Efficiency Video Coding", IETF internet draft, http://datatracker.ietf.org/doc/draft-schierl-payload-rtp-h265/

Bản chất kỹ thuật của sáng chế

Theo đó, mục đích của sáng chế là đề xuất khái niệm mã hóa dòng dữ liệu video

mà khái niệm này là hiệu quả hơn để cho phép độ trễ đầu đến cuối thấp và/hoặc thực hiện nhận dạng các phần của dòng dữ liệu cho vùng quan tâm hoặc cho các ô nhất định dễ dàng hơn.

Mục đích này đã đạt được nhờ đối tượng yêu cầu bảo hộ trong các điểm yêu cầu bảo hộ độc lập kèm theo.

Một ý tưởng mà đơn được dựa trên ý tưởng này, đó là thông tin định thời gian phục hồi bộ giải mã, thông tin ROI và thông tin nhận dạng ô được chuyển trong dòng dữ liệu video tại mức mà mức này cho phép truy cập dễ dàng bởi các thiết bị mạng như MANE hoặc các bộ giải mã và, để đạt được mức này, thông tin của các loại như vậy nên được chuyển trong dòng dữ liệu video bởi các gói được đặt rải rác bên trong các gói của các đơn vị truy cập dòng dữ liệu video. Theo một phương án, các gói được đặt rải rác thuộc loại gói loại bỏ được, tức là, sự loại bỏ các gói được đặt rải rác này duy trì khả năng của bộ giải mã bao hàm hoàn toàn nội dung video được chuyển qua dòng dữ liệu video.

Theo một khía cạnh của sáng chế, việc đạt được độ trễ đầu đến cuối thấp được kết xuất hiệu quả hơn bằng cách sử dụng các gói được đặt rải rác để chuyển thông tin về các thời gian phục hồi bộ nhớ đệm bộ giải mã cho các đơn vị giải mã được tạo ra bởi các gói tải trọng mà các gói tải trọng này đi theo gói điều khiển định thời gian tương ứng trong dòng dữ liệu video nằm trong đơn vị truy cập hiện thời. Bằng cách này, bộ mã hóa được phép xác định các thời gian phục hồi bộ nhớ đệm bộ giải mã chạy trong khoảng thời gian mã hóa khung hiện thời, do đó có thể, trong khi mã hóa khung hiện thời, xác định một cách liên tục tốc độ bit được dùng thực sự cho phần của khung hiện thời đã được mã hóa thành các gói tải trọng và được truyền, hoặc được gửi, được đặt trước với các gói điều khiển định thời gian, mặt khác, và do đó thích hợp với sự phân bố tốc độ bit còn lại có sẵn cho khung hiện thời trên phần còn lại của khung hiện thời mà vẫn chưa được mã hóa. Bằng cách này, tốc độ bit khả dụng được khai thác một cách hiệu quả và tuy nhiên độ trễ được duy trì ngắn hơn do đó bộ mã hóa không cần đợi hoàn thành mã hóa khung hiện thời một cách trọn vẹn.

Theo khía cạnh nữa của sáng chế, các gói được đặt rải rác bên trong gói tải trọng của đơn vị truy cập được khai thác để chuyển thông tin trên vùng quan tâm, do đó cho

phép như được chỉ ra bên trên sự truy cập dễ dàng vào thông tin này bởi các thiết bị mạng vì chúng không phải kiểm tra các gói tải trọng trung gian. Hơn nữa, bộ mã hóa vẫn là tự do để xác định các gói thuộc về ROI khi mã hóa khung hiện thời trong quy trình hoạt động mà không phải xác định trước sự chia nhỏ khung hiện thời thành các phần con và các gói tải trọng tương ứng. Hơn nữa, theo phương án mà theo đó các gói được đặt rải rác là loại gói có thể loại bỏ, thông tin ROI có thể không được xét đến bởi người nhận dòng dữ liệu video không quan tâm đến thông tin ROI, hoặc không thể xử lý thông tin ROI.

Các sự suy xét tương tự được khai thác trong đơn theo khía cạnh khác mà theo khía cạnh này các gói được đặt rải rác chuyển thông tin lên trên ô mà các gói nhất định nằm trong đơn vị truy cập thuộc về ô này.

Mô tả vắn tắt các hình vẽ

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

Các hình vẽ từ Fig.1 đến Fig.10b thể hiện trạng thái hiện tại của HEVC với Fig.1 thể hiện cú pháp thông báo SEI giai đoạn đệm, Fig.2 thể hiện cú pháp thông báo SEI định thời gian ảnh, Fig.3 thể hiện cú pháp tham số VUI, Fig.4 thể hiện cú pháp tham số HRD, Fig.5 thể hiện cú pháp RBSP dữ liệu điền đầy, Fig.6 thể hiện cấu trúc của các dòng byte và các dòng đơn vị NAL cho các sự kiểm tra tính tương hợp HRD, Fig.7 thể hiện mô hình đệm HRD, Fig.8 thể hiện cú pháp đoạn đầu lát, Fig.9 thể hiện cú pháp RBSP tập hợp tham số ảnh, Fig.10a thể hiện giản đồ minh họa khối cây mã hóa lân cận không gian T có thể được sử dụng để dẫn quy trình suy ra khả dụng khối cây mã hóa đối với khối cây mã hóa hiện thời và Fig.10b thể hiện sự định nghĩa cấu trúc đơn vị truy cập;

Fig.11 thể hiện giản lược cặp bộ mã hóa và bộ giải mã được kết nối nhờ mạng để minh họa các vấn đề xảy ra trong việc truyền dòng dữ liệu video;

Fig.12 thể hiện sơ đồ khối giản lược của bộ mã hóa theo phương án sử dụng các gói điều khiển định thời gian;

Fig.13 thể hiện lưu đồ minh họa chế độ hoạt động của bộ mã hóa của Fig.12 theo một phương án;

Fig.14 thể hiện sơ đồ khối của phương án của bộ giải mã để giải thích chức năng của nó kết nối với dòng dữ liệu video được tạo ra bởi bộ mã hóa theo Fig.12;

Fig.15 thể hiện sơ đồ khối giản lược minh họa bộ mã hóa, thiết bị mạng và dòng dữ liệu video theo phương án khác nữa sử dụng các gói ROI;

Fig.16 thể hiện sơ đồ khối giản lược minh họa bộ mã hóa, thiết bị mạng và dòng dữ liệu video theo một phương án nữa sử dụng các gói nhận dạng ô;

Fig.17 thể hiện cấu trúc của đơn vị truy cập theo một phương án. Đường đứt nét phản ánh trường hợp đơn vị NAL tiền tố lát không bắt buộc.

Fig.18 thể hiện việc sử dụng sự tạo tín hiệu các ô trong vùng quan tâm;

Fig.19 thể hiện cú pháp/phiên bản đơn giản thứ nhất 1;

Fig.20 thể hiện cú pháp/phiên bản kéo dài 2 bao gồm phép tạo tín hiệu tile_id, ký hiệu nhận dạng khởi đầu đơn vị giải mã, ID tiền tố lát và dữ liệu đoạn đầu lát ngoài khái niệm thông báo SEI;

Fig.21 thể hiện các lớp loại đơn vị NAL và mã loại đơn vị NAL;

Fig.22 thể hiện cú pháp có thể sử dụng cho đoạn đầu lát, trong đó các phần tử cú pháp nhất định có mặt trong đoạn đầu lát theo phiên bản hiện tại được thay đổi thành phần tử cú pháp thứ bậc thấp hơn, được đề cập đến là slice_header_data();

Fig.23 thể hiện bảng trong đó tất cả các phần tử cú pháp đã được loại bỏ khỏi đoạn đầu lát được tạo tín hiệu qua dữ liệu đoạn đầu lát phần tử cú pháp;

Fig.24 thể hiện cú pháp thông báo thông tin nâng cao bổ sung;

Fig.25 thể hiện cú pháp tải trọng SEI được làm thích ứng để đưa vào lát mới hoặc các loại thông báo SEI ảnh con;

Fig.26 thể hiện ví dụ cho thông báo SEI đệm ảnh con;

Fig.27 thể hiện ví dụ cho thông báo SEI định thời gian ảnh con;

Fig.28 thể hiện thông báo SEI lát ảnh có thể trông như thế nào;

Fig.29 thể hiện ví dụ thông báo SEI thông tin ô ảnh con.

Fig.30 thể hiện ví dụ cú pháp cho thông báo SEI thông tin kích thước ô ảnh con.

Fig.31 thể hện biến thể thứ nhất của ví dụ cú pháp cho thông báo SEI vùng quan tâm trong đó mỗi ROI được tạo tín hiệu trong một thông báo SEI riêng biệt.

Fig.32 thể hiện biến thể thứ hai của ví dụ cú pháp cho thông báo SEI vùng quan tâm trong đó tất cả các ROI được tạo tín hiệu trong một thông báo SEI đơn lẻ;

Fig.33 thể hiện cú pháp khả thi cho gói điều khiển định thời gian theo một phương án nữa;

Fig.34 thể hiện cú pháp khả thi cho gói nhận dạng ô theo một phương án nữa;

Các hình vẽ từ Fig.35 đến Fig.38 thể hiện sự chia nhỏ có thể của ảnh theo các thiết lập chia nhỏ khác nhau theo một phương án; và

Fig.39 thể hiện ví dụ của phần ngoài dòng dữ liệu video theo một phương án sử dụng các gói điều khiển định thời gian được đặt rải rác giữa các gói tải trọng của đơn vị truy cập.

Mô tả chi tiết sáng chế

Đối với Fig.12, bộ mã hóa 10 theo một phương án của đơn và chế độ hoạt động của nó được mô tả. Bộ mã hóa 10 được tạo cấu hình để mã hóa nội dung video 16 thành dòng dữ liệu video 22. Bộ mã hóa được tạo cấu hình để thực hiện điều này trong các đơn vị của các phần con của các khung/ảnh 18 của nội dung video 16, trong đó các phần con có thể, ví dụ, là các lát 24 mà các ảnh 18 được phân chia thành các lát này, hoặc một số đoạn không gian khác, như, ví dụ các ô 26 hoặc các dòng con WPP 28, tất cả chúng được minh họa trên Fig.12 chỉ nhằm cho các mục đích minh họa hơn là gợi ý rằng bộ mã hóa 10 cần là có thể đỡ ô hoặc xử lý song song WPP, ví dụ, hoặc rằng các phần con cần là các lát.

Trong quy trình mã hóa nội dung video 16 trong các đơn vị phần con 24, bộ mã hóa 10 có thể tuân thủ thứ tự giải mã - hoặc thứ tự mã hóa - được định ra giữa các phần con 24, ví dụ, đi ngang qua các ảnh 18 của video 16 theo thứ tự giải mã ảnh mà thứ tự này, ví dụ, không cần thiết trùng với thứ tự tái tạo 20 được định ra giữa các ảnh

18, và đi ngang qua trong từng ảnh 18 mà các ảnh 18 được phân chia thành các khối, theo thứ tự quét mành, với các phần con 24 thể hiện các hoạt động liên tục của các khối này theo thứ tự giải mã. Cụ thể, bộ mã hóa 10 có thể được tạo cấu hình để tuân theo thứ tự giải mã này trong việc xác định khả năng sử dụng các phần lân cận không gian và/hoặc thời gian của các phần hiện tại được mã hóa để sử dụng các thuộc tính mô tả các phần lân cận này trong việc mã hóa dự báo và/hoặc mã hóa entrôpi như, ví dụ, để xác định ngữ cảnh dự báo và/hoặc entrôpi: Chỉ các phần (đã được mã hóa/đã được giải mã) đã đến thăm trước của video là có thể sử dụng. Nếu không, các thuộc tính vừa đề cập được thiết lập đến các giá trị mặc định hoặc một số phương pháp thay thế khác được lấy ra.

Mặt khác, bộ mã hóa 10 không cần mã hóa từng kỳ các phần con 24 theo thứ tự giải mã. Hơn nữa, bộ mã hóa 10 có thể sử dụng xử lý song song để tăng tốc quy trình mã hóa, hoặc để có thể thực hiện sự mã hóa phức tạp hơn trong thời gian thực. Nói cách khác, bộ mã hóa 10 có thể hoặc không thể được tạo cấu hình để truyền hoặc gửi ra ngoài dữ liệu mã hóa các phần con theo thứ tự giải mã. Ví dụ, bộ mã hóa 10 có thể xuất ra/truyền dữ liệu đã được mã hóa ở một thứ tự khác nào đó, ví dụ như, theo thứ tự mà ở thứ tự này sự mã hóa các phần con được hoàn thành bởi bộ mã hóa 10 mà bộ mã hóa này, do xử lý song song, có thể ví dụ lệch khỏi thứ tự giải mã vừa nêu.

Để kết xuất các phiên bản đã được mã hóa của các phần con 24 thích hợp để truyền trên mạng, bộ mã hóa 10 mã hóa từng phần con 24 thành một hoặc nhiều gói tải trọng của chuỗi các gói của dòng dữ liệu video 22. Trong trường hợp các phần con 24 là các lát, bộ mã hóa 10 có thể, ví dụ, được tạo cấu hình để đặt từng dữ liệu lát, cụ thể là từng lát đã được mã hóa, vào trong một hoặc nhiều gói tải trọng, như đơn vị NAL. Sự tạo gói này có thể làm cho dòng dữ liệu video 22 thích hợp để truyền qua mạng. Do đó, các gói có thể là những đơn vị nhỏ nhất mà tại đó mà dòng dữ liệu video 22 có thể diễn ra, tức là, các đơn vị nhỏ nhất có thể được gửi đi riêng biệt bởi bộ mã hóa 10 để truyền qua mạng đến người nhận.

Ngoài các gói tải trọng và các gói điều khiển định thời gian được đặt rải rác giữa các gói tải trọng và được thảo luận dưới đây, các gói khác, cụ thể là các gói của loại khác cũng có thể tồn tại, như các gói dữ liệu điền đầy, các gói tập hợp tham số chuỗi

hoặc ảnh để truyền các phần tử cú pháp hiếm khi biến đổi hoặc các gói EOF (end of file - kết thúc tệp) hoặc AUE (access unit end - đầu đơn vị truy cập) hoặc tương tự.

Bộ mã hóa thực hiện mã hóa thành các gói tải trọng sao cho chuỗi các gói được chia thành chuỗi các đơn vị truy cập 30 và mỗi đơn vị truy cập tập hợp các gói tải trọng 32 đề cập đến một ảnh 18 của nội dung video 16. Tức là, chuỗi 34 gồm các gói tạo thành dòng dữ liệu video 22 được chia nhỏ thành các phần không chồng lên, được gọi là các đơn vị truy cập 30, mỗi đơn vị được kết hợp với một ảnh tương ứng trong số các ảnh 18. Chuỗi các đơn vị truy cập 30 có thể đi theo thứ tự giải mã của các ảnh 18 mà đơn vị truy cập 30 đề cập đến. Fig.12 minh họa, ví dụ, đơn vị truy cập 30 được sắp đặt ở giữa phần dòng dữ liệu 22, đơn vị truy cập này gồm có một gói tải trọng 32 trên mỗi phần con 24 mà ảnh 18 được chia nhỏ thành phần con này. Tức là, mỗi gói tải trọng 32 mang phần con 24 tương ứng. Bộ mã hóa 10 được tạo cấu hình để đặt rải rác các gói điều khiển định thời gian 36 bên trong chuỗi 34 gồm các gói để các gói điều khiển định thời gian chia nhỏ các đơn vị truy cập 30 thành các đơn vị giải mã 38 để ít nhất một số đơn vị truy cập 30, như đơn vị truy cập ở giữa được thể hiện trên Fig.12, được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã 38, mỗi gói điều khiển định thời gian tạo tín hiệu thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã 38, các gói tải trọng 32 của chuỗi các gói theo sau gói điều khiển định thời gian tương ứng trong chuỗi 34 gồm các gói. Nói cách khác, bộ mã hóa 10 đứng trước các chuỗi con của chuỗi các gói tải trọng 32 nằm trong một đơn vị truy cập 30 với gói điều khiển định thời gian 36 tương ứng tạo tín hiệu cho chuỗi các gói tải trọng tương ứng được đặt trước bởi gói điều khiển định thời gian 36 tương ứng và tạo ra đơn vị giải mã 38, thời gian phục hồi bộ nhớ đệm bộ giải mã. Fig.12, ví dụ, minh họa trường hợp trong đó mỗi gói thứ hai 32 biểu diễn gói tải trọng thứ nhất của đơn vị giải mã 38 của đơn vị truy cập. Như được minh họa trên Fig.12, lượng dữ liệu hay tốc độ bit được tiêu thụ cho mỗi đơn vị giải mã 38 thay đổi và thời gian phục hồi bộ nhớ đệm bộ giải mã có thể tương quan với sự thay đổi tốc độ bit giữa các đơn vị giải mã 38 ở chỗ thời gian phục hồi bộ nhớ đệm bộ giải mã của đơn vị giải mã 38 có thể theo sau thời gian phục hồi bộ nhớ đệm bộ giải mã được tạo tín hiệu bởi gói điều khiển thời gian 36 của đơn vị giải mã 38 ngay trước đó cộng với khoảng thời gian tương ứng với sự tiêu thụ tốc độ bit cho đơn vị giải mã 38 ngay trước đó này.

Tức là, bộ mã hóa 10 có thể hoạt động như được thể hiện trên Fig.13. Cụ thể, như đã được nêu trên, bộ mã hóa 10 có thể, trong bước 40, đưa ra phần con 24 của ảnh hiện thời 18 để mã hóa. Như đã đề cập bên trên, bộ mã hóa 10 có thể quay vòng liên tục qua các phần con 24 theo thứ tự giải mã đã nêu trên như được minh họa bởi mũi tên 42, hoặc bộ mã hóa 10 có thể sử dụng việc xử lý song song nào đó như WPP và/hoặc xử lý ô để mã hóa đồng thời một số "phần con 24 hiện thời". Không quan tâm đến việc sử dụng sự xử lý song song hay không, bộ mã hóa 10 tạo ra đơn vị giải mã trong số một hoặc vài phần con vừa được mã hóa trong bước 40 và tiến hành bước 44, trong đó bộ mã hóa 10 thiết lập thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã này và truyền đơn vị giải mã này được đứng trước với gói điều khiển thời gian tạo tín hiệu thời gian phục hồi bộ nhớ đệm bộ giải mã vừa được thiết lập cho đơn vị giải mã này. Ví dụ, bộ mã hóa 10 có thể định ra thời gian phục hồi bộ đệm bộ giải mã trong bước 44 trên cơ sở của tốc độ bit được tiêu thụ để mã hóa các phần con đã được mã hóa thành các gói tải trọng tạo ra đơn vị giải mã hiện thời bao gồm, ví dụ, tất cả các gói trung gian nữa nằm trong đơn vị giải mã này, nếu có thể, cụ thể, "các gói đứng trước".

Tiếp đó, trong bước 46, bộ mã hóa 10 có thể thích ứng tốc độ bit khả dụng trên cơ sở tốc độ bit được dùng cho đơn vị giải mã vừa đã được truyền trong bước 44. Nếu, ví dụ, nội dung ảnh trong đơn vị giải mã vừa được truyền trong bước 44 là hoàn toàn phức tạp về mặt tốc độ nén, thì bộ mã hóa 10 có thể làm giảm tốc độ bit khả dụng cho đơn vị giải mã kế tiếp để tuân theo một số tốc độ bit đích được thiết lập bên ngoài đã được xác định, ví dụ, trên cơ sở trường hợp băng thông hiện tại kết hợp với mạng truyền dòng dữ liệu video 22. Các bước 40 đến 46 tiếp đó được lặp lại. Bằng cách thức này, các ảnh 18 được mã hóa và được truyền, cụ thể, được gửi đi, trong các đơn vị của các đơn vị giải mã, mỗi đơn vị giải mã được đứng trước bởi gói điều khiển định thời gian tương ứng.

Nói cách khác, bộ mã hóa 10, trong quy trình mã hóa ảnh hiện thời 18 của nội dung video 16, mã hóa 40 phần con hiện thời 24 của ảnh hiện thời 18 thành gói tải trọng hiện thời 32 của đơn vị giải mã hiện thời 38, truyền 44, trong dòng dữ liệu, đơn vị giải mã hiện thời 38 được đặt trước với gói điều khiển định thời gian hiện thời 36 với thiết lập thời gian phục hồi bộ nhớ đệm bộ giải mã được tạo tín hiệu bởi gói điều

khiển định thời gian hiện thời 36, tại thời điểm đầu tiên, và mã hóa 44, bằng cách vòng lại từ bước 46 đến 40, phần con 24 khác của ảnh hiện thời 18 tại thời điểm thứ hai - lần thứ hai đến bước 40 -, muộn hơn so với thời điểm thứ nhất - lần thứ nhất đến bước 44.

Vì bộ mã hóa có thể gửi ra ngoài đơn vị giải mã trước khi mã hóa phần còn lại của ảnh hiện tại mà đơn vị giải mã thuộc về phần còn lại này, bộ mã hóa 10 có thể làm thấp độ trễ đầu đến cuối. Mặt khác, bộ mã hóa 10 không cần lãng phí tốc độ bit khả dụng, vì bộ mã hóa 10 có thể tác động đến bản chất cụ thể của nội dung của ảnh hiện tại và sự phân bố không gian của độ phức.

Mặt khác, các thiết bị mạng trung gian, chịu trách nhiệm truyền dẫn dòng dữ liệu video 22 thêm từ bộ mã hóa đến bộ giải mã, có thể sử dụng các gói điều khiển định thời gian 36 để đảm bảo rằng bộ giải mã bất kỳ nhận dòng dữ liệu video 22 nhận các đơn vị giải mã kịp thời để có thể đạt được thuận lợi của việc mã hóa theo đơn vị giải mã và truyền dẫn bởi bộ mã hóa 10. Xem, ví dụ Fig.14 thể hiện ví dụ bộ giải mã để giải mã dòng dữ liệu video 22. Bộ giải mã 12 nhận dòng dữ liệu video 22 tại CPB bộ nhớ đệm ảnh đã được mã hóa 48 nhờ mạng mà thông qua mạng này bộ mã hóa 10 đã truyền dòng dữ liệu video 22 đến bộ giải mã 12. Cụ thể, vì mạng 14 được cho là có thể hỗ trợ ứng dụng độ trễ thấp, mạng 10 kiểm tra các thời gian phục hồi bộ nhớ đệm bộ giải mã để chuyển chuỗi 34 các gói dòng dữ liệu video 22 đến bộ nhớ đệm ảnh đã được mã hóa 48 của bộ giải mã 12 để mỗi đơn vị giải mã có mặt trong bộ nhớ đệm ảnh đã được mã hóa 48 trước khi thời gian phục hồi bộ nhớ đệm bộ giải mã được tạo tín hiệu bởi gói điều khiển định thời gian đặt trước đơn vị giải mã tương ứng. Bằng cách này, bộ giải mã có thể, mà không cần dừng, cụ thể, mà không chạy ngoài các gói tải trọng khả dụng trong bộ nhớ đệm ảnh đã được mã hóa 48, sử dụng các thời gian phục hồi bộ nhớ đệm bộ giải mã trong các gói điều khiển định thời gian để làm rỗng bộ nhớ đệm ảnh đã được mã hóa 48 của bộ giải mã trong các đơn vị của các đơn vị giải mã hơn là hoàn thành các đơn vị truy cập. Fig.14 thể hiện cho các mục đích minh họa, đơn vị xử lý 50 được kết nối với đầu ra của bộ nhớ đệm ảnh đã được mã hóa 48, đầu vào của nó nhận dòng dữ liệu video 22. Tương tự với bộ mã hóa 10, bộ giải mã 12 có thể thực hiện xử lý song song ví dụ như, sử dụng xử lý/giải mã song song ô và/hoặc xử lý/giải mã song song WPP.

Như được chỉ ra chi tiết hơn bên dưới, các thời gian phục hồi bộ nhớ đệm bộ giải mã được đề cập không nhất thiết đề cập đến các thời gian phục liên quan đến bộ nhớ đệm ảnh đã được mã hóa 48 của bộ giải mã 12. Đúng hơn, các gói điều khiển định thời gian có thể, ngoài ra hoặc theo phương án khác, điều khiển hướng phục hồi dữ liệu ảnh đã được giải mã của bộ nhớ đệm ảnh đã được giải mã tương ứng của bộ giải mã 12. Fig.14 thể hiện, ví dụ, bộ giải mã 12 gồm có bộ nhớ đệm ảnh cho bộ giải mã trong đó phiên bản đã được giải mã của nội dung video, thu được bởi bộ xử lý 50 bằng cách giải mã dòng dữ liệu video 22, được nhớ đệm, cụ thể được lưu trữ và được xuất ra, trong các đơn vị của các phiên bản đã được giải mã của các đơn vị giải mã. Bộ nhớ đệm ảnh đã được giải mã 22 của bộ giải mã do đó có thể được kết nối giữa đầu ra của bộ giải mã 12 và đầu ra của bộ xử lý 50. Bởi có khả năng thiết lập các thời gian phục hồi để xuất ra các bản đã được giải mã của các đơn vị giải mã từ bộ nhớ đệm ảnh đã được giải mã 52, bộ mã hóa 10 được có cơ hội để, trong quy trình hoạt động, cụ thể trong quy trình mã hóa ảnh hiện tại, điều khiển sự tái tạo, hoặc độ trễ đầu đến cuối của việc tái tạo nội dung video tại phía giải mã thậm chí tại độ hạt nhỏ hơn so với tỷ lệ ảnh hoặc tỷ lệ khung. Rõ ràng, phân đoạn ảnh cực nhỏ từng ảnh 18 thành một lượng lớn phần con 24 ở phía mã hóa có ảnh hưởng tiêu cực tới tốc độ bit để truyền dòng dữ liệu video 22, mặc dù mặt khác, độ trễ đầu đến cuối có thể được giảm đến mức nhỏ nhất vì thời gian được cần để mã hóa và truyền và giải mã và xuất ra đơn vị giải mã này được giảm thiểu đến mức nhỏ nhất. Mặt khác, việc tăng kích thước các phần con 24 làm tăng độ trễ đầu đến cuối. Do đó, phải tìm ra một sự thỏa hiệp. Sử dụng các thời gian phục hồi bộ nhớ đệm bộ giải mã vừa nêu để định hướng thời gian xuất ra các phiên bản đã được giải mã của các phần con 24 trong các đơn vị giải mã, cho phép bộ mã hóa 10 hoặc đơn vị khác nào đó tại phía mã hóa thích ứng quy ước này trong không gian trên nội dung của ảnh hiện tại. Bằng cách này, có thể điều khiển độ trễ đầu đến cuối theo cách thức này, rằng độ trễ đầu đến cuối thay đổi theo không gian qua nội dung của các ảnh hiện tại.

Để thực hiện các phương án nêu trên, có thể sử dụng, làm các gói điều khiển định thời gian, là các gói của loại gói có thể loại bỏ. Các gói thuộc loại gói có thể loại bỏ là không cần thiết để phục hồi nội dung video tại phía giải mã. Dưới đây, các gói này được gọi là các gói SEI. Hơn nữa các gói thuộc loại gói có thể loại bỏ có thể tồn tại

cũng như, tức là, các gói có thể loại bỏ của loại khác như, nếu được truyền trong dòng, các gói thừa. Theo phương án khác, các gói điều khiển định thời gian có thể là các gói thuộc loại gói có thể loại bỏ đã biết, tuy nhiên, mang thêm trường loại gói SEI đã biết. Ví dụ, các gói điều khiển định thời gian có thể là các gói SEI với mỗi gói SEI mang một hoặc nhiều thông báo SEI, và chỉ các gói SEI đó gồm có thông báo SEI của loại đã biết tạo ra các gói điều khiển định thời gian nêu trên.

