(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) **1-0042357**

(51)[2020.01] **G10L 19/038**; H03M 7/30

(13) **B**

(21) 1-2020-06658
(22) 25/06/2015
(86) PCT/SE2015/050743 25/06/2015
(87) W02016/018185 04/02/2016
(30) 62/029,586 28/07/2014 US
(45) 27/01/2025 442
(43) 25/02/2021 395
(73) TELEFONAKTIEBOLAGET LM ERICSSON (PUBL) (SE)
SE-164 83 Stockholm, Sweden
(72) SVEDBERG, Jonas (SE).
(74) Công ty Luật TNHH T&G (TGVN)

(54) BỘ XỬ LÝ TÍN HIỆU SỐ VÀ PHƯƠNG PHÁP TÌM KIẾM HÌNH DẠNG BỘ LƯỢNG TỬ HÓA VECTƠ HÌNH THÁP, VÀ THIẾT BỊ TRUYỀN THÔNG BAO GỒM BỘ XỬ LÝ TÍN HIỆU SỐ NÀY

(21) 1-2020-06658

(57) Sáng chế đề xuất phương pháp tìm kiếm hình dạng PVQ (Pyramid Vector Quantizer - Bộ lượng tử hóa vectơ hình tháp), PVQ lấy vectơ đích x làm đầu vào và suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong. Phương pháp này bao gồm bước, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị, xác định, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, theo cách không tổn hao trong vòng lặp chiều bên trong sắp tới. Biến $enloop_y$ liên quan đến năng lượng được tích lũy của vectơ y. Việc thực hiện phương pháp này cho phép bộ mã hóa giữ độ phức tạp của sự tìm kiếm ở mức hợp lý. Ví dụ, nó cho phép bộ mã hóa áp dụng vòng lặp có độ chính xác tăng chỉ khi nào việc này có thể là cần thiết, nhờ phân tích liệu "kịch bản cho trường hợp xấu nhất" trong vòng lặp bên trong sắp tới có yêu cầu vòng lặp bên trong với độ chính xác cao hơn so với vòng lặp được sử dụng hiện thời. Sáng chế cũng đề cập đến bộ xử lý tín hiệu số được tạo kết cấu để tìm kiếm hình dạng Lượng tử hóa vectơ hình tháp (Pyramid Vector Quantization - PVQ) và thiết bị truyền thông bao gồm bộ xử lý tín hiệu số.

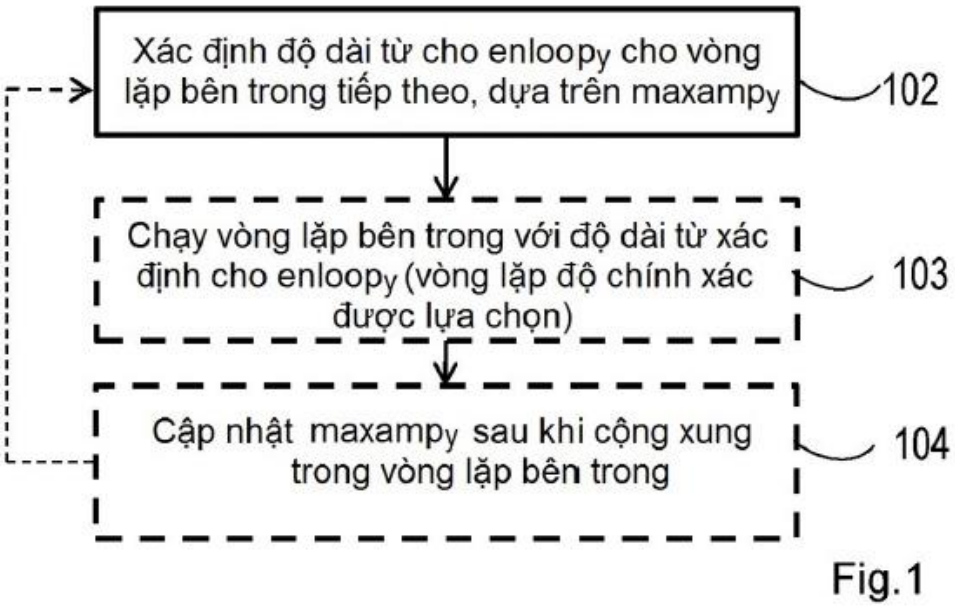


Fig.1

Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập chung đến VQ (vector quantization - lượng tử hóa vectơ) được thực hiện bởi bộ mã hóa.

