(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) **1-0026359**

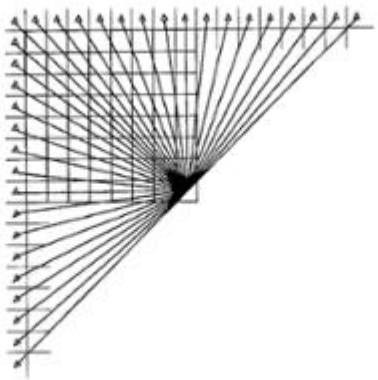
(51)[7] **H04N 7/34**

(13) **B**

(21) 1-2016-03064 (22) 14/07/2011
(62) 1-2013-00108
(86) PCT/US2011/044014 14/07/2011 (87) WO 2012/009540 A1 19/01/2012
(30) 61/364,322 14/07/2010 US; 61/388,541 30/09/2010 US
(45) 25/11/2020 392 (43) 25/04/2013 301A
(73) NTT DOCOMO, INC. (JP)
Sanno Park Tower, 11-1, Nagatacho 2-chome Chiyoda-ku Tokyo, 100-6150, Japan
(72) BOSSEN, Frank, Jan (NL); TAN, Thiow, Keng (MY).
(74) Công ty TNHH một thành viên Sở hữu trí tuệ VCCI (VCCI-IP CO.,LTD)

(54) PHƯƠNG PHÁP MÃ HOÁ VIĐEO VÀ PHƯƠNG PHÁP GIẢI MÃ VIĐEO

(57) Sáng chế đề cập đến phương pháp dự đoán trong duy nhất nhằm nâng cao hiệu quả mã hóa viđeo. H.264/AVC sử dụng các điểm ảnh tham chiếu ở đường biên ngang nằm ngay trên khối đích cần được dự đoán và các điểm ảnh tham chiếu ở đường biên thẳng đứng nằm ngay bên trái khối đích. Theo sáng chế, ít nhất một vài điểm ảnh của một dãy điểm ảnh ở đường biên ngang và dãy điểm ảnh ở đường biên thẳng đứng được lấy ra. Sau đó, các điểm ảnh được lấy ra được bổ sung vào các điểm ảnh ở đường biên khác để mở rộng dãy của nó. Dự đoán trong được thực hiện, chỉ dựa vào dãy điểm ảnh ở đường biên được mở rộng.



Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến mã hóa viđeo, cụ thể là sáng chế đề cập đến dự đoán trong khung trong đó khối mẫu được dự đoán, nhờ sử dụng các điểm ảnh được mã hóa và được tái cấu trúc trước đó từ cùng khung viđeo.

Tình trạng kỹ thuật của sáng chế

Viđeo số yêu cầu lượng lớn dữ liệu để biểu diễn mỗi khung và mọi khung của chuỗi viđeo số (chẳng hạn, dãy các khung) theo cách không được nén. Hầu hết các ứng dụng đều không thể truyền các viđeo số không được nén qua các mạng máy tính do sự giới hạn băng thông. Ngoài ra, viđeo số không được nén cần một lượng lớn dung lượng lưu trữ. Viđeo số thường được mã hóa theo một vài cách để làm giảm các yêu cầu lưu trữ và các yêu cầu về băng thông.

Một kỹ thuật để mã hóa viđeo số là dự đoán liên khung, hoặc dự đoán liên đới. Dự đoán liên đới khai thác sự dư thừa theo thời gian trong số các khung khác nhau. Các khung liền kề theo thời gian của viđeo thường bao gồm các khối điểm ảnh, mà còn lại về cơ bản là giống nhau. Trong suốt quy trình mã hóa, vectơ chuyển động có quan hệ với sự dịch chuyển của khối điểm ảnh trong một khung đến khối điểm ảnh tương tự trong khung khác. Do đó, hệ thống không cần mã hóa khối điểm ảnh hai lần, mà mã hóa khối điểm ảnh chỉ một lần và đưa ra vectơ chuyển động để dự đoán khối điểm ảnh khác.

Kỹ thuật khác để mã hóa viđeo số là dự đoán trong khung hoặc dự đoán trong. Dự đoán trong mã hóa khung hoặc một phần của nó mà không dựa vào các điểm ảnh trong các khung khác. Dự đoán trong khai thác sự dư thừa theo không gian trong số các khối điểm ảnh ở trong khung. Do các khối điểm ảnh liền kề theo không gian thường có các thuộc tính giống nhau, nên hiệu quả của quy trình mã hóa được nâng cao nhờ việc dựa vào mối tương quan theo không gian giữa các khối liền kề. Mối tương quan này có thể được khai thác nhờ việc dự đoán khối đích dựa vào các chế độ dự đoán được sử dụng trong các khối liền

kề.

Bản chất kỹ thuật của sáng chế

Sáng chế đề xuất quy trình dự đoán trong duy nhất nhằm nâng cao hiệu quả mã hóa viđeo. H.264/AVC sử dụng các điểm ảnh tham chiếu ở đường biên ngang nằm ngay trên khối đích cần được dự đoán và các điểm ảnh tham chiếu ở đường biên thẳng đứng nằm ngay bên trái của khối đích. Theo sáng chế, ít nhất một vài điểm ảnh trong số dãy điểm ảnh ở đường biên ngang hoặc dãy điểm ảnh ở đường biên thẳng đứng được lấy ra. Sau đó, các điểm ảnh được lấy ra được bổ sung vào các điểm ảnh ở đường biên khác để mở rộng dãy của nó. Dự đoán trong được thực hiện, chỉ dựa vào các dãy các điểm ảnh ở biên được mở rộng. Theo phương án của sáng chế, ít nhất một vài điểm ảnh trong số các điểm ảnh ở đường biên thẳng đứng được lấy ra và được bổ sung vào các điểm ảnh ở đường biên ngang để mở rộng dãy của nó.

Sáng chế loại bỏ quy trình quyết định lựa chọn hoặc đường biên ngang hoặc đường biên thẳng đứng mà từ đó các điểm ảnh tham chiếu được lấy ra. Sáng chế cũng loại bỏ quy trình lặp lại việc tính toán vị trí đường biên thẳng đứng giao với chiều dự đoán, trong đó quy trình tính toán lặp lại thường gồm phép chia. Việc loại bỏ các quy trình này cho phép quy trình dự đoán trong được thực hiện trên các cấu trúc một lệnh nhiều dữ liệu (Single-Instruction Multiple Data - SIMD), nhờ đó nâng cao hiệu quả tính toán của việc mã hóa viđeo.

Trong một phương án theo sáng chế, ít nhất một vài điểm ảnh trong số các điểm ảnh ở đường biên thẳng đứng được lấy ra, sử dụng bộ nhận dạng điểm ảnh theo chiều thẳng đứng được biểu diễn bởi:

$$[\frac{size \times col}{angle}],$$

trong đó *size* là kích cỡ của khối đích được dự đoán, *angle* là chiều dự đoán còn *col* là bộ đếm mà bị giảm đi 1 từ -1 đến *angle*. Các điểm ảnh được lấy ra được bổ sung vào các điểm ảnh theo chiều ngang ở vị trí được nhận dạng bởi bộ nhận dạng điểm ảnh theo chiều ngang [*col*].

Theo phương án khác, trong thời gian lấy ra ít nhất một vài điểm ảnh ở đường biên thẳng đứng, *InvAngle* được tính từ:

$$\frac{N \times size}{angle},$$

trong đó N là lũy thừa nguyên của 2. Sau đó, ít nhất một vài điểm ảnh trong số các điểm ảnh ở đường biên thẳng đứng được lấy ra, sử dụng bộ nhận dạng điểm ảnh theo chiều thẳng đứng mà được biểu diễn bởi [$col \times InvAngle >> \log_2 N$]. Các điểm ảnh được lấy ra được bổ sung vào các điểm ảnh theo chiều ngang ở vị trí được nhận dạng bởi bộ nhận dạng điểm ảnh theo chiều ngang [*col*].

Theo phương án khác, *InvAngle* thu được từ bảng tìm kiếm mà nó liệt kê các giá trị của *InvAngle* liên quan đến các giá trị của *angle*.

Theo phương án khác, điểm ảnh được nhận dạng trong số các điểm ảnh ở đường biên thẳng đứng, sử dụng bộ nhận dạng điểm ảnh theo chiều thẳng đứng [*row*], trong đó *row* là bộ đếm được tăng lên 1 từ 0 đến *size*. Điểm ảnh được lấy ra được bổ sung vào các điểm ảnh ở đường biên ngang ở vị trí được xác định nhờ vào bộ nhận dạng điểm ảnh theo chiều ngang [*int* + 1], trong đó *int* biểu diễn số nguyên của vị trí điểm ảnh giao với chiều dự đoán.

Sáng chế cũng đề xuất bộ mã hóa và bộ giải mã mà thực hiện hoạt động dự đoán trong mà trong đó ít nhất một vài điểm ảnh trong số dãy các điểm ảnh ở đường biên ngang hoặc dãy các điểm ảnh ở đường biên thẳng đứng được lấy ra. Sau đó, các điểm ảnh được lấy ra được bổ sung vào các điểm ảnh ở đường biên khác để mở rộng dãy của nó. Dự đoán trong được thực hiện, chỉ dựa vào dãy các điểm ảnh ở biên được mở rộng .

Mô tả vắn tắt các hình vẽ

Fig.1 là sơ đồ khối thể hiện cấu trúc phần cứng ví dụ mà trên đó sáng chế có thể được thực hiện.

Fig.2 là sơ đồ khối thể hiện tổng thể bộ mã hóa viđeo mà sáng chế có thể được áp dụng.

Fig.3 là sơ đồ khối thể hiện tổng thể bộ giải mã viđeo mà sáng chế có thể

được áp dụng.

Fig.4 là sơ đồ khối thể hiện các môđun chức năng của bộ mã hóa theo một phương án của sáng chế.

Fig.5 là lưu đồ thể hiện quy trình dự đoán trong được thực hiện bởi môđun dự đoán trong theo một phương án của sáng chế.

Fig.6 là sơ đồ khối thể hiện môđun chức năng của bộ giải mã theo một phương án của sáng chế.

Fig.7 là sơ đồ thể hiện các chiều dự đoán minh họa các chế độ dự đoán trong 4×4 được hỗ trợ trong H.264/AVC.

Fig.8 là sơ đồ thể hiện các chiều dự đoán được đề xuất trong tài liệu số JCT-VC A119.

Fig.9 là lưu đồ thể hiện quy trình, được đề xuất trong JCT-VC A119, tạo ra khối được dự đoán dọc theo mốt trong số các chiều dự đoán được thể hiện trên Fig.7.

Fig.10 là lưu đồ thể hiện quy trình dự đoán trong ít phức tạp được thực hiện theo một phương án của sáng chế.

Fig.11A là hình vẽ dưới dạng giản đồ thể hiện khối dự đoán và các dãy điểm ảnh ở đường biên ngang và đường biên thẳng đứng.

Fig.11B là hình vẽ dưới dạng giản đồ thể hiện dãy các điểm ảnh ở đường biên ngang được mở rộng với các điểm ảnh ở đường biên thẳng đứng.

Fig.12 là lưu đồ thể hiện quy trình mở rộng dãy các điểm ảnh ở đường biên ngang được thực hiện theo một phương án của sáng chế.

Fig.13 là lưu đồ thể hiện phương án khác về việc mở rộng dãy các điểm ảnh ở đường biên ngang.

Fig.14 là lưu đồ thể hiện quy trình dự đoán trong ít phức tạp được thực hiện theo phương án khác của sáng chế.

Mô tả chi tiết sáng chế

Fig.1 thể hiện cấu trúc phần cứng ví dụ của máy tính 100 mà trên đó sáng chế được thực hiện. Xin lưu ý là cấu trúc phần cứng được thể hiện trên Fig.1 có

thể là thông thường trong cả bộ mã hóa viđeo và bộ giải mã viđeo mà thực hiện các phương án của sáng chế. Máy tính 100 bao gồm bộ xử lý 101, bộ nhớ 102, thiết bị lưu trữ 105, và một hoặc nhiều thiết bị đầu ra và/hoặc đầu vào 106 (Input/Output - I/O) (hoặc các thiết bị ngoại vi) mà được ghép với nhau thông qua giao diện cục bộ 107. Giao diện cục bộ 107 có thể là, chẳng hạn, nhưng không bị giới hạn, một hoặc nhiều bus hoặc dây dẫn khác hoặc các liên kết không dây, như được biết trong lĩnh vực này.

Bộ xử lý 101 là thiết bị phần cứng để thực hiện phần mềm, cụ thể là được lưu trữ trong bộ nhớ 102. Bộ xử lý 101 có thể là bộ xử lý bất kỳ theo yêu cầu hoặc bộ xử lý có sẵn trên thị trường, bộ xử lý trung tâm (Central Processing Unit - CPU), bộ xử lý phụ trong số một vài bộ xử lý được kết hợp với máy tính 100, bộ vi xử lý trên cơ sở chất bán dẫn (theo dạng của mạch vi xử lý hoặc vi mạch), hoặc thiết bị thông thường bất kỳ để thực hiện các lệnh phần mềm.

Bộ nhớ 102 bao gồm phương tiện có thể đọc được bằng máy tính mà có thể bao gồm một hệ thống bất kỳ hoặc hệ thống các phần tử nhớ khả biến (chẳng hạn, bộ nhớ truy cập ngẫu nhiên (Random Access Memory - RAM, như DRAM, SRAM, SDRAM v.v.)) và các phần tử nhớ bất khả biến (chẳng hạn, ROM, ổ cứng, băng từ, CDROM v.v.). Hơn nữa, bộ nhớ 102 có thể có dạng điện tử, từ tính, quang học, và/hoặc dạng phương tiện lưu trữ khác. Phương tiện có thể đọc được bằng máy tính có thể là phương tiện bất kỳ mà có thể lưu trữ, truyền thông, lan truyền hoặc truyền chương trình để sử dụng hoặc kết nối với hệ thống, thiết bị hoặc cơ cấu thực hiện lệnh. Lưu ý là bộ nhớ 102 có thể có cấu trúc được phân bố sao cho các bộ phận khác nhau nằm tách biệt nhau, nhưng có thể được truy cập bởi bộ xử lý 101.