Do đó, phương án được mô tả đến giờ đối với các hình vẽ từ Fig.12 đến Fig.14, theo một phương án nữa, được áp dụng trên tiêu chuẩn HEVC, do đó tạo ra khái niệm có thể để làm cho HEVC hiệu quả hơn để đạt được độ trễ đầu đến cuối thấp. Để thực hiện điều này, các gói nêu trên được tạo ra bởi các đơn vị NAL và các gói tải trọng nêu trên là các đơn vị NAL VCL của dòng đơn vị NAL với các lát tạo ra các phần con nêu trên.

Tuy nhiên, trước khi sự mô tả phương án được chi tiết hơn, các phương án khác được mô tả trùng khớp với các phương án được nêu trên ở chỗ các gói được đặt rải rác được sử dụng để chuyển, theo cách thức hiệu quả, thông tin mô tả dòng dữ liệu video, nhưng loại thông tin khác với các phương án nêu trên ở chỗ các gói điều khiển định thời gian được chuyển thông tin định thời gian phục hồi bộ nhớ đệm bộ giải mã. Theo các phương án được mô tả thêm bên dưới, loại thông tin được truyền qua các gói được đặt rải rác được đặt rải rác bên trong các gói tải trọng thuộc về đơn vị truy cập, đề cập tới thông tin vùng quan tâm (region of interest - ROI) và/hoặc thông tin nhận dạng ô. Các phương án được mô tả thêm bên dưới có thể hoặc không thể được kết hợp với các phương án đã được mô tả đối với các hình vẽ từ Fig.12 đến Fig.14.

Fig.15 thể hiện bộ mã hóa 10 hoạt động tương tự với bộ mã hóa đã được giải thích bên trên đối với Fig.12, ngoại trừ việc đặt rải rác các gói điều khiển định thời gian và chức năng được mô tả bên trên đối với Fig.13 là tùy ý đối với bộ mã hóa 10 của Fig.15. Tuy nhiên, bộ mã hóa 10 của Fig.15 được tạo cấu hình để mã hóa nội dung video 16 thành dòng dữ liệu video 22 trong các đơn vị phần con 24 của các ảnh 18 của nội dung video 16 vừa như được giải thích bên trên đối với Fig.11. Trong việc mã hóa nội dung video 16, bộ mã hóa 10 được quan tâm về việc chuyển thông tin, cùng với dòng dữ liệu video 22, trên vùng quan tâm ROI 60 về phía giải mã. ROI 60 là vùng

con không gian của ảnh hiện thời 18 mà bộ giải mã ví dụ, quan tâm đặc biệt đến vùng này. Vị trí không gian của ROI 60 có thể nhập vào bộ mã hóa 10 từ phía ngoài, như được minh họa bằng nét đứt 62, bởi sự nhập vào của người dùng, hoặc có thể được xác định một cách tự động bởi bộ mã hóa 10 hoặc bởi thực thể khác nào đó, đang hoạt động trong khoảng thời gian mã hóa ảnh hiện thời 18. Hoặc trong trường hợp, bộ mã hóa 10 đối mặt với các vấn đề sau đây: chỉ ra vị trí của ROI 60 theo nguyên tắc không có vấn đề nào cho bộ mã hóa 10. Để thực hiện điều này, bộ mã hóa 10 có thể dễ dàng chỉ ra vị trí của ROI 60 nằm trong dòng dữ liệu 22. Tuy nhiên, để làm cho thông tin này dễ dàng có thể truy cập, bộ mã hóa 10 của Fig.15 sử dụng sự đặt rải rác các gói ROI giữa các gói tải trọng của các đơn vị truy cập 10 do đó bộ mã hóa 10, trên cơ sở trực tuyến, thoải mái lựa chọn việc chia đoạn ảnh hiện thời 18 thành các phần con 24 và/hoặc nhiều gói tải trọng mà các phần con 24 được tạo gói thành các gói tải trọng này, không gian phía ngoài và không gian phía trong ROI 60. Sử dụng các gói ROI được đặt rải rác, thiết bị mạng bất kỳ có thể dễ dàng nhận dạng các gói tải trọng thuộc ROI. Mặt khác, trong trường hợp sử dụng loại gói có thể loại bỏ cho các gói ROI này, các gói ROI này có thể dễ dàng không được chú ý đến bởi thiết bị mạng bất kỳ.

Fig.15 thể hiện ví dụ đặt rải rác gói ROI 64 giữa các gói tải trọng 32 của đơn vị truy cập 30. Gói ROI 64 chỉ ra nơi mà trong chuỗi 34 gồm các gói của dòng dữ liệu video 2, dữ liệu đã được mã hóa nằm trong chuỗi các gói đề cập đến, tức là, mã hóa ROI 60. Làm thế nào gói ROI 64 chỉ ra vị trí của ROI 60 có thể được cung cấp theo các cách đa dạng. Ví dụ, sự tồn tại/sự xuất hiện thuần của gói ROI 64 có thể chỉ ra sự kết hợp của dữ liệu đã được mã hóa đề cập đến ROI 60 trong một hoặc nhiều gói tải trọng 32 tiếp theo, đi theo thứ tự liên tiếp của chuỗi 34 gồm các gói, tức là, thuộc về các gói tải trọng đứng trước. Hoặc, phần tử cú pháp bên trong gói ROI 64 có thể chỉ ra liệu một hoặc nhiều gói tải trọng 32 tiếp theo có đề cập đến, tức là, ít nhất mã hóa một phần, ROI 60 hay không. Lượng lớn sự thay đổi cũng xuất phát từ các sự thay đổi có thể đề cập đến "phạm vi" của gói ROI 64 tương ứng, tức là, số lượng gói tải trọng đứng trước được đứng trước bởi một gói ROI 64. Ví dụ, việc chỉ ra sự kết hợp hoặc không kết hợp của dữ liệu đã được mã hóa bất kỳ đề cập đến ROI 60 trong một gói ROI, có thể liên quan đến tất cả các gói tải trọng 32, tức là đi theo sau trong thứ tự liên tiếp của chuỗi 34 tùy theo sự xuất hiện của gói ROI 64 kế tiếp, hoặc có thể đơn thuần

liên quan đến gói tải trọng 32 ngay theo sau, tức là các gói tải trọng 32 ngay theo sau gói ROI 64 tương ứng trong thứ tự liên tiếp của chuỗi 34. Trên Fig.15, đồ thị 66 minh họa ví dụ trường hợp trong đó các gói ROI 64 chỉ ra ROI liên quan, tức là sự kết hợp của dữ liệu đã được mã hóa bất kỳ đề cập đến ROI 60, hoặc ROI không liên quan, tức là không có dữ liệu đã được mã hóa bất kỳ đề cập đến ROI 60, liên quan đến tất cả các gói tải trọng 32 thấy ở dòng xuôi của gói ROI 64 tương ứng đến sự xuất hiện của ROI 64 kế tiếp hoặc kết thúc của đơn vị truy cập 30 hiện tại bất kể thấy sớm hơn dọc chuỗi 34 gồm các gói. Cụ thể, Fig.15 minh họa trường hợp trong đó gói ROI 64 có phần tử cú pháp bên trong, chỉ ra liệu các gói tải trọng 32 có đi theo thứ tự tiếp sau của chuỗi 34 gồm các gói có dữ liệu đã được mã hóa bất kỳ đề cập đến ROI 60 bên trong hay không. Phương án này cũng được mô tả dưới đây. Tuy nhiên, một khả năng khác là, như vừa được đề cập, rằng mỗi gói ROI 64 đơn thuần chỉ ra, nhờ sự có mặt của nó trong chuỗi gói 34, rằng (các) gói tải trọng 32 thuộc về "phạm vi" của gói ROI 64 tương ứng, có dữ liệu đề cập đến ROI 60 bên trong, tức là, dữ liệu đề cập đến ROI 60. Theo phương án được mô tả chi tiết hơn dưới đây, gói ROI 64 thậm chí chỉ ra vị trí của một phần của ROI 60 được mã hóa thành (các) gói tải trọng 32 thuộc về "phạm vi" của nó.

Thiết bị mạng 68 bất kỳ nhận dòng dữ liệu video 22 có thể khai thác sự biểu lộ của ROI liên quan như được nhận ra bằng cách sử dụng gói ROI 64 để xử lý, ví dụ, các phần liên quan đến ROI của chuỗi 34 gồm các gói với tính ưu tiên cao hơn so với các phần khác của chuỗi 34 gồm các gói. Hoặc, thiết bị mạng 68 có thể sử dụng thông tin liên quan ROI để thực hiện các nhiệm vụ khác đề cập đến, ví dụ, việc sự truyền dẫn dòng dữ liệu video 22. Thiết bị mạng 68 có thể, ví dụ là MANE hoặc bộ giải mã để giải mã và phát lại nội dung video 60 như được chuyển qua dòng dữ liệu video 22. 28. Nói cách khác, thiết bị mạng 68 có thể sử dụng kết quả nhận dạng các gói ROI để quyết định các nhiệm vụ truyền dẫn đề cập dòng dữ liệu video. Các nhiệm vụ truyền dẫn có thể gồm có các yêu cầu truyền dẫn lại đề cập đến các gói khiếm khuyết. Thiết bị mạng 68 có thể được tạo cấu hình để xử lý vùng quan tâm 70 với tính ưu tiên tăng lên và gán tính ưu tiên cao cho các gói ROI 72 và các gói tải trọng kết hợp của chúng, cụ thể các gói tải trọng được đặt trước bởi nó, chúng được tạo tín hiệu là chồng lên vùng quan tâm, so với các gói ROI và các gói tải trọng kết hợp của chúng, chúng được

tạo tín hiệu là không chồng lên ROI. Thiết bị mạng theo điểm 68 có thể trước tiên yêu cầu truyền lại các gói tải trọng có tính ưu tiên cao hơn được gán cho nó, trước khi yêu cầu bất kỳ sự truyền lại nào các gói tải trọng có ưu tiên thấp hơn được gán vào đó.

Phương án của Fig.15 có thể dễ dàng được kết hợp với phương án được mô tả trước đó với các hình vẽ từ Fig.12 đến Fig.14. Ví dụ, các gói ROI 64 như nêu trên có thể cũng là các gói SEI có loại thông báo SEI nhất định nằm trong đó, cụ thể thông báo ROI SEI. Tức là, gói SEI có thể, ví dụ, là gói điều khiển định thời gian và đồng thời là gói ROI, cụ thể nếu gói SEI tương ứng gồm cả thông tin điều khiển định thời gian cũng như thông tin chỉ ra ROI. Hoặc, gói SEI có thể là một trong gói điều khiển định thời gian và gói ROI, hơn là gói còn lại, hoặc có thể là gói ROI hoặc gói điều khiển định thời gian.

Theo phương án được thể hiện trên Fig.16, việc đặt rải rác các gói giữa các gói tải trọng của các đơn vị truy cập được sử dụng để chỉ ra, theo cách thức dễ truy cập cho các thiết bị mạng 68 để xử lý dòng dữ liệu video 22, mà ô hoặc các ô của ảnh hiện thời 18, mà đơn vị truy cập hiện thời 30 đề cập đến, được chồng lên bởi phần con bất kỳ được mã hóa thành bất kỳ gói tải trọng 32 nào mà các gói tương ứng đứng trước cho các gói tải trọng này. Trên Fig.16, ví dụ, ảnh hiện thời 18 được thể hiện là được chia nhỏ thành bốn ô 70, được tạo ra bởi bốn góc của ảnh hiện thời 18. Sự chia nhỏ ảnh hiện thời 18 thành các ô 70 có thể, ví dụ, được tạo tín hiệu bên trong dòng dữ liệu video trong các đơn vị gồm các chuỗi ảnh như, ví dụ, trong các gói VPS hoặc SPS cũng được đặt rải rác trong chuỗi 34 gồm các gói. Như sẽ được mô tả chi tiết hơn bên dưới, sự chia nhỏ ô của ảnh hiện thời 18 có thể là sự chia nhỏ đều ảnh 18 thành các cột và các hàng gồm các ô. Số lượng cột và số lượng hàng cũng như độ rộng cột và chiều cao hàng của các ô có thể biến thiên. Cụ thể, độ rộng và chiều cao của các cột/các hàng của các ô có thể là khác nhau cho các hàng khác nhau và các cột khác nhau tương ứng. Fig.16 còn thể hiện ví dụ trong đó các phần con 24 là các lát của ảnh 18. Các lát 24 chia nhỏ ảnh 18. Như được chỉ ra chi tiết hơn bên dưới, sự chia nhỏ ảnh 18 thành các lát 24 có thể phải chịu sự bắt buộc mà theo đó mỗi lát 24 có thể được chứa hoàn toàn trong một ô đơn lẻ 70 hoặc hoàn toàn bao phủ hai hoặc nhiều hơn hai ô 70. Fig.16 minh họa trường hợp trong đó ảnh 18 được chia nhỏ thành năm lát 24. Một phần tư số lát 24 này theo thứ tự giải mã được đề cập bên trên bao phủ hai ô 70 đầu tiên, trong khi

lát thứ năm hoàn toàn bao phủ các ô thứ ba và thứ tư 70. Hơn nữa, Fig.16 minh họa trường hợp trong đó mỗi lát 24 được mã hóa riêng biệt thành gói tải trọng tương ứng 32. Hơn nữa, Fig.16 minh họa trường hợp trong đó mỗi gói tải trọng 32 được đặt trước bởi gói nhận dạng ô đặt trước 72. Mỗi gói nhận dạng ô 72, lần lượt, chỉ ra gói tải trọng 32 ngay tiếp theo của gói nhận dạng ô mà đối với gói tải trọng của các ô 70 này, phần con 24 được mã hóa thành gói tải trọng 32 này xếp chồng lên. Do đó, trong khi hai gói nhận dạng ô 72 đầu tiên bên trong đơn vị truy cập 30 đề cập đến ảnh hiện thời 18 cho biết ô thứ nhất, gói nhận dạng ô thứ ba và thứ tư 72 cho biết ô thứ hai 70 của ảnh 18, và gói nhận dạng ô thứ năm 72 cho biết các ô thứ ba và thứ tư 70. Đối với phương án của Fig.16, các biến đổi tương tự là dễ dàng như được mô tả bên trên đối với Fig.15. Tức là, "phạm vi" của các gói nhận dạng ô 72 có thể, ví dụ, chỉ bao gồm gói tải trọng 32 ngay tiếp theo thứ nhất hoặc các gói tải trọng 32 ngay sau đến sự hiện diện của gói nhận dạng ô kế tiếp.

Đối với các ô, bộ mã hóa 10 có thể được tạo cấu hình để mã hóa mỗi ô 70 sao cho, qua các biên ô, không có sự dự báo không gian nào hoặc không có sự lựa chọn ngữ cảnh nào diễn ra. Bộ mã hóa 10 có thể, ví dụ, mã hóa ô 70 song song. Tương tự, bộ giải mã bất kỳ như thiết bị mạng 68 có thể giải mã các ô 70 song song.

Thiết bị mạng 68 có thể là MANE hoặc bộ giải mã hoặc thiết bị khác nào đó giữa bộ mã hóa 10 và bộ giải mã, và có thể được tạo cấu hình để sử dụng thông tin được chuyển bởi các gói nhận dạng ô 72 để quyết định các nhiệm vụ truyền dẫn nhất định. Ví dụ, thiết bị mạng 68 có thể xử lý một ô nhất định của ảnh hiện thời 18 của video 16 với ưu tiên cao hơn, tức là, có thể chuyển các gói tải trọng tương ứng đã được chỉ ra là đề cập đến ô này sớm hơn hoặc sử dụng sự bảo vệ FEC an toàn hơn hoặc tương tự. Nói cách khác, thiết bị mạng 68 có thể sử dụng kết quả nhận dạng để quyết định các nhiệm vụ truyền đề cập dòng dữ liệu video. Các nhiệm vụ truyền dẫn có thể gồm có các yêu cầu truyền dẫn lại đề cập đến các gói được nhận trong trạng thái thiếu sót - tức là với việc vượt quá sự bảo vệ FEC bất kỳ của dòng dữ liệu video, nếu có. Thiết bị mạng có thể xử lý, ví dụ, các ô 70 khác nhau với ưu tiên khác nhau. Để đạt được điều này, thiết bị mạng có thể gán ưu tiên cao hơn cho các gói nhận dạng ô 72 và các gói tải trọng của chúng, tức là nhờ đó các gói nhận dạng ô đứng trước, đề cập đến các ô có ưu tiên cao hơn, so với các gói nhận dạng ô 72 và các gói tải trọng của chúng đề cập đến

các ô có ưu tiên thấp hơn. Thiết bị mạng theo điểm 68 có thể, ví dụ, trước tiên yêu cầu truyền lại các gói tải trọng có ưu tiên cao hơn được gán vào đó, trước khi yêu cầu sự truyền lại bất kỳ các gói tải trọng có ưu tiên thấp hơn được gán vào đó.

Các phương án được mô tả cho đến giờ có thể được xây dựng thành khung HEVC như được mô tả trong phần mở đầu của bản mô tả của đơn như được mô tả sau đây.

Cụ thể, các thông báo SEI có thể được gán cho các lát của các đơn vị giải mã trong trường hợp CPB/HRD ảnh con. Tức là, khoảng thời gian nhớ đệm và các thông báo SEI định thời gian có thể được gán cho các đơn vị NAL chứa các lát của đơn vị giải mã. Điều này có thể đạt được bởi loại đơn vị NAL mới không phải là đơn vị VCL-NAL được phép đứng trước trực tiếp một hoặc nhiều đơn vị lát/NAL VCL của đơn vị giải mã. Đơn vị NAL mới này có thể được gọi là đơn vị NAL tiền tố lát. Fig.17 minh họa cấu trúc của đơn vị truy cập bỏ qua các đơn vị NAL thử nghiệm bất kỳ để kết thúc chuỗi và dòng.

Theo Fig.17, đơn vị truy cập 30 được hiểu như sau: theo thứ tự liên tiếp của các gói của chuỗi 34 gồm các gói, đơn vị truy cập 30 có thể bắt đầu với sự diễn ra của loại gói đặc biệt, cụ thể là dấu phân cách đơn vị truy cập 80. Tiếp đó một hoặc nhiều gói SEI 82 của loại gói SEI đề cập đến toàn bộ đơn vị truy cập có thể theo sau trong đơn vị truy cập 30. Cả hai loại gói 80 và 82 là tùy ý. Tức là, không có gói loại này có thể xuất hiện trong đơn vị truy cập 30. Tiếp đó, chuỗi các đơn vị giải mã 38 theo sau. Từng đơn vị giải mã 38 tùy ý bắt đầu với đơn vị NAL tiền tố lát 84, bao gồm trong đó, ví dụ thông tin điều khiển định thời gian hoặc theo phương án của Fig.15 hoặc Fig.16, thông tin ROI hoặc thông tin ô hoặc, thậm chí chung hơn, thông báo SEI ảnh con 86 tương ứng. Tiếp theo, dữ liệu lát thực 88 trong các gói tải trọng tương ứng hoặc các đơn vị NAL VCL theo sau như được chỉ ra trong 88. Do đó, mỗi đơn vị giải mã 38 gồm có chuỗi các đơn vị NAL tiền tố lát 84 sau đó là (các) đơn vị NAL dữ liệu lát tương ứng 88. Mũi tên đường vòng 90 trên Fig.17, đi vòng qua đơn vị NAL tiền tố lát, sẽ chỉ ra rằng trong trường hợp không có sự chia nhỏ đơn vị giải mã của đơn vị truy cập hiện thời 30 có thể không có đơn vị NAL tiền tố lát 84 nào.

Như được chỉ ra bên trên, tất cả thông tin được tạo tín hiệu trong tiền tố lát và các

thông báo SEI ảnh con kết hợp có thể là có giá trị cho tất cả các đơn vị NAL VCL trong đơn vị truy cập hoặc cho đến khi xuất hiện đơn vị NAL tiền tố thứ hai hoặc cho đơn vị NAL VCL sau đây theo thứ tự giải mã, phụ thuộc vào cờ được đưa ra trong đơn vị NAL tiền tố lát.

Đơn vị NAL VCL lát mà thông tin được tạo tín hiệu cho trong tiền tố lát là có giá trị được đề cập đến là các lát được đứng trước sau đây. Các lát đứng trước kết hợp với một lát được đứng trước không nhất thiết tạo nên một đơn vị giải mã hoàn thiện mà có thể là một phần của đơn vị giải mã hoàn thiện. Tuy nhiên, một tiền tố lát đơn lẻ không thể có giá trị cho nhiều đơn vị giải mã (các ảnh con) và sự bắt đầu đơn vị giải mã được tạo tín hiệu trong tiền tố lát. Nếu phương tiện để tạo tín hiệu không được đưa ra qua cú pháp tiền tố lát (như trong "cú pháp đơn giản"/phiên bản 1 được chỉ ra bên dưới) sự diễn ra các đơn vị NAL tiền tố lát tạo tín hiệu bắt đầu đơn vị giải mã. Chỉ các thông báo SEI nhất định (được nhận dạng thông qua payloadType trong sự mô tả cú pháp bên dưới) có thể được gửi riêng nằm trên mức ảnh con nằm trong đơn vị NAL tiền tố lát, trong khi một số thông báo SEI khác có thể được gửi trong đơn vị NAL tiền tố lát nằm trên mức ảnh con hoặc là thông báo SEI thông thường nằm trên mức đơn vị truy cập.

Như đã thảo luận bên trên đối với Fig.16, ngoài ra hoặc theo một lựa chọn khác, sự tạo tín hiệu nhận dạng ô/thông báo SEI nhận dạng ô có thể được thấy trong cú pháp mức cao. Trong các thiết kế ban đầu của HEVC, đoạn đầu lát/dữ liệu lát chứa ký hiệu nhận dạng cho ô được chứa trong lát tương ứng. Ví dụ, các ngữ nghĩa học của dữ liệu lát đọc:

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

Tuy nhiên, tham số này không được xem là hữu dụng vì ID này có thể dễ dàng được suy ra từ địa chỉ lát và kích thước lát được tạo tín hiệu trong tập hợp tham số ảnh, nếu tiles_or_entropy_coding_sync_idc bằng 1.

Mặc dù ID ô có thể được suy ra hoàn toàn trong quy trình giải mã, kiến thức về tham số này trên lớp ứng dụng cũng là quan trọng đối với các trường hợp sử dụng

khác nhau, ví dụ như trong kịch bản hội nghị video ở đó các ô khác nhau có thể có quyền ưu tiên khác nhau cho việc phát lại (các ô đó thường tạo ra vùng quan tâm chứa người nói trong trường hợp sử dụng đàm thoại) có thể có quyền ưu tiên cao hơn so với các ô khác. Trong trường hợp mất các gói mạng trong việc truyền nhiều ô, các gói mạng đó chứa các ô thể hiện vùng quan tâm có thể được truyền lại với quyền ưu tiên cao hơn để giữ chất lượng trải nghiệm tại đầu cuối máy thu cao hơn trong trường hợp truyền lại các ô mà không có bất kỳ thứ tự ưu tiên nào. Trường hợp sử dụng khác có thể là để gán các ô, nếu các kích thước và vị trí của chúng được biết, cho các màn hình khác nhau, ví dụ, trong kịch bản hội nghị bằng video.

Để cho phép lớp ứng dụng này xử lý các ô với quyền ưu tiên nhất định trong các kịch bản truyền dẫn, tile_id có thể được cung cấp là ảnh con hoặc thông báo SEI lát riêng hoặc trong đơn vị NAL đặc biệt ở phía trước một hoặc nhiều đơn vị NAL của ô hoặc trong phần đoạn đầu đặc biệt của đơn vị NAL của ô.

Như được mô tả bên trên đối với Fig.15, vùng của các thông báo SEI quan tâm cũng có thể được cung cấp thêm vào hoặc như một lựa chọn khác. Thông báo SEI này có thể cho phép tạo tín hiệu vùng quan tâm (region of interest - ROI), cụ thể tạo tín hiệu ROI mà tile_id/ô nhất định thuộc về ROI này. Thông báo có thể cho phép đưa ra các ID vùng quan tâm cùng với độ ưu tiên của vùng quan tâm.

Fig.18 minh họa việc sử dụng sự tạo tín hiệu vùng quan tâm.

Ngoài việc đã được mô tả bên trên, sự tạo tín hiệu đoạn đầu lát có thể được thực hiện. Đơn vị NAL tiền tố lát cũng có thể chứa đoạn đầu lát cho các lát phụ thuộc sau đây, tức là, các lát được đứng trước bởi tiền tố lát tương ứng. Nếu đoạn đầu lát chỉ được cung cấp trong đơn vị NAL tiền tố lát, loại lát thực sự cần được dẫn bởi loại đơn vị NAL của đơn vị NAL chứa lát phụ thuộc tương ứng hoặc bởi cờ trong tiền tố lát tạo tín hiệu liệu dữ liệu lát sau đây thuộc về loại lát đóng vai trò như đơn vị truy cập ngẫu nhiên.

Hơn nữa, đơn vị NAL tiền tố lát có thể mang lát hoặc các thông báo SEI cụ thể ảnh con để chuyển thông tin không bắt buộc như sự định thời gian ảnh con hoặc bộ nhận dạng ô. Thông báo cụ thể ảnh con không bắt buộc không được hỗ trợ trong thông số kỹ thuật HEVC được mô tả trong phần mở đầu của bản mô tả của đơn, nhưng là

quyết định đối với các ứng dụng nhất định.

Dưới đây, cú pháp có thể để thực hiện khái niệm chỉ ra bên trên của lát đứng trước được mô tả. Cụ thể, được mô tả các sự thay đổi có thể đáp ứng trên mức lát khi sử dụng trạng thái HEVC như được chỉ ra trong phần mở đầu của đơn của đơn hiện tại.

Cụ thể, sau đây, hai phiên bản của cú pháp tiền tố lát có thể được thể hiện, một phiên bản với tính năng của duy nhất thông báo SEI, và một phiên bản với tính năng mở rộng của sự tạo tín hiệu một phần đoạn đầu lát cho các lát sau đây. Cú pháp đơn giản thứ nhất/phiên bản 1 được thể hiện trên Fig.19.

Như một lưu ý mở đầu, Fig.19 do đó thể hiện một khả năng thực hiện khả thi để thực hiện bất kỳ phương án nào được mô tả bên trên đối với các hình vẽ từ Fig.11 đến Fig.16. Các gói được đặt rải rác được thể hiện trong các phương án có thể được phân tích như được thể hiện trên Fig.19 và sau đây Fig.19 được mô tả chi tiết hơn với các ví dụ thực hiện cụ thể.

Cú pháp được kéo dài/phiên bản 2 bao gồm tạo tín hiệu tile_id, ký hiệu nhận dạng khởi đầu đơn vị giải mã, ID tiền tố lát và dữ liệu đoạn đầu lát ngoài khái niệm thông báo SEI được đưa ra trong bảng của Fig.20.