Tình trạng kỹ thuật của sáng chế

Đã biết là sự lượng tử hóa vectơ không ràng buộc là phương pháp lượng tử hóa tối ưu cho các mẫu được nhóm, nghĩa là các vectơ, có độ dài nhất định. Tuy nhiên, việc thực hiện sự lượng tử hóa vectơ không ràng buộc kéo theo các yêu cầu cao liên quan đến độ phức tạp và dung lượng bộ nhớ. Mong muốn để cho phép thực hiện sự lượng tử hóa vectơ trong cả các tình huống có các giới hạn về bộ nhớ và độ phức tạp tìm kiếm, đã dẫn đến sự phát triển bộ lượng tử hóa được gọi là bộ lượng tử hóa vectơ có cấu trúc. Các cấu trúc khác nhau đưa ra các sự cân bằng khác nhau liên quan đến các yêu cầu về độ phức tạp tìm kiếm và bộ nhớ. Một phương pháp như vậy được gọi là lượng tử hóa vectơ độ khuếch đại-hình dạng, trong đó vectơ đích $\boldsymbol{t}$ được biểu diễn bằng cách sử dụng vectơ hình dạng $\boldsymbol{x}$ và trị số độ khuếch đại $\boldsymbol{G}$:

$$x = \frac{t}{G} \qquad \text{(Phương trình 0)}$$

Khái niệm của sự lượng tử hóa vectơ độ khuếch đại-hình dạng là để lượng tử hóa cặp $\{\mathbf{x}, G\}$ thay vì trực tiếp lượng tử hóa vectơ đích $\boldsymbol{t}$. Các thành phần độ khuếch đại($\boldsymbol{G}$) và hình dạng($\boldsymbol{x}$) được mã hóa sử dụng bộ lượng tử hóa hình dạng mà được điều chỉnh cho đầu vào hình dạng được chuẩn hóa, và bộ lượng tử hóa độ khuếch đại mà xử lý động lực học của tín hiệu. Cấu trúc độ khuếch đại-hình dạng này thường xuyên được sử dụng trong mã hóa âm thanh do sự phân chia thành động lực học và hình dạng, cũng được biểu thị là cấu trúc tinh, phù hợp với kiểu nghe cảm tính. Khái niệm độ khuếch đại-hình dạng cũng có thể được áp dụng cho các hệ số Biến đổi cosin rời rạc hoặc các hệ số khác được sử dụng trong mã hóa viđeo.

Nhiều bộ mã hóa-giải mã (codec) tiếng nói và âm thanh như ITU-T G.718 và IETF Opus (RFC 6716) sử dụng VQ độ khuếch đại-hình dạng dựa trên PVQ có cấu trúc để mã hóa các hệ số phổ của tín hiệu tiếng nói/âm thanh đích.

Khái niệm mã hóa-PVQ được giới thiệu bởi R. Fischer trong quãng thời gian 1983-1986 và đã phát triển để sử dụng trên thực tế kể từ đó với sự tiếp đến của các DSP (Digital Signal Processor - Bộ xử lý tín hiệu số) có hiệu quả hơn. Khái niệm mã hóa PVQ bao gồm tìm kiếm, xác định vị trí và sau đó mã hóa điểm trên siêu hình tháp N-chiều với chuẩn-L1 nguyên của K xung đơn vị. Cái được gọi là chuẩn-L1 là tổng của các trị số tuyệt đối của vectơ, nghĩa là tổng tuyệt đối của vectơ PVQ số nguyên có dấu được giới hạn vào chính xác là K, trong đó xung đơn vị được biểu diễn bởi trị số nguyên là "1". Số nguyên có dấu có khả năng biểu diễn các số nguyên âm, so với không dấu chỉ có thể biểu diễn các số nguyên không âm.

Một trong các lợi ích đáng quan tâm đối với phương pháp tiếp cận mã hóa-PVQ có cấu trúc mà ngược lại với nhiều VQ có cấu trúc khác là không có giới hạn riêng liên quan đến chiều N, nên các phương pháp tìm kiếm được phát triển cho mã hóa-PVQ sẽ có thể áp dụng được đối với chiều N bất kỳ và đối với trị số K bất kỳ.