Phần mềm 103 trong bộ nhớ 102 có thể bao gồm một hoặc nhiều chương trình riêng biệt, từng chương trình chứa danh sách các lệnh có thể thực hiện được biểu thị để thực hiện các chức năng theo lôgic của máy tính 100 như được mô tả bên dưới. Theo ví dụ của Fig.1, phần mềm 103 trong bộ nhớ 102 xác định chức năng mã hóa viđeo và giải mã viđeo của máy tính 100 phù hợp với sáng chế. Ngoài ra, mặc dù không cần nhưng bộ nhớ 102 có thể chứa hệ điều hành

(Operating System - O/S) 104. Hệ điều hành 104 cần điều khiển sự thực hiện các chương trình máy tính và cung cấp sự lập lịch, điều khiển đầu vào-đầu ra, quản lý dữ liệu và tệp tin, quản lý bộ nhớ, kiểm soát truyền thông và các dịch vụ liên quan.

Thiết bị lưu trữ 105 của máy tính 100 có thể là một trong số nhiều loại thiết bị lưu trữ khác nhau, bao gồm thiết bị lưu trữ cố định hoặc thiết bị lưu trữ di động. Chẳng hạn, thiết bị lưu trữ 105 có thể là băng từ, đĩa từ, bộ nhớ tác động nhanh, bộ nhớ tức thời, hoặc thiết bị lưu trữ khác. Ngoài ra, thiết bị lưu trữ 105 có thể là thẻ nhớ số an toàn hoặc thiết bị lưu trữ di động bất kỳ 105 khác.

Các thiết bị I/O 106 có thể bao gồm các thiết bị đầu vào, chẳng hạn, nhưng không giới hạn với màn hình cảm ứng, bàn phím, chuột, máy quét, micrô hoặc thiết bị đầu vào khác. Ngoài ra, các thiết bị I/O 106 có thể cũng bao gồm các thiết bị đầu ra, chẳng hạn, nhưng không giới hạn ở màn hình hoặc các thiết bị đầu ra khác. Các thiết bị I/O 106 có thể còn bao gồm các thiết bị mà truyền thông thông qua cả các đầu vào và các đầu ra, chẳng hạn, nhưng không giới hạn bộ điều biến/bộ giải điều (môđem; ví dụ thiết bị, hệ thống, hoặc mạng khác), tần số rađio (Radio Frequency - RF), bộ thu-phát không dây hoặc bộ khác, giao diện điện thoại, cầu nối, bộ định tuyến hoặc các thiết bị khác mà có chức năng của cả đầu ra và đầu vào.

Như đã biết bởi người có trình độ trung bình trong lĩnh vực, việc nén viđeo đạt được bằng cách loại bỏ các thông tin dư trong chuỗi viđeo. Nhiều tiêu chuẩn mã hóa viđeo khác hiện có, bao gồm MPEG-1, MPEG-2, MPEG-4, H.261, H.263, và H.264/AVC chẳng hạn. Cần lưu ý là sáng chế không có mục đích giới hạn ứng dụng bất kỳ tiêu chuẩn mã hóa viđeo cụ thể nào. Tuy nhiên, phần mô tả tiếp theo của sáng chế được tạo ra, sử dụng ví dụ của tiêu chuẩn H.264/AVC, mà được kết hợp ở đây để tham khảo. H.264/AVC là tiêu chuẩn mã hóa viđeo mới nhất và đạt được sự cải thiện hiệu quả đáng kể so với các tiêu chuẩn mã hóa viđeo trước đó như MPEG-1, MPEG-2, H.261 và H.263.

Ở H.264/AVC, mỗi khung hoặc hình ảnh của viđeo có thể được phân chia thành các lát. Các lát sau đó được phân chia thành các khối điểm ảnh 16×16

được gọi là các khối macrô, mà sau đây có thể còn được phân chia thành các khối điểm ảnh 8×16, 16×8, 8×8, 4×8, 8×4, xuống đến 4×4. Có năm loại lát được hỗ trợ nhờ H.264/AVC. Trong các lát I, tất cả các khối macrô được mã hóa sử dụng dự đoán trong. Trong các lát P, các khối macrô có thể được mã hóa sử dụng dự đoán trong hoặc dự đoán liên đới. Các lát P cho phép chỉ một dự đoán bù chuyển động (Motion Compensated Prediction - MCP) báo hiệu mỗi khối macrô được sử dụng. Trong các lát B, các khối macrô có thể được mã hóa sử dụng dự đoán trong hoặc dự đoán liên đới. Hai tín hiệu MCP có thể được sử dụng cho mỗi dự đoán. Các lát SP cho phép các lát P được chuyển đổi giữa các chuỗi viđeo khác một cách hiệu quả. Lát SI là sự ăn khớp chính xác cho lát SP truy cập ngẫu nhiên hoặc khôi phục lỗi, trong khi chỉ sử dụng dự đoán trong.

Fig.2 thể hiện hình vẽ tổng thể về bộ mã hóa viđeo mà sáng chế có thể áp dụng. Các khối được thể hiện trên hình vẽ thể hiện các môđun chức năng được thực hiện nhờ bộ xử lý 101 thực hiện phần mềm 103 trong bộ nhớ 102. Bức ảnh của khung viđeo 200 được cấp tới bộ mã hóa viđeo 201. Bộ mã hóa viđeo xử lý bức ảnh 200 trong các bộ phận của các khối macrô 200A. Mỗi khối macrô chứa các điểm ảnh khác nhau của bức ảnh 200. Trên mỗi khối macrô, sự chuyển đổi thành các hệ số biến đổi được thực hiện sau đó nhờ việc lượng tử hoá thành các mức độ của hệ số biến đổi. Ngoài ra, dự đoán trong hoặc dự đoán liên đới được sử dụng, sao cho không thực hiện các bước mã hóa trực tiếp trên dữ liệu điểm ảnh mà trên sự chênh lệch của cùng các giá trị điểm ảnh được dự đoán, nhờ đó đạt được các giá trị nhỏ mà được nén một cách dễ dàng hơn.

Đối với mỗi lát, bộ mã hóa 201 tạo ra một vài phần tử cú pháp, mà tạo thành kiểu được mã hóa của các khối macrô của lát tương ứng. Tất cả các phần tử dữ liệu dư trong các phần tử cú pháp, mà liên quan tới việc mã hóa của các hệ số biến đổi, như là các mức độ của hệ số biến đổi hoặc bản đồ quan trọng biểu thị các mức độ của hệ số biến đổi được bỏ qua, được gọi là các phần tử cú pháp dữ liệu dư. Ngoài các phần tử cú pháp dữ liệu dư, các phần tử cú pháp được tạo ra nhờ bộ mã hóa 201 chứa các phần tử cú pháp thông tin điều khiển mà chứa thông tin điều khiển như cách mỗi khối macrô được mã hóa và được giải mã

tương ứng. Nói cách khác, các phần tử cú pháp có thể phân chia thành hai loại. Loại thứ nhất, các phần tử cú pháp thông tin điều khiển, chứa các chi tiết liên quan đến loại khối macrô, loại khối macrô phụ và thông tin trên các chế độ dự đoán ở cả loại theo không gian và loại theo thời gian, cũng như thông tin điều khiển dựa vào khối macrô và dựa vào lát chẳng hạn. Trong loại thứ hai, tất cả các phần tử dữ liệu dư, như bản đồ quan trọng biểu thị các vị trí của các hệ số quan trọng bên trong khối của các hệ số biến đổi được lượng tử hoá và các giá trị của các hệ số quan trọng, mà được biểu thị trong các bộ phận của các mức độ tương ứng với các bước lượng tử hoá, được kết hợp và trở thành các phần tử cú pháp dữ liệu dư.

Bộ mã hóa 201 bao gồm bộ mã hóa entrôpi mà mã hóa các phần tử cú pháp và tạo ra các từ mã thuật toán cho mỗi lát. Khi tạo ra các từ mã thuật toán cho lát, bộ mã hóa entrôpi khai thác các phần phụ thuộc thống kê giữa các giá trị dữ liệu của các phần tử cú pháp trong dòng bit tín hiệu viđeo. Bộ mã hóa 201 cung cấp tín hiệu viđeo được mã hóa cho lát của hình ảnh 200 tới bộ giải mã viđeo 301 được thể hiện trên Fig.3.

Fig.3 thể hiện hình vẽ tổng thể của bộ giải mã viđeo mà sáng chế có thể được áp dụng. Hơn nữa, các khối thể hiện trên hình vẽ thể hiện các môđun chức năng được thực hiện nhờ bộ xử lý 101 thực hiện phần mềm 103 trong bộ nhớ 102. Bộ giải mã viđeo 301 nhận tín hiệu viđeo được mã hóa và tín hiệu giải mã entrôpi thứ nhất quay lại các phần tử cú pháp. Bộ giải mã 301 sử dụng các phần tử cú pháp để tái cấu trúc, từng khối macrô và sau đó là từng lát, các mẫu hình ảnh 300A của các điểm ảnh trong hình ảnh 300.

Fig.4 thể hiện các môđun chức năng của bộ mã hóa viđeo 201. Các môđun chức năng này được thực hiện nhờ bộ xử lý 101 thực hiện phần mềm 103 trong bộ nhớ 102. Hình ảnh viđeo được đưa vào là khung hoặc trường hình ảnh viđeo tự nhiên (không được nén) được xác định nhờ các điểm mẫu thể hiện các thành phần của các màu gốc, như độ thuần khiết màu sắc ("sắc độ") và cường độ sáng ("độ sáng") (các thành phần khác có thể có, chẳng hạn, màu sắc, độ bão hoà và sự phối màu). Hình ảnh viđeo đầu vào được phân chia thành các khối macrô 400

mà đại diện cho từng vùng hình ảnh vuông bao gồm các điểm ảnh 16×16 của thành phần độ sáng của màu sắc hình ảnh. Hình ảnh viđeo đầu vào cũng được phân chia thành các khối macrô mà mỗi khối này thể hiện các điểm ảnh 8×8 của mỗi trong số hai thành phần sắc độ của màu sắc hình ảnh. Trong hoạt động của bộ mã hóa thông thường, các khối macrô được đưa vào có thể được dự đoán theo không gian hoặc theo thời gian sử dụng dự đoán trong hoặc dự đoán liên đới. Tuy nhiên các khối macrô 400 được giả định nhằm mục đích cho rằng nó là tất các khối macrô loại lát I và chỉ được đưa ra để thực hiện dự đoán trong.

Dự đoán trong đã hoàn thành ở môđun dự đoán trong 401 sẽ được mô tả ở bên dưới về chi tiết hoạt động. Môđun dự đoán trong 401 tạo ra khối dự đoán 402 từ các điểm ảnh ở đường biên ngang và đường biên thẳng đứng của các khối liền kề, mà đã được mã hóa, được tái cấu trúc, và được lưu trữ trước đó trong bộ nhớ khung 403. Phần dư 404 của khối dự đoán 402, mà có sự chênh lệch giữa khối đích 400 và khối dự đoán 402, được biến đổi, được thu nhỏ và được lượng tử hoá ở môđun biến đổi/lượng tử hoá 405, sử dụng các phương pháp và các kỹ thuật đã biết đối với người có trình độ trung bình trong lĩnh vực mã hóa viđeo. Các hệ số biến đổi được lượng tử hoá 406 sau đó được mã hóa entrôpi ở môđun mã hóa entrôpi 407 và được truyền (cùng với các thông tin liên quan tới dự đoán trong khác) như tín hiệu viđeo được mã hóa 408.

Bộ mã hóa viđeo 201 chứa chức năng giải mã để thực hiện dự đoán trong trên các khối đích. Chức năng giải mã bao gồm môđun lượng tử hoá/biến đổi ngược 409, mà thực hiện lượng tử hoá ngược và biến đổi ngược trên các hệ số biến đổi được lượng tử hoá 406 để tạo ra phần dư dự đoán được giải mã 410, mà được bổ sung vào khối dự đoán 402. Tổng của phần dư dự đoán được giải mã 410 và khối dự đoán 402 là khối được tái cấu trúc 411, mà được lưu trữ trong bộ nhớ khung 403 và sẽ được đọc và được sử dụng bởi môđun dự đoán trong 401 để tạo ra khối dự đoán 402 dùng cho việc giải mã khối đích 400 tiếp theo.

Fig.5 là lưu đồ thể hiện các quy trình được thực hiện bởi môđun dự đoán trong 401. Theo tiêu chuẩn H.264/AVC, dự đoán trong liên quan đến việc dự đoán mỗi điểm ảnh của khối đích 400 dưới các chế độ dự đoán, sử dụng các sự

nội suy của các điểm ảnh ở đường biên ("các điểm ảnh tham chiếu") của các khối liền kề được mã hóa và được tái cấu trúc trước đó. Các chế độ dự đoán được nhận dạng nhờ các số nguyên dương 0, 1, 2 v.v. mỗi số được kết hợp với một lệnh hoặc thuật toán khác nhau để dự đoán các điểm ảnh cụ thể trong khối đích 400. Môđun dự đoán trong 401 thực hiện dự đoán trong dưới các chế độ dự đoán tương ứng và tạo ra các khối dự đoán khác nhau. Bằng thuật toán tìm kiếm toàn bộ ("full search FS"), từng khối dự đoán được tạo ra được so sánh với khối đích 400 để tìm ra chế độ dự đoán tối ưu, mà giảm thiểu phần dư dự đoán 404 hoặc tạo ra phần dư dự đoán 404 ít hơn trong số các chế độ dự đoán. Sự xác định chế độ dự đoán tối ưu được nén lại và được gửi tới bộ giải mã 301 với các phần tử cú pháp thông tin điều khiển khác.

Mỗi chế độ dự đoán có thể được mô tả bởi chiều dự đoán thông thường như được mô tả bằng lời nói (nghĩa là, chiều ngang hướng lên, chiều thẳng đứng và chiều bên trái hướng xuống theo đường chéo). Chiều dự đoán có thể được mô tả bằng đồ thị nhờ chiều góc mà được biểu thị bằng sơ đồ với các mũi tên như được thể hiện trên Fig.7. Trong loại sơ đồ này, mỗi mũi tên có thể được xem xét thể hiện một chiều dự đoán hoặc một chế độ dự đoán. Góc tương ứng với chế độ dự đoán có quan hệ thông thường với chiều từ vị trí bình quân gia quyền của các điểm ảnh tham chiếu được sử dụng để dự đoán điểm ảnh mục tiêu tới vị trí điểm ảnh mục tiêu. Lưu ý là các chế độ dự đoán bao gồm chế độ dự đoán DC mà không được kết hợp với bất kỳ chiều dự đoán nào và, do đó, không thể được mô tả bằng lời nói trong sơ đồ không giống như các chế độ dự đoán khác. Trong chế độ dự đoán DC, khối dự đoán 402 được tạo ra sao cho mỗi điểm ảnh trong khối dự đoán 402 được thiết lập đồng nhất với giá trị trung bình của các điểm ảnh tham chiếu.