Các ngữ nghĩa học có thể được định nghĩa như sau:

rap_flag với giá trị 1 chỉ ra rằng đơn vị truy cập chứa tiền tố lát là ảnh RAP. rap_flag với giá trị 0 chỉ ra rằng đơn vị truy cập chứa tiền tố lát không phải là ảnh RAP.

decoding_unit_start_flag chỉ ra sự bắt đầu đơn vị giải mã trong đơn vị truy cập, do đó rằng các lát dưới đây cho đến đầu cuối của đơn vị truy cập hoặc bắt đầu đơn vị giải mã khác thuộc cùng đơn vị giải mã.

single_slice_flag với giá trị 0 chỉ ra rằng thông tin được cung cấp trong đơn vị NAL lát tiền tố và các thông báo SEI ảnh con kết hợp là có giá trị đối với tất cả các đơn vị NAL VCL sau đây cho đến khi bắt đầu đơn vị truy cập tiếp theo, sự kiện tiền tố lát khác hoặc đoạn đầu lát trọn vẹn khác single_slice_flag với giá trị 1 chỉ ra rằng tất cả thông tin được cung cấp trong đơn vị NAL tiền tố lát và các thông báo SEI ảnh con kết hợp là có giá trị duy nhất đối với đơn vị NAL VCL kế tiếp theo thứ tự giải mã.

tile_idc chỉ ra số ô được có mặt trong lát dưới đây. tile_idc bằng 0 chỉ ra rằng không có ô nào được sử dụng trong lát tiếp theo. tile_idc bằng 1 chỉ ra rằng một ô đơn lẻ được sử dụng trong lát tiếp theo và do đó ký hiệu nhận dạng ô của nó được tạo tín hiệu. tile_idc có giá trị bằng 2 chỉ ra rằng nhiều ô được sử dụng trong lát tiếp theo và số ô và ký hiệu nhận dạng ô thứ nhất theo đó được tạo tín hiệu.

prefix_slice_header_data_present_flag chỉ ra rằng dữ liệu của đoạn đầu lát, tương ứng với các lát tiếp theo đi theo thứ tự giải mã được tạo tín hiệu trong tiền tố lát đã cho.

slice_header_data() được định nghĩa sau đây trong ngữ cảnh. Nó chứa thông tin đoạn đầu lát liên quan, nó không được bao hàm bởi đoạn đầu lát, nếu **dependent_slice_flag** được thiết lập bằng 1.

Lưu ý rằng sự giải ghép của đoạn đầu lát và dữ liệu lát thực sự cho phép các sơ đồ truyền dẫn linh động hơn của đoạn đầu và dữ liệu lát.

num_tiles_in_prefixed_slices_minus1 chỉ ra số ô được sử dụng trong đơn vị giải mã sau đây trừ đi 1.

first_tile_id_in_prefixed_slices chỉ ra ký hiệu nhận dạng ô của ô thứ nhất trong đơn vị giải mã tiếp theo.

Đối với cú pháp đơn giản/phiên bản 1 của tiền tố lát, các phần tử cú pháp sau đây có thể được thiết lập đến các giá trị mặc định như sau, nếu không có mặt:

- **decoding_unit_start** bằng 1, tức là tiền tố lát thường chỉ ra sự bắt đầu đơn vị giải mã.
- **single_slice_flag** bằng 0, tức là tiền tố lát là có giá trị đối với tất cả các lát trong đơn vị giải mã.

Đơn vị NAL tiền tố lát được đề xuất có loại đơn vị NAL 24 và bảng tổng quát loại đơn vị NAL được mở rộng theo Fig.21.

Tức là, tóm lại trong các hình vẽ từ Fig.19 đến Fig.21, các chi tiết cú pháp được thể hiện trong đó bộc lộ rằng loại gói nhất định có thể được đóng góp vào các gói được đặt rải rác được nhận dạng bên trên, ở đây minh họa loại đơn vị NAL 24. Hơn nữa, đặc biệt ví dụ cú pháp của Fig.20 làm cho rõ ràng rằng các biến thể được mô tả bên trên đề

cập đến "phạm vi" của các gói được đặt rải rác, cơ cấu chuyển mạch được điều khiển bởi phần tử cú pháp tương ứng trong các gói được đặt rải rác này, ở đây minh họa single_slice_flag, có thể được sử dụng để điều khiển phạm vi này, cụ thể, để chuyển giữa các biến thể khác nhau cho định nghĩa phạm vi này, một cách lần lượt. Hơn nữa, nhận thấy rằng các phương án được mô tả bên trên của các hình vẽ từ Fig.1 đến Fig.16 có thể được mở rộng ở chỗ các gói được đặt rải rác cũng gồm có dữ liệu đoạn đầu lát thông thường cho các lát 24 nằm trong các gói thuộc "phạm vi" của các gói được đặt rải rác tương ứng. Tức là, có thể có cơ chế được điều khiển bởi cờ tương ứng nằm trong các gói được đặt rải rác này, cơ chế chỉ ra liệu dữ liệu đoạn đầu lát thông thường đư trong gói được đặt rải rác tương ứng hay không.

Tất nhiên, khái niệm vừa được thể hiện theo phần của dữ liệu đoạn đầu lát được chuyển thành tiền tố đoạn đầu lát, yêu cầu các thay đổi cho các đoạn đầu lát như được chỉ ra trong phiên bản hiện tại của HEVC. Bảng trên Fig.22 thể hiện cú pháp có thể cho đoạn đầu lát đã nêu, trong đó phần tử cú pháp nhất định có mặt trong đoạn đầu lát theo phiên bản hiện tại, được dịch chuyển đến phần tử cú pháp thứ bậc thấp hơn được đề cập đến là slice_header_data(). Cú pháp này của đoạn đầu lát và dữ liệu đoạn đầu lát chỉ áp dụng cho tùy chọn mà theo tùy chọn này khái niệm đơn vị NAL đứng trước đoạn đầu lát mở rộng được sử dụng.

Trên Fig.22, slice_header_data_present_flag chỉ ra rằng dữ liệu đoạn đầu lát cho lát hiện tại sẽ được dự báo từ các giá trị được tạo tín hiệu trong đơn vị NAL tiền tố lát cuối cùng trong đơn vị truy cập, tức là, đơn vị NAL tiền tố lát diễn ra gần đây nhất.

Tất cả các phần tử cú pháp được loại bỏ khỏi đoạn đầu lát được tạo tín hiệu qua dữ liệu đoạn đầu lát phần tử cú pháp như được đưa ra trong bảng của Fig.23.

Tức là, chuyển khái niệm của Fig.22 và Fig.23 trên các phương án của các hình vẽ từ Fig.12 đến Fig.16, các gói được đặt rải rác được mô tả trong đó có thể được mở rộng bởi khái niệm kết hợp vào trong các gói được đặt rải rác này, một phần của cú pháp đoạn đầu lát của các lát (các phần con) 24 được mã hóa thành các gói tải trọng, tức là, các đơn vị NAL VCL. Sự kết hợp có thể là tùy ý. Tức là, phần tử cú pháp tương ứng trong gói được đặt rải rác có thể chỉ ra liệu cú pháp đoạn đầu lát này có nằm trong gói được đặt rải rác tương ứng hay không. Nếu được kết hợp, dữ liệu đoạn đầu lát

tương ứng được kết hợp vào trong gói được đặt rải rác tương ứng có thể áp dụng cho tất cả các lát nằm trong gói thuộc "phạm vi" của gói được đặt rải rác tương ứng. Liệu dữ liệu đoạn đầu lát nằm trong gói được đặt rải rác có được thích ứng bởi lát được mã hóa thành bất kỳ gói tải trọng nào thuộc phạm vi của gói được đặt rải rác này hay không, có thể được tạo tín hiệu bởi cờ tương ứng, như slice_header_data_present_flag của Fig.22. Bằng cách này, các đoạn đầu lát của các lát được mã hóa thành các gói thuộc "phạm vi" của gói được đặt rải rác tương ứng có thể được giảm kích cỡ do sử dụng cờ nêu trên trong đoạn đầu lát và bộ giải mã bất kỳ nhận dòng dữ liệu video, như các thiết bị mạng được thể hiện trong các hình vẽ từ Fig.12 đến Fig.16, đáp lại cờ vừa được nêu trong các lát đoạn đầu lát để sao chép dữ liệu đoạn đầu lát được kết hộp trong gói được đặt rải rác bên trong đoạn đầu lát của lát đã được mã hóa thành gói tải trọng thuộc về phạm vi của gói được đặt rải rác này trong trường hợp cờ tương ứng nằm trong lát đó tạo tín hiệu dịch chuyển dữ liệu đoạn đầu lát đến tiền tố lát, tức là, gói được đặt rải rác tương ứng.

Bắt đầu thêm với ví dụ cú pháp để thực hiện các phương án của các hình vẽ từ Fig.12 đến Fig.16, cú pháp thông báo SEI có thể là như được thể hiện trên Fig.24. Để đưa vào lát hoặc các loại thông báo SEI ảnh con, cú pháp tải trọng SEI có thể được thích ứng như được thể hiện trong bảng của Fig.25. Duy nhất thông báo SEI với payloadType nằm trong khoảng từ 180 đến 184 có thể được gửi ngoại trừ hai giá trị đầu mút trên mức ảnh con nằm trong đơn vị NAL tiền tố lát. Ngoài ra, các thông báo SEI vùng quan tâm với payloadType bằng 140 có thể được gửi đi trong đơn vị NAL tiền tố lát trên mức ảnh con, hoặc thông báo SEI đều đặn trên mức đơn vị truy cập.

Tức là, khi chuyển các chi tiết được thể hiện trên Fig.25 và Fig.24 trên các phương án đã được mô tả bên trên đối với các hình vẽ từ Fig.12 đến Fig.16, các gói được đặt rải rác được thể hiện trong các phương án này của các hình vẽ từ Fig.12 đến Fig.16 có thể được nhận ra bằng cách sử dụng các đơn vị NAL tiền tố lát với loại đơn vị NAL đã biết, ví dụ 24, gồm có loại đã biết của thông báo SEI được tạo tín hiệu bởi payloadType tại lúc bắt đầu mỗi thông báo SEI trong đơn vị NAL tiền tố lát. Trong phương án cú pháp cụ thể được mô tả bên dưới, payloadType = 180 và payloadType = 181 dẫn đến gói điều khiển định thời gian theo các phương án của các hình vẽ từ Fig.11 đến Fig.14, trong khi payloadType = 140 dẫn đến gói ROI theo phương án của

Fig.15, và payloadType = 182 dẫn đến gói nhận dạng ô theo phương án của Fig.16. Ví dụ cú pháp cụ thể được mô tả bên dưới có thể gồm có chỉ một hoặc tập con của các tùy chọn payloadType vừa được đề cập. Ngoài điều này, Fig.25 bộc lộ rằng bất kỳ phương án nào đã được mô tả bên trên của các hình vẽ từ Fig.11 đến Fig.16 có thể được kết hợp với nhau. Thậm chí hơn nữa, Fig.25 bộc lộ rằng bất kỳ phương án nào bên trên của các hình vẽ từ Fig.12 đến Fig.16, hoặc sự kết hợp bất kỳ của chúng, có thể được mở rộng bởi gói được đặt rải rác thêm, được giải thích thêm với payloadType = 184. Như được mô tả bên trên, sự mở rộng được mô tả bên dưới đối với payloadType = 183 kết thúc trong khả năng rằng các gói được đặt rải rác bất kỳ có thể được kết hợp trong dữ liệu đoạn đầu lát thông thường cho các đoạn đầu lát của các lát đã được mã hóa thành gói tải trọng bất kỳ thuộc phạm vi của chúng.

Các bảng trong các hình vẽ tiếp theo xác định các thông báo SEI mà các thông báo này có thể được sử dụng trên lát hoặc hoặc mức ảnh con. Thông báo SEI vùng quan tâm cũng được có mặt có thể được sử dụng trên mức ảnh con và đơn vị truy cập.

Fig.26 ví dụ thể hiện ví dụ về thông báo SEI nhớ đệm ảnh con xuất hiện không kể đến đơn vị NAL tiền tố lát thuộc loại đơn vị NAL 24 có loại thông báo SEI 180 nằm trong đó, do đó tạo ra gói điều khiển định thời gian.

Các ngữ nghĩa học có thể được định nghĩa như sau:

seq_parameter_set_id chỉ ra tập hợp tham số chuỗi chứa các thuộc tính HRD chuỗi. Giá trị của seq_parameter_set_id sẽ bằng với giá trị của seq_parameter_set_id trong tập hợp tham số ảnh được tham khảo bởi ảnh đã được mã hóa đầu tiên được kết hợp với thông báo SEI giai đoạn đệm. Giá trị của seq_parameter_set_id sẽ nằm trong khoảng từ 0 đến 31, bao gồm hai giá trị đầu mút.

initial_cpb_removal_delay[SchedSelIdx] và**initial_alt_cpb_removal_delay**[SchedSelIdx] chỉ ra các độ trễ loại bỏ CPB ban đầu đối với CPB thứ SchedSelIdx của đơn vị giải mã (ảnh con). Các phần tử cú pháp có độ dài tính bằng bit được đưa ra bởi initial_cpb_removal_delay_length_minus1 + 1, và là trong các đơn vị của đồng hồ 90kHz. Các giá trị của các phần tử cú pháp sẽ không bằng 0 và sẽ không vượt quá 90000 * (CpbSize[SchedSelIdx] ÷ BitRate[SchedSelIdx]), sự tương đương thời

gian của kích cỡ CPB trong các đơn vị đồng hồ 90kHz.

Trên toàn bộ chuỗi video đã được mã hóa, tổng của initial_cpb_removal_delay[SchedSelIdx] và initial_cpb_removal_delay_offset[SchedSelIdx] trên mỗi đơn vị giải mã (ảnh con) sẽ là không đổi cho mỗi giá trí của SchedSelIdx, và tổng của initial_alt_cpb_removal_delay[SchedSelIdx] và initial_alt_cpb_removal_delay_offset[SchedSelIdx] sẽ là không đổi cho mỗi giá trị của SchedSelIdx.

Fig.27 cũng thể hiện ví dụ cho thông báo SEI định thời gian ảnh con, trong đó các ngữ nghĩa có thể được định nghĩa như sau:

du_cpb_removal_delay chỉ ra có bao nhiêu nhịp đồng hồ để đợi sau khi loại bỏ khỏi CPB của đơn vị giải mã (ảnh con) được kết hợp với thông báo SEI khoảng thời gian nhớ đệm ảnh con gần nhất trong đơn vị truy cập đứng trước trong cùng đơn vị giải mã (ảnh con), nếu có mặt, nói cách khác được kết hợp với thông báo SEI khoảng thời gian nhớ đệm ảnh con gần nhất trong đơn vị truy cập đứng trước, trước khi loại bỏ khỏi bộ nhớ đệm dữ liệu đơn vị giải mã (ảnh con) được kết hợp với thông báo SEI định thời gian ảnh con. Giá trị này cũng được sử dụng để tính toán thời gian đến sớm nhất có thể của dữ liệu đơn vị giải mã (ảnh con) trong CPB cho HSS ((Hypothetical Stream Scheduler [2]0). Phần tử cú pháp có mã chiều dài cố định mà chiều dài của nó tính bằng bit được đưa ra bởi cpb_removal_delay_length_minus1 + 1. cpb_removal_delay là phần còn lại của bộ đếm modulo $2^{(cpb\_removal\_delay\_length\_minus1 + 1)}$.

du_dpb_output_delay được sử dụng để tính toán thời gian xuất ra DPB của đơn vị giải mã (ảnh con). Nó chỉ ra bao nhiêu nhịp đồng hồ để đợi sau khi loại bỏ bỏ đơn vị giải mã được giải mã (ảnh con) khỏi CPB trước khi đơn vị giải mã (ảnh con) của ảnh được xuất ra khỏi DPB.

Lưu ý rằng điều này cho phép các sự cập nhật ảnh con. Trong kịch bản này, các đơn giải mã không được cập nhật có thể vẫn không bị thay đổi ảnh đã được giải mã cuối cùng, cụ thể, chúng vẫn có thể nhìn thấy được.

Tóm tắt lại Fig.26 và Fig.27 và truyền các chi tiết cụ thể nằm trong các hình vẽ này lên phương án của các hình vẽ từ Fig.12 đến Fig.14, có thể nói rằng thời gian phục

hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã có thể được tạo tín hiệu trong gói điều khiển định thời gian kết hợp theo cách thức đã được mã hóa khác biệt, cụ thể gia tăng đối với thời gian phục hồi bộ nhớ đệm giải mã khác. Tức là, để thu được thời gian phục hồi bộ nhớ đệm bộ giải mã cho một đơn vị giải mã nhất định, bộ giải mã nhận dòng dữ liệu video bổ sung cộng thời gian phục hồi bộ giải mã được thu từ gói điều khiển định thời gian đứng trước đơn vị giải mã nhất định, vào thời gian phục hồi bộ giải mã của đơn vị giải mã đi ngay trước, tức là thời gian phục hồi bộ giải mã đứng trước đơn vị giải mã nhất định, và tiến hành theo cách thức này với các đơn vị giải mã tiếp theo. Tại lúc bắt đầu các chuỗi video đã được mã hóa của vài ảnh mà mỗi ảnh hoặc các phần của ảnh, gói điều khiển định thời gian có thể còn hoặc theo cách khác, gồm có giá trị thời gian phục hồi bộ nhớ đệm bộ giải mã được mã hóa tuyệt đối so với khác đối với bất kỳ thời gian phục hồi bộ nhớ đệm bộ giải mã của đơn vị giải mã đứng trước nào.

Fig.28 thể hiện việc làm thế nào thông báo SEI thông tin lát ảnh con có thể nhìn thế nào. Các ngữ nghĩa học có thể được định nghĩa như sau:

slice_header_data_flag với giá trị là 1 chỉ ra rằng dữ liệu đoạn đầu lát được thể hiện trong thông báo SEI. Dữ liệu đoạn đầu lát được cung cấp trong SEI là có giá trị đối với tất cả các lát tiếp theo trong thứ tự giải mã cho đến khi kết thúc đơn vị truy cập, sự xuất hiện của dữ liệu lát trong thông báo SEI khác, đơn vị NAL lát hoặc đơn vị NAL tiền tố lát.

Fig.29 thể hiện ví dụ thông báo SEI thông tin ô ảnh con, trong đó các ngữ nghĩa có thể được định nghĩa như sau:

tile_priority chỉ ra sự ưu tiên tất cả các ô trong các lát được đứng trước đi theo thứ tự giải mã. Giá trị của tile_priority sẽ nằm trong khoảng từ 0 đến 7, bao gồm hai giá trị đầu mút, trong đó 7 chỉ ra sự ưu tiên cao nhất.

multiple_tiles_in_prefixed_slices_flag = 1 chỉ ra rằng có nhiều hơn 1 ô trong các lát đứng trước đi theo sau trong thứ tự giải mã. multiple_tiles_in_prefixed_slices_flag có giá trị là 0 chỉ ra rằng các lát đứng trước tiếp theo chứa duy nhất một ô.

num_tiles_in_prefixed_slices_minus1 chỉ ra số ô trong các lát đứng trước theo sau trong thứ tự giải mã.

first_tile_id_in_prefixed_slices chỉ ra tile_id của ô thứ nhất trong các lát đứng trước đi theo thứ tự giải mã.

Tức là, phương án của Fig.16 có thể được thực hiện sử dụng cú pháp của Fig.29 để thấy rõ các gói nhận dạng ô được đề cập trên Fig.16. Như được thể hiện trong đó, cờ nhất định, ở đây multiple_tiles_in_prefixed_slices_flag, có thể được sử dụng để tạo tín hiệu bên trong gói nhận dạng ô được đặt rải rác chỉ một ô hoặc nhiều hơn một ô được chứa bởi phần con bất kỳ của ảnh hiện thời 18 đã được mã hóa thành gói tải trọng bất kỳ thuộc phạm vi của gói nhận dạng ô được đặt rải rác tương ứng. Nếu cờ tạo tín hiệu xếp chồng nhiều hơn một ô, một phần tử cú pháp thêm nữa nằm trong gói được đặt rải rác tương ứng, ở đây minh họa là num_tiles_in_prefixed_slices_minus1 cho biết số ô được xếp chồng bởi phần con bất kỳ của gói tải trọng bất kỳ thuộc phạm vi của gói nhận dạng ô được đặt rải rác tương ứng. Cuối cùng, thêm một phần tử cú pháp nữa, ở đây minh họa là first_tile_id_in_prefixed_slices, cho biết ID ô giữa nhiều ô được chỉ ra bởi gói nhận dạng ô được đặt rải rác hiện tại, nó là phần tử cú pháp đầu tiên theo thứ tự giải mã. Chuyển cú pháp của Fig.29 lên phương án của Fig.16, gói nhận dạng ô 72 đứng trước gói tải trọng 32 thứ năm có thể, ví dụ, có tất cả ba phần tử cú pháp vừa được thảo luận với multiple_tiles_in_prefixed_slices_flag được thiết lập bằng 1, num_tiles_in_prefixed_slices_minus1 được thiết lập bằng 1, do đó chỉ ra rằng hai ô thuộc phạm vi hiện tại, và first_tile_id_in_prefixed_slices được thiết lập bằng 3, chỉ ra rằng sự chạy các ô theo thứ tự giải mã thuộc về phạm vi của gói nhận dạng ô 72 hiện tại bắt đầu ở ô thứ ba (có tile_id = 2).

Fig.29 cũng bộc lộ rằng gói nhận dạng ô 72 có thể cũng chỉ ra tile_priority, cụ thể, ưu tiên của các ô thuộc phạm vi của chúng. Tương tự với khía cạnh ROI, thiết bị mạng 68 có thể sử dụng thông tin ưu tiên này để điều khiển các nhiệm vụ truyền dẫn như yêu cầu truyền dẫn lại các gói tải trọng nhất định.

Fig.30 thể hiện ví dụ cú pháp cho thông báo SEI thông tin kích thước ô ảnh con, trong đó các ngữ nghĩa có thể được định nghĩa như sau:

multiple_tiles_in_prefixed_slices_flag có giá trị là 1 chỉ ra rằng có nhiều hơn 1

ô trong các lát đứng trước đi theo thứ tự giải mã. multiple_tiles_in_prefixed_slices_flag có giá trị là 0 chỉ ra rằng các lát đứng trước tiếp theo chứa duy nhất một ô.

num_tiles_in_prefixed_slices_minus1 chỉ ra số ô trong các lát được đứng trước đi theo thứ tự giải mã.

tile_horz_start[i] chỉ ra sự bắt đầu theo chiều ngang của ô thứ i trong các điểm ảnh trong ảnh.

tile_width[i] chỉ ra độ rộng của ô thứ i trong các điểm ảnh trong ảnh.

tile_vert_start[i] chỉ ra sự bắt đầu theo chiều ngang của ô thứ i trong các điểm ảnh trong ảnh.

tile_height[i] chỉ ra độ rộng của ô thứ i trong các điểm ảnh trong ảnh.

Lưu ý rằng thông báo SEI kích thước ô có thể được sử dụng để hiển thị các hoạt động, ví dụ, để gán ô cho màn hình trong nhiều kịch bản hiển thị màn hình.

Fig.30 do đó bộc lộ rằng ví dụ cú pháp thực hiện của Fig.29 với các gói nhận dạng của Fig.16 có thể được biến đổi ở chỗ các ô thuộc phạm vi của gói nhận dạng ô tương ứng được chỉ ra bởi vị trí của chúng trong ảnh hiện thời 18 hơn là so với ID ô của chúng. Tức là, đúng hơn là tạo tín hiệu nhận dạng ô ô thứ nhất theo thứ tự giải mã được bao gồm bởi các phần con tương ứng đã được mã hóa thành bất kỳ gói tải trọng nào thuộc phạm vi của gói nhận dạng ô được đặt rải rác tương ứng, cho mỗi ô thuộc gói nhận dạng ô hiện tại, vị trí của nó có thể được tạo tín hiệu bằng cách tạo tín hiệu, ví dụ, vị trí góc bên trái phía trên của mỗi ô i, ở đây minh họa bởi tile_horz_start và tile_vert_start, và độ rộng và chiều cao của ô i, ở đây minh họa bởi tile_width và tile_height.

Ví dụ cú pháp cho vùng thông báo SEI quan tâm được thể hiện trên Fig.31. Chính xác hơn, Fig.32 thể hiện biến thể thứ nhất. Cụ thể, ví dụ thông báo SEI vùng quan tâm có thể được sử dụng trên mức đơn vị truy cập hoặc trên mức ảnh để tạo tín hiệu một hoặc nhiều vùng quan tâm. Theo biến thể thứ nhất của Fig.32, ROI riêng biệt được tạo tín hiệu trên mỗi thông báo ROI SEI, hơn là tạo tín hiệu tất cả các ROI của gói ROI thuộc phạm vi gói ROI tương ứng bên trong một thông báo SEI ROI nếu

nhiều ROI là nằm trong phạm vi hiện tại.

Theo Fig.31, thông báo SEI vùng quan tâm tạo tín hiệu từng ROI riêng biệt. Các ngữ nghĩa học có thể được định nghĩa như sau:

roi_id chỉ ra ký hiệu nhận dạng của vùng quan tâm.

roi_priority chỉ ra sự ưu tiên của tất cả các ô thuộc vùng quan tâm trong các lát đứng trước hoặc tất cả các lát đi theo thứ tự giải mã phụ thuộc vào liệu thông báo SEI được gửi trên mức ảnh con hay mức đơn vị truy cập. Giá trị của roi_priority sẽ nằm trong khoảng từ 0 đến 7, bao gồm cả hai giá trị đầu mút, trong đó 7 chỉ ra sự ưu tiên cao nhất. Nếu cả hai, roi_priority trong thông báo SEI thông tin ROI và tile_priority trong các thông báo SEI thông tin ô ảnh con đã cho, giá trị cao nhất của cả hai có giá trị cho sự ưu tiên các ô riêng biệt.

num_tiles_in_roi_minus1 chỉ ra số ô trong các lát đứng trước đi theo thứ tự giải mã thuộc vùng quan tâm.

roi_tile_id[i] chỉ ra tile_id của ô thứ i thuộc vùng quan tâm trong các lát đứng trước đi theo thứ tự giải mã.

Tức là, Fig.31 chỉ ra rằng gói ROI như được thể hiện trên Fig.15 có thể tạo tín hiệu nhận dạng vùng quan tâm mà gói ROI và gói tải trọng tương ứng thuộc phạm vi của nó đề cập đến. Tùy ý, chỉ số ưu tiên ROI có thể được tạo tín hiệu cùng với ROI ID. Tuy nhiên, cả hai phần tử cú pháp là tùy ý. Tiếp đó, phần tử cú pháp num_tiles_in_roi_minus1 có thể chỉ ra số ô trong phạm vi của gói ROI tương ứng thuộc ROI 60 tương ứng. Tiếp đó, roi_tile_id chỉ ra tile-ID của các ô thứ i thuộc ROI 60. Hình dung rằng, ví dụ, ảnh 18 được chia nhỏ thành các ô 70 theo cách thức được thể hiện trên Fig.16, và rằng ROI 60 của Fig.15 tương ứng với nửa bên trái của ảnh 18, được tạo ra bởi ô thứ nhất và thứ ba theo thứ tự giải mã. Tiếp đó, gói ROI có thể được đặt trước gói tải trọng thứ nhất 32 của đơn vị truy cập 30 trên Fig.16, tiếp theo đó là gói ROI nữa nằm giữa các gói tải trọng thứ tư và thứ năm 32 của đơn vị truy cập 30 này. Tiếp đó, gói ROI thứ nhất nên có num_tile_in_roi_minus1 được thiết lập bằng 0 với roi_tile_id[0] bằng 0 (do đó đề cập đến ô thứ nhất trong thứ tự giải mã), trong đó gói ROI thứ hai ở phía trước gói tải trọng thứ năm 32 có num_tiles_in_roi_minus1

được thiết lập bằng 0 với roi_tile_id[0] được thiết lập bằng 2 (do đó chỉ ra ô thứ ba trong thứ tự giải mã ở phần tư đáy bên trái của ảnh 18).

Theo biến thể thứ hai, cú pháp của thông báo SEI vùng quan tâm có thể được thể hiện như trên Fig.32. Ở đây, tất cả các ROI trong thông báo SEI đơn lẻ được tạo tín hiệu. Cụ thể, cùng cú pháp như được thảo luận bên trên đối với Fig.31 cần được sử dụng, nhưng nhân các phần tử cú pháp cho từng ROI trong nhiều ROI mà thông báo SEI ROI tương ứng hoặc gói ROI đề cập đến, với số được tạo tín hiệu bởi phần tử cú pháp, ở đây ví dụ là num_rois_minus1. Một cách tùy ý, phần tử cú pháp khác, ở đây là roi_presentation_on_separate_screen, có thể tạo tín hiệu cho từng ROI xem ROI tương ứng có là thích hợp để được thể hiện trên màn hình riêng biệt không.