Một vấn đề đối với sự lượng tử hóa hình dạng PVQ có cấu trúc là tìm vectơ được lượng tử hóa tốt nhất có thể bằng cách sử dụng tổng độ phức tạp hợp lý. Để mã hóa tiếng nói và âm thanh ở tốc độ cao hơn, khi số lượng các xung đơn vị K được cho phép, có thể trở nên rất cao và chiều N cũng có thể cao, có nhu cầu còn mạnh mẽ hơn nữa để có được sự tìm kiếm PVQ có hiệu quả, trong khi duy trì được chất lượng, ví dụ liên quan đến SNR (Signal to Noise Ratio - Tỷ lệ tín hiệu so với nhiễu), của tiếng nói/âm thanh được cấu tạo lại.

Ngoài ra, việc sử dụng khái niệm PVQ không bị giới hạn vào lĩnh vực mã hóa tiếng nói và âm thanh. Hiện nay, nhóm được gọi là IETF (Internet Engineering Task Force - Nhóm đặc trách kỹ thuật Internet) đang theo đuổi việc phát triển bộ mã hóa-giải mã viđeo trong đó các hệ số DCT (Discrete Cosine Transform - Biến đổi cosin rời rạc) được mã hóa bằng cách sử dụng thuật toán dựa trên PVQ. Việc có được thủ tục tìm kiếm có hiệu quả trong mã hóa viđeo còn quan trọng hơn so với trong mã hóa âm thanh, do số lượng hệ số có thể trở nên rất lớn với các thiết bị hiển thị lớn.

Bản chất kỹ thuật của sáng chế

Đối với PVQ có cấu trúc, có mong muốn để cho phép sự tìm kiếm hình dạng có hiệu quả về mặt tính toán mà vẫn đưa ra tỷ lệ tín hiệu so với nhiễu cao. Đặc biệt là đối với việc thực hiện bao gồm DSP có độ chính xác cố định. Giải pháp được đưa ra ở đây cho phép sự tìm kiếm hình dạng PVQ có hiệu quả về mặt tính toán, nhờ đưa ra sự tìm kiếm tinh PVQ được cải tiến.

Theo khía cạnh thứ nhất, sáng chế đề xuất phương pháp tìm kiếm hình dạng PVQ, để được thực hiện bởi bộ mã hóa. PVQ được giả sử bao gồm việc lấy vectơ đích x làm đầu vào suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong. Phương pháp được đề xuất bao gồm bước, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị: xác định, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn, theo cách không tổn hao, biến, $enloop_y$. Biến $enloop_y$ liên quan đến năng lượng được tích lũy (accumulated energy) của y, trong vòng lặp chiều bên trong sắp tới.

Theo khía cạnh thứ hai, bộ mã hóa được đề xuất, để tìm kiếm hình dạng PVQ. PVQ được giả sử bao gồm việc lấy vectơ đích x làm đầu vào suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong. Bộ mã hóa được đề xuất được tạo kết cấu để, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị: xác định, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn, theo cách không tổn hao, biến, $enloop_y$. Biến $enloop_y$ liên quan đến năng lượng được tích lũy của y, trong vòng lặp chiều bên trong sắp tới.

Phương pháp có thể bao gồm bước và bộ mã hóa có thể được tạo kết cấu để, trước khi vào vòng lặp chiều bên trong tiếp theo để cộng xung đơn vị: xác định, dựa trên trị số tuyệt đối cực đại, $xabs_{max}$, của vectơ đầu vào, x, độ dịch lên có thể có, trong từ bit, của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo, $corr_{xy}$, giữa x và vectơ y.

Phương pháp có thể bao gồm bước và bộ mã hóa có thể được tạo kết cấu để, khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, thực hiện các tính toán vòng lặp bên trong sử dụng độ dài từ bit dài hơn để biểu diễn $enloop_y$.

Phương pháp có thể bao gồm bước và bộ mã hóa có thể được tạo kết cấu để, khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, thực hiện các tính toán vòng lặp bên trong sử dụng độ dài từ bit dài hơn để biểu diễn bình phương trị số tương quan trong vòng lặp được tích lũy, $corr_{xy}^2$, giữa x và vectơ y, trong vòng lặp bên trong.