Quay lại Fig.5, chế độ dự đoán được bắt đầu ở bước 501. Sau đó được xác định ở bước 502 rằng chế độ dự đoán có biểu thị dự đoán DC hay không. Nếu có thì quy trình chuyển sang bước 503, nơi mà khối dự đoán DC 402 được tạo ra với giá trị trung bình của các điểm ảnh tham chiếu ở bước 503. Nếu chế độ dự đoán biểu thị cách khác, thì khối dự đoán 402 được tạo ra theo lệnh hoặc thuật

toán được kết hợp với chế độ dự đoán ở bước 504, mà quy trình sẽ được mô tả chi tiết dưới đây. Sau bước 503 hoặc 504, quy trình chuyển sang bước 505, mà xác định rằng các khối dự đoán có được tạo ra cho tất cả các chế độ dự đoán hay không. Nếu dự đoán trong được thực hiện dưới tất cả các chế độ dự đoán, quy trình chuyển sang bước 506. Nếu khác, chế độ dự đoán được tăng lên bước 507 và quy trình quay lại bước 502. Trong bước 506, từng khối dự đoán được tạo ra được so sánh với khối đích 400 để xác định chế độ dự đoán tối ưu, mà giảm thiểu phần dư dự đoán 404.

Fig.6 thể hiện các môđun chức năng của bộ giải mã viđeo 301. Các môđun chức năng này được thực hiện nhờ bộ xử lý 101 thực hiện phần mềm 103 trong bộ nhớ 102. Tín hiệu viđeo được mã hóa từ bộ mã hóa 201 được tiếp nhận trước tiên bởi bộ giải mã entrôpi 600 và được giải mã entrôpi để trở lại thành các hệ số biến đổi được lượng tử hoá 601. Các hệ số biến đổi được lượng tử hoá 601 được lượng tử hoá và được biến đổi ngược nhờ môđun lượng tử hoá/biến đổi ngược 602 để tạo ra phần dư dự đoán 603. Môđun dự đoán trong 604 cho biết chế độ dự đoán được lựa chọn bởi bộ mã hóa 201. Theo chế độ dự đoán được chọn, môđun dự đoán trong 604 thực hiện quy trình dự đoán trong tương tự như được thực hiện ở các bước 502, 503 và 504 của Fig.5 để tạo ra khối dự đoán 605, sử dụng các điểm ảnh ở biên của các khối liền kề được tái cấu trúc và được lưu trữ trước đó trong bộ nhớ khung 606. Khối dự đoán 605 được bổ sung vào phần dư dự đoán 603 để tái cấu trúc khối 607 của tín hiệu viđeo được giải mã. Khối được tái cấu trúc 607 được lưu trữ trong bộ nhớ khung 606 để sử dụng trong việc dự đoán của các khối tiếp theo.

Phần mô tả chi tiết sẽ được đưa ra sau đây trên quy trình của bước 504 được thực hiện nhờ các môđun dự đoán trong 401 và 604 để tạo ra khối dự đoán dưới một trong số các chế độ dự đoán, ngoại trừ chế độ dự đoán DC. H.264/AVC hỗ trợ dự đoán trong 4×4, dự đoán trong 8×8 và dự đoán trong 16×16. Dự đoán trong 4×4 được sử dụng phổ biến khi có chi tiết quan trọng trong hình ảnh. Dự đoán trong 4×4 dự đoán mười sáu khối độ sáng 4×4 ở trong một khối macrô riêng biệt. Dự đoán trong 4×4 được thực hiện dưới chín chế độ

dự đoán, bao gồm một chế độ dự đoán DC. Các chiều dự đoán theo không gian dọc theo dự đoán trong 4×4 được thực hiện được thể hiện trên Fig.7. Dự đoán trong 8×8 được thực hiện dưới chín chế độ dự đoán, bao gồm một chế độ dự đoán DC. Dự đoán trong 16×16 được thực hiện dưới bốn chế độ dự đoán, bao gồm một chế độ dự đoán DC.

Các nghiên cứu gần đây thể hiện rằng việc tăng số chiều dự đoán, hoặc tăng số chế độ dự đoán, thường góp phần nâng cao hiệu quả nén trong việc mã hóa viđeo. Chẳng hạn xem tài liệu số JCT-VC A119 ("dự đoán trong góc") và JCT-VC A124 ("bên trong chiều bất kỳ") được đăng ký với Nhóm hợp tác chung về việc mã hóa viđeo (Joint Collaborative Team on Video Coding - JCT-VC), rằng cả hai đều được kết hợp ở đây để tham khảo. Việc tăng số chiều dự đoán dẫn đến tăng số khoảng cách góc của các chiều dự đoán sẵn có và, do đó, dẫn đến tăng các sự tham gia của khối dự đoán. Các sự tham gia của khối dự đoán được tăng lên đơn giản làm tăng cơ hội có khối dự đoán mà gần giống như khối đích được mã hóa. Fig.8 là sơ đồ thể hiện các chiều dự đoán được đề xuất trong tài liệu số JCT-VC A119. Trên Fig.8, các điểm ảnh tham chiếu bao gồm mười bẩy (17) điểm ảnh theo chiều ngang và mười bẩy (17) điểm ảnh theo chiều thẳng đứng, trong đó điểm ảnh phía trái bên trên là thông dụng với cả các đường biên ngang đường biên thẳng đứng. Do đó, 33 chiều dự đoán khác nhau sẵn sàng để tạo ra các điểm ảnh dự đoán ở khối 8×8. JCT-VC A124 đề xuất dự đoán trong theo chiều bất kỳ mà trong đó số chiều dự đoán được điều chỉnh theo kích cỡ của khối được dự đoán.

Fig.9 là lưu đồ thể hiện quy trình, được đề xuất trong JCT-VC A119, mà nó tạo ra khối dự đoán dọc theo một trong số các chiều dự đoán được thể hiện trên Fig.8. Trong phần mô tả tiếp theo của quy trình, một vài thuật toán được đơn giản hoá để giải thích dễ dàng hơn. Ngoài ra, quy trình được mô tả bị giới hạn ở dự đoán trong dọc theo chiều dự đoán mà chủ yếu là theo chiều thẳng đứng. Dự đoán trong dọc theo chiều dự đoán mà chủ yếu là theo chiều ngang có thể được thực hiện đối xứng với quy trình được thể hiện trên Fig.9, như được chứng minh trong phần mềm được cung cấp bởi JCT-VC A119. Mặc dù Fig.8

thể hiện khối 8×8 được đự đoán, nhưng quy trình được thể hiện trên Fig.9 có thể được mở rộng để được áp dụng cho số các điểm ảnh khác nhau theo các kết cấu khác nhau. Chẳng hạn, khối được đự đoán có thể bao gồm dãy các điểm ảnh 4×4. Khối dự đoán cũng có thể bao gồm dãy các điểm ảnh 8×8, 16×16, hoặc các dãy điểm ảnh lớn hơn. Các kết cấu điểm ảnh khác, bao gồm cả các dãy hình chữ nhật và hình vuông, cũng có thể tạo ra khối dự đoán.

Trong bước 900 trên Fig.9, các điểm ảnh tham chiếu ở các đường biên ngang hoặc đường biên thẳng đứng, mà nằm ngay bên trên và bên trái của khối đích, một cách tương ứng, được đọc từ các khối liền kề mà được mã hóa, được tái cấu trúc và được lưu trữ trước đó trong bộ nhớ khung, như bộ nhớ 403 được thể hiện trên Fig.4. Các điểm ảnh từ đường biên ngang được lưu trữ trong vùng bộ nhớ được gọi là "*refH*". Các điểm ảnh từ đường biên thẳng đứng được lưu trữ trong vùng bộ nhớ khác được gọi là "*refV*". Quay lại Fig.8, các điểm ảnh tham chiếu được xác định nhờ các tọa độ của nó trong hệ tọa độ có gốc ở vị trí điểm ảnh phía trên bên trái trong khối 8×8. Do đó, các điểm ảnh ở đường biên ngang có các tọa độ được biểu diễn bởi p[x, y] với x = 0, 1…16 và y = 0. Các điểm ảnh ở đường biên thẳng đứng có các tọa độ được biểu diễn bởi p[x, y] với x = 0, y = 0, -1, -2…-16.

Giả định rằng các điểm ảnh ở đường biên ngang được lưu trữ trong vùng bộ nhớ *refH* được xác định bởi các địa chỉ lôgic (x) với x = 0, 1…16 và các điểm ảnh ở đường biên thẳng đứng được lưu trữ trong vùng bộ nhớ *refV* được xác định cũng như thế bởi địa chỉ lôgic (y) với y = 0, -1, -2…-16, trong đó mỗi điểm ảnh được lưu trữ trong địa chỉ có số tọa độ mà được đọc. Do đó, như các điểm ảnh theo chiều ngang và theo chiều thẳng đứng được thể hiện bằng đồ thị trên Fig.8, các vùng bộ nhớ *refH* và *refV* có thể được xem xét mở rộng theo đường thẳng và vuông góc với nhau và mỗi vùng có chiều dài 2 × *size* + 1, trong đó "*size*" là tham số thể hiện kích cỡ của khối đích. Giả định rằng *size* có giá trị bằng với lũy thừa nguyên của 2, như là 4, 8, 16 v.v.. Bộ lọc thông thấp như được mô tả trong phần 8.3.2.2.1 trong H.264/AVC có thể được áp dụng tối ưu đối với các điểm ảnh trong *refH* và *refV*.

Trong bước 901, bộ đếm được gọi là "*row*" được thiết lập về không ("0"). Bộ đếm *row* nhận giá trị từ 0 tới *size* và biểu thị vị trí hàng của điểm ảnh dự đoán trong khối dự đoán. Trong bước 902, tham số được gọi là "*pos*" được tính bởi *angle* × (*row* + 1). *angle* là tham số có số thập phân trong việc biểu diễn điểm cố định. Như vậy, *angle* được tạo thành với phần nguyên và phần thập phân, và phần thập phân bao gồm số cố định của các chữ số nhị phân. *angle* thể hiện một trong số các chiều dự đoán được thể hiện trên Fig.8. Ví dụ, "*angle* = -*size*" xác định chiều dự đoán đi qua tọa độ [x = 0, y = 0] trên Fig.8. *angle* có giá trị dương xác định chiều dự đoán chỉ cắt đường biên ngang, trái lại *angle* có giá trị âm xác định chiều dự đoán cắt cả đường biên ngang và đường biên thẳng đứng. *angle* thay thay đổi ở trong khoảng được xác định bởi số chiều dự đoán mong muốn sử dụng. Như được cung cấp trong JCT-VC A124, số chiều dự đoán được sử dụng có thể được xác định theo kích cỡ của khối được dự đoán. Trong phần mô tả tiếp theo, giả định rằng *angle* nhận số thập phân mà thay đổi trong khoảng từ "-*size*" tới "*size*". Lưu ý là khoảng giới hạn của *angle* có thể được xác định với các giá trị khác.

Như *angle*, tham số *pos* bao gồm phần nguyên và phần thập phân, và phần thập phân của nó bao gồm số cố định của số nhị phân, mà bằng với logarit cơ số 2 của khoảng giới hạn của *angle*, mà có thể được tính bằng *log2_size* theo giả thiết bên trên mà khoảng giới hạn của *angle* được thiết lập ở *size*. *pos* xác định vị trí giao giữa đường biên ngang và chiều dự đoán được thể hiện bởi *angle*. Quay lại bước 902, phép tính "*pos* >> *log2_size*" xác định số nguyên trong *pos*, mà được lưu trữ trong tham số "*int*", và phép tính "*pos* & (*size* - 1)" xác định số thập phân trong *pos*, mà được lưu trữ trong tham số "*frac*". Toán tử ">>" gọi là phép dịch phải của số nhị phân. Toán tử "&" gọi là phép tính "và" từng bit.

Trong bước 903, *angle* được xác nhận xem có giá trị lớn hơn hoặc bằng không ("0") hay không. Nếu *angle* có giá trị lớn hơn hoặc bằng không, quy trình chuyển sang bước 904. Nếu khác thì quy trình chuyển sang bước 913. *angle* lớn hơn hoặc bằng không thể hiện rằng chỉ các điểm ảnh tham chiếu nằm ở đường biên ngang, hoặc được lưu trữ trong *refH*, mới có thể được dựa vào để lấy ra các

điểm ảnh dự đoán trong khối dự đoán. Mặt khác, *angle* nhỏ hơn không đề xuất rằng các điểm ảnh tham chiếu nằm ở đường biên thẳng đứng, hoặc được lưu trữ trong *refV*, là cần thiết để lấy ra các điểm ảnh dự đoán trong khối dự đoán.

Trong bước 904, *frac* được xác định xem có khác không hay không. Nếu *frac* khác không, quy trình chuyển sang bước 905. Nếu *frac* là không, quy trình chuyển sang bước 906. *frac* bằng không thể hiện rằng điểm ảnh dự đoán trong khối dự đoán có thể được sao chép trực tiếp từ điểm ảnh tham chiếu trong đường biên ngang. *frac* khác không đề xuất rằng chiều dự đoán cắt đường biên ngang ở vị trí không phải số nguyên, và việc bổ sung vào nhiều hơn một điểm ảnh tham chiếu là cần thiết để lấy ra điểm ảnh dự đoán trong khối dự đoán.

Trong bước 905, bộ đếm được gọi là "*col*" được thiết lập về không ("0"). Bộ đếm *col* được sử dụng để chỉ định điểm ảnh tham chiếu trong *refH*. Trong bước 907, hai điểm ảnh tham chiếu được xác định bằng "*int* + *col* + 1" và "*int* + *col* + 2" được lấy ra từ *refH*. Hai điểm ảnh tham chiếu này là bình quân gia quyền hoặc được nội suy với *frac* để lấy ra điểm ảnh dự đoán *v*. Cụ thể, điểm ảnh tham chiếu trong *refH* được xác định bởi "*int* + *col* + 1" được nhân với "*size* - *frac*" và được lưu trữ trong tham số *a*. Điểm ảnh tham chiếu trong *refH* được xác định bởi "*int* + *col* + 2" được nhân với "*frac*" và được lưu trữ trong tham số *b*. Các tham số *a* và *b* sau đó được cộng vào và được chia cho *size*, nghĩa là, (*size* - *frac*) + *frac*. Việc chia cho *size* có thể được thay thế bằng sự dịch phải bởi *log2_size*. Điểm ảnh dự đoán *v* được lấy ra được lưu trữ trong dãy của các vùng bộ nhớ được gọi là "*pred,*" mà thể hiện khối dự đoán cho khối đích theo chiều dự đoán cụ thể. Mỗi vùng bộ nhớ trong *pred* được xác định bằng các tham số *row* và *col*. Sau đó, *col* được tăng lên 1 trong bước 908 và được so sánh với *size* trong bước 909. Chừng nào mà *col* nhỏ hơn *size*, các bước 907 và 908 được lặp lại. Khi *col* bằng với *size*, quy trình chuyển sang bước 920.