Các ngữ nghĩa học có thể được định nghĩa:

num_rois_minus1 chỉ ra số lượng ROI trong các lát đứng trước hoặc các lát thông thường đi theo thứ tự giải mã.

roi_id[i] chỉ ra ký hiệu nhận dạng của vùng quan tâm thứ i.

roi_priority[i] chỉ ra sự ưu tiên của tất cả các ô thuộc vùng quan tâm thứ i trong các lát đứng trước hoặc tất cả các lát đi theo thứ tự giải mã phụ thuộc vào liệu thông báo SEI được gửi trên mức ảnh con hay mức đơn vị truy cập. Giá trị của roi_priority sẽ nằm trong khoảng từ 0 đến 7, bao gồm cả hai giá trị đầu mút, trong đó 7 chỉ ra sự ưu tiên cao nhất. Nếu cả hai, roi_priority trong thông báo SEI roi_info và tile_priority trong các thông báo SEI thông tin ô ảnh con được đưa ra, giá trị cao nhất của cả hai có giá trị cho sự ưu tiên các ô riêng biệt.

num_tiles_in_roi_minus1[i] chỉ ra số ô trong các lát đứng trước đi theo thứ tự giải mã thuộc vùng quan tâm thứ i.

roi_tile_id[i][n] chỉ ra tile_id của ô thứ n thuộc vùng quan tâm thứ i trong các lát đứng trước theo thứ tự giải mã.

roi_presentation_on_seperate_screen [i] chỉ ra rằng vùng quan tâm, kết hợp với roi_id thứ i là thích hợp để thể hiện trên màn hình riêng biệt.

Do đó, tóm lại một cách ngắn gọn các phương án khác nhau được mô tả đến giờ, chiến lược tạo tín hiệu cú pháp mức cao bổ sung được đưa ra cho phép áp dụng các

thông báo SEI cũng như các mục cú pháp mức cao bổ sung vượt các cái nằm trong đoạn đầu đơn vị NAL trên mỗi mức lát. Do đó, chúng tôi đã mô tả được đơn vị NAL tiền tố lát. Cú pháp và các ngữ nghĩa học của tiền tố lát và slice_level/các thông báo SEI ảnh con được mô tả cùng với các trường hợp sử dụng cho các hoạt động độ trễ thấp/CPB ảnh con, sự tạo tín hiệu ô và tạo tín hiệu ROI. Cú pháp mở rộng được đưa ra để tạo tín hiệu một phần của đoạn đầu lát của các lát tiếp theo trong tiền tố lát.

Cho mục đích trọn vẹn, Fig.33 thể hiện ví dụ khác nữa cho cú pháp mà cú pháp này có thể được sử dụng cho gói điều khiển định thời gian theo phương án của các hình vẽ từ Fig.12 đến Fig.14. Các ngữ nghĩa có thể là:

du_spt_cpb_removal_delay_increment chỉ ra khoảng thời gian, trong các đơn vị nhịp con đồng hồ, giữa các thời gian CPB không đáng kể của đơn vị giải mã cuối cùng trong đơn vị truy cập hiện tại và đơn vị giải mã được kết hợp với thông báo SEI thông tin đơn vị giải mã. Giá trị này cũng được sử dụng để tính toán thời gian đến sớm nhất có thể của dữ liệu đơn vị giải mã trong CPB cho HSS, như được chỉ ra trong Phụ lục C. Phần tử cú pháp được mô tả bởi mã độ dài định trước mà chiều dài của nó tính bằng bit được đưa ra bởi du_cpb_removal_delay_increment_length_minus1 + 1. Khi đơn vị giải mã được kết hợp với thông báo SEI thông tin đơn vị giải mã là đơn vị giải mã cuối cùng trong đơn vị truy cập hiện tại, giá trị của du_spt_cpb_removal_delay_increment sẽ là bằng 0.

dpb_output_du_delay_present_flag bằng 1 chỉ ra sự có mặt của phần tử cú pháp pic_spt_dpb_output_du_delay trong thông báo SEI thông tin đơn vị giải mã. dpb_output_du_delay_present_flag bằng 0 chỉ ra sự vắng mặt của phần tử cú pháp pic_spt_dpb_output_du_delay trong thông báo SEI thông tin đơn vị giải mã.

pic_spt_dpb_output_du_delay được sử dụng để tính toán thời gian xuất ra DPB của ảnh khi SubPicHrdFlag là bằng 1. Nó chỉ ra bao nhiêu nhịp đồng hồ con để đợi sau khi loại bỏ đơn vị giải mã cuối cùng trong đơn vị truy cập khỏi CPB trước khi ảnh đã được giải mã được xuất ra khỏi DPB. Khi không xuất hiện, giá trị của pic_spt_dpb_output_du_delay được suy ra là bằng pic_dpb_output_du_delay. Chiều dài của phần tử cú pháp pic_spt_dpb_output_du_delay được tính bằng bit bởi dpb_output_delay_du_length_minus1 + 1.

Nó là yêu cầu của sự tương thích dòng bit rằng tất cả các thông báo SEI thông tin đơn vị giải mã được kết hợp với cùng đơn vị truy cập, áp dụng cho cùng điểm hoạt động, và có dpb_output_du_delay_present_flag bằng 1 sẽ có cùng giá trị là pic_spt_dpb_output_du_delay. Thời gian xuất ra được dẫn ra từ pic_spt_dpb_output_du_delay của ảnh bất kỳ được xuất ra từ bộ giải mã tương thích sự định thời gian xuất ra đứng trước thời gian xuất ra nhận được từ pic_spt_dpb_output_du_delay của tất cả các ảnh trong CVS liên tục bất kỳ theo thứ tự giải mã.

Thứ tự xuất ảnh được thiết lập bởi các giá trị của phần tử cú pháp này sẽ là cùng thứ tự được thiết lập bởi các giá trị của PicOrderCntVal.

Đối với các ảnh không được xuất ra bởi quy trình "va đụng" bởi vì chúng đứng trước, theo thứ tự giải mã, ảnh IRAP với NoRaslOutputFlag bằng 1 có no_output_of_prior_pics_flag bằng 1 hoặc được suy ra là bằng 1, các thời gian xuất ra được thu từ pic_spt_dpb_output_du_delay sẽ tăng dần lên với việc tăng giá trị của PicOrderCntVal liên quan đến tất cả các ảnh nằm trong cùng CVS. Đối với hai ảnh bất kỳ trong CVS, sự khác biệt giữa các thời gian xuất ra của hai ảnh khi SubPicHrdFlag bằng 1 sẽ là giống với cùng sự khác biệt giữa các thời gian xuất ra của hai ảnh khi SubPicHrdFlag bằng 0.

Hơn nữa, Fig.34 thể hiện một ví dụ khác để tạo tín hiệu vùng ROI sử dụng các gói ROI. Theo Fig.34, cú pháp của gói ROI gồm có chỉ một cờ cho biết liệu tất cả các phần con của ảnh 18 được mã hóa thành gói tải trọng 32 bất kỳ thuộc phạm vi của nó thuộc ROI hay không. "Phạm vi" mở rộng đến sự xuất hiện của gói ROI hoặc thông báo SEI region_refresh_info. Nếu cờ là 1, vùng được chỉ ra được mã hóa thành (các) gói tải trọng liên tục tương ứng, và nếu là 0 áp dụng ngược lại, cụ thể, các phần con tương ứng của ảnh 18 không thuộc về ROI 60.

Trước khi thảo luận một số phương án nêu trên lại lần nữa, nói cách khác với việc giải thích thêm một số thuật ngữ được sử dụng bên trên như ô, lát và sự chia nhỏ dòng con WPP, cần lưu ý rằng sự tạo tín hiệu mức cao của các phương án trên có thể theo cách khác được xác định trong các đặc tả kỹ thuật vận chuyển như [3-7]. Nói cách khác, các gói được đề cập bên trên và tạo ra chuỗi 34 gồm các gói có thể là các gói vận

chuyển mà một số trong số các gói vận chuyển này có các phần con của lớp áp dụng như các lát, được kết hợp, như được tạo gói đầy đủ hoặc được chia đoạn, trong đó, một số gói vận chuyển khác lại được đặt rải rác giữa chuỗi các gói theo một cách thức, và với mục đích được thảo luận bên trên. Nói cách khác, các gói được đặt rải rác nêu trên không bị giới hạn được định ra là các thông báo SEI của các loại đơn vị NAL khác, được định ra trong bộ mã hóa-giải mã video của lớp áp dụng, nhưng có thể là gói vận chuyển vượt quá được định ra trong các giao thức vận chuyển.

Nói cách khác, theo một khía cạnh của đơn sáng chế, các phương án trên đã bộc lộ dòng dữ liệu video có nội dung video được mã hóa trong đó, trong các đơn vị phần con (xem các khối cây mã hóa hoặc các lát) của các ảnh của nội dung video, mỗi phần con được mã hóa tương ứng thành một hoặc nhiều gói tải trọng (xem các đơn vị NAL VCL) của chuỗi các gói (các đơn vị NAL) của dòng dữ liệu video, chuỗi các gói được chia thành chuỗi các đơn vị truy cập để mỗi đơn vị truy cập tập hợp các gói tải trọng đề cập đến ảnh tương ứng của nội dung video, trong đó chuỗi các gói được đặt rải rác vào đó các gói điều khiển định thời gian (tiền tố lát) để các gói điều khiển định thời gian chia nhỏ các đơn vị truy cập do đó ít nhất một số đơn vị truy cập được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã, thu được từ mỗi gói điều khiển định thời gian, thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã, các gói tải trọng của đơn vị giải mã đi theo gói điều khiển định thời gian tương ứng trong chuỗi các gói, và phục hồi các đơn vị giải mã từ bộ nhớ đệm của bộ giải mã được lập biểu ở các thời gian được định ra bởi các thời gian phục hồi bộ nhớ đệm bộ giải mã cho các đơn vị giải mã.

Như mô tả bên trên, miền mà đối với miền này, nội dung video được mã hóa thành dòng dữ liệu trong các đơn vị phần con của các ảnh, có thể bao phủ các phần tử cú pháp đề cập đến mã hóa dự báo như các chế độ mã hóa (như trong chế độ, liên chế độ, thông tin chia nhỏ và tương tự), các tham số dự báo (như các vectơ chuyển động, các hướng ngoại suy hoặc tương tự) và/hoặc dữ liệu phần còn lại (như các mức hệ số biến đổi, với các phần tử cú pháp này được kết hợp với các phần cục bộ của ảnh như các khối cây mã hóa, các khối dự báo và các khối còn lại (như biến đổi) tương ứng.

Như được mô tả bên trên, các gói tải trọng mỗi gói có thể bao gồm một hoặc

nhiều lát (trọn vẹn, tương ứng). Các lát có thể được giải mã độc lập hoặc có thể thể hiện mối tương quan gây trở ngại cho việc giải mã độc lập các lát. Ví dụ, các lát entrôpi có thể được giải mã entrôpi độc lập, nhưng sự dự báo ngoài các biên lát có thể bị cấm. Các lát phụ thuộc có thể cho phép xử lý WPP, tức là mã hóa/giải mã sử dụng mã hóa hóa entrôpi và dự báo vượt ra ngoài các biên lát với khả năng mã hóa/giải mã song song các lát phụ thuộc theo cách thức xếp chồng thời gian với, tuy nhiên, việc bắt đầu lệch thủ tục mã hóa/giải mã của các lát phụ thuộc riêng biệt và lát/các lát được đề cập tới bởi các lát phụ thuộc.

Thứ tự liên tiếp mà tại đó các gói tải trọng của đơn vị truy cập được sắp đặt trong đơn vị truy cập tương ứng có thể được biết đến cho bộ giải mã trước. Ví dụ, thứ tự mã hóa/giải mã có thể được định ra giữa các phần con của các ảnh như thứ tự quét giữa các khối cây mã hóa trong các ví dụ bên trên.

Xem, ví dụ, hình vẽ bên dưới. Ảnh 100 đã được mã hóa/được giải mã hiện thời có thể được chia ra thành các ô, trên Fig.35 và Fig.36, các ô này tương ứng với bốn góc của ảnh 110 và được chỉ ra với các số tham chiếu 112a-112d. Tức là, toàn bộ ảnh 110 có thể tạo ra một ô như trong trường hợp của Fig.37 hoặc có thể được chia đoạn thành nhiều hơn một ô. Sự chia đoạn ô có thể được giới hạn đối với các ô đều đặn trong đó các ô được sắp đặt chỉ trong các cột và các hàng. Các ví dụ khác nhau được thể hiện dưới đây:

Như có thể được thấy, ảnh 110 còn được chia nhỏ thành các khối (cây) mã hóa (các hộp nhỏ trong hình và được gọi là CTB trên) 114 thứ tự mã hóa 116 được định ra giữa các khối mã hóa này (ở đây, theo thứ tự quét mành, nhưng cũng có thể là khác). Sự chia nhỏ ảnh thành các ô 112a-d có thể được giới hạn sao cho các ô là các tập hợp tháo rời gồm các khối 114. Ngoài ra, cả hai khối 114 và các ô 112a-d có thể được giới hạn đối với sự sắp đặt đều đặn thành trong các cột và các hàng.

Nếu các ô (tức là, nhiều hơn một ô) có mặt, thì thứ tự mã hóa (giải mã) 116 quét mành ô hoàn thiện thứ nhất trước tiên sau đó chuyển - cũng theo thứ tự ô quét mành - đến ô kế tiếp theo thứ tự ô.

Vì các ô có thể được mã hóa/giải mã độc lập với nhau do không đi qua các biên ô bởi các dự báo không gian và các lựa chọn ngữ cảnh được suy ra từ vùng lân cận

không gian, bộ mã hóa 10 và bộ giải mã 12 có thể mã hóa/giải mã ảnh được chia nhỏ thành các ô 112 (trước đây được chỉ ra bởi 70), song song, độc lập với nhau – ngoại trừ, ví dụ, lọc vòng lặp hoặc lọc sau có thể được phép để qua các biên ô.

Ảnh 110 có thể còn được chia nhỏ thành các lát 118a-d, 180 - trước đây được chỉ ra sử dụng số tham chiếu 24. Lát có thể chứa chỉ một phần của ô, một ô hoàn thiện hoặc nhiều hơn một ô hoàn thiện. Do đó, sự chia thành các lát cũng có thể chia nhỏ các ô như trong trường hợp của Fig.35. Mỗi lát gồm có ít nhất một khối mã hóa 114 hoàn thiện và được tạo nên bởi các khối mã hóa liên tiếp 114 theo thứ tự giải mã 116 do đó thứ tự được định ra giữa các lát 118a-d mà theo sau đó các chỉ số trong hình vẽ được gán. Sự chia lát trong các hình vẽ từ Fig.35 đến Fig.37 được chọn chỉ cho các mục đích minh họa. Các biên ô có thể được tạo tín hiệu trong dòng dữ liệu. Ảnh 110 có thể tạo ra một ô đơn lẻ như được mô tả trên Fig.37.

Bộ mã hóa 10 và bộ giải mã 12 có thể được tạo cấu hình để tuân theo các biên ô trong đó dự báo không gian không được áp dụng qua các biên ô. Sự thích ứng ngữ cảnh, cụ thể, các thích ứng xác suất của các ngữ cảnh (số học) entrôpi khác nhau có thể tiếp tục trên toàn bộ các lát. Tuy nhiên, liệu lát đi qua - theo thứ tự mã hóa 116 - biên ô (nếu xuất hiện bên trong phía trong của lát) như trên Fig.36 đối với các lát 118a, b thì lát, lần lượt, được chia nhỏ thành các phần con (các dòng con hoặc các ô) với lát gồm các điểm (xem entry_point_offset) chỉ ra các sự bắt đầu của mỗi phần con. Trong các bộ lọc vòng bộ giải mã có thể được phép qua các biên ô. Các bộ lọc này có thể bao gồm một hoặc nhiều bộ lọc phân rã khối, bộ lọc độ lệch thích ứng mẫu (SAO - Sample Adaptive Offset) và bộ lọc vòng thích ứng (ALF - Adaptive loop filter). ALF có thể được áp dụng cho các biên ô/lát nếu được kích hoạt.

Mỗi phần con thứ hai và tiếp theo tùy ý có thể có phần mở đầu của chúng được định vị byte được sắp hàng trong lát với con trỏ chỉ ra độ lệch từ lúc bắt đầu một phần con đến lúc bắt đầu phần con kế tiếp. Các phần con được sắp đặt trong các lát theo thứ tự quét 116. Fig.38 thể hiện ví dụ với lát 180c của Fig.37 được chia nhỏ thành các đoạn con 119_i.

Đối với các hình vẽ lưu ý rằng các lát tạo thành các phần con của các ô không phải kết thúc bằng hàng trong ô 112a. Xem, ví dụ lát 118a trên Fig.37 và Fig.38.

Hình vẽ bên dưới thể hiện phần ví dụ của dòng dữ liệu đề cập đến đơn vị truy cập được gắn với ảnh 110 của Fig.38 phía trên). Ở đây, mỗi gói tải trọng 122a-d – trước đây được chỉ ra bởi ký hiệu tham chiếu 32 – chỉ cung cấp một lát 118a. Hai gói điều khiển định thời gian 124a,b – trước được chỉ ra bởi số tham chiếu 36 -được thể hiện là được đặt rải rác trong đơn vị truy cập 120 cho các mục đích minh họa: 124a đặt trước gói 122a trong thứ tự gói 126 (tương ứng với trục thời gian giải mã/mã hóa) và 124b đặt trước gói 122c. Do đó, đơn vị truy cập 120 được chia thành hai đơn vị giải mã 128a,b – trước đây được chỉ ra bởi ký hiệu tham chiếu 38 - , đơn vị giải mã thứ nhất gồm có các gói 122a,b (cùng với các gói dữ liệu điền đầy tùy ý (lần lượt kế tiếp các gói thứ nhất và thứ hai 122a,b) và đơn vị truy cập tùy ý dẫn các gói SEI (đứng trước gói thứ nhất 122a)) và đơn vị giải mã thứ hai gồm có các gói 118c,d (cùng với các gói dữ liệu điền đầy tùy ý (lần lượt kế tiếp các gói 122c,d)).

Như được mô tả bên trên, mỗi gói của chuỗi các gói có thể được gán cho chính xác một gói trong số nhiều loại gói (nal_unit_type). Các gói tải trọng và các gói điều khiển định thời gian (và dữ liệu điền đầy tùy ý và các gói SEI), ví dụ, là các loại gói khác nhau. Các sự tạo nấc các gói của một loại gói nhất định trong chuỗi các gói có thể là tùy theo các giới hạn nhất định. Các giới hạn này có thể định ra thứ tự giữa các loại gói (xem Fig.17) mà thứ tự này được tuân theo bởi các gói nằm trong mỗi đơn vị truy cập do đó các biên đơn vị truy cập 130a,b có thể dò tìm được, và vẫn ở cùng vị trí trong chuỗi các gói, dù nếu các gói thuộc loại gói có thể loại bỏ bất kỳ được loại bỏ khỏi dòng dữ liệu video. Ví dụ, các gói tải trọng thuộc loại gói không có thể loại bỏ. Tuy nhiên, các gói điều khiển định thời gian, các gói dữ liệu điền đầy và các gói SEI có thể, như thảo luận bên trên, là loại gói có thể loại bỏ, cụ thể, chúng có thể là các đơn vị NAL không phải VCL.

Trong ví dụ trên, các gói điều khiển định thời gian được minh họa một cách rõ ràng bên trên bởi cú pháp slice_prefix_rbsp().

Bằng cách sử dụng việc đặc rải rác các gói điều khiển định thời gian như vậy, bộ mã hóa được phép điều chỉnh sự lập lịch biểu bộ nhớ đệm tại phía bộ giải mã trong quy trình mã hóa các ảnh riêng biệt của nội dung video. Ví dụ, bộ mã hóa được phép tối ưu hóa sự lập lịch biểu bộ nhớ đệm để giảm đến mức tối thiểu độ trễ đầu đến cuối.

Về điều này, bộ mã hóa được phép tính đến việc phân phối riêng biệt việc mã hóa phức tạp trên vùng ảnh của nội dung video cho các ảnh riêng biệt của nội dung video. Cụ thể, bộ mã hóa có thể liên tục xuất ra chuỗi các gói 122, 122a-d, 122a-d_{1-3} trên cơ sở gói theo gói (tức là, ngay khi gói hiện tại được hoàn thành nó là đầu ra). Bằng cách sử dụng các gói điều khiển định thời gian, bộ mã hóa có thể điều chỉnh sự lập lịch biểu bộ nhớ đệm tại phía giải mã tại các thời điểm mà ở đó một số phần con của ảnh hiện tại đã được mã hóa thành các gói tải trọng tương ứng với việc giữ lại các phần con, tuy nhiên, vẫn chưa được mã hóa.

Theo đó, bộ mã hóa để mã hóa nội dung video thành dòng dữ liệu video trong các đơn vị phần con (xem các khối cây mã hóa, các ô hoặc các lát) của các ảnh của nội dung video, với việc mã hóa lần lượt mỗi phần con thành một hoặc nhiều gói tải trọng (xem các đơn vị NAL VCL) của chuỗi các gói (các đơn vị NAL) của dòng dữ liệu video để chuỗi các gói được chia nhỏ thành chuỗi các đơn vị truy cập và mỗi đơn vị truy cập tập hợp các gói tải trọng đề cập đến ảnh tương ứng của nội dung video, có thể được tạo cấu hình để đặt rải rác các gói điều khiển định thời gian (tiền tố lát) bên trong các chuỗi các gói do đó các gói điều khiển định thời gian chia nhỏ các đơn vị truy cập thành các đơn vị giải mã do đó ít nhất một số đơn vị truy cập được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã, với mỗi gói điều khiển định thời gian tạo tín hiệu thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã, các gói tải trọng của chúng đi theo gói điều khiển định thời gian tương ứng trong chuỗi các gói.

Bộ giải mã bất kỳ nhận dòng dữ liệu video vừa được chỉ ra là tự do để khai thác thông tin lập lịch biểu nằm trong gói điều khiển định thời gian hoặc không. Tuy nhiên, trong khi bộ giải mã là tự do khai thác thông tin, bộ giải mã tương thích với mức bộ mã hóa bộ giải mã phải có thể giải mã dữ liệu đi theo sự định thời gian được chỉ ra. Nếu sự khai thác diễn ra, bộ giải mã nạp bộ nhớ đệm bộ giải mã của nó và làm rỗng bộ nhớ đệm bộ giải mã của nó trong các đơn vị của các đơn vị giải mã. "Bộ nhớ đệm bộ giải mã" như được mô tả bên trên, có thể bao gồm bộ nhớ đệm cho ảnh đã được giải mã và/hoặc bộ nhớ đệm cho ảnh đã được mã hóa.

Theo đó, bộ giải mã để giải mã dòng dữ liệu video có nội dung video được mã hóa trong dòng dữ liệu video, trong các nhóm phần con (xem các khối cây mã hóa, các

ô hoặc các lát) của các ảnh của nội dung video, mỗi phần con được mã hóa tương ứng thành một hoặc nhiều gói tải trọng (xem các đơn vị NAL VCL) của chuỗi các gói (các đơn vị NAL) của dòng dữ liệu video, chuỗi các gói được chia thành chuỗi các đơn vị truy cập để mỗi đơn vị truy cập tập hợp các gói tải trọng đề cập đến ảnh tương ứng của nội dung video, có thể được tạo cấu hình để tìm kiếm các gói điều khiển định thời gian được đặt rải rác trong chuỗi các gói, chia nhỏ các đơn vị truy cập do đó ít nhất một số đơn vị truy cập được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã, suy ra từ mỗi gói điều khiển định thời gian, thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã, các gói tải trọng của đơn vị giải mã đi theo gói điều khiển định thời gian tương ứng trong chuỗi các gói, và phục hồi các đơn vị giải mã từ bộ nhớ đệm của bộ giải mã được lập biểu ở các thời gian được định ra bởi các thời gian phục hồi bộ nhớ đệm bộ giải mã cho các đơn vị giải mã.

Tìm kiếm gói điều khiển định thời gian có thể bao gồm bộ giải mã kiểm tra đoạn đầu đơn vị NAL và phần tử cú pháp được bao gồm trong đó, cụ thể nal_unit_type. Nếu giá trị của cờ về sau bằng với giá trị nào đó, tức là, phù hợp với các ví dụ bên trên, 124, tiếp đó gói được kiểm tra hiện tại là gói điều khiển định thời gian. Tức là, gói điều khiển định thời gian gồm hoặc chuyển thông tin được giải thích bên trên đối với mã giả subpic_buffering cũng như subpic_timing. Tức là, các gói điều khiển định thời gian có thể chuyển hoặc chỉ ra các độ trễ loại bỏ CPB ban đầu cho bộ giải mã hoặc chỉ ra có bao nhiêu nhịp đồng hồ để đợi sau khi loại bỏ khỏi CPB của đơn vị giải mã tương ứng.

Để cho phép truyền lặp đi lặp lại các gói điều khiển định thời gian mà không chia nhỏ thêm một cách không định trước đơn vị truy cập thành thêm các đơn vị giải mã, cờ nằm trong các gói điều khiển định thời gian có thể tạo tín hiệu một cách rõ ràng liệu gói điều khiển định thời gian hiện tại có tham gia vào sự chia nhỏ đơn vị truy cập thành các đơn vị mã hóa hay không (so sánh decoding_unit_start_flag = 1 chỉ ra sự bắt đầu đơn vị giải mã, và decoding_unit_start_flag = 0 tạo tín hiệu trường hợp ngược lại).

Khía cạnh về sử dụng đơn vị giải mã được đặt rải rác đề cập đến thông tin nhận dạng ô là khác với khía cạnh về sử dụng đơn vị giải mã được đặt rải rác đề cập đến các gói điều khiển định thời gian ở chỗ các gói nhận dạng ô được đặt rải rác bên trong

dòng dữ liệu. Các gói điều khiển định thời gian được đề cập bên trên có thể hơn nữa được đặt rải rác trong dòng dữ liệu hoặc các thời gian phục hồi bộ nhớ đệm bộ giải mã có thể được truyền cùng với thông tin nhận dạng ô được giải thích bên dưới trong cùng một gói thông thường. Do đó, các chi tiết được thể hiện trong phần trên có thể được sử dụng để làm rõ các vấn đề trong sự mô tả bên dưới.

Khía cạnh nữa của đơn sáng chế có thể nhận được từ các phương án nêu trên bộc lộ dòng dữ liệu video có nội dung video được mã hóa trong đó, sử dụng mã hóa dự báo và mã hóa entrôpi, trong các đơn vị lát mà các ảnh của nội dung video được chia nhỏ không gian thành các đơn vị lát này, sử dụng thứ tự mã hóa giữa các lát, với việc giới hạn các dự báo của mã hóa dự báo và/hoặc mã hóa entrôpi đối với bên trong của các ô mà các ảnh của nội dung video được chia nhỏ không gian thành các ô này, trong đó chuỗi các lát theo thứ tự mã hóa được tạo gói thành các gói tải trọng của chuỗi các gói (đơn vị NAL) của dòng dữ liệu video theo thứ tự mã hóa, chuỗi các gói được chia thành chuỗi các đơn vị truy cập do đó mỗi đơn vị truy cập tập hợp các gói tải trọng đã được tạo gói trong đó các lát đề cập đến ảnh tương ứng của nội dung video, trong đó chuỗi các gói có các gói nhận dạng ô được đặt rải rác trong đó nhận dạng các ô (chỉ gói nhận dạng ô tiềm năng) mà các gói nhận dạng được chồng lên bởi các lát được tạo gói thành một hoặc nhiều gói tải trọng ngay theo sau gói nhận dạng ô tương ứng trong chuỗi các gói.