Phương pháp còn có thể bao gồm bước và bộ mã hóa có thể được tạo kết cấu để, khi không cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$:

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ nhất sử dụng độ dài từ bit thứ nhất để biểu diễn $enloop_y$, và:

khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$:

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ hai sử dụng độ dài từ bit dài hơn, so với vòng lặp cộng xung đơn vị thứ nhất, để biểu diễn $enloop_y$.

Bước xác định, dựa trên $maxamp_y$, về liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$ có thể bao gồm xác định các đặc tính của trường hợp khi, trong vòng lặp tìm kiếm bên trong sắp tới, xung đơn vị được cộng vào vị trí trong y mà được kết hợp với $maxamp_y$.

Phương pháp còn có thể bao gồm bước và bộ mã hóa có thể được tạo kết cấu để, trong vòng lặp tìm kiếm chiều bên trong để cộng xung đơn vị:

- xác định vị trí, n_{best}, trong y để cộng xung đơn vị nhờ ước lượng nhân chéo, cho mỗi vị trí n trong y, của trị số năng lượng và tương quan cho n hiện thời; và bình phương tương quan, BestCorrSq và trị số năng lượng, bestEn, được lưu từ các trị số trước của n, như:

$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$

trong đó

$$\left.\begin{array}{c} n_{best} = n \\ bestEn = enloop_y \\ BestCorrSq = corr_{xy}^2 \end{array}\right\} \text{khi } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

Phương pháp cũng có thể bao gồm bước và bộ mã hóa được tạo kết cấu để theo dõi $maxamp_y$ khi trị số cuối cùng của K, được kết hợp với vectơ đích x, vượt quá trị số ngưỡng. Ở đây, phương pháp có thể bao gồm bước và bộ mã hóa có thể được tạo kết cấu để tính toán số dư năng lượng, en_margin, chỉ khi nào trị số hiện thời của K vượt quá trị số ngưỡng mà có thể là trị số ngưỡng được đề cập trong câu trước.

Theo khía cạnh thứ ba, thiết bị truyền thông được đề xuất, thiết bị truyền thông này bao gồm bộ mã hóa theo khía cạnh thứ hai.

Theo khía cạnh thứ tư, chương trình máy tính được đề xuất, chương trình máy tính này bao gồm các lệnh mà, khi được chạy trên ít nhất một bộ xử lý, như DSP, làm cho ít nhất một bộ xử lý thực hiện phương pháp theo khía cạnh thứ nhất.

Theo khía cạnh thứ năm, vật mang được đề xuất, chứa chương trình máy tính theo khía cạnh thứ tư, vật mang này là một trong số tín hiệu điện tử, tín hiệu quang học, tín hiệu rađio, hoặc vật ghi đọc được bởi máy tính.

Theo khía cạnh thứ sáu, và bộ mã hóa được đề xuất, bộ mã hóa này được tạo kết cấu để tìm kiếm hình dạng PVQ; PVQ lấy vectơ đích x làm đầu vào và suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong. Bộ mã hóa được đề xuất bao gồm bộ phận xác định thứ nhất để, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị, xác định, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn, theo cách không tổn hao, biến, $enloop_y$, liên quan đến năng lượng được tích lũy của y, trong vòng lặp chiều bên trong sắp tới.

Bộ mã hóa theo khía cạnh thứ sáu có thể bao gồm bộ phận xác định thứ hai để, trước khi vào vòng lặp chiều bên trong tiếp theo để cộng xung đơn vị, xác định, dựa trên trị số tuyệt đối cực đại, $xabs_{max}$, của vectơ đầu vào, x, độ dịch lên có thể có, trong từ bit, của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo, $corr_{xy}$, giữa x và vectơ y.

Bộ mã hóa theo khía cạnh thứ sáu có thể bao gồm bộ phận tìm kiếm tinh để thực hiện các tính toán vòng lặp bên trong sử dụng độ dài từ bit dài hơn để biểu diễn $enloop_y$, khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$.

Bộ mã hóa theo khía cạnh thứ sáu có thể bao gồm bộ phận tìm kiếm tinh để:

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ nhất sử dụng độ dài từ bit thứ nhất khi không cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, và:

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ hai sử dụng độ dài từ bit dài hơn so với vòng lặp cộng xung đơn vị thứ nhất khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$.