Nếu *frac* được xác định là không trong bước 904, bộ đếm *col* được thiết lập về không trong bước 906. Trong bước 910, điểm ảnh dự đoán *v* được sao chép trực tiếp từ *refH* (*int* + *col* + 1) và sau đó được lưu trữ trong vùng bộ nhớ tương ứng trong *pred*. *col* sau đó được tăng lên 1 trong bước 911 và được so

sánh với *size* trong bước 912. Chừng nào mà *col* nhỏ hơn *size*, các bước 910 và 911 được lặp lại. Khi *col* bằng với *size*, quy trình chuyển sang bước 920.

Quay lại bước 903, *angle* nhỏ hơn không cần các điểm ảnh tham chiếu từ *refV* để lấy ra các điểm ảnh tham chiếu trong khối dự đoán. Bộ đếm *col* được thiết lập về không trong bước 913. Sau đó được xác nhận trong bước 914 xem "*int* + *col* + 1" có nhỏ hơn không hay không. "*int* + *col* + 1" lớn hơn hoặc bằng không thể hiện rằng chỉ các điểm ảnh tham chiếu được lưu trữ trong *refH* vẫn có thể được lựa chọn để lấy ra các điểm ảnh dự đoán trong khối dự đoán, và quy trình chuyển sang bước 915. Quy trình được thực hiện trong bước 915 giống với quy trình được thực hiện ở bước 907, và phần mô tả của nó sẽ không được nhắc lại ở đây. *col* sau đó được tăng lên 1 trong bước 916 và được so sánh với *size* trong bước 917. Chừng nào mà *col* nhỏ hơn *size*, các bước 914, 915 và 916 được lặp lại. Khi *col* bằng với *size*, quy trình chuyển sang bước 920.

Nếu "*int* + *col* + 1" được xác định nhỏ hơn không trong bước 914, các điểm ảnh được lưu trữ trong *refV* là cần thiết để lấy ra các điểm ảnh dự đoán trong khối dự đoán. Trong bước 918, vị trí giao giữa đường biên thẳng đứng và chiều dự đoán được xác định trước tiên. Trong bước 918, vị trí này được thể hiện bởi *pos2*. Lưu ý là trong bước 902, *pos*, nghĩa là, vị trí giao giữa đường biên ngang và chiều dự đoán, được xác định bởi "*angle* × (*row* + 1)". Biết rằng *angle* thể hiện tỷ lệ khác biệt theo chiều ngang và theo chiều thẳng đứng, "*angle*$^{-1}$ × (*col* + 1)", thay vì "*angle* × (*row* + 1)", được tính toán để xác định vị trí giao giữa đường biên thẳng đứng và chiều dự đoán. Như giả định ở trên, *angle* trong phạm vi của *-size* tới *size* (*-size* ≤ *angle* ≤ *size*). Do đó, tỷ lệ α giữa *angle* và *size* được xác định bằng:

$$\alpha = \frac{angle}{size} \ (-1 \leq \alpha \leq 1).$$

Sau đó, *angle*$^{-1}$ được xác định bằng:

$$angle^{-1} = \frac{size}{\alpha} \text{ hoặc } \frac{size^2}{angle}.$$

Như vậy, *pos2* được xác định trong bước 918 bằng bình phương của *size* nhân với *col* + 1 và sau đó được chia cho giá trị tuyệt đối của *angle* như sau:

$$pos2 = \frac{size^2 \times (col+1)}{|angle|}.$$

Giống như *pos*, *pos2* có số thập phân trong việc thể hiện điểm cố định được tạo thành từ phần nguyên và phần thập phân. Phần thập phân bao gồm số của các chữ số nhị phân được xác định bởi *log2_size*. Phần nguyên của *pos2* được lưu trữ trong tham số *int2*, và phần thập phân của *pos 2* được lưu trữ trong tham số *frac2*. Trong bước 919, hai điểm ảnh được xác định bằng "*int2* + *row* + 1" và "*int2* + *row* + 2" được lấy ra từ *refV*. Hai điểm ảnh tham chiếu này là bình quân gia quyền hoặc được nội suy với *frac2* để lấy ra điểm ảnh dự đoán *v*. Cụ thể, điểm ảnh tham chiếu từ *refV* (*int2* + *row* + 1) được nhân với "*size* - *frac2*" và được lưu trữ trong tham số *a*. Điểm ảnh tham chiếu từ *refV* (*int2* + *row* + 2) được nhân với "*frac2*" và được lưu trữ trong tham số *b*. Các tham số *a* và *b* sau đó được cộng vào và được chia cho *size* hoặc được dịch phải bởi *log2_size*. Điểm ảnh dự đoán được lấy ra *v* được lưu trữ trong vùng bộ nhớ tương ứng của *pred*. Các bước 914, 918, 919 và 916 được lặp lại cho đến khi *col* bằng với *size* trong bước 917.

Trong bước 920, *row* được tăng lên 1. Sau đó được xác định trong bước 921 xem *row* có nhỏ hơn *size* hay không. Chừng nào mà *row* nhỏ hơn *size*, các bước từ bước 902 được lặp lại để lấy ra điểm ảnh dự đoán trong khối dự đoán. Quy trình kết thúc khi *row* bằng với *size* trong bước 921.

Như đã nêu trên, việc tăng các sự tham gia của khối dự đoán góp phần nâng cao hiệu quả mã hóa, trái lại việc làm tăng sự tham gia của khối dự đoán dẫn tới việc tăng khối lượng công việc tính toán. Do đó, để tăng các sự tham gia của khối dự đoán mà nhờ đó nâng cao hiệu quả mã hóa, quy trình tạo nên sự tham gia của khối dự đoán cần được xem xét lại để đạt được hơn nữa hiệu quả của quy trình. Trong việc xem xét lại quy trình được thể hiện trên Fig.9, hai nút tính toán có thể được xác định. Nút tính toán thứ nhất là phép so sánh và chia nhánh của bước 914, mà được lặp lại ở trong vòng lặp. Nút tính toán thứ hai là phép chia của bước 918, mà cũng được lặp lại ở trong vòng lặp.

Hiện nay, một lệnh nhiều dữ liệu (SIMD) sẵn sàng để tính toán có hiệu

quả. SIMD cho phép các máy tính với nhiều phần tử xử lý thực hiện cùng phép tính trên nhiều dữ liệu một cách đồng thời. Tuy nhiên, các kết cấu dạng SIMD không hỗ trợ thực hiện phép chia và sự tính toán/sự chia nhánh trong vòng lặp lại và, do đó, không thể được sử dụng để thực hiện quy trình được thể hiện trên Fig.9 bởi bì nó bao gồm các bước 914 và 918 trong vòng lặp lại, mặc dù các vòng lặp lại bắt đầu từ các bước 907 và 910 đủ điều kiện để được thực hiện với SIMD. Do đó đối tượng của sáng chế chuyển sang các nút tính toán từ quy trình được thể hiện trên Fig.9 và cung cấp sự dự đoán trong ít phức tạp, mà cho phép các cấu trúc dạng SIMD thực hiện việc xử lý song song theo tất cả các chiều dự đoán được thể hiện trên Fig.8.

Fig.10 là lưu đồ thể hiện quy trình dự đoán trong ít phức tạp theo một phương án của sáng chế, mà được thiết kế để thay thế quy trình của Fig.9 trong việc thực hiện của quy trình trong bước 504 của Fig.5. Trên Fig.10, các bước tiến trình giống nhau như được thực hiện trên Fig.9 được xác định bằng các số biểu thị bước giống nhau như được sử dụng trên Fig.9, như các bước 900, 901, 902, 904, 905, 906, 907, 908, 909, 910, 911, 912, 920 và 921. Phần mô tả về các bước thông dụng này không được nhắc lại ở đây. Các bước 1000 và 1001 là các bước riêng đối với quy trình của Fig.10. Rõ ràng là từ sự so sánh với quy trình được thể hiện trên Fig.9, quy trình của Fig.10 loại trừ bước so sánh của bước 903 và tất cả các bước được phân nhánh sang trái từ bước 903, mà được thực hiện khi *angle* nhỏ hơn không, do đó loại trừ các nút tính toán của các bước 914 và 918.

Trong các bước được bổ sung vào 1000 và 1001, *angle* được xác định có thể lớn hơn hoặc bằng -1. Khi *angle* lớn hơn hoặc bằng -1, các điểm ảnh tham chiếu nằm ở đường biên ngang đủ để tạo ra điểm ảnh dự đoán trong khối dự đoán, và các điểm ảnh tham chiếu ở đường biên thẳng đứng là không cần thiết. Mặt khác, *angle* nhỏ hơn -1, các điểm ảnh tham chiếu ở đường biên thẳng đứng cần thiết để tạo ra điểm ảnh dự đoán trong khối dự đoán. Trong bước 1001, các điểm ảnh tham chiếu được lưu trữ trong *refH* được mở rộng theo chiều âm, sử dụng ít nhất một vài điểm ảnh được lưu trữ trong *refV*. Các Fig.11A và Fig.11B

là các sự thể hiện dưới dạng giản đồ thể hiện sự mở rộng của *refH* được thực hiện trong bước 1001. Trong Fig.11A, các điểm ảnh 1102 được lưu trữ trong *refH* là từ đường biên ngang nằm trên khối đích 1101. Các điểm ảnh 1103 được lưu trữ trong *refV* là từ đường biên thẳng đứng nằm bên trái của khối đích 1101. Như được thể hiện trên Fig.11B, sau bước 1001 của Fig.10, một vài điểm ảnh trong *refV* được sao chép vào *refH*, và *refH* có phần được mở rộng 1104 mở rộng theo chiều âm.

Fig.12 là lưu đồ thể hiện chi tiết quy trình được thực hiện trong bước 1001. Trong bước 1201, bộ đếm *col* được thiết lập về -1. *col* được sử dụng để xác định địa chỉ của phần được mở rộng của *refH*. Trong bước 1202, điểm ảnh tham chiếu trong *refV* được sao chép vào phần được mở rộng của *refH* được xác định bằng:

$$\frac{size \times col}{angle}.$$

Phép chia trong phương trình trên là phép chia số nguyên, và phương trình này cho kết quả là một số nguyên. Phương trình này có chức năng tương tự đối với quy trình của bước 918 được thể hiện trên Fig.9. Trong bước 918, giá trị nguyên của *pos2* được tính bằng:

$$\frac{(size^2 \times (col+1))}{angle} >> \log 2\_size.$$

Lưu ý là sự dịch phải bằng *log2_size* tương đương với phép chia cho *size*.

Trong bước 1203, *col* bị giảm đi 1. Sau đó nó được xác định trong bước 1204 xem *col* có bằng với *angle* hay không. Nếu *col* không bằng với *angle*, quy trình quay lại bước 1202. Các bước 1202 và 1203 được lặp lại cho đến khi *col* bằng với *angle*. Do đó, các điểm ảnh tham chiếu được đọc từ *refV* theo thứ tự tăng dần, hoặc từ trên xuống dưới của đường biên thẳng đứng, và được sao chép vào *refH* cũng theo thứ tự tăng dần, hoặc từ trái sang phải của đường biên ngang. Tuy nhiên, không phải tất cả các điểm ảnh tham chiếu trong *refV* đều được sao chép vào *refH*. Chỉ có các điểm ảnh tham chiếu nằm trong khoảng từ đầu đến điểm giao của chiều dự đoán mới được sao chép từ *refV* vào *refH*.

Quay lại Fig.10, các bước tiến trình bắt đầu từ bước 902 được sao chép từ

Fig.9, và bao gồm các bước để tạo ra các điểm ảnh dự đoán được chia nhánh sang bên phải từ bước so sánh của bước 903 trên Fig.9. Tuy nhiên lưu ý là các bước trên Fig.10 để tạo ra các điểm ảnh dự đoán sử dụng *refH* được mở rộng (tổng của các phần 1102 + 1104 trên Fig.11B), trái lại các bước tương ứng trên Fig.9 sử dụng *refH* nguyên bản (phần 1102 trên Fig.10A). Do *refH* được mở rộng theo chiều âm, hoạt động dự đoán trong tách biệt được thiết kế cụ thể để sử dụng các điểm ảnh tham chiếu được lưu trữ trong *refV*, như được chia nhánh sang trái từ bước 903 trên Fig.9, là không cần thiết bất chấp dấu hiệu của *angle*.

Fig.13 là lưu đồ thể hiện phương án khác của quy trình để mở rộng *refH*, sử dụng các điểm ảnh tham chiếu trong *refV*. Quy trình được thể hiện trên các Fig.11 và Fig.12 loại trừ các nút thắt của các bước 914 và 918 được thể hiện trên Fig.9 và, do đó, được kỳ vọng để nâng cao hiệu quả của quy trình dự đoán trong. Quy trình này được thể hiện trên Fig.13 loại trừ phép chia được thực hiện trong bước 1202 của Fig.12 từ vòng lặp lại để sao chép các điểm ảnh tham chiếu từ *refV* vào *refH*. Bằng cách loại trừ phép chia từ vòng lặp lại, quy trình được thể hiện trên Fig.13 được kỳ vọng nâng cao hơn nữa hiệu quả của quy trình dự đoán trong.