Xem, ví dụ, hình vẽ đứng ngay trước thể hiện dòng dữ liệu. Các gói 124a và 124b sẽ mô tả các gói nhận dạng ô. Hoặc bằng cách tạo tín hiệu rõ ràng (so sánh single_slice_flag = 1) hoặc trên mỗi quy ước, gói nhận dạng ô có thể chỉ nhận dạng các ô mà các ô này được xếp chồng bởi các lát được tạo gói thành gói tải trọng ngay theo sau 122a. Hoặc, bằng cách tạo tín hiệu rõ ràng hoặc trên mỗi quy ước, gói nhận dạng ô 124a có thể nhận dạng các ô mà các ô này được xếp chồng bởi các lát được tạo gói thành một hoặc nhiều gói tải trọng ngay theo sau gói nhận dạng ô 124 tương ứng trong chuỗi các gói cho đến khi sớm kết thúc 130b của đơn vị truy cập hiện thời 120 và bắt đầu đơn vị giải mã 128b kế tiếp tương ứng. Xem, ví dụ, Fig.35: nếu mỗi lát 118a-d_{1-3} được tạo gói riêng biệt thành gói tương ứng 122a-d_{1-3}, với sự chia nhỏ thành các đơn vị giải mã sao cho các gói được nhóm thành ba đơn vị giải mã theo {$122a_{1-3}$},{$122b_{1-3}$} và {$122c_{1-3}$, $122d_{1-3}$}, thì các lát {$118c_{1-3}$, $118d_{1-3}$} được tạo gói thành các

gói {$122c_{1-3}$, $122d_{1-3}$} của đơn vị giải mã thứ ba, ví dụ, xếp chồng các ô 112c và 112d, và tiền tố lát tương ứng chỉ ra "c" và "d", cụ thể, các ô 112c và 112d này khi đề cập đến đơn vị giải mã hoàn thiện.

Do đó, thiết bị mạng được đề cập thêm bên dưới có thể sử dụng sự tạo tín hiệu rõ ràng này hoặc sự quy ước để kết hợp một cách chính xác mỗi gói nhận dạng ô với một hoặc nhiều gói tải trọng theo ngay sau gói nhận dạng trong chuỗi các gói. Cách nhận dạng có thể được tạo tín hiệu được mô tả minh họa phía trên bởi mã giả subpic_tile_info. Các gói tải trọng được kết hợp được đề cập bên trên là "các lát đứng trước". Thường, ví dụ có thể được biến đổi. Ví dụ, các phần tử cú pháp "tile_priority" có thể được để lại. Hơn nữa, thứ tự giữa các phần tử cú pháp có thể được chuyển đổi và bộ mô tả đề cập đến các độ dài bit và các nguyên tắc mã hóa của các phần tử cú pháp chỉ mang tính minh họa.

Thiết bị mạng nhận dòng dữ liệu video (tức là, dòng dữ liệu video có nội dung video được mã hóa trong đó, sử dụng mã hóa dự báo và entrôpi, trong các đơn vị lát mà các ảnh của nội dung video có thể được chia nhỏ không gian thành các lát này, sử dụng thứ tự mã hóa giữa các lát, với giới hạn các sự dự báo của mã hóa dự báo và/hoặc mã hóa entrôpi đối với bên trong của các ô mà các ảnh của nội dung video được chia nhỏ không gian thành các ô này, trong đó chuỗi các lát theo thứ tự mã hóa được tạo gói thành các gói tải trọng của chuỗi các gói (các đơn vị NAL) của dòng dữ liệu video theo thứ tự mã hóa, chuỗi các gói được chia thành chuỗi các đơn vị truy cập do đó mỗi đơn vị truy cập tập hợp các gói tải trọng đã được tạo gói trong đó các lát đề cập đến ảnh tương ứng của nội dung video, trong đó chuỗi các gói có các gói nhận dạng ô được đặt rải rác trong đó) có thể được tạo cấu hình để nhận dạng, dựa trên các gói nhận dạng ô, các ô được xếp chồng bởi các lát được tạo gói thành một hoặc nhiều gói tải trọng ngay theo sau gói nhận dạng ô tương ứng trong chuỗi các gói. Thiết bị mạng có thể sử dụng kết quả nhận dạng để quyết định các nhiệm vụ truyền dẫn. Ví dụ, thiết bị mạng có thể xử lý các ô khác nhau với ưu tiên khác nhau để phát lại. Ví dụ, trong trường hợp tổn hao gói, các gói tải trọng đó liên quan đến các ô có tính ưu tiên cao hơn có thể được ưu tiên để truyền lại trên các gói tải trọng liên quan đến các ô có ưu tiên thấp hơn. Tức là, thiết bị mạng có thể trước tiên yêu cầu truyền lại các gói tải trọng đã bị tổn hao liên quan đến các ô có ưu tiên cao hơn. Chỉ trong trường hợp đủ thời gian còn

lại (phụ thuộc vào tốc độ truyền) thiết bị mạng tiến hành yêu cầu truyền lại các gói tải trọng đã bị mất liên quan đến các ô có ưu tiên thấp hơn. Tuy nhiên, thiết bị mạng có thể cũng là khối phát lại mà khối này có thể gán các ô hoặc các gói tải trọng liên quan đến các ô nhất định cho các màn hình khác nhau.

Đối với khía cạnh sử dụng vùng thông tin quan tâm được đặt rải rác, cần lưu ý rằng các gói ROI được đề cập bên dưới có thể cùng tồn tại với các gói điều khiển định thời gian được đề cập bên trên và/hoặc các gói nhận dạng ô, hoặc bằng cách kết hợp nội dung thông tin của nó trong các gói thông thường như được mô tả bên trên đối với các tiền tố lát, hoặc trong dạng các gói riêng biệt.

Nói cách khác, khía cạnh sử dụng vùng thông tin quan tâm được đặt rải rác như được mô tả bên trên bộc lộ dòng dữ liệu video có nội dung video được mã hóa trong đó, sử dụng mã hóa dự báo và entrôpi, trong các đơn vị lát mà các ảnh của nội dung video được chia nhỏ không gian thành các lát này, sử dụng thứ tự mã hóa giữa các lát, với giới hạn các dự báo và/hoặc mã hóa entrôpi của mã hóa dự báo đối với bên trong của các ô mà các ảnh của nội dung video được chia thành các ô này, trong đó chuỗi các lát theo thứ tự mã hóa được tạo gói thành các gói tải trọng của chuỗi các gói (đơn vị NAL) của dòng dữ liệu video theo thứ tự mã hóa, chuỗi các gói được chia thành chuỗi các đơn vị truy cập sao cho mỗi đơn vị truy cập tập hợp các gói tải trọng đã được tạo gói thành các các gói trong đó các lát đề cập đến ảnh tương ứng của nội dung video, trong đó chuỗi các gói có các gói ROI được đặt rải rác trong đó nhận dạng các ô của các ảnh thuộc về ROI của các ảnh tương ứng.

Đối với các gói ROI, các nhận xét tương tự có giá trị như những quy định trước đối với các gói nhận dạng ô: các gói ROI có thể nhận dạng các ô của các ảnh mà các ảnh này thuộc về ROI của ảnh chỉ nằm giữa các ô mà ô này được chồng lên bởi các lát nằm trong một hoặc nhiều gói tải trọng, mà thuộc về một ROI của hình ảnh chỉ đơn thuần giữa các quân bài đó được phủ lên bằng các lát chứa trong một hoặc nhiều gói tải trọng mà gói tin ROI tương ứng đề cập đến bằng cách của mình ngay trước một hoặc nhiều gói tải trọng như đã nêu trên đối với các "lát đứng trước".

Các gói ROI có thể cho phép nhận dạng nhiều hơn một gói ROI trên mỗi lát đứng trước với nhận dạng các ô kết hợp trên mỗi ROI này (xem num_rois_minus1). Tiếp đó,

đối với mỗi ROI, sự ưu tiên có thể được truyền dẫn cho phép xếp hạng ROI theo tính ưu tiên (xem roi_priority[i]). Để cho phép "bám sát" các ROI theo thời gian trong chuỗi ảnh của video, mỗi ROI có thể được lập chỉ dẫn bởi chỉ số ROI để các ROI được chỉ dẫn trong các gói ROI có thể được kết hợp với nhau để vượt qua/qua các biên ảnh, cụ thể theo thời gian (xem roi_id[i]).

Thiết bị mạng nhận dòng dữ liệu video (tức là, dòng dữ liệu video có nội dung video được mã hóa trong đó, sử dụng mã hóa dự báo và entrôpi, trong các đơn vị lát mà ảnh của nội dung video được chia nhỏ không gian thành các đơn vị lát này, sử dụng thứ tự mã hóa giữa các lát, với việc giới hạn các sự dự báo của mã hóa dự báo đối với bên trong của các ô mà các ảnh của nội dung video được chia ra thành các ô này, trong khi tiếp tục sự thích ứng xác suất mã hóa entrôpi trên toàn bộ các lát, trong đó chuỗi lát theo thứ tự mã hóa được phân chia thành các gói tải trọng của chuỗi các gói (các đơn vị NAL) của dòng dữ liệu video theo thứ tự mã hóa, chuỗi các gói được chia thành chuỗi các đơn vị truy cập do đó mỗi đơn vị truy cập tập hợp các gói tải trọng đã được phân chia thành các lát liên quan đến ảnh tương ứng của nội dung video có thể được tạo cấu hình để nhận dạng, dựa trên các gói nhận dạng ô, các gói phân chia các lát mà các lát này chồng lên các ô mà các ô này thuộc về ROI của các ảnh.

Thiết bị mạng có thể khai thác thông tin được chuyển bởi gói ROI theo cách thức tương tự như được giải thích bên trên trong phần trước đề cập đến các gói nhận dạng ô.

Đối với phần hiện thời cũng như phần trước, cần lưu ý rằng thiết bị mạng bất kỳ, như MANE hoặc bộ giải mã, có thể xác định được ô hoặc các ô được phủ bởi lát hoặc các lát của gói tải trọng được xem xét hiện thời, đơn giản bằng cách xem xét thứ tự lát của các lát của các ảnh và xem xét sự phát triển của một phần của ảnh hiện thời mà các lát này bao phủ, đối với vị trí của các ô trong ảnh, các lát này có thể được tạo tín hiệu một cách rõ ràng trong dòng dữ liệu như được giải thích bên trên hoặc có thể được biết đến đối với bộ mã hóa và bộ giải mã bởi quy ước. Hoặc, từng lát (ngoại trừ lát thứ nhất của ảnh theo thứ tự quét) có thể được cung cấp chỉ dẫn/chỉ số (slice_address được đo trong các đơn vị của các khối cây mã hóa) của khối cây mã hóa thứ nhất (ví dụ, CTB) cùng đề cập đến (cùng các mã) do đó bộ giải mã có thể đặt từng lát (sự tái tạo của nó)

vào trong ảnh từ khối mã hóa thứ nhất theo hướng của thứ tự quét. Do đó, nó có thể đáp ứng nếu các gói thông tin ô nêu trên chỉ bao gồm chỉ số của ô thứ nhất (first_tile_id_in_prefixed_slices) được chồng lên bởi lát bất kỳ của một hoặc nhiều gói tải trọng kết hợp ngay theo sau gói nhận dạng ô tương ứng vì nó là rõ ràng đối với thiết bị mạng nhờ gặp gói nhận dạng ô kế tiếp trong hàng mà nếu chỉ số được chuyển bởi gói nhận dạng ô về sau khác với gói nhận dạng ô trước bởi nhiều hơn một, thì các gói tải trọng giữa hai gói nhận dạng ô đó bao gồm các ô có chỉ số ô giữa chúng. Như được đề cập bên trên, điều này là đúng nếu cả sự chia nhỏ ô và sự chia nhỏ khối mã hóa, ví dụ, được dựa trên sự chia nhỏ từng hàng/cột có thứ tự quét mành được định ra giữa chúng, đối với các ô và các khối mã hóa thì thứ tự quét mành được định ra ví dụ theo từng hàng, cụ thể chỉ số ô tăng lên theo thứ tự quét mành này cũng như các lát đi theo nhau phù hợp với thứ tự lát dọc theo thứ tự quét mành này giữa các khối mã hóa.

Khía cạnh của sự tạo tín hiệu đoạn đầu lát được đặt rải rác và được tạo gói đã được mô tả có thể nhận được từ các phương án nêu trên, khía cạnh này cũng có thể kết hợp với bất kỳ khía cạnh nào trong số các khía cạnh nêu trên hoặc sự kết hợp bất kỳ giữa các khía cạnh. Các tiền tố lát được mô tả rõ ràng trước đây, ví dụ, theo phiên bản 2 đã thống nhất được tất cả các khía cạnh. Thuận lợi của khía cạnh hiện tại là khả năng làm cho dữ liệu đoạn đầu lát có thể sử dụng được dễ dàng hơn cho các thiết bị mạng vì các thiết bị mạng được vận chuyển trong các gói độc lập nằm bên ngoài các lát đứng trước/các gói tải trọng, và cho phép truyền lặp đi lặp lại dữ liệu đoạn đầu lát.

Do đó, một khía cạnh nữa của đơn sáng chế là khía cạnh sự tạo tín hiệu đoạn đầu lát được được tạo gói và đặt rải rác và nói cách khác, khía cạnh này có thể được thấy là sự bộc lộ dòng dữ liệu video có nội dung video được mã hóa trong đó trong các đơn vị phần con (xem các khối cây mã hóa hoặc các lát) của các ảnh của nội dung video, mỗi phần con được mã hóa tương ứng thành một hoặc nhiều gói tải trọng (xem các đơn vị NAL VCL) của chuỗi các gói (các đơn vị NAL) của dòng dữ liệu video, chuỗi các gói được chia thành chuỗi các đơn vị truy cập do đó mỗi đơn vị truy cập tập hợp các gói tải trọng đề cập đến ảnh tương ứng của nội dung video, trong đó chuỗi các gói đặt rải rác trong đó các gói đoạn đầu lát (tiền tố lát) chuyển dữ liệu đoạn đầu lát cho, và bỏ lỡ trong, một hoặc nhiều gói tải trọng đi sau gói đoạn đầu lát tương ứng trong chuỗi các gói.

Thiết bị mạng nhận dòng dữ liệu video (tức là, dòng dữ liệu video có nội dung video đã được mã hóa trong đó trong các đơn vị phần con (xem các khối cây mã hóa hoặc các lát) của các ảnh của nội dung video, mỗi phần con được mã hóa tương ứng thành một hoặc nhiều gói tải trọng (xem các đơn vị NAL VCL) của chuỗi các gói (các đơn vị NAL) của dòng dữ liệu video, chuỗi các gói được chia thành chuỗi các đơn vị truy cập do đó mỗi đơn vị truy cập tập hợp các gói tải trọng đề cập đến ảnh tương ứng của nội dung video, trong đó chuỗi các gói đặt rải rác ở đó bên trong các gói đoạn đầu lát) có thể được tạo cấu hình để đọc đoạn đầu lát cùng với dữ liệu tải cho các lát từ các gói với, tuy nhiên, suy ra từ các gói đoạn đầu lát dữ liệu đoạn đầu lát và nhảy cách việc đọc đoạn đầu lát cho một hoặc nhiều gói tải trọng đi theo gói đoạn đầu lát tương ứng trong chuỗi các gói, nhưng thích ứng đoạn đầu lát được nhận từ gói đoạn đầu lát mà một hoặc nhiều gói tải trọng đi theo.

Là đúng với các khía cạnh nêu trên, có thể các gói, ở đây là các gói đoạn đầu lát, cũng có thể có chức năng chỉ ra thiết bị mạng bất kỳ như MANE hoặc bộ giải mã, bắt đầu đơn vị giải mã hoặc bắt đầu việc vận hành một hoặc nhiều gói tải trọng được đứng trước bởi gói tương ứng. Do đó, thiết bị mạng theo khía cạnh hiện tại có thể nhận dạng các gói tải trọng mà việc đọc đoạn đầu lát cho các gói tải trọng này phải được bỏ quãng dựa trên các phần tử cú pháp nêu trên trong gói này, cụ thể single_slice_flag, kết hợp, ví dụ, decoding_unit_start_flag, cờ về sau giữa chúng cho phép, như thảo luận bên trên, truyền dẫn lại các bản sao của các gói đoạn đầu lát nhất định nằm trong các đơn vị giải mã. Điều này là hữu dụng, ví dụ, làm đoạn đầu lát của các lát nằm trong một đơn vị giải mã có thể thay đổi dọc theo chuỗi các lát, và do đó, trong khi các gói đoạn đầu lát ở lúc bắt đầu các đơn vị giải mã có thể có decoding_unit_start_flag được thiết lập (là bằng 1), các gói đoạn đầu lát được đặt giữa chúng có thể có cờ này không được thiết lập, để ngăn thiết bị mạng bất kỳ không phiên dịch sai lầm sự cố của gói đoạn đầu lát này là bắt đầu đơn vị giải mã mới này.

Mặc dù một vài khía cạnh đã được mô tả trong ngữ cảnh của thiết bị, rõ ràng rằng các khía cạnh này cũng thể hiện mô tả của phương pháp tương ứng, trong đó khối hoặc thiết bị tương ứng với bước xử lý của phương pháp hoặc đặc điểm của bước xử lý của phương pháp. Tương tự, các khía cạnh được mô tả trong ngữ cảnh của bước xử lý của phương pháp cũng thể hiện mô tả của khối hoặc mục tương ứng hoặc đặc điểm của

thiết bị tương ứng. Một vài hoặc tất cả các bước phương pháp có thể được thực hiện bởi (hoặc sử dụng) thiết bị phần cứng, như ví dụ, bộ vi xử lý, máy tính khả trình hoặc mạch điện tử. Theo một số phương án, một số một hoặc nhiều bước phương pháp quan trọng nhất có thể được thực hiện bởi thiết bị này.

Dòng dữ liệu video theo sáng chế có thể được lưu trữ trên vật lưu trữ số hoặc có thể được truyền trong vật ghi truyền như vật ghi truyền không dây hoặc vật ghi truyền có dây như Internet.

Phụ thuộc vào các yêu cầu thực hiện đã biết, các phương án của sáng chế có thể được thực hiện trong phần cứng hoặc trong phần mềm. Phương án có thể được thực hiện sử dụng vật lưu trữ số, ví dụ, đĩa mềm, DVD, Blu-Ray, CD, ROM, PROM, EPROM, EEPROM hoặc bộ nhớ FLASH, có các tín hiệu điều khiển có thể đọc đường bằng điện được lưu trữ trên đó, mà kết hợp (hoặc có thể kết hợp) với hệ thống máy tính khả trình sao cho phương pháp tương ứng được thực hiện. Do đó, môi trường lưu trữ dạng số có thể đọc được bởi máy tính.

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

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

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

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

Phương án khác của các phương pháp theo sáng chế là, vật mang dữ liệu (hoặc vật ghi lưu trữ số, hoặc vật ghi đọc được bằng máy tính) gồm có, đã được ghi lại trên

đó, chương trình máy tính để thực hiện một trong các phương pháp được mô tả ở đây. Vật mang dữ liệu, vật ghi lưu trữ số hoặc vật ghi đã được ghi thường là hữu hình và/hoặc không tạm thời.

Phương án khác của các phương pháp theo sáng chế là, dòng dữ liệu hoặc chuỗi các tín hiệu thể hiện chương trình máy tính để thực hiện một trong các phương pháp đã được mô tả ở đây. Dòng dữ liệu hoặc chuỗi các tín hiệu có thể ví dụ được tạo cấu hình để được truyền nhờ sự kết nối truyền thông dữ liệu, ví dụ thông qua Liên mạng.

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

Phương án khác gồm có máy tính mà một chương trình máy tính sau khi được lắp đặt trên máy tính để thực hiện một trong các phương pháp được mô tả ở đây.

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

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

Các phương án được mô tả bên trên chỉ mang tính minh hoạ cho các nguyên lý của sáng chế. Cần hiểu rằng các sự biến thể và biến đổi của các phương án và các chi tiết được mô tả ở đây sẽ là rõ ràng đối với người có trình độ trong lĩnh vực kỹ thuật tương ứng. Đó là ý định, do đó, bị giới hạn chỉ bởi phạm vi của các điểm yêu cầu bảo hộ và không bởi các chi tiết cụ thể được thể hiện bởi bản mô tả và sự giải thích các phương án ở đây.

YÊU CẦU BẢO HỘ

1. Vật ghi lưu trữ dạng số chứa trên đó dòng dữ liệu video có nội dung video (16) được mã hóa trên đó theo các đơn vị của các phần con (24) của các ảnh (18) của nội dung video (16), mỗi phần con (24) lần lượt được mã hóa thành một hoặc nhiều gói tải trọng (22) của chuỗi (34) gồm các gói của dòng dữ liệu video (22), chuỗi (34) gồm các gói được chia thành chuỗi các đơn vị truy cập (30) để mỗi đơn vị truy cập (30) thu các gói tải trọng (32) đề cập đến ảnh tương ứng (18) của nội dung video (16), trong đó chuỗi (34) gồm các gói đã được đặt rải rác vào trong các gói điều khiển định thời gian (36) để các gói điều khiển định thời gian (36) chia nhỏ các đơn vị truy cập (30) thành các đơn vị giải mã (38) để ít nhất một số đơn vị truy cập (30) được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã (38), với mỗi gói điều khiển định thời gian (38) tạo tín hiệu thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã (38), các gói tải trọng (32) của đơn vị này sẽ theo sau gói điều khiển định thời gian (38) tương ứng trong chuỗi (34) gồm các gói.

2. Vật ghi lưu trữ dạng số chứa trên đó dòng dữ liệu video theo điểm 1, trong đó các phần con (24) là các lát, và mỗi gói tải trọng (32) bao gồm một hoặc nhiều lát.

3. Vật ghi lưu trữ dạng số chứa trên đó dòng dữ liệu video theo điểm 2, trong đó các lát bao gồm các lát được giải mã độc lập và các lát phụ thuộc cho phép, đối với sự xử lý WPP, giải mã sử dụng giải mã entrôpi và dự báo vượt qua các biên lát.

4. Vật ghi lưu trữ dạng số chứa trên đó dòng dữ liệu video theo điểm bất kì trong số các điểm từ 1 đến 3, trong đó mỗi gói của chuỗi (34) gồm các gói có thể được gán chính xác một loại gói khỏi nhiều loại gói với các gói tải trọng (32) và các gói điều khiển định thời gian (36) là các loại gói khác nhau, trong đó sự xuất hiện của các gói có nhiều loại gói trong chuỗi (34) gồm các gói phải chịu các giới hạn nhất định mà xác định thứ tự trong số các loại gói mà được tuân theo bởi các gói trong mỗi đơn vị truy cập (30) để các biên đơn vị truy cập là có thể phát hiện sử dụng các giới hạn bằng cách phát hiện các nấc mà các giới hạn bị phản đối, và duy trì tại cùng vị trí trong chuỗi các gói, thậm chí nếu các gói của bất kì loại gói có thể loại bỏ được loại bỏ khỏi dòng dữ liệu video, trong đó các gói tải trọng (32) là loại gói không thể loại bỏ, và các gói điều khiển định thời gian (36) là loại gói có thể loại bỏ.

5. Vật ghi lưu trữ dạng số chứa trên đó dòng dữ liệu video theo điểm bất kì trong số các điểm 1 đến 4, trong đó mỗi gói bao gồm một phần phần tử cú pháp chỉ ra loại gói.

6. Vật ghi lưu trữ dạng số chứa trên đó dòng dữ liệu video theo điểm 5, trong đó loại gói biểu thị một phần phần tử cú pháp bao gồm trường loại gói trong phần đầu gói của gói tương ứng, nội dung của gói này phân biệt các gói tải trọng và các gói điều khiển định thời gian, và, đối với các gói điều khiển định thời gian, trường loại gói SEI phân biệt một mặt là các gói điều khiển định thời gian và mặt khác là các gói SEI có loại khác.

7. Bộ mã hóa để mã hóa vào dòng dữ liệu video (22) nội dung video (16) trong các đơn vị của các phần con (24) của các ảnh (18) của nội dung video (16), với mã hóa lần lượt mỗi phần con (24) thành một hoặc nhiều gói tải trọng (22) của chuỗi (34) gồm các gói của dòng dữ liệu video (22) để chuỗi (34) gồm các gói được chia thành chuỗi các đơn vị truy cập (30) và mỗi đơn vị truy cập (30) thu các gói tải trọng (32) đề cập đến ảnh tương ứng (18) của nội dung video (16), bộ mã hóa được tạo cấu hình để đặt rải rác các gói điều khiển định thời gian (36) vào chuỗi (34) gồm các gói để các gói điều khiển định thời gian (36) chia nhỏ các đơn vị truy cập (30) thành các đơn vị giải mã (38) để ít nhất một số đơn vị truy cập (30) được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã (38), với mỗi gói điều khiển định thời gian (36) tạo tín hiệu thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã (38), các gói tải trọng (32) của đơn vị này sẽ theo sau gói điều khiển định thời gian (36) tương ứng trong chuỗi (34) gồm các gói.

8. Bộ mã hóa theo điểm 7 trong đó ở đây bộ mã hóa được tạo cấu hình để, trong khi mã hóa ảnh hiện thời của nội dung âm thanh,

mã hóa phần con hiện thời (24) của ảnh hiện thời (18) thành gói tải trọng hiện thời (32) của đơn vị giải mã hiện thời (38),

truyền dẫn, trong dòng dữ liệu, bộ phận giải mã hiện thời (38) được đặt trước với gói điều khiển định thời gian hiện thời (36) với thiết lập thời gian phục hồi bộ nhớ đệm bộ giải mã được tạo tín hiệu bằng gói điều khiển định thời gian hiện thời (36), tại nấc thời gian thứ nhất; và

mã hóa phần con nữa của ảnh hiện thời tại nấc thời gian thứ hai, sau nấc thời gian thứ nhất.

9. Phương pháp mã hóa vào dòng dữ liệu video (22) nội dung video (16) trong các đơn vị của các phần con (24) của các ảnh (18) của nội dung video (16), với mã hóa lần lượt mỗi phần con (24) thành một hoặc nhiều gói tải trọng (22) của chuỗi (34) gồm các gói của dòng dữ liệu video (22) để chuỗi (34) gồm các gói được chia thành chuỗi các đơn vị truy cập (30) và mỗi đơn vị truy cập (30) thu các gói tải trọng (32) đề cập đến ảnh tương ứng (18) của nội dung video (16), phương pháp bao gồm đặt rải rác các gói điều khiển định thời gian (36) vào chuỗi (34) gồm các gói để các gói điều khiển định thời gian (36) chia nhỏ các đơn vị truy cập (30) thành các đơn vị giải mã (38) để ít nhất một số đơn vị truy cập (30) được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã (38), với mỗi gói điều khiển định thời gian (36) tạo tín hiệu thời gian phục hồi bộ nhớ đệm bộ giải mã cho đơn vị giải mã (38), các gói tải trọng (32) của đơn vị này sẽ theo sau gói điều khiển định thời gian (36) tương ứng trong chuỗi (34) gồm các gói.

10. Bộ giải mã để giải mã dòng dữ liệu video (22) có nội dung video (16) được mã hóa trong đó trong các đơn vị của các phần con (24) của các ảnh (18) của nội dung video (16), mỗi phần con được mã hóa lần lượt vào một hoặc nhiều gói tải trọng (32) của chuỗi (34) gồm các gói của dòng dữ liệu video (22), chuỗi (34) gồm các gói được chia thành chuỗi các đơn vị truy cập (30) để mỗi đơn vị truy cập (30) thu các gói tải trọng (32) đề cập đến ảnh tương ứng (18) của nội dung video (16), bộ giải mã bao gồm bộ nhớ đệm để nhớ đệm dòng dữ liệu video hoặc khôi phục nội dung video thu được từ đó bằng cách giải mã dòng dữ liệu video và được tạo cấu hình để tìm kiếm các gói điều khiển định thời gian (36) được phân tán trong chuỗi các gói, chia nhỏ các đơn vị truy cập (30) thành các gói giải mã (38) tại các gói điều khiển định thời gian (36) để ít nhất một số đơn vị truy cập được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã và làm rỗng bộ nhớ đệm trong các đơn vị giải mã.