Bộ mã hóa theo khía cạnh thứ sáu có thể bao gồm bộ phận tìm kiếm tinh để

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ nhất, có độ chính xác nhất định, khi không cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$; và để

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ hai, có độ chính xác cao hơn so với vòng lặp cộng xung đơn vị thứ nhất, khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$.

Bộ mã hóa theo khía cạnh thứ sáu có thể được tạo kết cấu để thực hiện việc xác định, dựa trên $maxamp_y$, về liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$ nhờ xác định các đặc tính của trường hợp khi, trong vòng lặp tìm kiếm bên trong sắp tới, xung đơn vị được cộng vào vị trí trong y mà được kết hợp với $maxamp_y$.

Bộ mã hóa theo khía cạnh thứ sáu có thể bao gồm bộ phận tìm kiếm tinh để, trong vòng lặp tìm kiếm chiều bên trong để cộng xung đơn vị:

- xác định vị trí, n_{best}, trong y để cộng xung đơn vị nhờ ước lượng nhân chéo, cho mỗi vị trí n trong y, của trị số năng lượng và tương quan cho n hiện thời; và tương quan, BestCorrSq, và trị số năng lượng, bestEn, được lưu từ các trị số trước của n, như:

$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$

trong đó

$$\left.\begin{array}{c} n_{best} = n \\ bestEn = enloop_y \\ BestCorrSq = corr_{xy}^2 \end{array}\right\} \text{khi } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

Bộ mã hóa theo khía cạnh thứ sáu có thể bao gồm bộ phận lưu trữ để theo dõi $maxamp_y$ khi số lượng các xung đơn vị cuối cùng, K, được kết hợp với vectơ đích x, vượt quá trị số ngưỡng.

Theo khía cạnh thứ bảy, thiết bị truyền thông được đề xuất, thiết bị truyền thông này bao gồm bộ mã hóa theo khía cạnh thứ sáu.

Mô tả vắn tắt các hình vẽ

Các mục đích, dấu hiệu, và ưu điểm nêu trên và các mục đích, dấu hiệu, và ưu điểm khác của sáng chế được bộc lộ ở đây sẽ trở nên rõ ràng từ phần mô tả chi tiết dưới đây của các phương án như được minh họa trên các hình vẽ kèm theo. Các hình vẽ không nhất thiết theo cùng tỷ lệ, thay vào đó sự nhấn mạnh sẽ được đặt vào khi minh họa các nguyên lý của sáng chế được bộc lộ ở đây.

Fig.1-Fig.4 là các hình vẽ minh họa phương pháp tìm kiếm hình dạng PVQ (tìm kiếm tinh), theo các phương án làm ví dụ khác nhau.

Fig.5 là hình vẽ thể hiện các bước của một phương án để tìm kiếm hình dạng PVQ (tìm kiếm tinh), theo một phương án làm ví dụ.

Fig.6 là hình vẽ thể hiện chi tiết hơn về các bước để tìm kiếm hình dạng PVQ (tìm kiếm tinh) trên Fig.5, theo một phương án làm ví dụ.

Fig.7 là hình vẽ minh họa các phương án để tìm kiếm hình dạng PVQ.

Fig.8 là hình vẽ thể hiện thiết bị truyền thông theo một phương án được trang bị bộ mã hóa EVS.

Fig.9 là hình vẽ thể hiện thiết bị truyền thông theo một phương án, và

Fig.10 cũng là hình vẽ thể hiện thiết bị truyền thông theo một phương án.

Fig.11a-Fig.11c là các hình vẽ thể hiện bộ mã hóa theo các phương án làm ví dụ.

Fig.12 là hình vẽ thể hiện ví dụ về hệ thống mã hóa âm thanh PVQ, trong đó ít nhất một phần của hệ thống được bao gồm trong bộ mã hóa và/hoặc bộ mã hóa-giải mã mà đến lượt được bao gồm trong thiết bị truyền thông, như điện thoại di động.

Mô tả chi tiết sáng chế

Trong số học dấu phẩy động không có vấn đề lớn nào liên quan đến việc thiết lập động lực học của các thông số bước lặp để tìm kiếm hình dạng PVQ vòng lặp bên trong, tuy nhiên trong các DSP có độ chính xác cố định với, ví dụ, các bộ tích lũy (accumulator) giới hạn 16/32 bit (thanh ghi trong đó lưu trữ các kết quả số học và/hoặc lôgic trung gian) và các biến, điều rất quan trọng là dùng các phương pháp tìm kiếm có hiệu quả trong đó dải động giới hạn của các biến DSP được cực đại hóa và độ chính xác được cực đại hóa, trong khi có thể sử dụng nhiều nhất có thể các hoạt động DSP dải động giới hạn nhanh sẵn có.