Quy trình được thể hiện trên Fig.13 thay thế bước 1202 của Fig.12 bằng các bước 1301 và 1302. Bước 1302 ở trong vòng lặp để sao chép các điểm ảnh tham chiếu từ *refV* vào *refH,* trái lại bước 1301 ở ngoài vòng lặp. Bước 1301 giới thiệu tham số mới được gọi là "*InvAngle*". *InvAngle* được xác định bằng:

$$256 \times \frac{size}{angle}.$$

Việc nhân với 256 tương đương với sự dịch trái bởi 8 và chắc chắn rằng mọi bit kết quả từ phép chia của "*size/angle*" giải thích cho việc tính toán xác định điểm ảnh tham chiếu trong *refV*. Trong bước 1302, địa chỉ của điểm ảnh tham chiếu trong *refV* được sao chép vào phần được mở rộng của *refH* được xác định bằng:

$$col \times InvAngle >> 8 \ .$$

Kết quả của "*col* × *InvAngle*" được dịch phải bởi 8 đối với phép dịch trái

tái cấu trúc lại được thực hiện trong bước 1301. Lưu ý là phép dịch phải trong bước 1302 có chức năng làm tròn xuống kết quả của "*col* × *InvAngle*". Để làm tròn về phía số nguyên gần nhất, độ lệch làm tròn của 128 có thể được bổ sung vào kết quả của "*col* × *InvAngle*" trước khi phép dịch phải được thực hiện. Cần lưu ý là số "256" là ví dụ, và bước 1301 có thể sử dụng số lệch khác, tốt hơn là lũy thừa nguyên của 2, chừng nào mà số đủ lớn để duy trì tất cả các bit kết quả từ phép tính của "*size/angle*". Ví dụ, số có thể là 64 trong bước 1301, thay vì 256, và số dịch phải là 6 trong bước 1302, thay vì 8. Nếu 64 được chấp nhận, độ lệch làm tròn nên là 32.

Sự tính toán được thực hiện trong bước 1301 có thể được thay thế bằng hoạt động tìm kiếm để làm giảm bớt hơn nữa khối lượng công việc tính toán. Nói cách khác, bảng tìm kiếm được sửa mà lưu các giá trị của *InvAngle* liên quan tới các giá trị của *angle*. Bảng 1 bố trí bên dưới là bảng mẫu để tìm kiếm trong bước 1301:

Bảng 1

angle	4	5	6	7	8
InvAngle	512	410	341	293	256

Giả định rằng trong bảng trên, *size* là 8, và *angle* nhận các giá trị nguyên từ 4 đến 8. Tuy nhiên cần lưu ý rằng *size* không bị giới hạn ở 8 mà có thể nhận giá trị khác, như 4 và 16. Do đó *angle* có thể là số thập phân trong việc thể hiện điểm cố định được xác định bên trên.

Khi điểm ảnh tham chiếu được sao chép từ *refV* đến *refH* trong bước 1202 của Fig.12 hoặc bước 1302 của Fig.13, điểm ảnh tham chiếu có thể đi qua bộ lọc thông thấp để làm giảm sai số có thể trong khối dự đoán. Điểm mạnh của bộ lọc thông thấp là có thể thay đổi theo giá trị của *angle*. Chẳng hạn, khi *angle* bằng với -*size*, bộ lọc thông thấp yếu có thể được áp dụng, và khi *angle* bằng -2, bộ lọc thông thấp mạnh có thể được áp dụng.

Như được giải thích ở trên, không phải tất cả các điểm ảnh tham chiếu trong *refV* đều được sao chép vào *refH*. Do không phải tất cả các điểm ảnh tham chiếu trong *refV* đều được sao chép, một vài thông tin sẽ bị mất đi khi các điểm

ảnh được sao chép. Để giảm thiểu việc mất thông tin, độ phân giải của các điểm ảnh tham chiếu trong *refH* và *refV* có thể được gấp đôi sao cho *refH* và *refV* chứa không chỉ các điểm ảnh từ các khối được mã hóa và được tái cấu trúc trước đó mà còn chứa một điểm ảnh trong số hai điểm ảnh được tái cấu trúc liền kề, mà được tạo ra bằng cách nội suy hai điểm ảnh liền kề. Hai điểm ảnh liền kề có thể đơn giản được tính trung bình để tạo ra điểm ảnh nội suy. Quy trình nội suy có thể được thực hiện khi các điểm ảnh được đọc trong bước 900 của Fig.9. Khi độ phân giải của các điểm ảnh được gấp đôi trong *refH* và *refV*, việc xác định các địa chỉ của các điểm ảnh tham chiếu được lưu trữ trong *refH* và *refV*, như được thực hiện trong các bước 907, 910, 915 và 919 trên Fig.9, và bước 1001 trên Fig.10, cần được thu nhỏ. Ví dụ, "*int* + *col* + 1" được thực hiện trong các bước 907, 910 và 915 cần được thay đổi thành "*int* + 2×*col* + 2". "*int* + *col* + 2" được thực hiện trong các bước 907, 910, 915 cần được đổi thành "*int* + 2×*col* + 3". "*int2* + row + 1" và "*int2* + row + 2" được thực hiện trong bước 919 cần được đổi lần lượt thành "*int2* + 2×row + 2" và "*int2* + 2×row + 3".

Theo phương án khác, quy trình của bước 1202 trên Fig.12 có thể đơn giản được đổi thành "*refH* [*col*]←*refV* [-*col*]" để làm đơn giản hơn nữa quy trình sao chép. Mặc dù làm giảm độ chính xác của các dự đoán, nhưng phương án này lại cung cấp độ phức tạp thấp nhất cho hoạt động dự đoán trong.

Fig.11B thể hiện phần được mở rộng 1104 được bổ sung vào *refH*. Phần được mở rộng 1104 không cần được tạo thành với các điểm ảnh tham chiếu từ *refV*. Phần được mở rộng 1104 có thể được tạo thành với các điểm ảnh từ vùng của khối được tái cấu trúc trước đó, mà tương ứng về mặt không gian với vị trí của phần được mở rộng 1104. Trên Fig.11B, do được mở rộng theo chiều âm, *refH* được mở rộng (các phần 1102 và 1104) nằm trong phạm vi từ -*size* + 1 đến 2×*size*. Phạm vi của *refH* được mở rộng có thể được mở rộng tới phạm vi từ 0 đến 3×*size* - 1 bằng cách cộng vào phần bù thích hợp khi biểu thị các điểm ảnh tham chiếu trong *refH* được mở rộng. Điều này cũng đúng cho phạm vi mở rộng của *refV*.

Theo phương án khác, phạm vi giới hạn của *angle* có thể được chọn tuỳ ý.

Theo các phương án ở trên, giả định rằng *angle* nhận giá trị trong phạm vi từ -*size* đến *size* (-*size* ≤ *angle* ≤ *size*). Nói cách khác, theo các phương án ở trên, phạm vi giới hạn của *angle* được xác định với kích cỡ của khối đích. Lưu ý là phạm vi giới hạn của *angle* có thể được xác định độc lập với kích cỡ của khối đích, mặc dù tốt hơn là phạm vi giới hạn được xác định với lũy thừa nguyên của 2, vì vậy *log2_rangelimit* là số nguyên dương, và phương trình "*rangelimit* = 1 << *log2_rangelimit*" cũng đúng. Bằng cách chọn số lớn phù hợp cho *rangelimit*, số chiều dự đoán lớn có thể được tạo ra và được thể hiện bằng các giá trị của *angle* ở các khoảng góc đủ rộng.

Nếu phạm vi giới hạn của *angle* được xác định độc lập với kích cỡ của khối đích, *size* xuất hiện trên các Fig.9 và Fig.10 cần được thay thế bằng *rangelimit*, và *log2_size* cần được thay thế bằng *log2_rangelimit*, ngoại trừ các bước 909, 912, 917 và 921. Sự so sánh của "*angle* ≥ -1" được thực hiện trong bước 1000 của Fig.10 cũng cần được thay thế bằng "*angle*×*size*/*rangelimit* ≥ -1" hoặc "*angle*×*size* ≥ - *rangelimit*". Hơn nữa, *size* xuất hiện trong các bước 1202 và 1301 trên các Fig.12 và Fig.13 cần được thay thế bằng *rangelimit*, và sự so sánh của "*col* = *angle*?" được thực hiện trong bước 1204 cần được thay thế bằng "*col* = *angle*×*size*/*rangelimit*?".

Nếu *rangelimit* được đưa vào như phạm vi giới hạn của *angle*, bảng 1 (được cung cấp ở trên) có thể được thay đổi như sau:

Bảng 2

*angle**	13	17	21	26	32
InvAngle	630	482	390	315	256

Trong bảng 2, *rangelimit* được thiết lập ở 32. *angle** bằng xấp xỉ số nguyên của "*rangelimit* × tan (π/4 × *angle*/8)", trong đó *angle* = 4, 5, 6, 7 và 8. *InvAngle* bằng 256 × *rangelimit*/*angle**. Các giá trị trong bảng 2 là tất cả các số nguyên mà được lấy ra bằng cách làm tròn lên. Thay vì được làm tròn lên, các số có thể được làm tròn xuống. Trong bảng 3 được cung cấp bên dưới, *InvAngle* bằng 32 × *rangelimit*/*angle**. Do "32" được sử dụng, thay vì "256", nên tính chính xác của dự đoán chắc chắn sẽ thấp hơn so với của bảng 2.

Bảng 3

*angle**	13	17	21	26	32
InvAngle	78	60	48	39	32

Fig.14 là lưu đồ thể hiện phương án khác mà làm đơn giản hơn nữa quy trình được thể hiện trên Fig.10. Quy trình được thể hiện trên Fig.10 của việc sao chép các điểm ảnh tham chiếu từ *refV* vào *refH* được thực hiện trước khi quy trình nhập vào vòng lặp dự đoán chính, trái lại quy trình sao chép được thể hiện trên Fig.14 được thực hiện trong vòng lặp dự đoán chính. Do đó, quy trình được thể hiện trên Fig.14 loại trừ *InvAngle* có thể biến đổi. Các bước 900, 902 và 921 được thể hiện trên Fig.14 là từ các bước tương ứng trên Fig.10.

Trong bước 1401, bộ đếm *lastInt* được thiết lập ban đầu ở -1. *lastInt* thể hiện chỉ số của điểm ảnh cuối cùng mà được bổ sung vào *refH*. Trong bước 902, *pos* được tính bằng *angle* × (*row* + 1). Như đã giải thích ở trên, *pos* xác định vị trí giao giữa các đường biên và chiều dự đoán được thể hiện bằng *angle*. Trong bối cảnh của Fig.9, bước 902 cho kết quả là *pos*, mà xác định vị trí giao giữa đường biên ngang và chiều dự đoán được thể hiện bằng *angle*. Hơn nữa trong bước 902, phần nguyên trong *pos* được lưu trữ trong *int*, và phần thập phân trong *pos* được lưu trữ trong tham số "*frac*". Trong bước 1402, *int* được xác nhận xem có nhỏ hơn *lastInt* hay không. Nếu *int* nhỏ hơn *lastInt*, điểm ảnh tham chiếu trong *refV* được xác định bằng *row* được sao chép vào *refH* ở địa chỉ được xác định bởi "*int* + 1". Bước 1404 bao gồm các bước 904, 905, 906, 907, 908, 909, 910, 911 và 912 được thể hiện trên các Fig.9 và Fig.10, mà phần mô tả của nó không được nhắc lại ở đây. Trong bước 1405, *int* được sao chép vào *lastInt*. Hoạt động sao chép *int* vào *lastInt* có thể được thực hiện trong bước 1403, thay cho bước 1405.

Hoạt động sao chép trong bước 1403 dẫn đến việc sao chép cùng điểm ảnh như được sao chép trong các bước 1202 và 1302, trong đó việc làm tròn xuống được sử dụng ở các bước này. Bước 1403 có thể được biến đổi để làm tròn tới số nguyên gần nhất nhờ sử dụng điều kiện "*row* + 1", thay vì "row", trong bước 1403 khi vị trí phân đoạn của *frac* được tính trong bước 902 lớn hơn

offset, mà được xác định bằng *rangelimit* + (*angle* >> 1). Lưu ý là *angle* là -ve, và *frac* là +ve. Việc sử dụng "*row* + 1" dẫn đến việc làm tròn lên. Để thực hiện việc tăng có điều kiện của *row* lên 1, quy trình được thực hiện trong bước 1403 được thay đổi thành *refH*[*int* + 1] ← *refV*[*row* - ((*offset* - *frac*) >> 31)], giả định rằng trong 32 bit số học, sự dịch trái của "*offset* - *frac*" dẫn đến kết quả là -1 khi *frac* lớn hơn *offset* và nếu không kết quả là 0. Do đó, bộ nhận dạng địa chỉ "*row* - ((*offset* - *frac*) >> 31)" trở thành "*row* + 1" khi *frac* lớn hơn *offset* và nếu không trở thành "*row*". Nếu *offset* được thiết lập về *rangelimit*, "*offset* - *frac*" sẽ luôn là dương và do đó sẽ không phải làm tròn.