11. Bộ giải mã theo điểm 10, trong đó bộ giải mã được tạo cấu hình để, khi tìm kiếm các gói điều khiển định thời gian (36), kiểm tra, trong mỗi gói, chỉ ra một phần phần tử cú pháp loại gói với, nếu giá trị của một phần phần tử cú pháp chỉ ra loại gói

bằng giá trị được định trước, nhận biết gói tương ứng như gói điều khiển định thời gian (36).

12. Phương pháp giải mã dòng dữ liệu video (22) có nội dung video (16) được mã hóa trong đó trong các đơn vị của các phần con (24) của các ảnh (18) của nội dung video (16), mỗi phần con được mã hóa lần lượt vào một hoặc nhiều gói tải trọng (32) của chuỗi (34) gồm các gói của dòng dữ liệu video (22), chuỗi (34) gồm các gói được chia thành chuỗi các đơn vị truy cập (30) để mỗi đơn vị truy cập (30) thu các gói tải trọng (32) đề cập đến ảnh tương ứng (18) của nội dung video (16), phương pháp sử dụng bộ nhớ đệm để nhớ đệm dòng dữ liệu video hoặc khôi phục nội dung video thu được từ đó bằng cách giải mã dòng dữ liệu video và được tạo cấu hình để tìm kiếm các gói điều khiển định thời gian (36) được phân tán trong chuỗi các gói, chia nhỏ các đơn vị truy cập (30) thành các gói giải mã (38) tại các gói điều khiển định thời gian (36) để ít nhất một số đơn vị truy cập được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã và làm rỗng bộ nhớ đệm trong các đơn vị giải mã.

13. Thiết bị mạng để truyền dẫn dòng dữ liệu video (22) có nội dung video (16) được mã hóa trên đó theo các đơn vị của các phần con (24) của các ảnh (18) của nội dung video (16), mỗi phần con lần lượt được mã hóa thành một hoặc nhiều gói tải trọng (22) của chuỗi (34) gồm các gói của dòng dữ liệu video (22), chuỗi (34) gồm các gói được chia thành chuỗi các đơn vị truy cập (30) để mỗi đơn vị truy cập (30) thu các gói tải trọng (32) đề cập đến ảnh tương ứng (18) của nội dung video (16), bộ giải mã được tạo cấu hình để tìm kiếm các gói điều khiển định thời gian (36) được phân tán vào chuỗi các gói, chia nhỏ các đơn vị truy cập thành các đơn vị giải mã tại các gói điều khiển định thời gian (36) để ít nhất một số đơn vị truy cập (30) được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã (38), suy ra từ mỗi gói điều khiển định thời gian (36) thời gian phục hồi bộ nhớ đệm bộ mã hóa (38), các gói tải trọng (32) mà theo sau gói điều khiển định thời gian tương ứng (36) trong chuỗi (34) gồm các gói, và thực hiện sự truyền dẫn dòng dữ liệu video phụ thuộc vào các thời gian phục hồi bộ nhớ đệm bộ giải mã cho các đơn vị giải mã (38).

14. Phương pháp truyền dẫn dòng dữ liệu video (22) có nội dung video (16) được mã hóa trên đó theo các đơn vị của các phần con (24) của các ảnh (18) của nội dung

video (16), mỗi phần con lần lượt được mã hóa thành một hoặc nhiều gói tải trọng (22) của chuỗi (34) gồm các gói của dòng dữ liệu video (22), chuỗi (34) gồm các gói được chia thành chuỗi các đơn vị truy cập (30) để mỗi đơn vị truy cập (30) thu các gói tải trọng (32) đề cập đến ảnh tương ứng (18) của nội dung video (16), phương pháp bao gồm tìm kiếm các gói điều khiển định thời gian (36) được phân tán vào chuỗi các gói, chia nhỏ các đơn vị truy cập thành các đơn vị giải mã tại các gói điều khiển định thời gian (36) để ít nhất một số đơn vị truy cập (30) được chia nhỏ thành hai hoặc nhiều hơn hai đơn vị giải mã (38), suy ra từ mỗi gói điều khiển định thời gian (36) thời gian phục hồi bộ nhớ đệm bộ mã hóa (38), các gói tải trọng (32) mà theo sau gói điều khiển định thời gian tương ứng (36) trong chuỗi (34) gồm các gói, và thực hiện sự truyền dẫn dòng dữ liệu video phụ thuộc vào các thời gian phục hồi bộ nhớ đệm bộ giải mã cho các đơn vị giải mã (38).

15. Vật ghi có thể đọc được bằng máy tính bao gồm chương trình máy tính có mã chương trình để thực hiện, khi chạy trên máy tính, phương pháp theo điểm 9.

16. Vật ghi có thể đọc được bằng máy tính bao gồm chương trình máy tính có mã chương trình để thực hiện, khi chạy trên máy tính, phương pháp theo điểm 12.

17. Vật ghi có thể đọc được bằng máy tính bao gồm chương trình máy tính có mã chương trình để thực hiện, khi chạy trên máy tính, phương pháp theo điểm 14.

buffering_period(payloadSize) {	Ký hiệu
seq_parameter_set_id	ue(v)
if(!sub_pic_cpb_params_present_flag)	
rap_cpb_params_present_flag	u(1)
if(NalHrdBpPresentFlag) {	
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {	
initial_cpb_removal_delay[SchedSelIdx]	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx]	u(v)
if(sub_pic_cpb_params_present_flag || rap_cpb_params_present_flag) {	
initial_alt_cpb_removal_delay[SchedSelIdx]	u(v)
initial_alt_cpb_removal_delay_offset[SchedSelIdx]	u(v)
}	
}	
}	
if(VclHrdBpPresentFlag) {	
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {	
initial_cpb_removal_delay[SchedSelIdx]	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx]	u(v)
if(sub_pic_cpb_params_present_flag || rap_cpb_params_present_flag) {	
initial_alt_cpb_removal_delay[SchedSelIdx]	u(v)
initial_alt_cpb_removal_delay_offset[SchedSelIdx]	u(v)
}	
}	
}	
}	

FIG 1

pic_timing(payloadSize) {	Ký hiệu
if(CpbDpbDelaysPresentFlag) {	
cpb_removal_delay	u(v)
dpb_output_delay	u(v)
if(sub_pic_cpb_params_present_flag) {	
num_decoding_units_minus1	ue(v)
for(i = 0; i <= num_decoding_units_minus1; i++) {	
num_nalus_in_du_minus1[i]	ue(v)
du_cpb_removal_delay[i]	u(v)
}	
}	
}	
}	

FIG 2

vui_parameters( ) {	Ký hiệu
aspect_ratio_info_present_flag	u(1)
if(aspect_ratio_info_present_flag) {	
aspect_ratio_idc	u(8)
if(aspect_ratio_idc = = Extended_SAR) {	
sar_width	u(16)
sar_height	u(16)
}	
}	
overscan_info_present_flag	u(1)
if(overscan_info_present_flag)	
overscan_appropriate_flag	u(1)
video_signal_type_present_flag	u(1)
if(video_signal_type_present_flag) {	
video_format	u(3)
video_full_range_flag	u(1)
colour_description_present_flag	u(1)
if(colour_description_present_flag) {	
colour_primaries	u(8)
transfer_characteristics	u(8)
matrix_coefficients	u(8)
}	
}	
chroma_loc_info_present_flag	u(1)
if(chroma_loc_info_present_flag) {	
chroma_sample_loc_type_top_field	ue(v)
chroma_sample_loc_type_bottom_field	ue(v)
}	
neutral_chroma_indication_flag	u(1)

FIG 3A

FIG 3

FIG 3A

FIG 3B

field_seq_flag	u(1)
timing_info_present_flag	u(1)
if(timing_info_present_flag) {	
num_units_in_tick	u(32)
time_scale	u(32)
fixed_pic_rate_flag	u(1)
}	
nal_hrd_parameters_present_flag	u(1)
if(nal_hrd_parameters_present_flag)	
hrd_parameters( )	
vcl_hrd_parameters_present_flag	u(1)
if(vcl_hrd_parameters_present_flag)	
hrd_parameters( )	
if(nal_hrd_parameters_present_flag || vcl_hrd_parameters_present_flag)	
low_delay_hrd_flag	u(1)
sub_pic_cpb_params_present_flag	u(1)
if(sub_pic_cpb_params_present_flag)	
num_units_in_sub_tick	u(32)
}	
bitstream_restriction_flag	u(1)
if(bitstream_restriction_flag) {	
tiles_fixed_structure_flag	u(1)
motion_vectors_over_pic_boundaries_flag	u(1)
max_bytes_per_pic_denom	ue(v)
max_bits_per_mincu_denom	ue(v)
log2_max_mv_length_horizontal	ue(v)
log2_max_mv_length_vertical	ue(v)
}	
}	

FIG 3B

FIG 3	FIG 3A
	FIG 3B

hrd_parameters( ) {	Ký hiệu
cpb_cnt_minus1	ue(v)
bit_rate_scale	u(4)
cpb_size_scale	u(4)
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {	
bit_rate_value_minus1[SchedSelIdx]	ue(v)
cpb_size_value_minus1[SchedSelIdx]	ue(v)
cbr_flag[SchedSelIdx]	u(1)
}	
initial_cpb_removal_delay_length_minus1	u(5)
cpb_removal_delay_length_minus1	u(5)
dpb_output_delay_length_minus1	u(5)
time_offset_length	u(5)
}	

FIG 4

filler_data_rbsp( ) {	Ký hiệu
while(next_bits(8) = = 0xFF)	
ff_byte /* equal to 0xFF */	f(8)
rbsp_trailing_bits( )	
}	

FIG 5

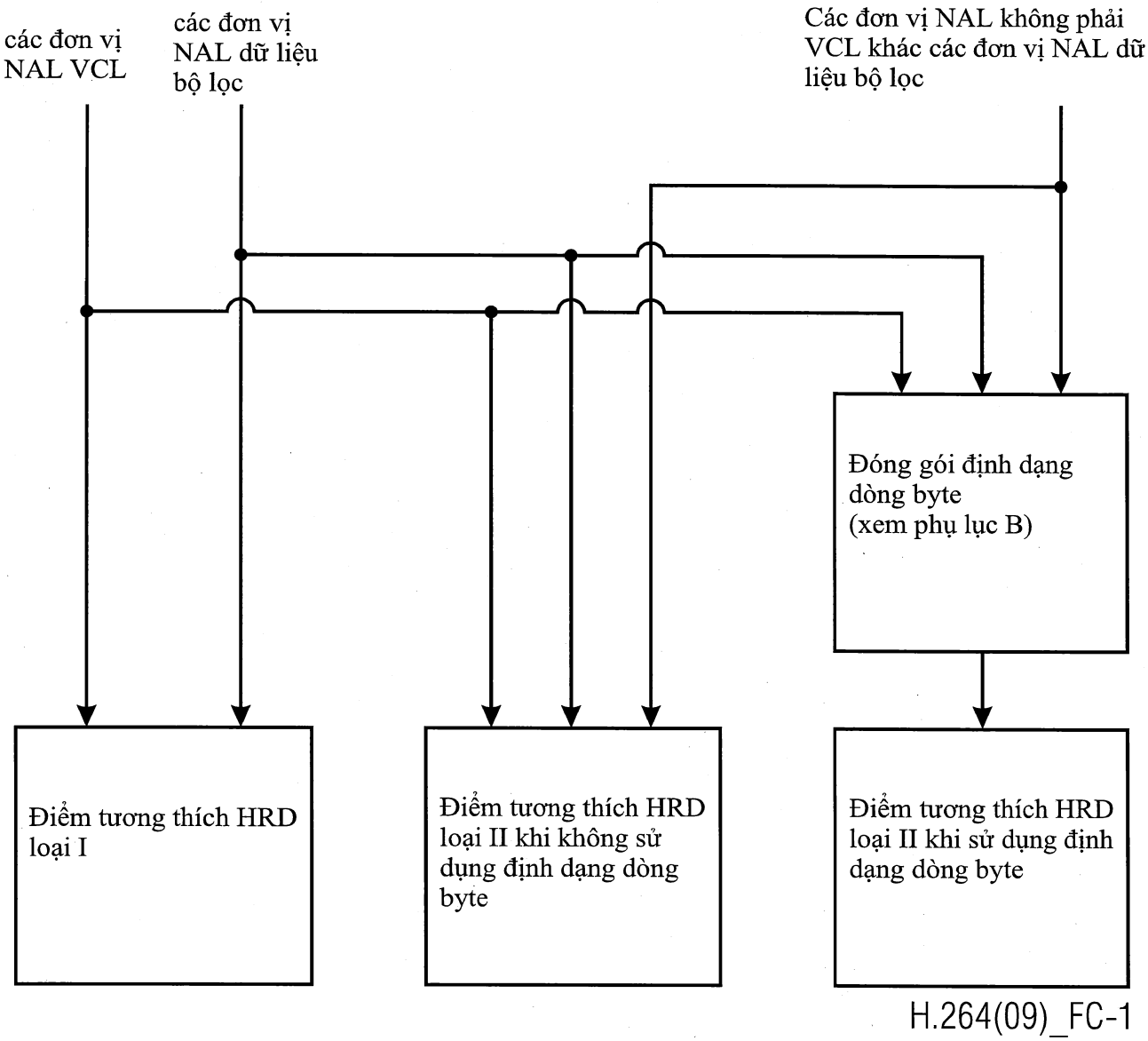


FIG 6

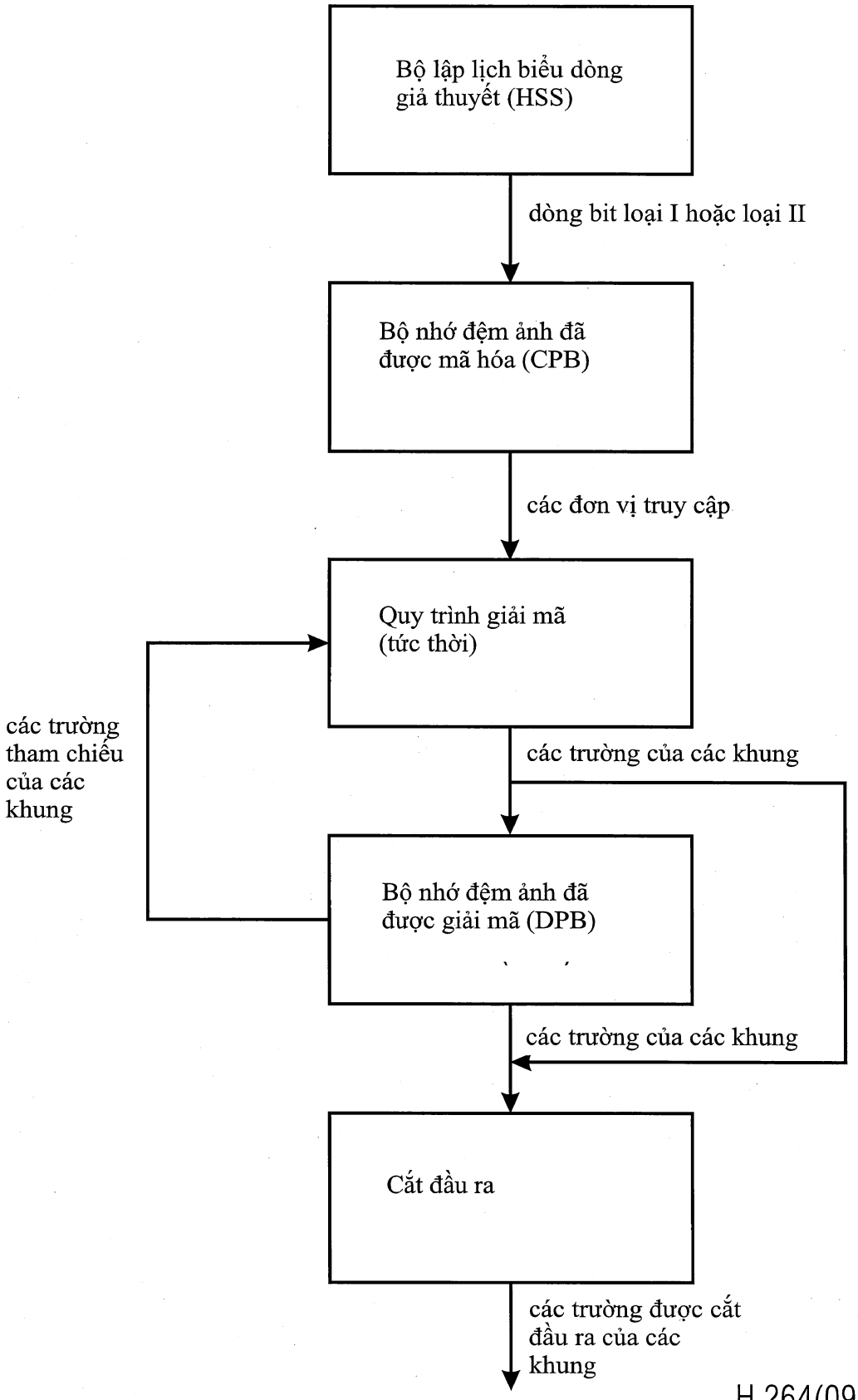

Bộ lập lịch biểu dòng giả thuyết (HSS)
dòng bit loại I hoặc loại II
Bộ nhớ đệm ảnh đã được mã hóa (CPB)
các đơn vị truy cập
Quy trình giải mã (tức thời)
các trường tham chiếu của các khung
các trường của các khung
Bộ nhớ đệm ảnh đã được giải mã (DPB)
các trường của các khung
Cắt đầu ra
các trường được cắt đầu ra của các khung
H.264(09)_FC-2

FIG 7

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

FIG 8

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

FIG 9

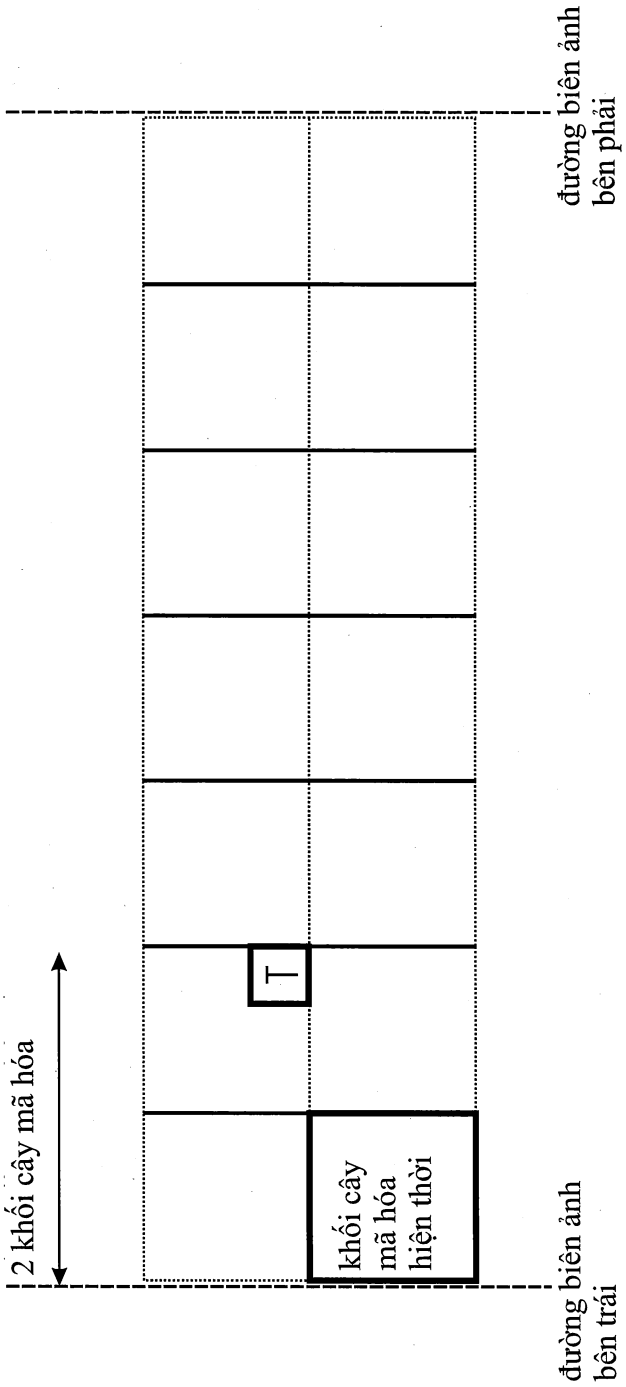

đường biên ảnh
bên phải
T
2 khối cây mã hóa
khối cây
mã hóa
hiện thời
đường biên ảnh
bên trái

FIG 10A

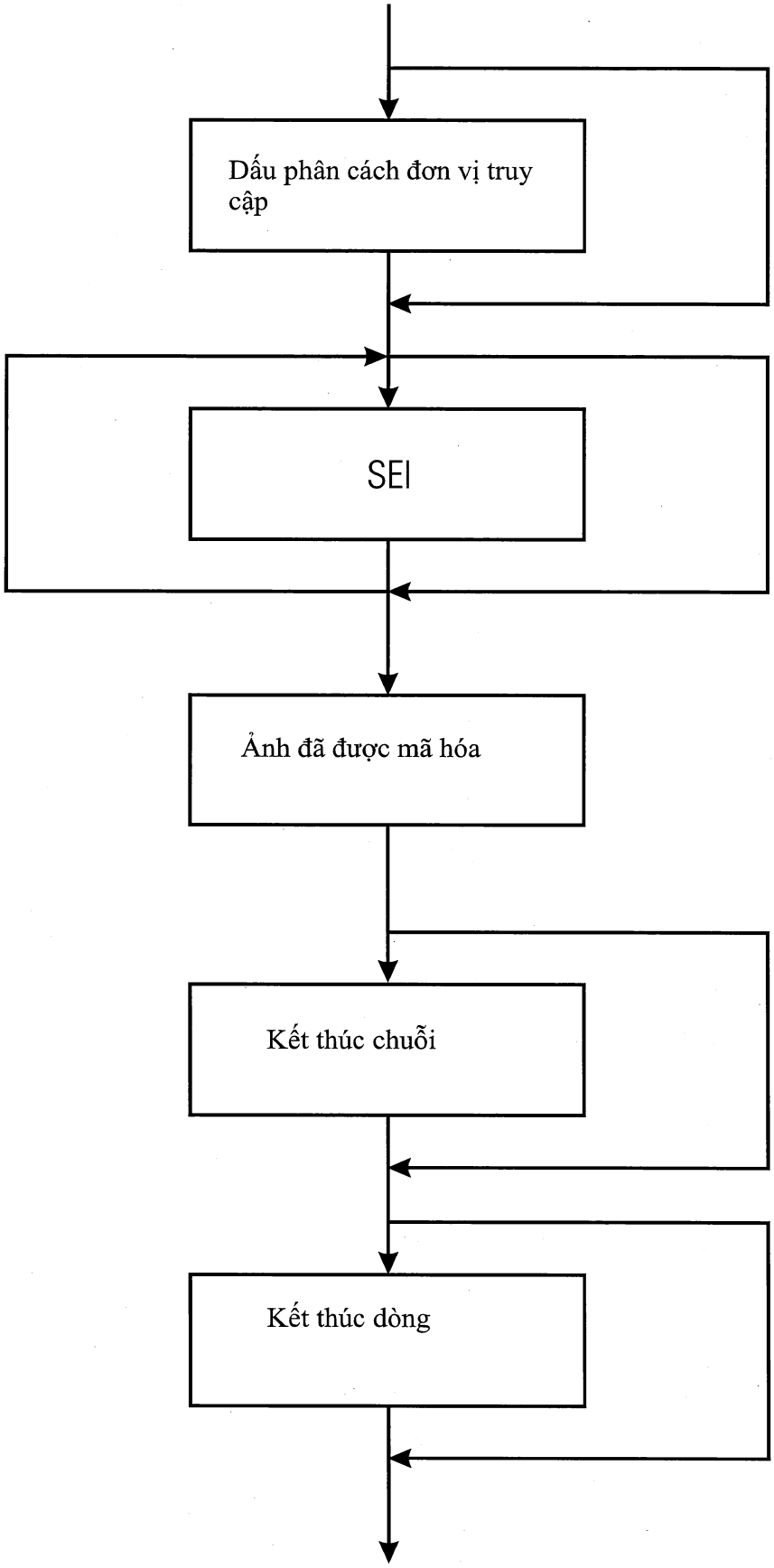

Dấu phân cách đơn vị truy cập
SEI
Ảnh đã được mã hóa
Kết thúc chuỗi
Kết thúc dòng