Thuật ngữ "độ chính xác" trên đây chỉ khả năng có thể biểu diễn các số nhỏ nhất có thể, nghĩa là số bit sau dấu phẩy thập phân cho độ dài từ cụ thể. Nói cách khác là, độ chính xác tương ứng với độ phân giải của sự biểu diễn, mà một lần nữa được định nghĩa bởi số chữ số thập phân hoặc nhị phân. Nguyên nhân cho việc độ chính xác theo các phương án được mô tả dưới đây có thể được nói là tương quan với số bit sau dấu phẩy thập phân và không nhất thiết với chính độ dài từ đó là, trong số học dấu phẩy tĩnh, có thể có các độ chính xác khác nhau cho cùng độ dài từ. Ví dụ, các định dạng dữ liệu 1Q15 và 2Q14 đề có độ dài từ 16, nhưng định dạng thứ nhất có 15 bit sau dấu phẩy thập phân và định dạng kia có 14 bit. Số nhỏ nhất có thể biểu diễn được khi đó sẽ lần lượt là 2^-15 và 2^-14.

Một cách thực hiện sự lượng tử hóa vectơ hình tháp của hình dạng được bộc lộ trong đoạn 3.2 của Valin và đồng tác giả, "A full-bandwidth audio codec with low complexity and very low delay", EUSIPCO, 2009. Trong tài liệu này bộ mã hóa-giải mã MDCT được thể hiện trong đó các chi tiết, nghĩa là hình dạng, trong mỗi băng được lượng tử hóa về đại số bằng cách sử dụng sách mã hình cầu và trong đó sự cấp phát bit được suy ra từ thông tin được chia sẻ giữa bộ mã hóa và bộ giải mã. Các khía cạnh và phương án theo sáng chế đề cập, ít nhất là theo cách không chặt chẽ, đến cách thức thực hiện sự tìm kiếm theo các phương trình 4-7 trong tài liệu của Valin và đồng tác giả, theo cách có hiệu quả theo dấu phẩy tĩnh được giới hạn vào, ví dụ, số học 16/32 bit thay vì các trị số động như trong tài liệu của Valin và đồng tác giả.

Theo một số khía cạnh và phương án được bộc lộ sau đây, cho vectơ đích $x(n)$ (t trong Phương trình 0) có chiều N nhất định, và cho số xung đơn vị K nhất định, hình dạng được phân tích và vectơ cấu tạo lại thích hợp $x_q(n)$=func($y(n)$), mà cực tiểu hóa lỗi lượng tử hóa hình dạng, và theo đó cực đại hóa chất lượng được cảm nhận ví dụ trong trường hợp mã hóa âm thanh, được xác định. Ít nhất trong số trong các khía cạnh và phương án được thực hiện để nhằm tìm sự tập hợp tối ưu của K xung đơn vị, trong vectơ $y(n)$ mà cần bám vào chuẩn L1, trong khi kiểm soát được độ phức tạp, nghĩa là thấp nhất có thể trên thực tế.

Thay vì sử dụng các phương pháp lặp hở trong tình trạng kỹ thuật để xác định các trị số xấp xỉ cho dải động vòng lặp bên trong và độ chính xác của bộ tích lũy, một số trong các khía cạnh và phương án được thiết kế để sử dụng sự phân tích trước "gần tối ưu" phí tổn thấp, liên quan đến các chu kỳ DSP cần thiết và liên quan đến ROM (Read-Only Memory - Bộ nhớ chỉ đọc) chương trình bổ sung cần thiết, của tử số trong trường hợp xấu nhất và/hoặc mẫu số trong trường hợp xấu nhất trước khi bắt đầu các ước lượng tốn kém của thương số biến dạng hình dạng PVQ trong vòng lặp tìm kiếm trong cùng. Sự phân tích trước "gần tối ưu" không nhằm mục đích định tỷ lệ các trị số vào dải động cực đại tối ưu chính xác, mà thay vào đó sự phân tích trước xác định lũy thừa gần tối ưu của hệ số tỷ lệ 2, do lũy thừa của sự định tỷ lệ 2 có thể được thực hiện như các phép dịch của số nhị phân và các phép dịch như vậy có phí tổn thấp trong các chu kỳ DSP và trong ROM DSP.