Mã nguồn được phát triển trong ngôn ngữ lập trình C++, mà xác định quy trình được thể hiện trên Fig.14, được liệt kê bên dưới. Mã nguồn được biến đổi từ hàm số TComPrediction::xPredIntraAng được tìm thấy trong tệp tin TComPrediction.cpp mà là bộ phận của phần mềm TMuC 0.7 được phát triển bởi JCT-VC, mà có sẵn ở http://hevc.kw.bbc.co.uk/svn/jctvc.a124/tags/0.7.

```
// Hàm số để suy ra các dự đoán trong góc được đơn giản hóa
Void TComPrediction::xPredIntraAng (Int* pSrc, Int iSrcStride, Pel*& rpDst,
Int iDstStride, UInt iWidth, UInt iHeight, UInt uiDirMode, Bool bAbove, Bool
bLeft){
        Int k,l;
        Int deltaInt, deltaFract, refMainIndex;
        Int intraPredAngle      = 0;
        Int absAng      = 0;
        Int signAng             = 0;
        Int blkSize     = iWidth;
        Bool modeDC             = false;
        Bool modeVer            = false;
        Bool modeHor            = false;
        Pel* pDst       = rpDst;
// Ánh xạ chỉ số chế độ tới chiều dự đoán và góc chính
```

```
if (uiDirMode == 0)
  modeDC = true;
else if (uiDirMode < 18)
  modeVer = true;
else
  modeHor = true;
intraPredAngle = modeVer ? uiDirMode - 9 : modeHor ? uiDirMode –
25 : 0;
absAng              = abs(intrapredAngle);
signAng      = intraPredAngle < 0 ? -1 : 1;
// Thiết lập những thay đổi bit và thu nhỏ tham số góc đến size2
Int iAngTable[9] = { 0, 2, 5, 9, 13, 17, 21, 26, 32};
absAng = iAngTable[absAng];
intraPredAngle = signAng * absAng;
// Thực hiện dự đoán DC
if (modeDC){
  Pel dcval = predIntraGetPredValDC(pSrc, iSrcStride, iWidth, iHeight,
bAbove, bLeft);
  for (k=0;k<blkSize;k++){
    for (l=0;l<blkSize;1++){
pDst(k*iDstStride+1] = dcval;
    }
  }
}
// Thực hiện các chiều dự đoán góc
else {
  Pel tmp;
  Int *pSrcTL = pSrc - iSrcStride - 1;
  Int iStepMain = (modeVer) ? 1 : iSrcStride;
  if (intraPredAngle == 0){
    for (k=0;k<blkSize;k++){
```

```
for (l=0;1<blkSize;1++){
 pDst [k*iDstStride+1] = pSrcTL[(1+1) * iStepMain];
    }
   }
  }
  else {
   Int iStepSide = (modeVer) ? iSrcStride 1;
   int lastDeltaInt = -1;
   Int iOffset = 32 + (intraPredAngle >> 1);// cho phép làm tròn đến sự tham chiếu phía gần nhất
 // Int iOffset = 32;                          // không làm tròn.
   Pel ref [2*MAX_CU_SIZE];
   Pel* refMain = ref + ((intraPredAngle < 0) ? blkSize : 0);
   if (intraPredAngle > 0){
    for (k = 0; k  <  2*blkSize; k++)
  refMain[k] = pSrcTL[(k+1)  *  iStepMain];
   }
   else {
    for (k = -1; k < blkSize; k++)    // phần còn lại được sao chép sau trong bước 1403, như khi được yêu cầu
   refMain[k] = pSrcTL[(k+1) * iStepMain];
   }
   for (k = 0; k < blkSize; k++){
   Int deltaPos = (k+1) * intraPredAngle;
   deltaInt   = deltaPos >> 5;
   deltaFract = deltaPos & (32 - 1);
   if (deltaInt < lastDeltaInt) { // bước 1402
    lastDeltaInt = deltaInt;
    refMain[deltaInt]          =          pSrcTL[(k-((iOffset-deltaFract)>>31))*iStepSide];  // bước 1403
   }
```

```
// bước 1404
            if (deltaFract){
// Thực hiện bộ lọc tuyến tính
              for (1=0;1<blkSize;1++){
                refMainIndex = 1+deltaInt;
                pDst[k*iDstStride+1]     =     (Pel)     (     ((32-deltaFract)     *
refMain[refMainIndex] + deltaFract * refMain[refMainlndex+1] + 16) >> 5 );
              }
            }
            else {
// chỉ sao chép các mẫu số nguyên
              for (1=0;1<<blkSize;l++) {
                pDst[k*iDstStride+1] = refMain[1+deltaInt];
              }
            }
          }
        }
// lật khối nếu đây là phương pháp theo chiều ngang
        if (modeHor){
          for (k=0;k<blkSize-1;k++){
            for (1=k+1;1<blkSize;1++){
              tmp = pDst[k*iDstStride+1];
            pDst(k*iDstStride+1] = pDst(1*iDstStride+k];
            pDst[1*iDstStride+k] = tmp;
            }
          }
        }
      }
```

Trong khi các sự thay đổi và các cải biến của sáng chế sẽ trở nên rõ ràng với người có trình độ trung bình trong lĩnh vực sau khi đọc phần mô tả nêu trên,

cần được hiểu rằng bất kỳ phương án cụ thể nào được thể hiện và được mô tả bằng cách minh họa đều không có mục đích giới hạn. Do đó, các sự tham khảo các chi tiết của các phương án khác nhau không có mục đích giới hạn phạm vi của các điểm yêu cầu bảo hộ, mà chúng chỉ mô tả các dấu hiệu được coi là cần thiết đối với sáng chế.

YÊU CẦU BẢO HỘ

1. Phương pháp mã hóa viđeo được thực hiện bởi bộ xử lý của bộ mã hóa viđeo, phương pháp này bao gồm:

bước lấy ra ít nhất một số điểm ảnh từ dãy các điểm ảnh ở đường biên thẳng đứng;

bước bổ sung các điểm ảnh được lấy ra vào dãy các điểm ảnh ở đường biên ngang, để mở rộng dãy các điểm ảnh ở đường biên ngang; và

bước thực hiện dự đoán trong dựa vào dãy được mở rộng của các điểm ảnh ở đường biên ngang,

trong đó bước lấy ra ít nhất một số điểm ảnh từ dãy các điểm ảnh ở đường biên thẳng đứng bao gồm các bước:

thu được *InvAngle* từ bảng tìm kiếm mà liệt kê các giá trị của *InvAngle* liên quan đến các giá trị của *angle** mà thể hiện chiều dự đoán; và

nhận dạng ít nhất một số điểm ảnh trong số các điểm ảnh ở đường biên thẳng đứng, bằng cách sử dụng bộ nhận dạng điểm ảnh theo chiều thẳng đứng mà được thể hiện bởi hàm sử dụng [*col* × *InvAngle*], trong đó *col* là bộ đếm mà giảm đi 1 từ -1 đến *(size* × *angle*/rangelimit)*, trong đó *size* là kích cỡ của khối đích và *rangelimit* xác định khoảng của *angle**, và

trong đó bước bổ sung các điểm ảnh được lấy ra vào dãy các điểm ảnh ở đường biên ngang bao gồm bước: bổ sung điểm ảnh được nhận dạng bởi bộ nhận dạng điểm ảnh theo chiều thẳng đứng vào các điểm ảnh ở đường biên ngang tại vị trí được nhận dạng bởi bộ nhận dạng điểm ảnh theo chiều ngang [*col*].

2. Phương pháp mã hóa viđeo theo điểm 1, trong đó *rangelimit* là lũy thừa nguyên của 2.

3. Phương pháp mã hóa viđeo theo điểm 1, trong đó *rangelimit* bằng 32 và *(size* × *angle*/rangelimit)* được thực hiện là *size* × *angle** >> *5*.

4. Phương pháp mã hóa viđeo theo điểm bất kỳ trong số các điểm từ 1 đến 3, trong đó độ lớn của *InvAngle* được thu từ bảng tìm kiếm sau đây:

angle*	13	17	21	26	32
InvAngle	630	482	390	315	256

5. Phương pháp giải mã viđeo được thực hiện bởi bộ xử lý của bộ giải mã viđeo, phương pháp này bao gồm:

bước lấy ra ít nhất một số điểm ảnh từ dãy các điểm ảnh ở đường biên thẳng đứng;

bước bổ sung các điểm ảnh được lấy ra vào dãy các điểm ảnh ở đường biên ngang, để mở rộng dãy các điểm ảnh ở đường biên ngang; và

bước thực hiện dự đoán trong dựa vào dãy được mở rộng của các điểm ảnh ở đường biên ngang,

trong đó bước lấy ra ít nhất một số điểm ảnh từ dãy các điểm ảnh ở đường biên thẳng đứng bao gồm các bước:

thu được *InvAngle* từ bảng tìm kiếm mà liệt kê các giá trị của *InvAngle* liên quan đến các giá trị của *angle** mà thể hiện chiều dự đoán; và

nhận dạng ít nhất một số điểm ảnh trong số các điểm ảnh ở đường biên thẳng đứng, bằng cách sử dụng bộ nhận dạng điểm ảnh theo chiều thẳng đứng mà được thể hiện bởi hàm sử dụng [*col* × *InvAngle*], trong đó *col* là bộ đếm mà giảm đi 1 từ -1 đến *(size* × *angle*/rangelimit)*, trong đó *size* là kích cỡ của khối đích và *rangelimit* xác định khoảng của *angle**, và

trong đó bước bổ sung các điểm ảnh được lấy ra vào dãy các điểm ảnh ở đường biên ngang bao gồm bước: bổ sung điểm ảnh được nhận dạng bởi bộ nhận dạng điểm ảnh theo chiều thẳng đứng vào các điểm ảnh ở đường biên ngang ở vị trí được nhận dạng bởi bộ nhận dạng điểm ảnh theo chiều ngang [*col*].

6. Phương pháp giải mã viđeo theo điểm 5, trong đó *rangelimit* là lũy thừa nguyên của 2.

7. Phương pháp giải mã viđeo theo điểm 5, trong đó *rangelimit* bằng 32 và *(size* × *angle*/rangelimit)* được thực hiện là *size* × *angle** >> *5*.

8. Phương pháp giải mã viđeo theo điểm bất kỳ trong số các điểm từ 5 đến 7, trong đó độ lớn của *InvAngle* được thu từ bảng tìm kiếm sau đây:

angle*	13	17	21	26	32
InvAngle	630	482	390	315	256

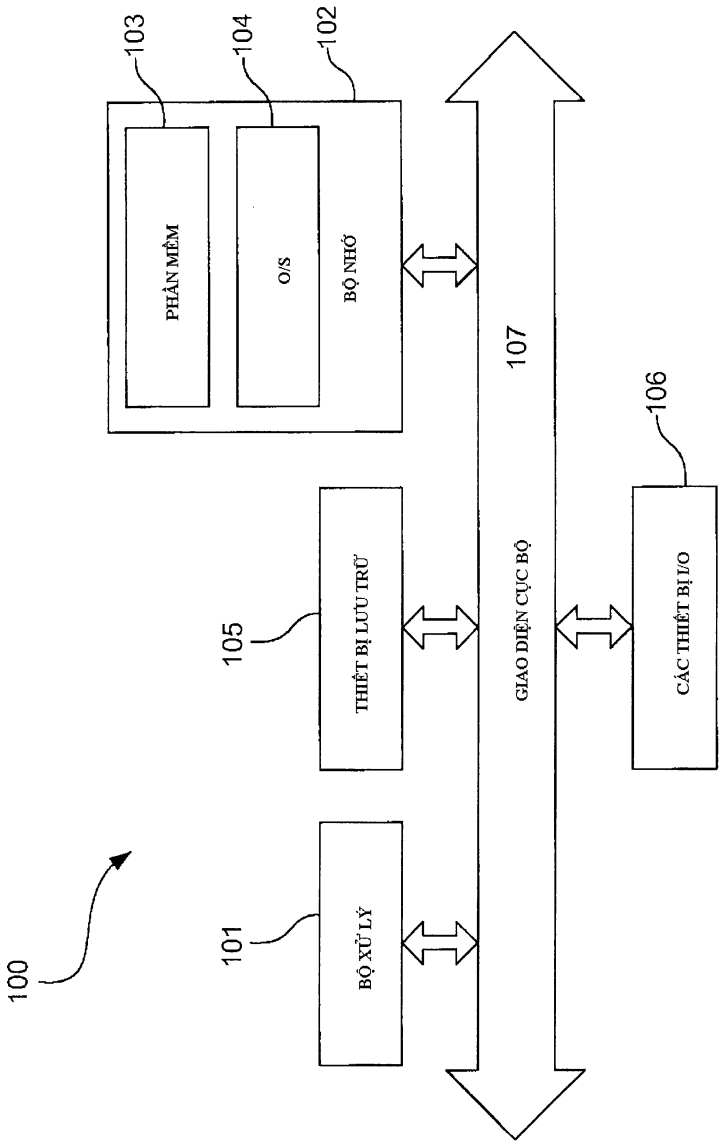

100
101
BỘ XỬ LÝ
105
THIẾT BỊ LƯU TRỮ
GIAO DIỆN CỤC BỘ
107
106
CÁC THIẾT BỊ I/O
102
BỘ NHỚ
O/S
104
PHẦN MỀM
103

Fig. 1

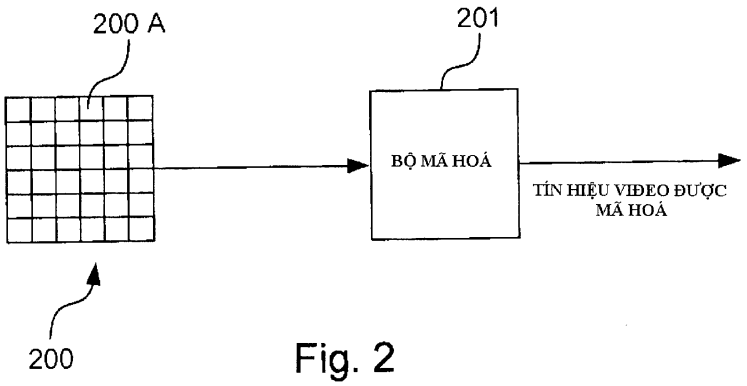

200 A
201
BỘ MÃ HOÁ
TÍN HIỆU VIĐEO ĐƯỢC MÃ HOÁ
200

Fig. 2

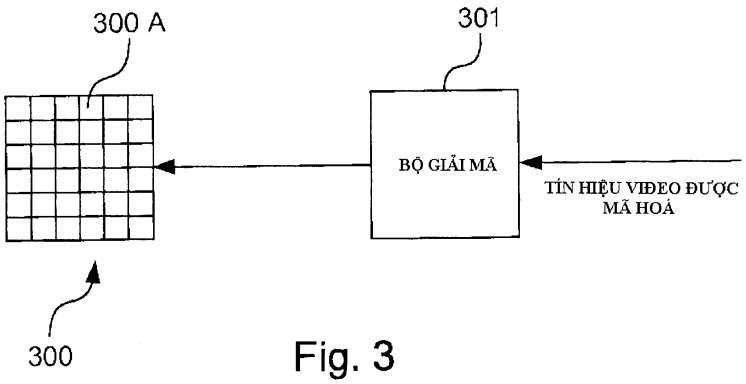

300 A
301
BỘ GIẢI MÃ
TÍN HIỆU VIĐEO ĐƯỢC MÃ HOÁ
300

Fig. 3

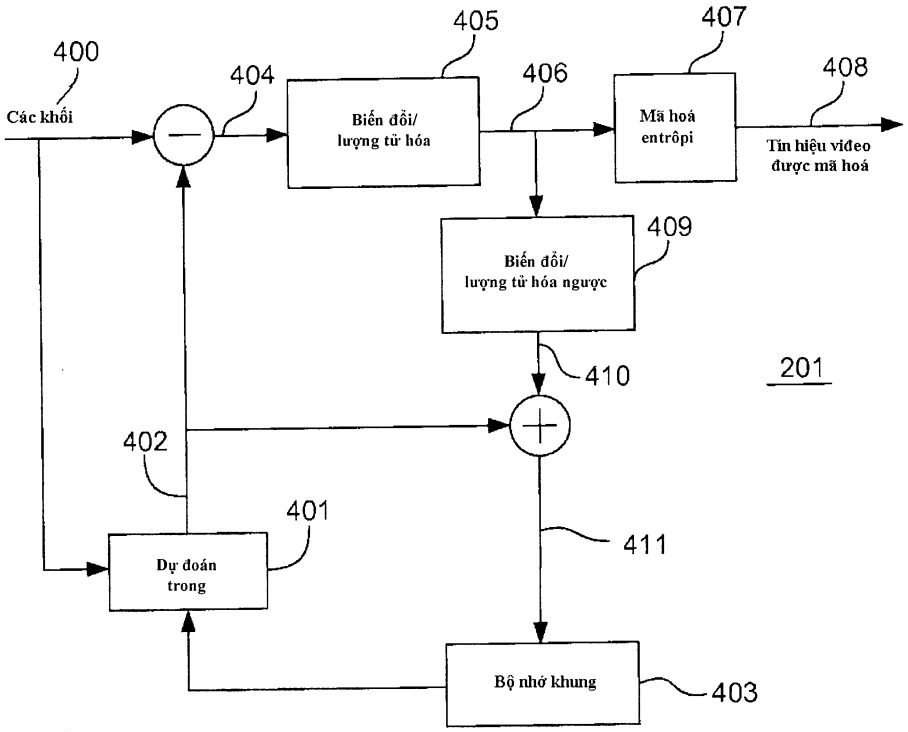

400
Các khối
404
405
Biến đổi/
lượng tử hóa
406
407
Mã hoá
entrôpi
408
Tín hiệu video
được mã hoá
409
Biến đổi/
lượng tử hóa ngược
410
201
402
401
Dự đoán
trong
411
Bộ nhớ khung
403