FIG 10B

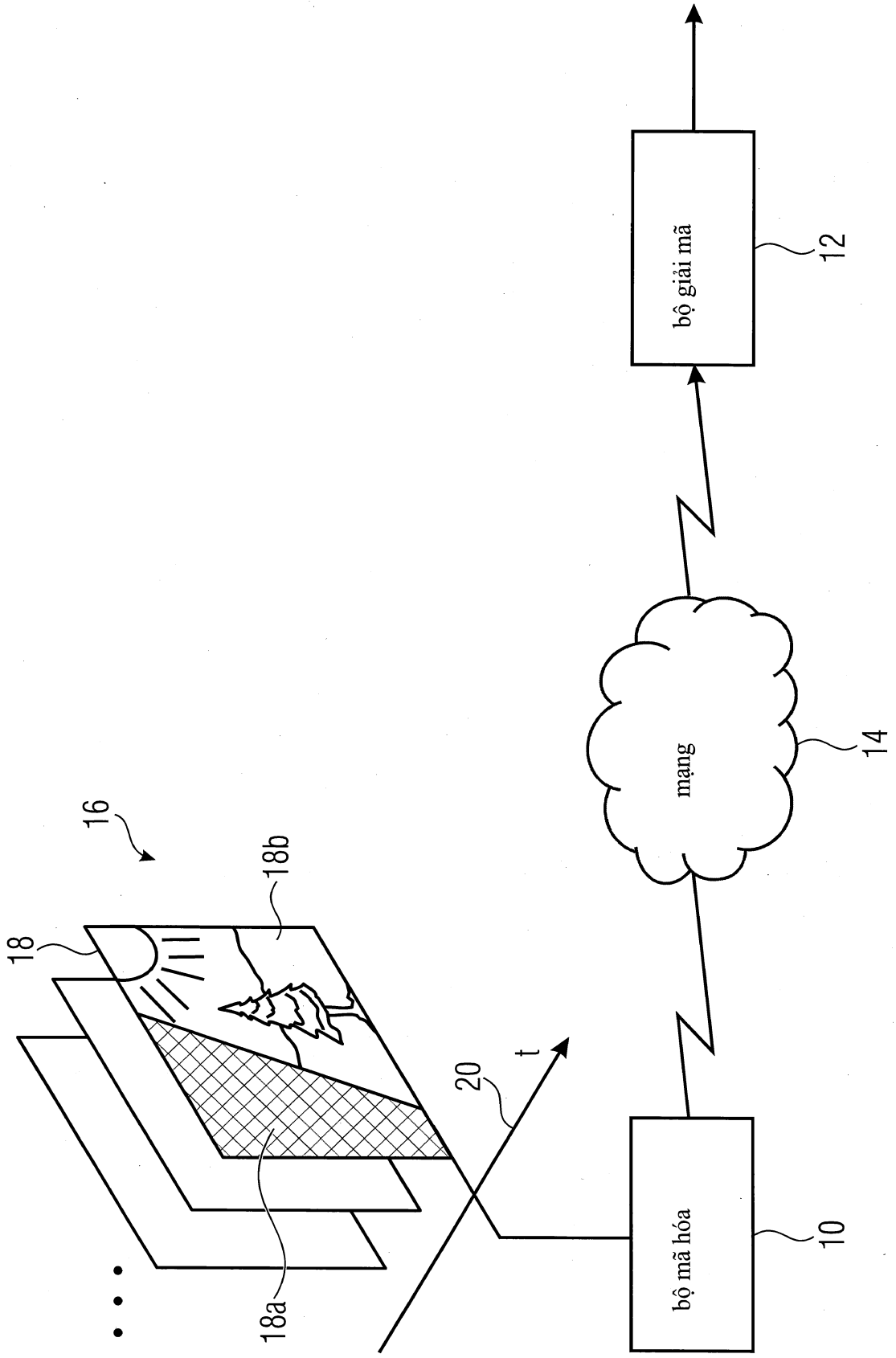


FIG 11

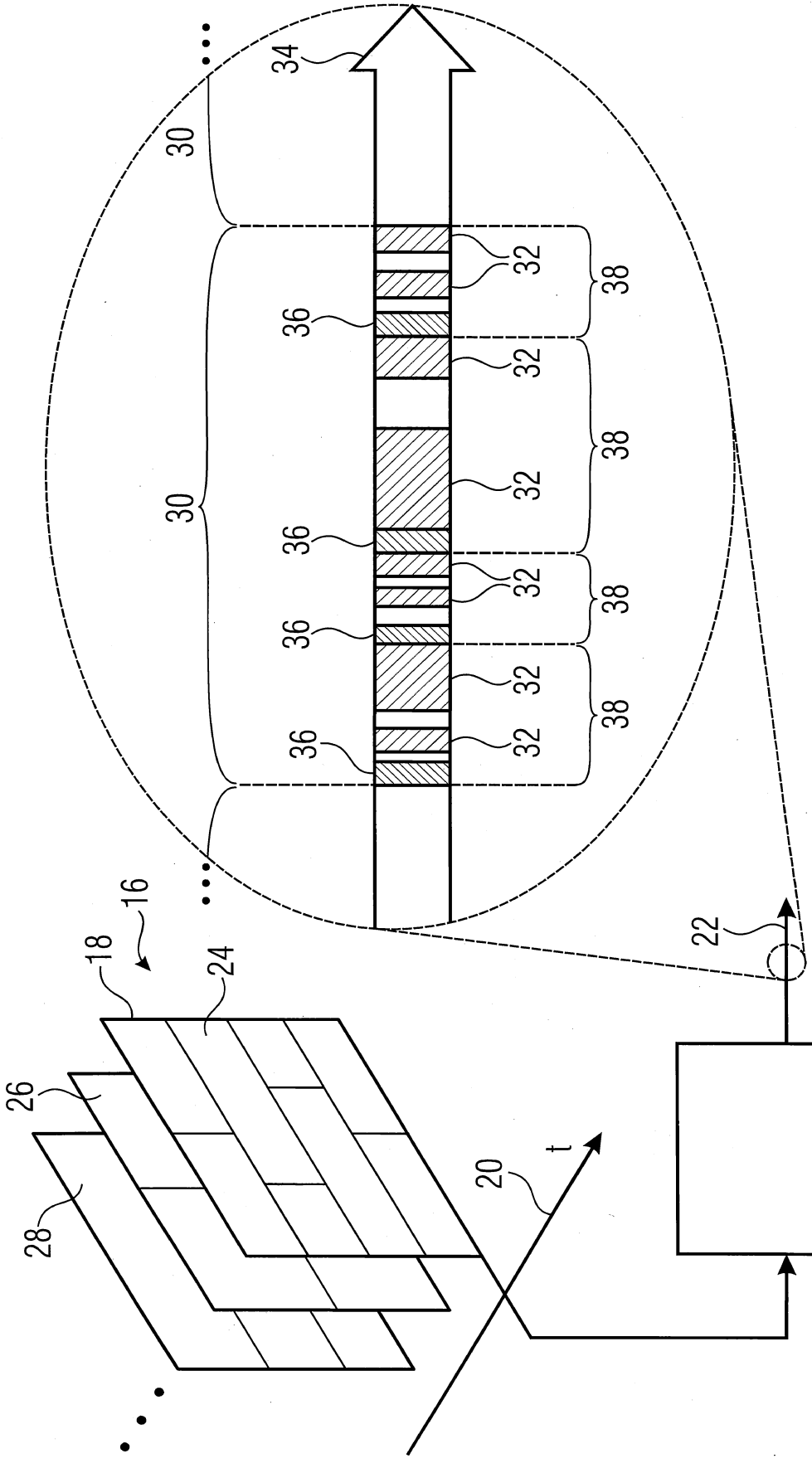

34
30
32
38
36
30
32
38
36
32
38
36
32
38
36
32
38
16
18
24
22
26
28
20
t
10

FIG 12

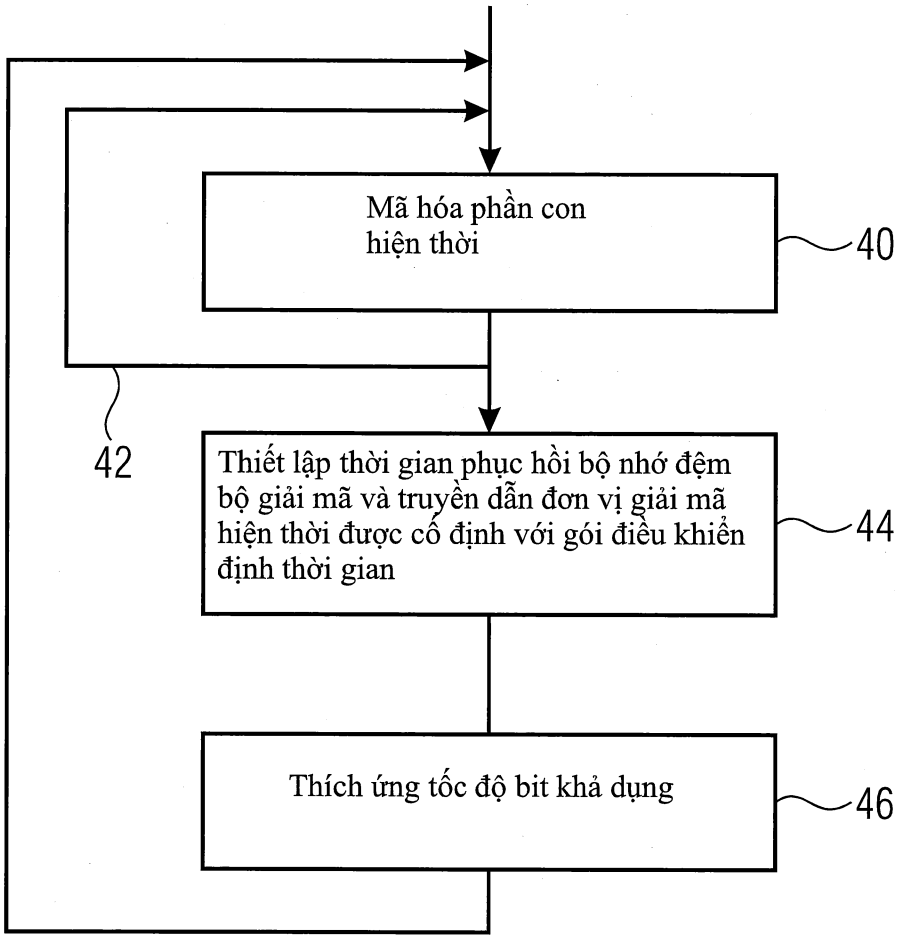

Mã hóa phần con hiện thời
40
42
Thiết lập thời gian phục hồi bộ nhớ đệm bộ giải mã và truyền dẫn đơn vị giải mã hiện thời được cố định với gói điều khiển định thời gian
44
Thích ứng tốc độ bit khả dụng
46

FIG 13

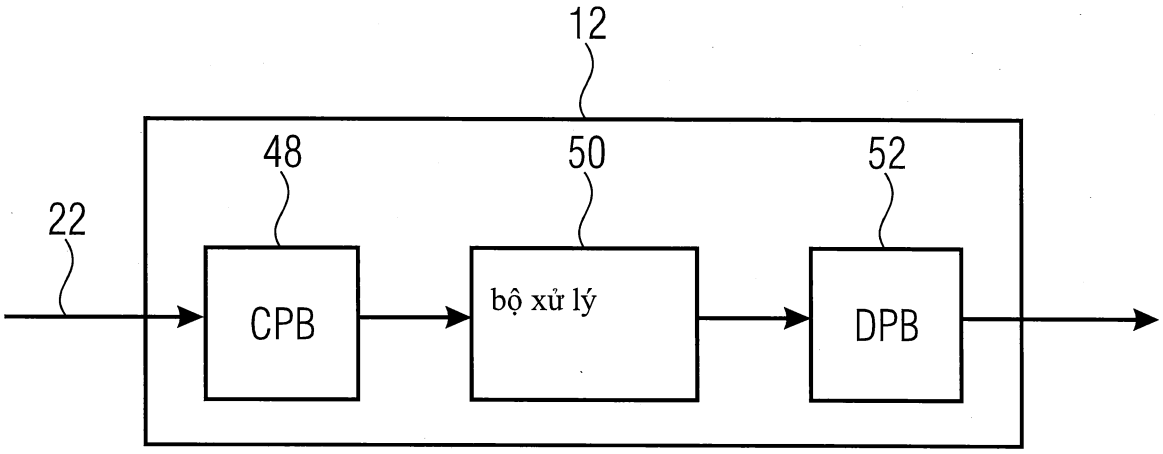

12
48
50
52
22
CPB
bộ xử lý
DPB

FIG 14

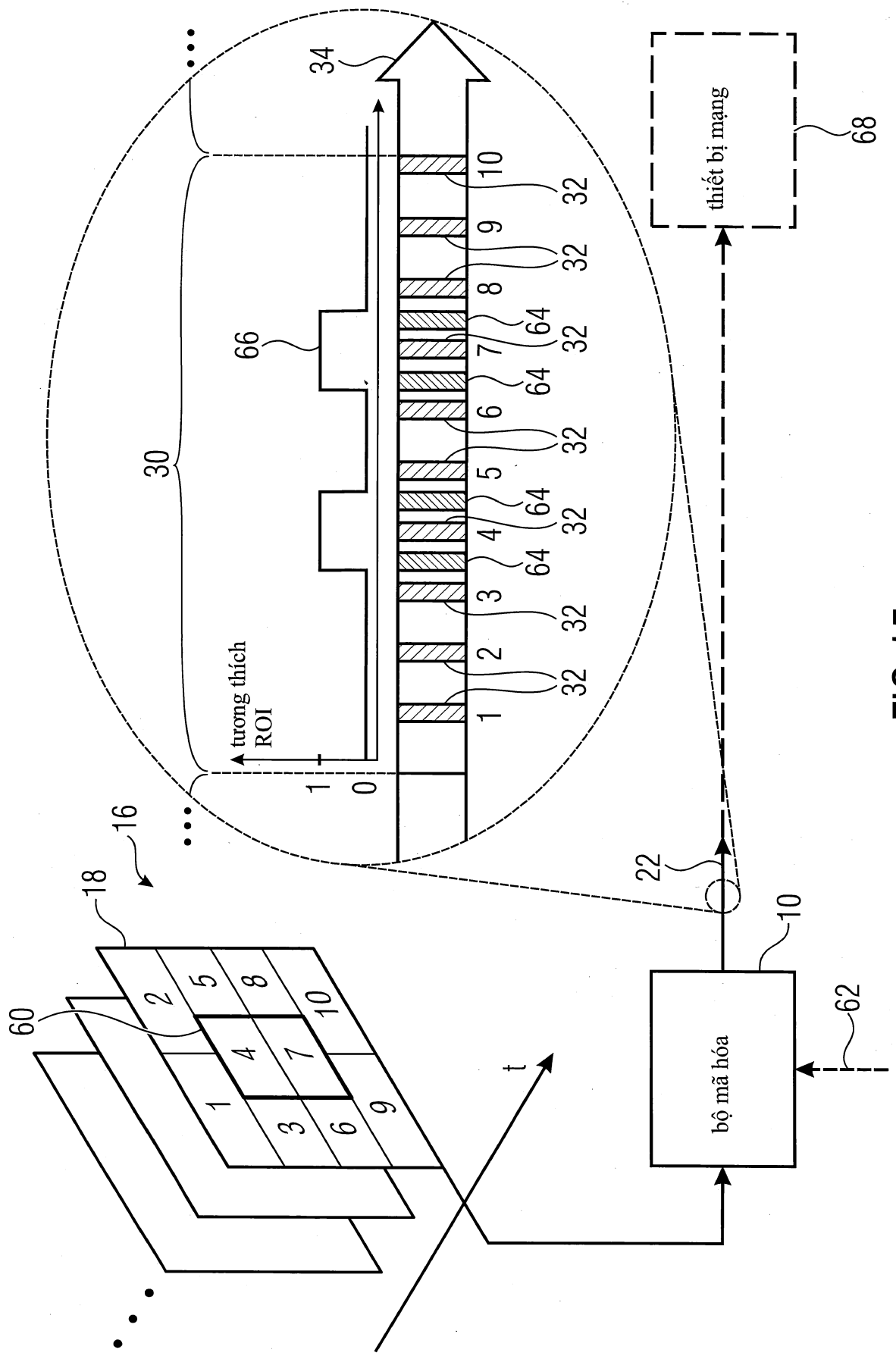

34
66
30
tương thích ROI
1
0
1
2
3
4
5
6
7
8
9
10
32
64
thiết bị mạng
68
22
16
18
60
10
bộ mã hóa
62
t

FIG 15

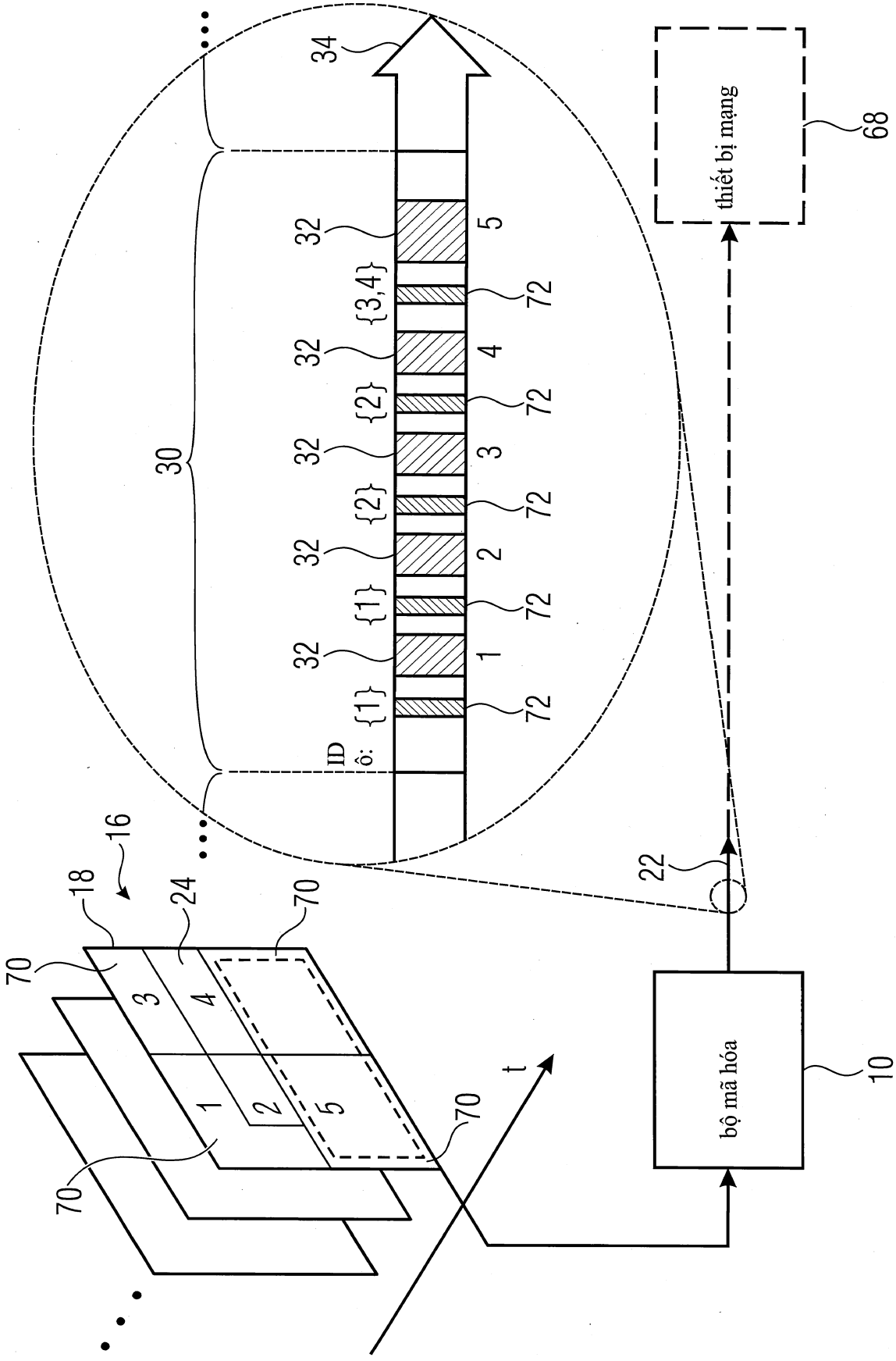

thiết bị mạng
68
bộ mã hóa
10
22
34
30
ID ô:
{1}
{2}
{3,4}
1
2
3
4
5
32
72
16
18
24
70
t

FIG 16

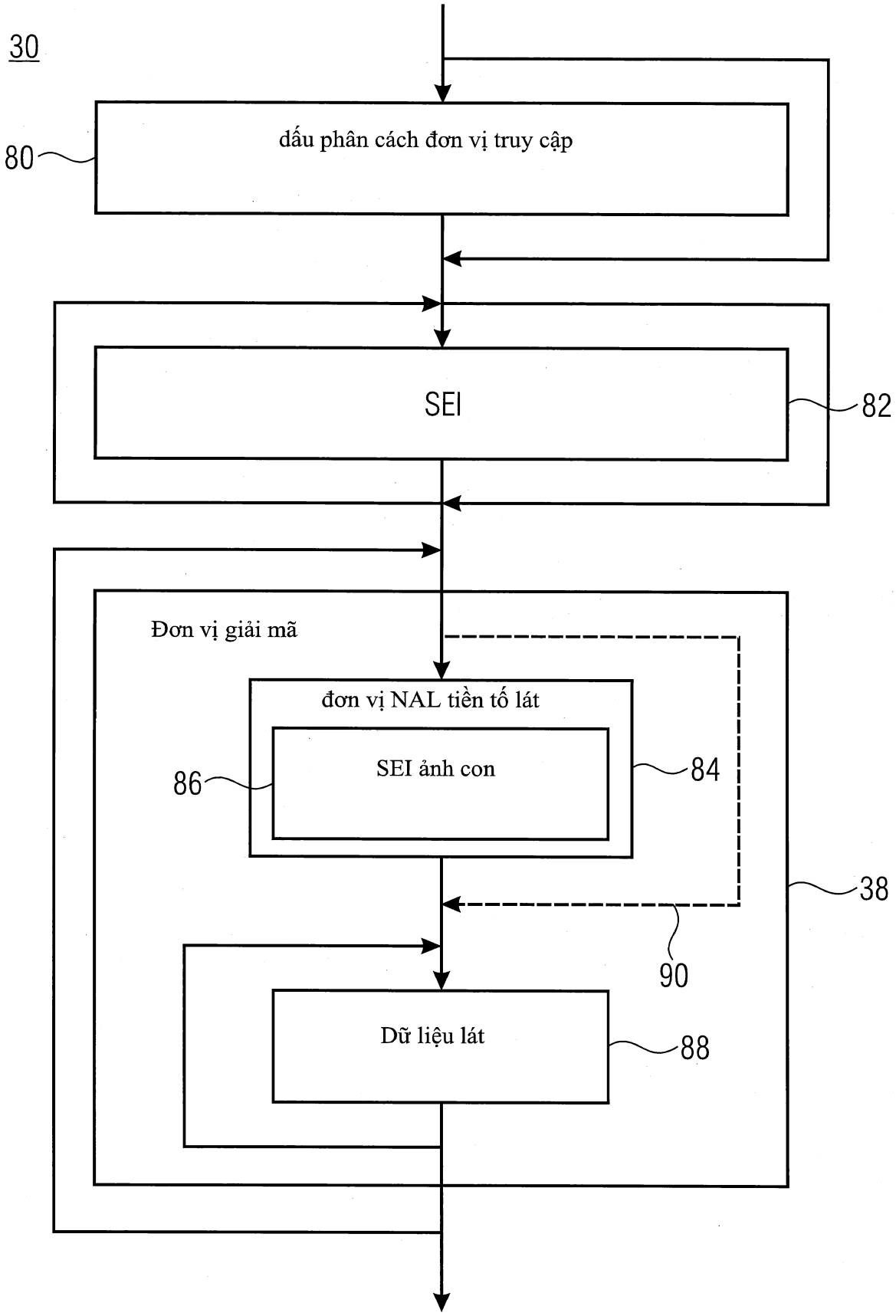

30
80
dấu phân cách đơn vị truy cập
SEI
82
Đơn vị giải mã
đơn vị NAL tiền tố lát
SEI ảnh con
86
84
38
90
Dữ liệu lát
88

FIG 17

Các ô trong vùng quan tâm

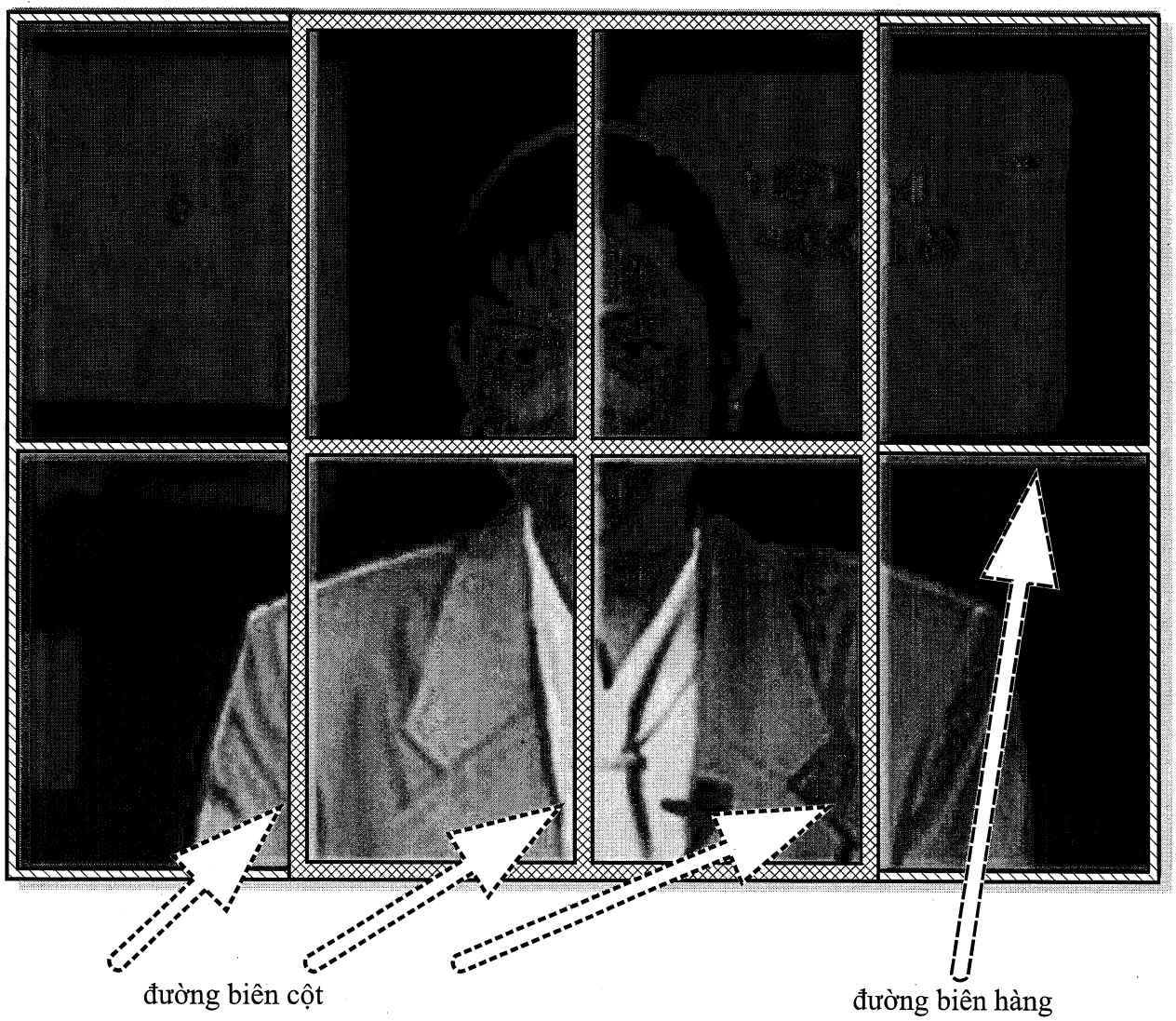


FIG 18

slice_prefix_rbsp( ) {	Ký hiệu
reserved_one_8bits	u(8)
do	
sei_message( )	
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

FIG 19

slice_prefix_rbsp( ) {	Ký hiệu
slice_prefix_id	u(2)
rap_flag	u(1)
decoding_unit_start_flag	u(1)
single_slice_flag	u(1)
tile_idc	u(2)
prefix_slice_header_data_present_flag	u(1)
if(prefix_slice_header_data_present_flag)	
slice_header_data()	
if(tile_idc > 0) {	
if(tile_idc == 2) /* multiple tiles in prefixed slices */	
num_tiles_in_prefixed_slices_minus1	u(16)
first_tile_id_in_prefixed_slices	u(16)
}	
do	
sei_message( )	
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

FIG 20

nal_unit_type	Nội dung của đơn vị NAL và cấu trúc cú pháp RBSP	Phân loại đầu vào đơn vị NAL
0	Không được chỉ rõ	không phải VCL
1	Lát được mã hóa của ảnh không phải RAp, không phải TFD và không phải TLA slice_layer_rbsp( )	VCL
2	Lát được mã hóa của ảnh TFD slice_layer_rbsp( )	VCL
3	Lát được mã hóa của ảnh TLA không phải TFD slice_layer_rbsp( )	VCL
4, 5	Lát được mã hóa của ảnh CRA slice_layer_rbsp( )	VCL
6, 7	Lát được mã hóa của ảnh BLA slice_layer_rbsp( )	VCL
8	Lát được mã hóa của ảnh IDR slice_layer_rbsp( )	VCL
9..234	Được bảo toàn	không xác định
24	Đơn vị NAL tiền tố lát	không phải VCL
25	Thiết lập tham số video video_parameter_set_rbsp( )	không phải VCL
26	Thiết lập tham số chuỗi sequence_parameter_set_rbsp( )	không phải VCL
27	Thiết lập tham số ảnh pic_parameter_set_rbsp( )	không phải VCL
28	Thiết lập tham số thích ứng aps_rbsp( )	không phải VCL
29	Dấu phân cách đơn vị truy cập access_unit_delimiter_rbsp( )	không phải VCL
30	Dữ liệu điền đầy filler_data_rbsp( )	không phải VCL
31	Thông tin nâng cao bổ sung (SEI) sei_rbsp( )	không phải VCL
32..47	Được bảo toàn	không xác định
48..63	Không được chỉ rõ	không phải VCL