Fig. 4

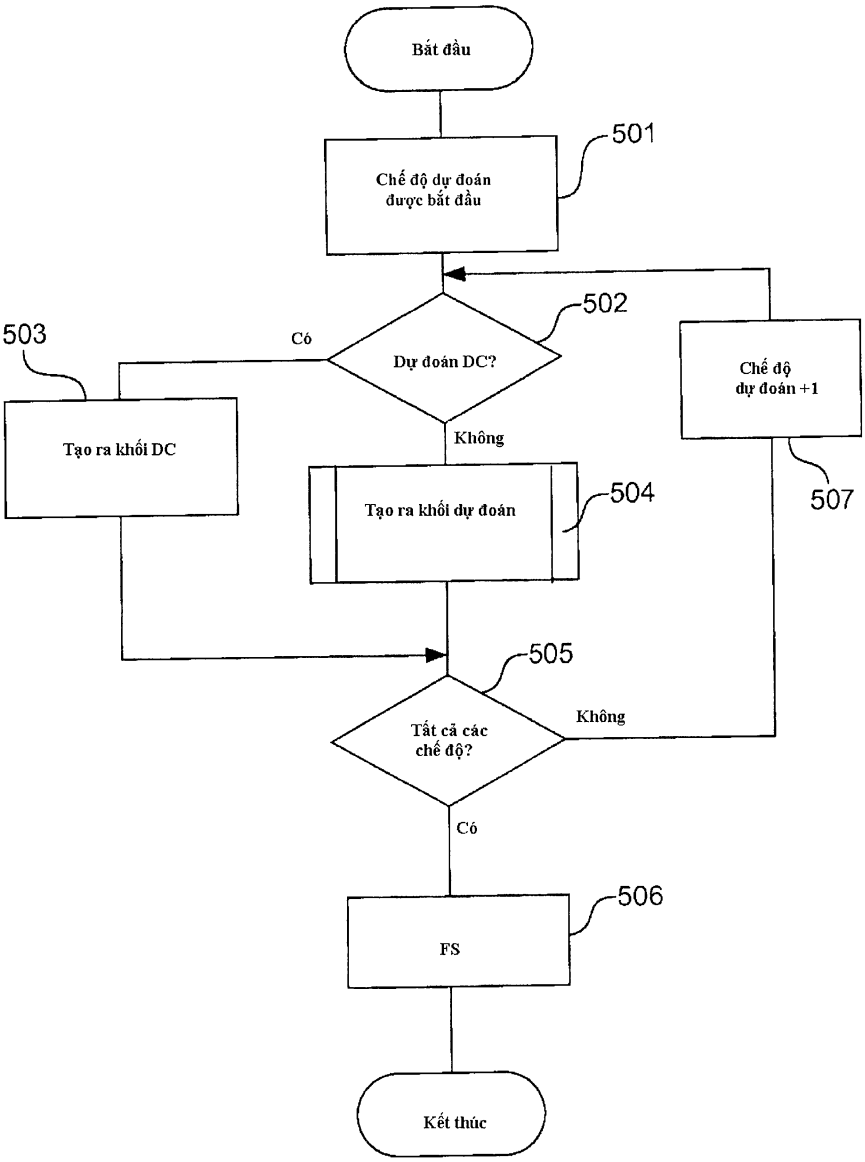

Bắt đầu
501
Chế độ dự đoán được bắt đầu
502
Dự đoán DC?
Có
503
Tạo ra khối DC
Không
504
Tạo ra khối dự đoán
Chế độ dự đoán +1
507
505
Tất cả các chế độ?
Không
Có
506
FS
Kết thúc

Fig. 5

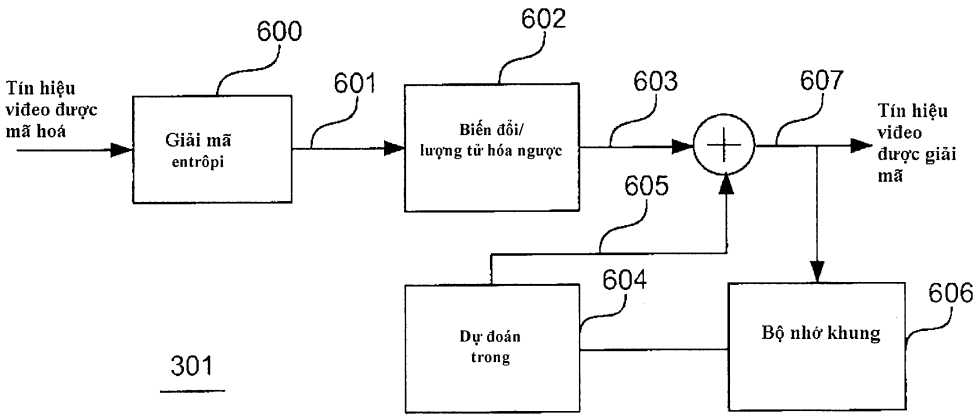

600
602
Tín hiệu
viđeo được
mã hoá
601
603
607
Giải mã
entrôpi
Biến đổi/
lượng tử hóa ngược
Tín hiệu
viđeo
được giải
mã
605
604
606
Dự đoán
trong
Bộ nhớ khung
301

Fig. 6

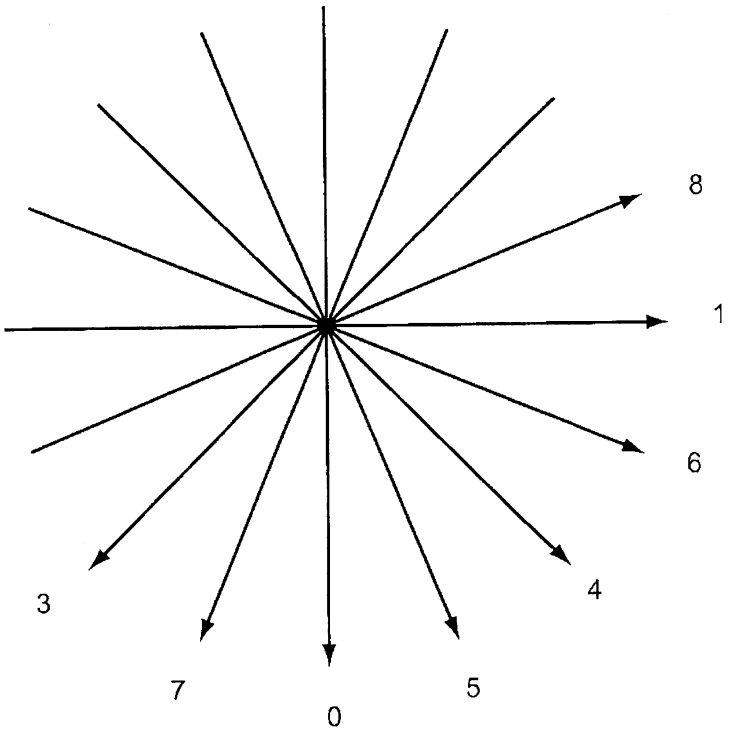


Fig. 7

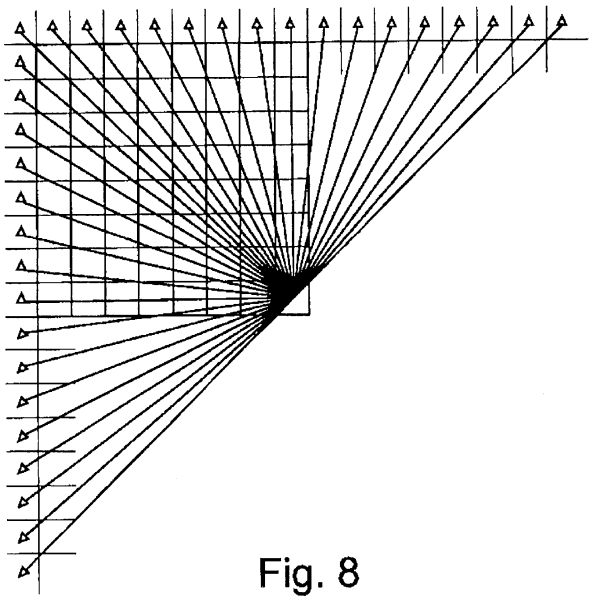


Fig. 8

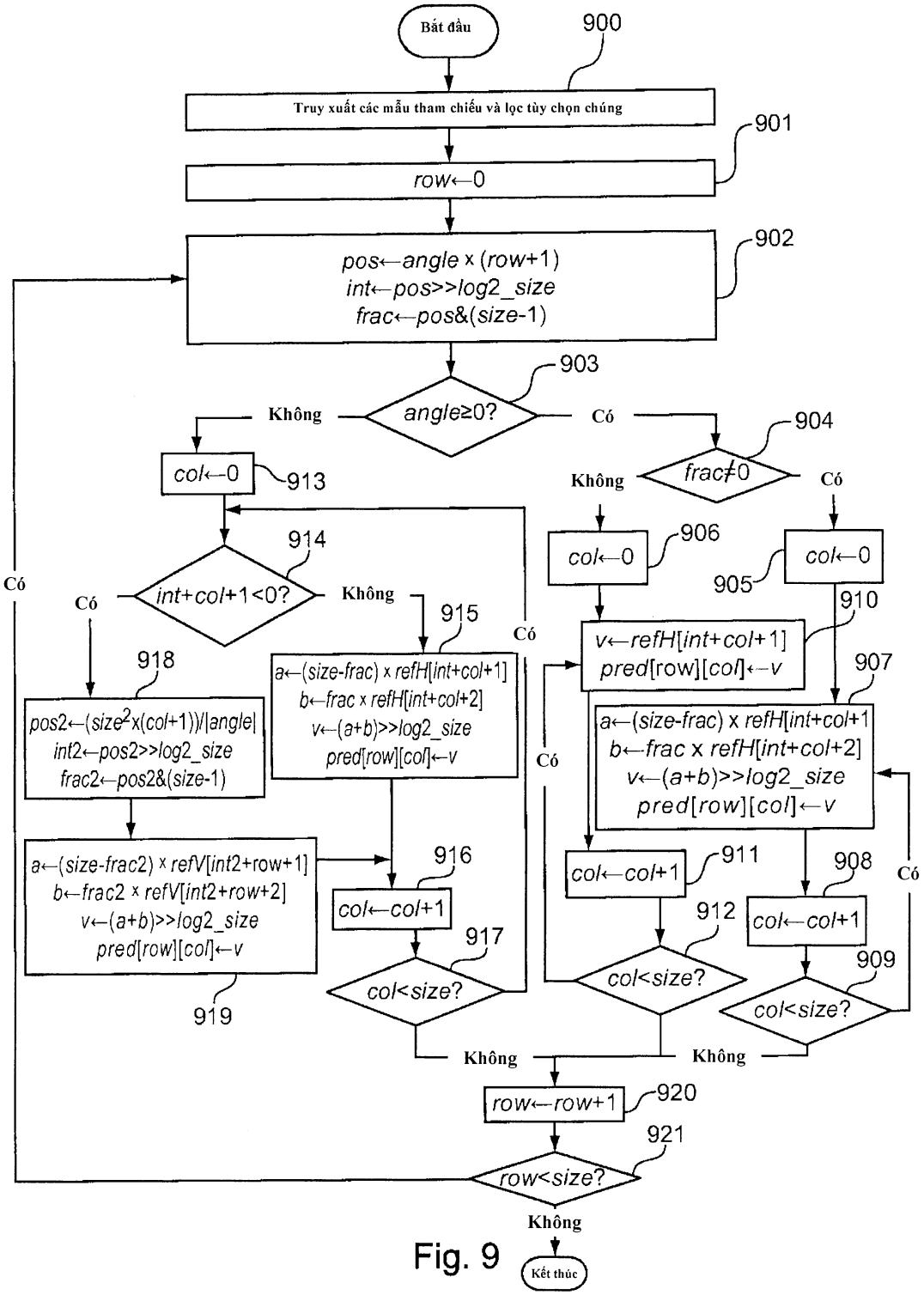

Bắt đầu
900
Truy xuất các mẫu tham chiếu và lọc tùy chọn chúng
901
row←0
902
pos←angle x (row+1)
int←pos>>log2_size
frac←pos&(size-1)
903
angle≥0?
Không
Có
904
frac≠0
Không
Có
col←0
913
914
int+col+1<0?
Có
Không
915
a←(size-frac) x refH[int+col+1]
b←frac x refH[int+col+2]
v←(a+b)>>log2_size
pred[row][col]←v
Có
918
pos2←(size2x(col+1))/|angle|
int2←pos2>>log2_size
frac2←pos2&(size-1)
a←(size-frac2) x refV[int2+row+1]
b←frac2 x refV[int2+row+2]
v←(a+b)>>log2_size
pred[row][col]←v
919
916
col←col+1
917
col<size?
Có
Không
906
col←0
v←refH[int+col+1]
pred[row][col]←v
905
col←0
910
907
a←(size-frac) x refH[int+col+1
b←frac x refH[int+col+2]
v←(a+b)>>log2_size
pred[row][col]←v
Có
911
col←col+1
912
col<size?
Không
908
col←col+1
909
col<size?
Có
Không
920
row←row+1
921
row<size?
Có
Không
Kết thúc

Fig. 9

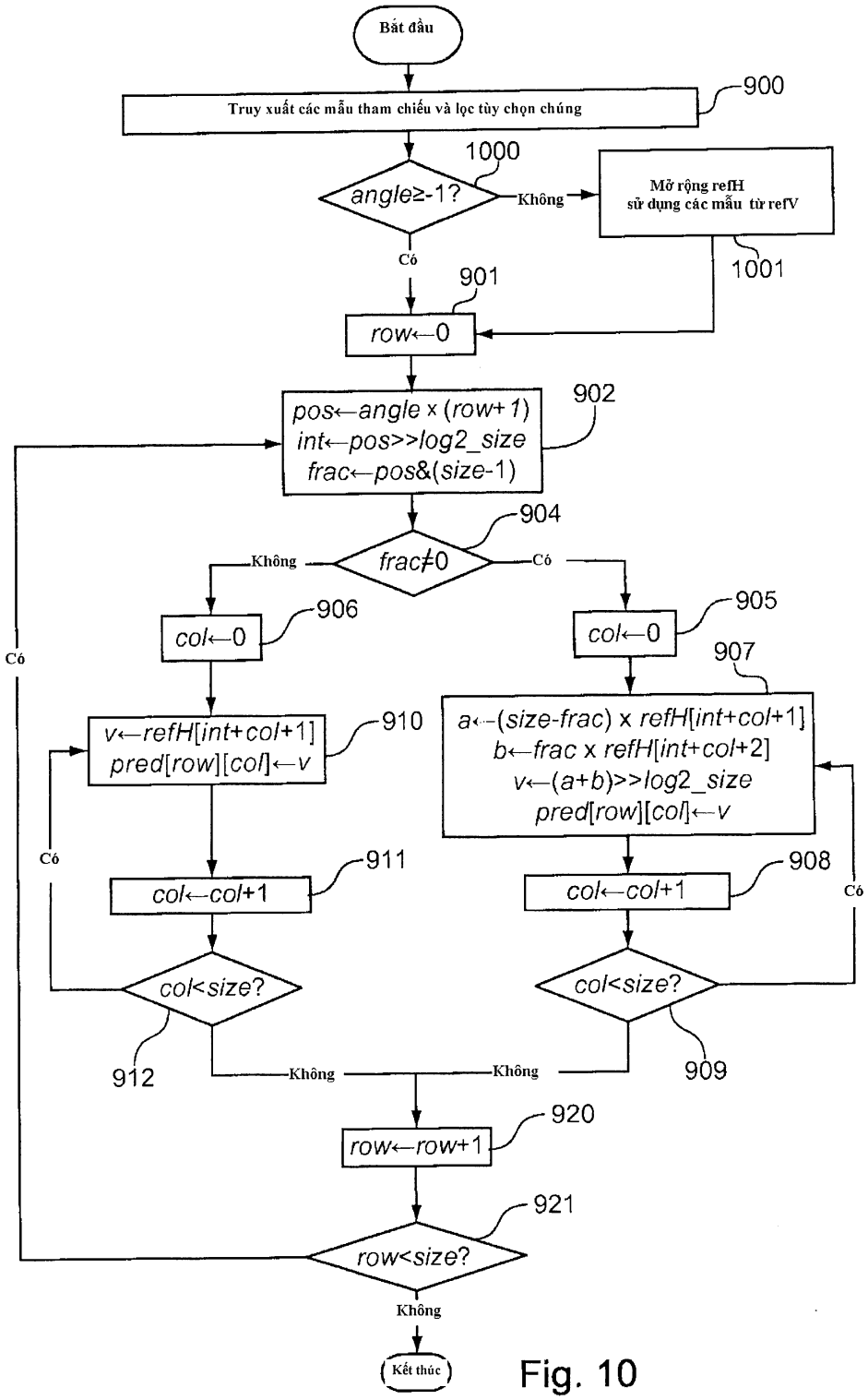

Bắt đầu
Truy xuất các mẫu tham chiếu và lọc tùy chọn chúng
900
angle≥-1?
1000
Không
Có
Mở rộng refH
sử dụng các mẫu từ refV
1001
row←0
901
pos←angle x (row+1)
int←pos>>log2_size
frac←pos&(size-1)
902
frac≠0
904
Không
Có
col←0
906
col←0
905
907
v←refH[int+col+1]
pred[row][col]←v
910
a←(size-frac) x refH[int+col+1]
b←frac x refH[int+col+2]
v←(a+b)>>log2_size
pred[row][col]←v
Có
Có
Có
col←col+1
911
col←col+1
908
col<size?
col<size?
912
909
Không
Không
row←row+1
920
row<size?
921
Không
Kết thúc

Fig. 10

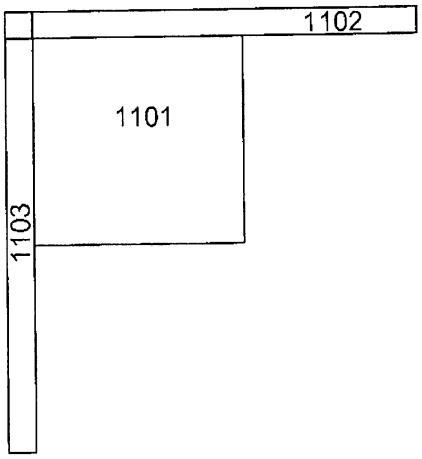


Fig. 11A

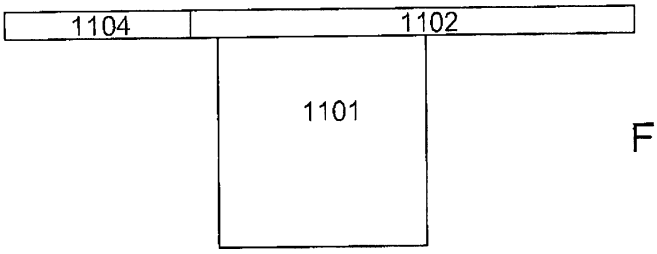


Fig. 11B

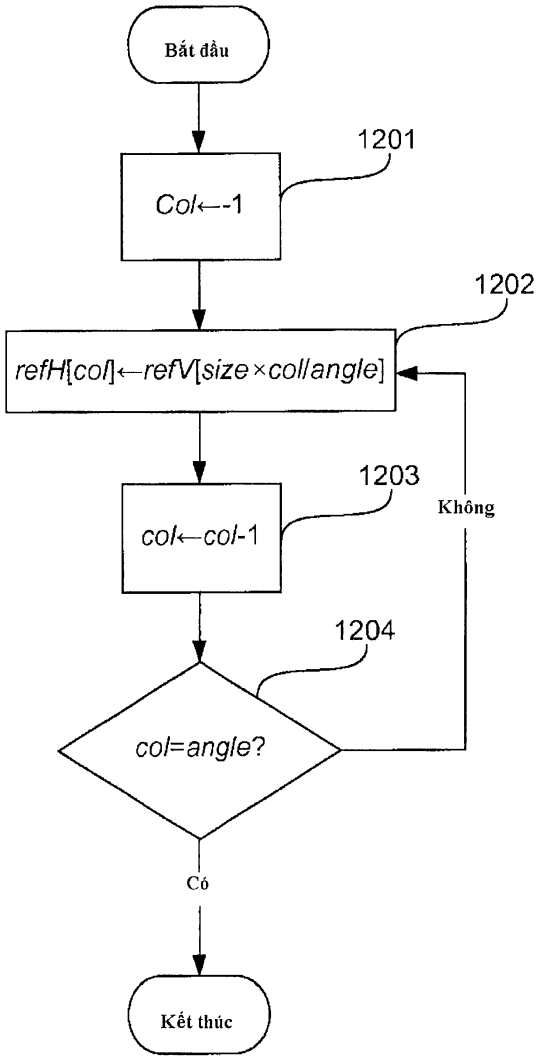

Bắt đầu
1201
Col←-1
1202
refH[col]←refV[size×col/angle]
1203
col←col-1
Không
1204
col=angle?
Có
Kết thúc

Fig. 12

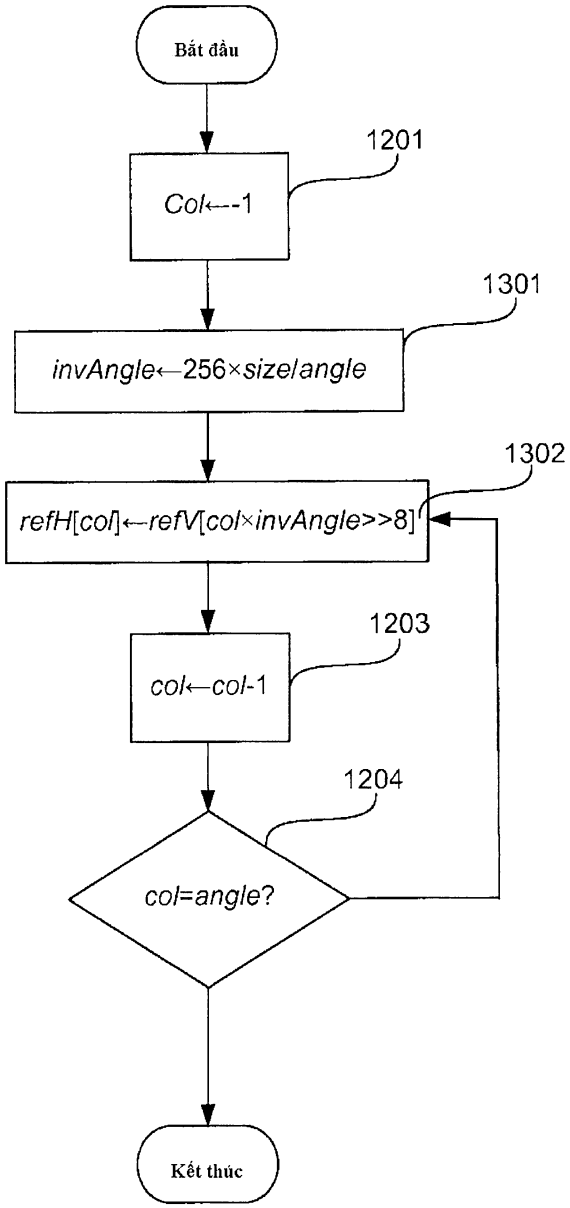

Bắt đầu
1201
Col←-1
1301
invAngle←256×size/angle
1302
refH[col]←refV[col×invAngle>>8]
1203
col←col-1
1204
col=angle?
Kết thúc

Fig. 13

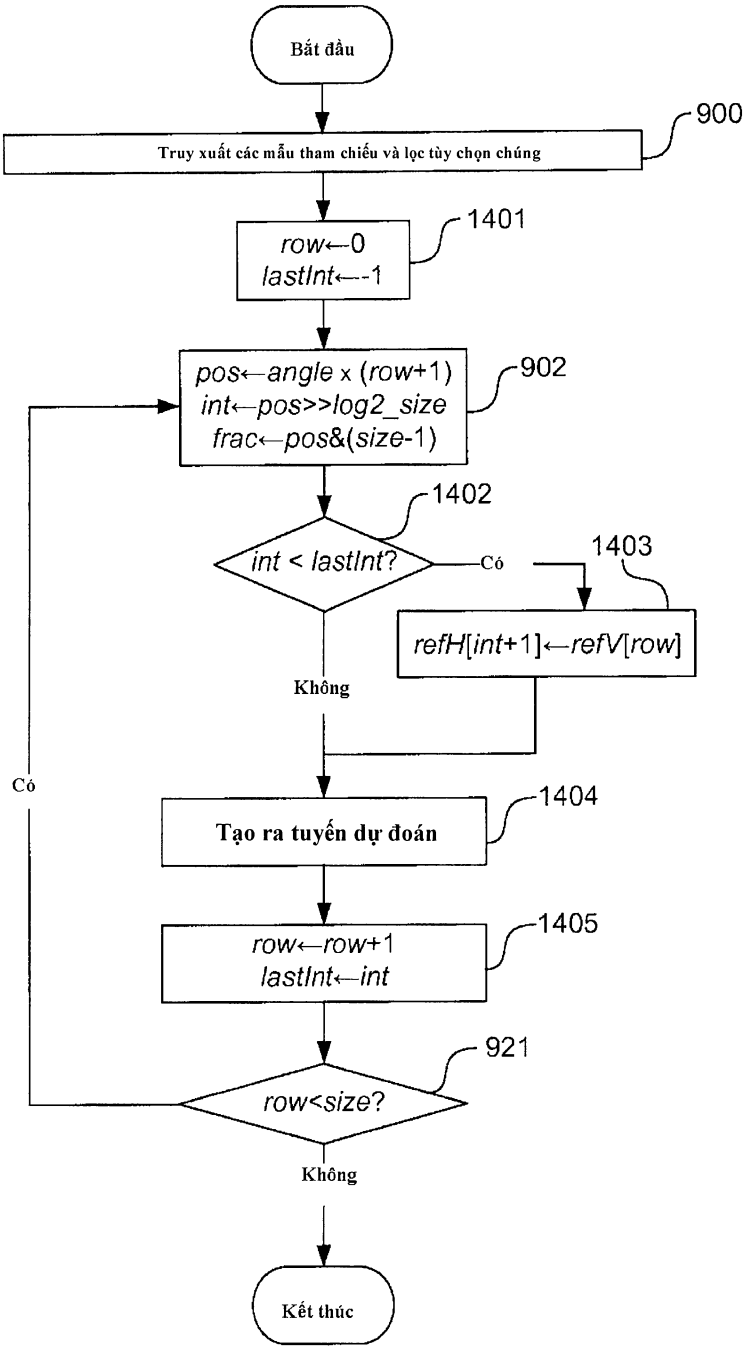

Bắt đầu
Truy xuất các mẫu tham chiếu và lọc tùy chọn chúng
900
row←0
lastInt←-1
1401
pos←angle x (row+1)
int←pos>>log2_size
frac←pos&(size-1)
902
1402
int < lastInt?
Có
1403
refH[int+1]←refV[row]
Không
Có
Tạo ra tuyến dự đoán
1404
row←row+1
lastInt←int
1405
921
row<size?
Không
Kết thúc

Fig. 14