FIG 21

slice_header( ) {	Ký hiệu
first_slice_in_pic_flag	u(1)
~~pic_parameter_set_id~~	~~ue(v)~~
if(!first_slice_in_pic_flag)	
slice_address	u(v)
if(dependent_slice_enabled_flag && !first_slice_in_pic_flag)	
dependent_slice_flag	u(1)
slice_header_data_present_flag	u(1)
if (slice_header_data_present_flag && !dependent_slice_flag)	
slice_header_data()	
if(tiles_or_entropy_coding_sync_idc == 1 || tiles_or_entropy_coding_sync_idc == 2) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
if(slice_header_extension_present_flag) {	
slice_header_extension_length	ue(v)
for(i = 0; i < slice_header_extension_length; i++)	
slice_header_extension_data_byte	u(8)
}	
byte_alignment( )	
}	

FIG 22

slice_header_data( ) {	Ký hiệu
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if(RapPicFlag) {	
rap_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
}	
if(!IdrPicFlag) {	
pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
poc_lsb_lt[i]	u(v)
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
}	
}	
if(sample_adaptive_offset_enabled_flag) {	

FIG 23A

FIG 23

FIG 23A
FIG 23B
FIG 23C

slice_sample_adaptive_offset_flag[0]	u(1)
if(slice_sample_adaptive_offset_flag[0]) {	
slice_sample_adaptive_offset_flag[1]	u(1)
slice_sample_adaptive_offset_flag[2]	u(1)
}	
}	
if(adaptive_loop_filter_enabled_flag)	
aps_id	ue(v)
if(slice_type = = P | | slice_type = = B) {	
if(sps_temporal_mvp_enable_flag)	
pic_temporal_mvp_enable_flag	u(1)
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
if(lists_modification_present_flag)	
ref_pic_list_modification( )	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag && slice_type != I)	
cabac_init_flag	u(1)
slice_qp_delta	se(v)
if(deblocking_filter_control_present_flag) {	
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)

FIG 23B

FIG 23	FIG 23A
	FIG 23B
	FIG 23C

if(deblocking_filter_override_flag) {	
slice_header_disable_deblocking_filter_flag	u(1)
if(!slice_header_disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
}	
}	
if(pic_temporal_mvp_enable_flag) {	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if(slice_type != I &&	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type = = P) | |	
pred_weight_table( )	
if(slice_type = = P | | slice_type = = B)	
five_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag &&	
alf_param( )	
if(slice_adaptive_loop_filter_flag	
alf_cu_control_param( )	
}	
if(seq_loop_filter_across_slices_enabled_flag &&	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	

FIG 23C

FIG 23

FIG 23A

FIG 23B

FIG 23C

sei_message( ) {	Ký hiệu
payloadType = 0	
while(next_bits(8) = = 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadType += 255	
}	
last_payload_type_byte	u(8)
payloadType += last_payload_type_byte	
payloadSize = 0	
while(next_bits(8) = = 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadSize += 255	
}	
last_payload_size_byte	u(8)
payloadSize += last_payload_size_byte	
sei_payload(payloadType, payloadSize)	
}	

FIG 24

sei_payload(payloadType, payloadSize) {	Ký hiệu
if(payloadType = = 0)	
buffering_period(payloadSize)	
else if(payloadType = = 1)	
pic_timing(payloadSize)	
else if(payloadType = = 2)	
pan_scan_rect(payloadSize)	
else if(payloadType = = 3)	
filler_payload(payloadSize)	
else if(payloadType = = 4)	
user_data_registered_itu_t_t35(payloadSize)	
else if(payloadType = = 5)	
user_data_unregistered(payloadSize)	
else if(payloadType = = 6)	
recovery_point(payloadSize)	
else if(payloadType = = 9)	
scene_info(payloadSize)	
else if(payloadType = = 15)	
full_frame_snapshot(payloadSize)	
else if(payloadType = = 16)	
progressive_refinement_segment_start(payloadSize)	
else if(payloadType = = 17)	
progressive_refinement_segment_end(payloadSize)	
else if(payloadType = = 19)	
film_grain_characteristics(payloadSize)	
else if(payloadType = = 20)	
deblocking_filter_display_preference(payloadSize)	
else if(payloadType = = 22)	
post_filter_hint(payloadSize)	
else if(payloadType = = 23)	
tone_mapping_info(payloadSize)	

FIG 25A

FIG 25	FIG 25A
	FIG 25B

```
        else if( payloadType == 45 )
            frame_packing_arrangement( payloadSize )
        else if( payloadType == 47 )
            display_orientation( payloadSize )
        else if( payloadType == 128 )
            sop_description( payloadSize )
        else if( payloadType == 129 )
            field_indication( payloadSize )
        else if( payloadType == 130)
            decoded_picture_hash( payloadSize )
        else if( payloadType == 140 )
            roi_info( payloadSize )
        else if( payloadType == 180 )
            subpic_buffering ( payloadSize )
        else if( payloadType == 181 )
            subpic_timing( payloadSize )
        else if( payloadType == 182 )
            subpic_tile_info ( payloadSize )
        else if( payloadType == 183 )
            subpic_slice_info( payloadSize )
        else if( payloadType == 184 )
            subpic_tile_dimensions_info( payloadSize )
        if( !byte_aligned( ) ) {
            bit_equal_to_one /* equal to 1 */
            while( !byte_aligned( ) )
                bit_equal_to_zero /* equal to 0 */
        }
    }
```

FIG 25B

FIG 25	FIG 25A
	FIG 25B

subpic_buffering(payloadSize) {	Ký hiệu
seq_parameter_set_id	ue(v)
if(NalHrdBpPresentFlag) {	
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {	
initial_cpb_removal_delay[SchedSelIdx]	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx]	u(v)
}	
}	
if(VclHrdBpPresentFlag) {	
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {	
initial_cpb_removal_delay[SchedSelIdx]	u(v)
initial_cpb_removal_delay_offset[SchedSelIdx]	u(v)
}	
}	
}	

FIG 26

subpic_timing(payloadSize) {	Ký hiệu
reserved_one_8bits	u(8)
du_cpb_removal_delay	u(v)
du_dpb_output_delay	u(v)
}	
}	

FIG 27

subpic_slice_info(payloadSize) {	Ký hiệu
reserved_one_7bits	u(7)
slice_header_data_flag	u(1)
if(slice_header_data_flag)	
slice_header_data( )	
byte_alignment()	
}	

FIG 28

subpic_tile_info(payloadSize) {	Ký hiệu
reserved_one_4bits	u(4)
tile_priority	u(3)
multiple_tiles_in_prefixed_slices_flag	u(1)
if(multiple_tiles_in_prefixed_slices_flag)	
num_tiles_in_prefixed_slices_minus1	u(16)
first_tile_id_in_prefixed_slices	u(16)
}	

FIG 29

subpic_tile_dimensions _info(payloadSize) {	Ký hiệu
reserved_one_7bits	u(7)
multiple_tiles_in_prefixed_slices_flag	u(1)
if(multiple_tiles_in_prefixed_slices_flag)	
num_tiles_in_prefixed_slices_minus1	u(16)
for(i = 0; i <= num_tiles_in_roi_minus1 ; i++) {	
tile_horz_start[i]	u(16)
tile_width[i]	u(16)
tile_vert_start[i]	u(16)
tile_height[i]	u(16)
}	
}	

FIG 30

roi_info(payloadSize) {	Ký hiệu
reserved_one_8bits	u(8)
roi_id	u(5)
roi_priority	u(3)
num_tiles_in_roi_minus1	u(16)
for(i = 0; i <= num_tiles_in_roi_minus1 ; i++) {	
roi_tile_id[i]	u(16)
}	
}	

FIG 31

roi_info(payloadSize) {	Ký hiệu
reserved_one_3bits	u(3)
num_rois_minus1	u(5)
for(i = 0; i <= num_ roi_minus1 ; i++) {	u(5)
roi_id[i]	u(5)
roi_priority[i]	u(3)
num_tiles_in_roi_minus1[i]	u(16)
for(n = 0; n <= num_tiles_in_roi_minus1 ; n++) {	
roi_tile_id[i] [n]	u(16)
}	
roi_presentation_on_seperate_screen [i]	u(1)
}	
}	

FIG 32

decoding_unit_info(payloadSize) {	Ký hiệu
decoding_unit_idx	ue(v)
if(!sub_pic_cpb_params_in_pic_timing_sei_flag)	
du_spt_cpb_removal_delay_increment	u(v)
dpb_output_du_delay_present_flag	u(1)
if(dpb_output_du_delay_present_flag)	
pic_spt_dpb_output_du_delay	u(v)
}	

FIG 33

region_refresh_info(payloadSize) {	Ký hiệu
refreshed_region_flag	u(1)
}	

FIG 34

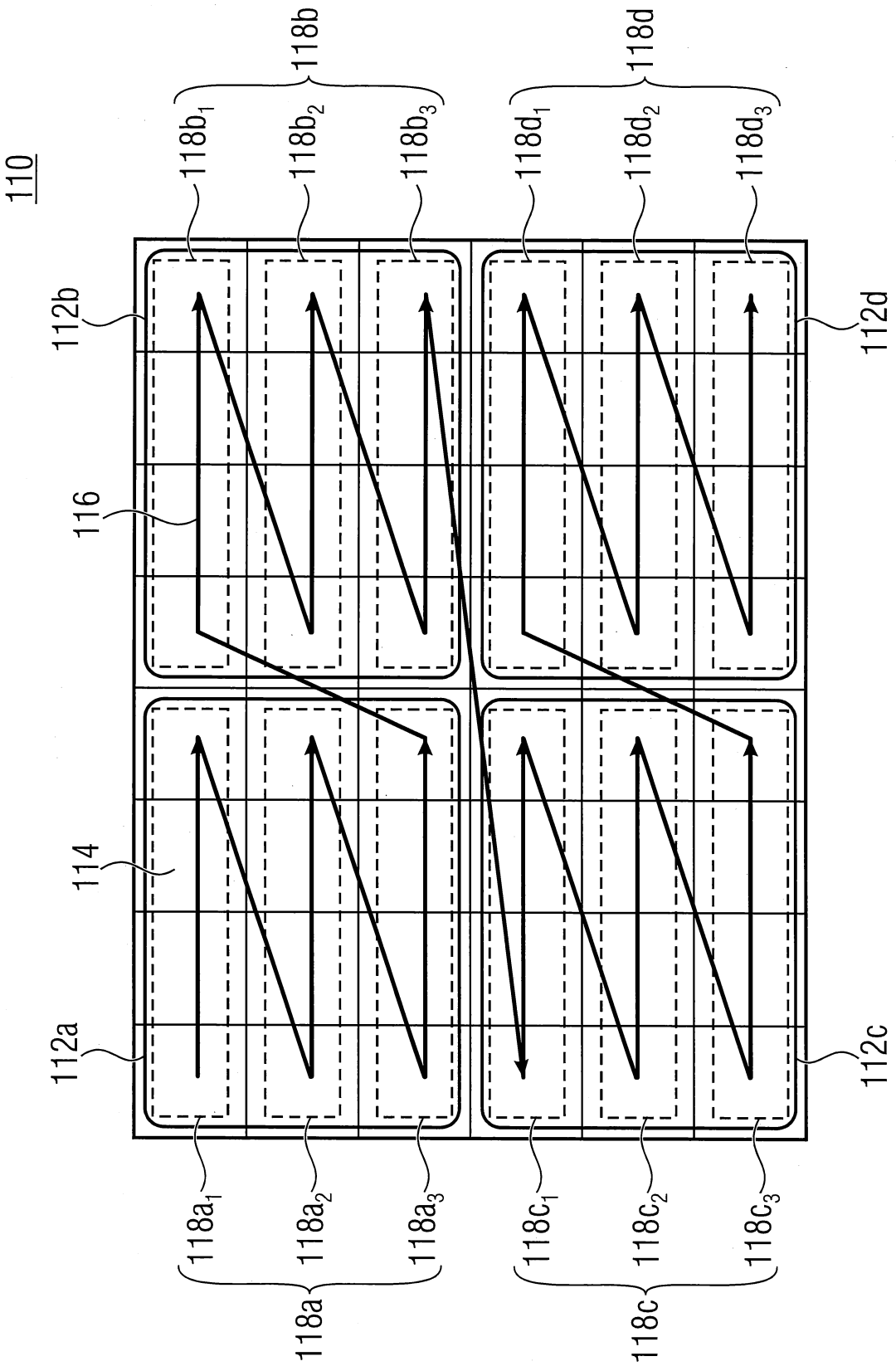

110
118b
118d
$118b_1$
$118b_2$
$118b_3$
$118d_1$
$118d_2$
$118d_3$
112b
112d
116
114
112a
112c
$118a_1$
$118a_2$
$118a_3$
$118c_1$
$118c_2$
$118c_3$
118a
118c

FIG 35

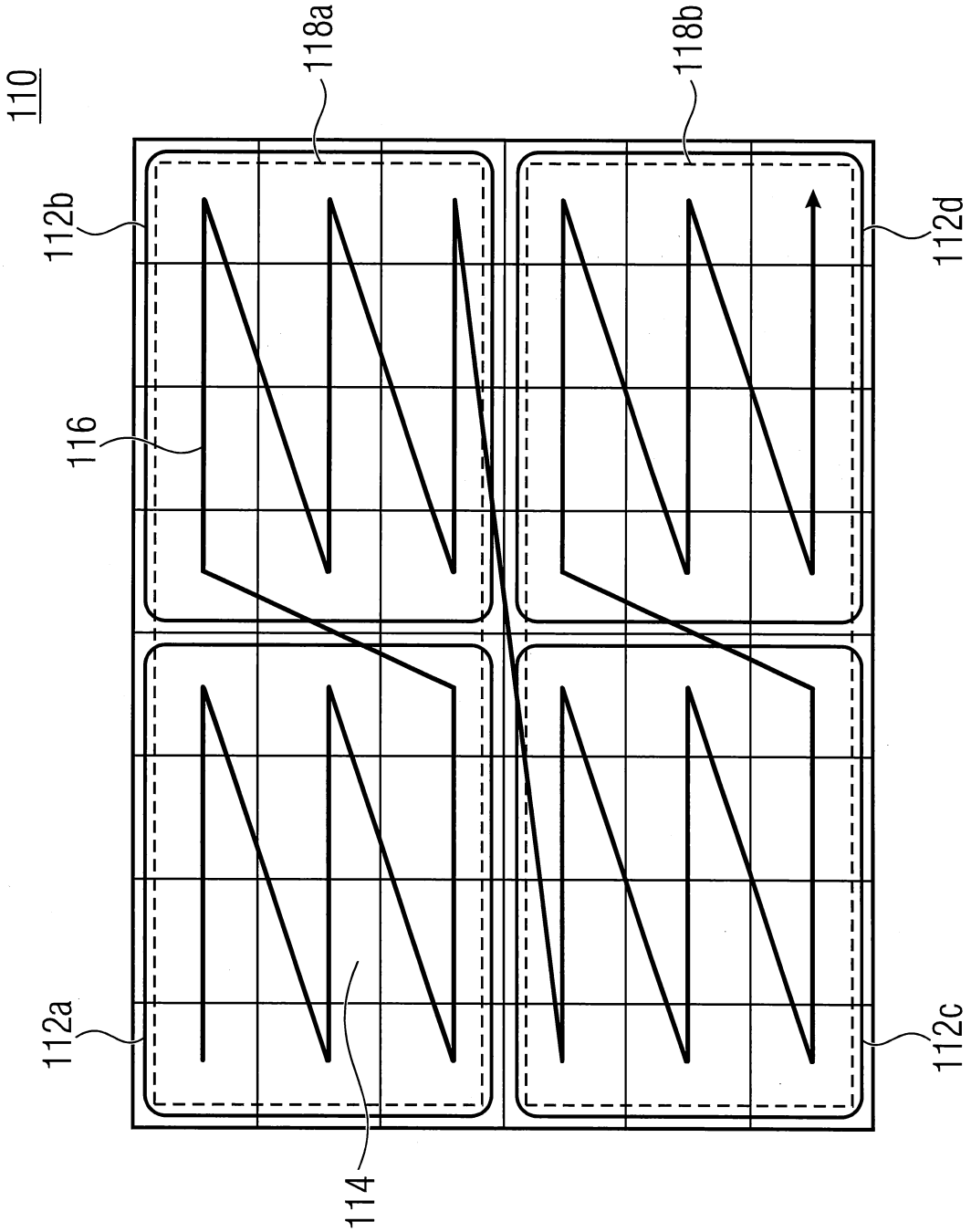


FIG 36

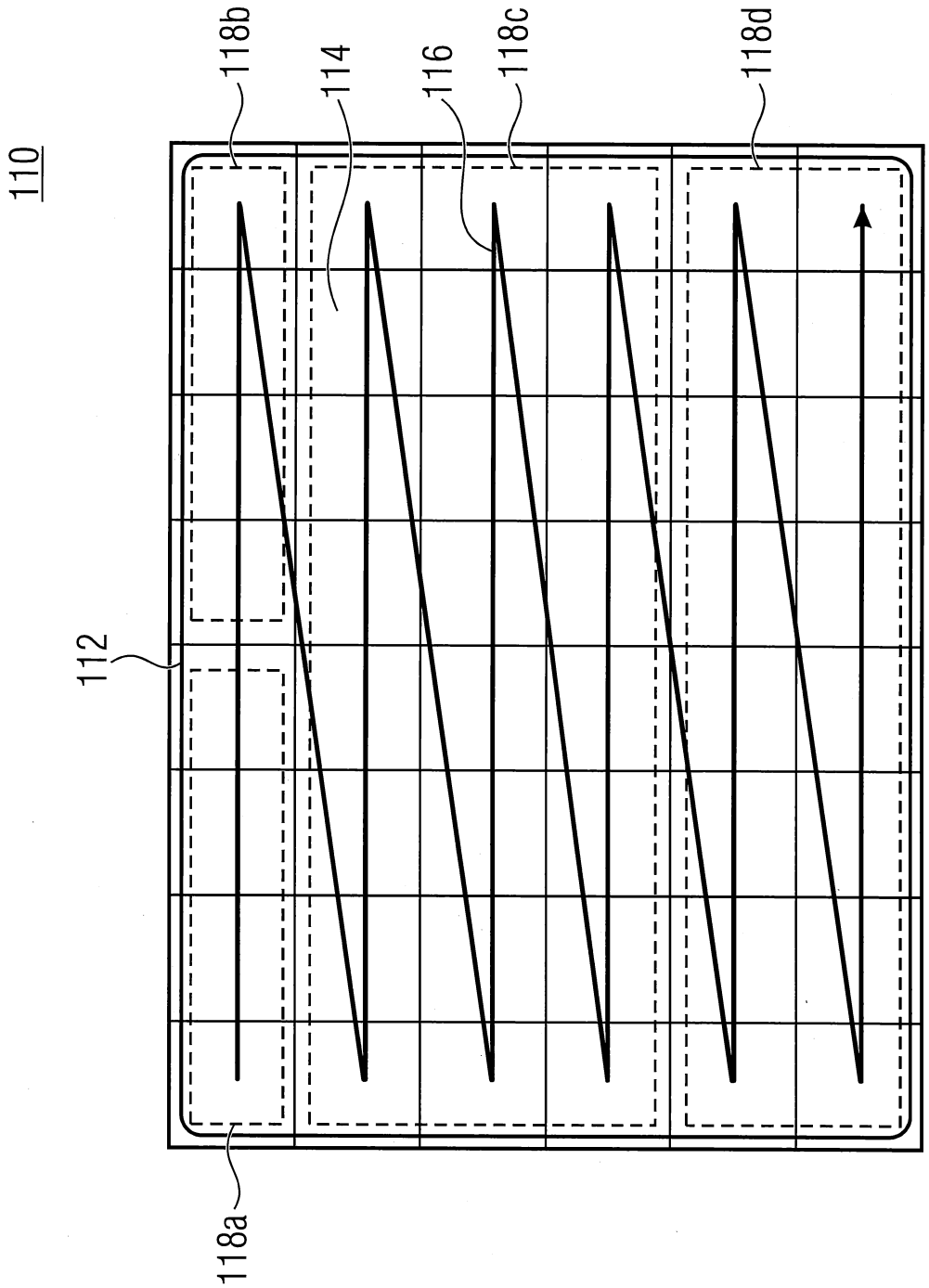


FIG 37

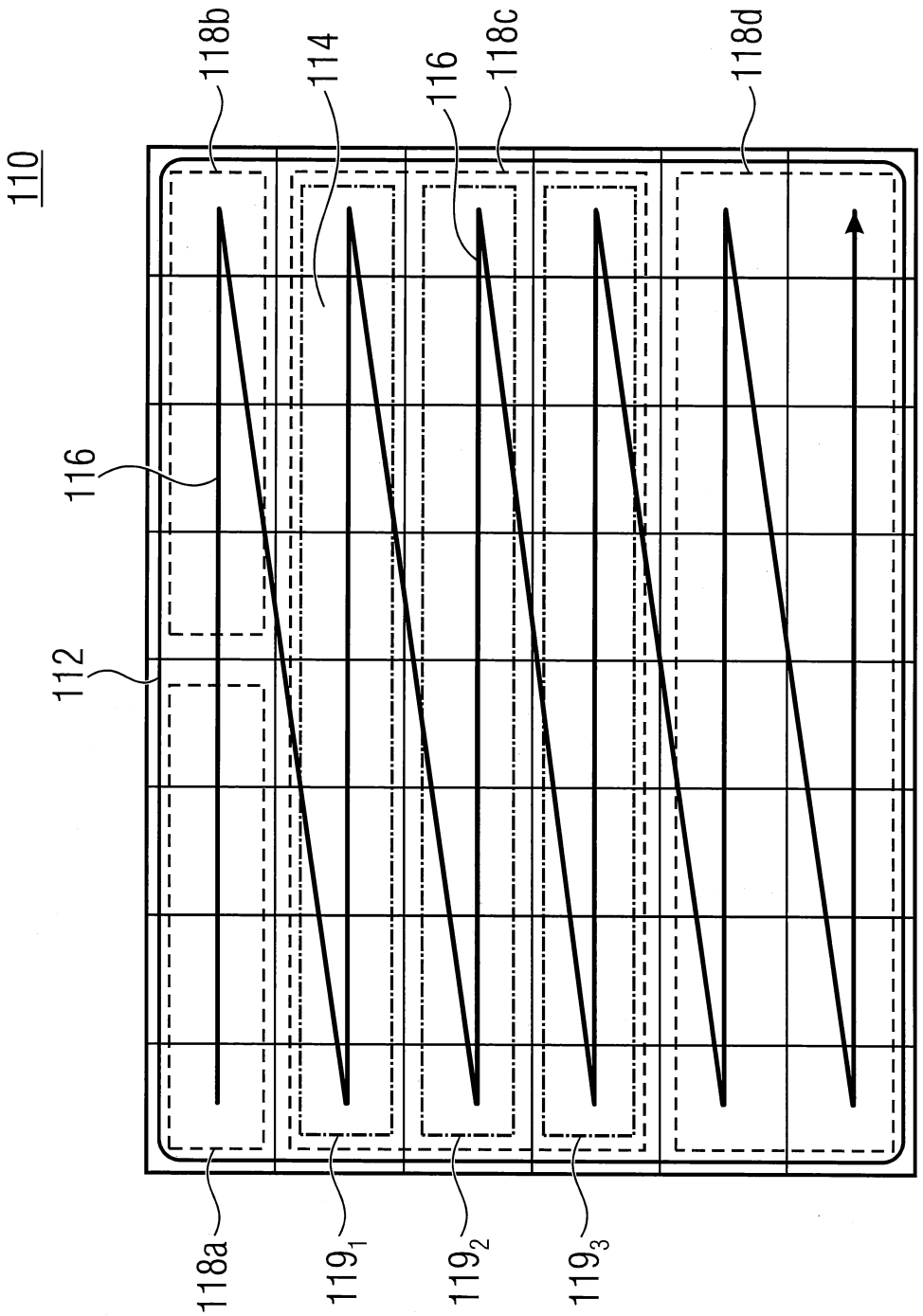


FIG 38

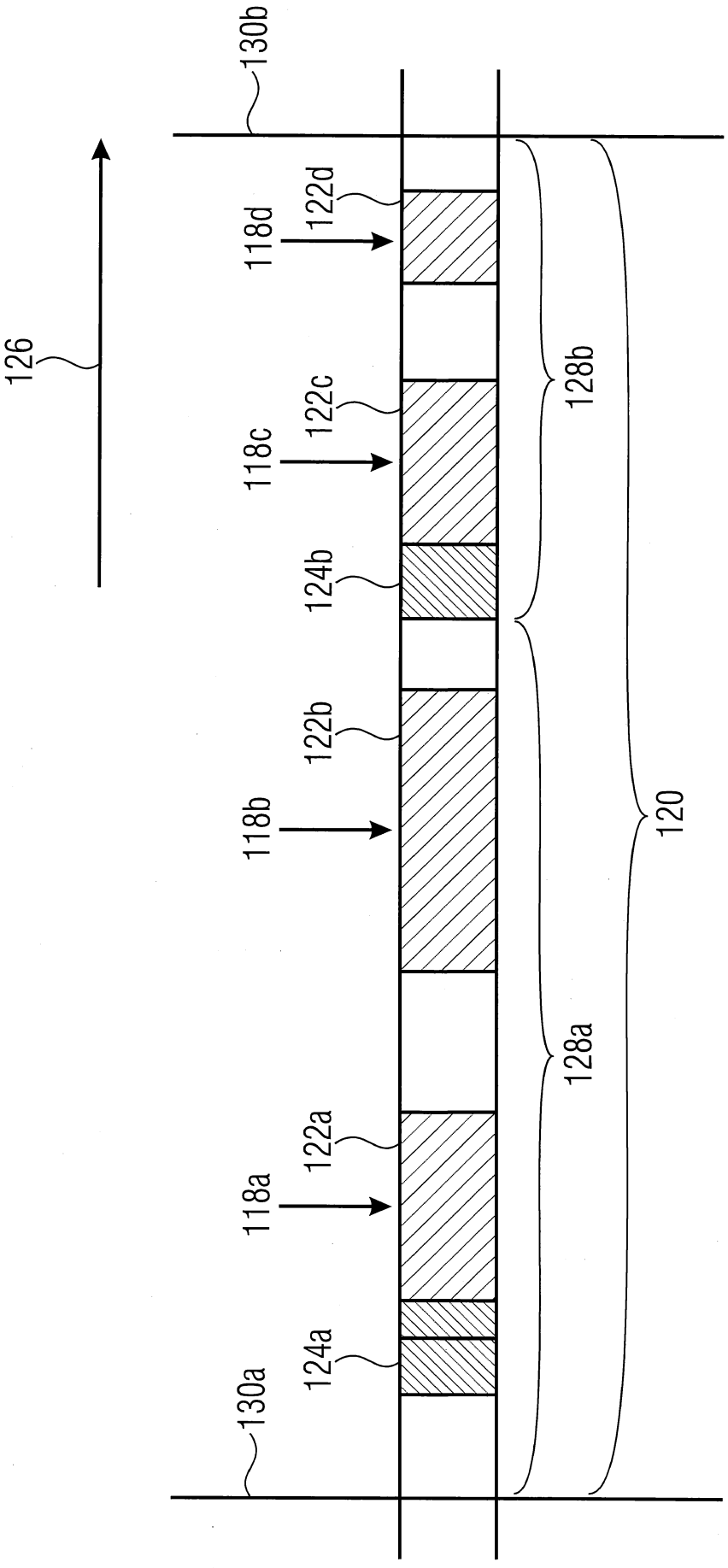

126
130b
118d
122d
118c
122c
124b
118b
122b
120
128b
128a
118a
122a
124a
130a

FIG 39