Sự lựa chọn độ chính xác của mẫu số được thúc đẩy về cảm tính do các vùng đỉnh (peaky region) về phổ sẽ được cấp phát chính xác hơn so với các vùng bằng phẳng (flatter region).

Trong khi một số trong các khái niệm chính được mô tả trong bản mô tả này bao hàm các biến đổi khác nhau và các cấu trúc thay thế, các phương án của các khía cạnh được thể hiện trên các hình vẽ và mã làm ví dụ và sẽ được mô tả chi tiết sau đây.

Giới thiệu sự tối ưu hóa tổng quát của tìm kiếm PVQ

Bộ lượng tử hóa PVQ có cấu trúc chuẩn-L1 cho phép một số sự tối ưu hóa tìm kiếm, trong đó sự tối ưu hóa chính là di chuyển đích đến "góc phần tư" (quadrant) dương toàn bộ (cũng có thể được biểu thị là orthant hoặc siêu góc phần tám (hyper octant))

trong không gian N-chiều và sự tối ưu hóa thứ hai là sử dụng phép chiếu chuẩn-L1 như sự xấp xỉ bắt đầu cho ***y(n)***. Chuẩn-L1 của ***K*** cho ***PVQ(N,K)*** có nghĩa là tổng tuyệt đối của tất cả các phần tử trong vectơ PVQ ***y(n)*** phải là ***K***, đúng như tổng tuyệt đối của tất cả các phần tử trong vectơ hình dạng đích x(n).

Sự tối ưu hóa thứ ba là cập nhật lặp lại Q_{PVQ} các số hạng thương $corr_{xy}^2$ và $energy_y$, thay vì tính toán lại Phương trình 4 (dưới đây) qua toàn bộ không gian vectơ N, cho mọi thay đổi ứng viên đối với vectơ ***y(n)*** để theo đuổi việc đạt được chuẩn-L1 ***K***, mà được yêu cầu cho bước đánh chỉ số tiếp theo.

Ba bước tối ưu hóa lớn trên đây là các sự tối ưu hóa mà nói chung có thể tồn tại trong các sự thực hiện PVQ trong quá khứ như CELT và IETF-Opus, và một phần trong G.718, tuy nhiên để làm phần mô tả các khía cạnh và phương án trở nên đầy đủ, các bước này cũng được phác họa vắn tắt dưới đây.

Sự tìm kiếm hình dạng vectơ PVQ có hiệu quả

Tổng quan về hệ thống mã hóa và giải mã âm thanh áp dụng một phương án tìm kiếm hình dạng PVQ được bộc lộ trong bản mô tả này có thể được thấy trên Fig.12. Sự tìm kiếm hình dạng tổng quát sử dụng phép chiếu hình tháp theo sau bởi tiến trình tìm kiếm (hình dạng) tinh có thể được thấy, ví dụ, trên Fig.5. Một phương án khác của phần tìm kiếm tinh của sự tìm kiếm hình dạng được mô tả trên Fig.6. Sự tìm kiếm hình dạng PVQ có thể bao gồm phép chiếu hình tháp và tìm kiếm tinh. Khi không áp dụng phép chiếu hình tháp, thì sự tìm kiếm hình dạng chỉ bao gồm tìm kiếm tinh. Do đó, "tìm kiếm tinh" và "tìm kiếm hình dạng" đôi khi có thể được sử dụng hoán đổi trong bản mô tả này, do sự tìm kiếm tinh là một phần của tìm kiếm hình dạng, và khi không có sự tìm kiếm thô ban đầu, bởi phép chiếu hình tháp, thì việc thực hiện tìm kiếm hình dạng đúng là cùng một việc với thực hiện tìm kiếm tinh. Nói cách khác, sự tìm kiếm tinh đôi khi có thể là hoặc tạo thành sự tìm kiếm hình dạng, và khi áp dụng phép chiếu hình tháp, thì sự tìm kiếm tinh là một phần của tìm kiếm hình dạng.

Giới thiệu sự tìm kiếm PVQ

Mục tiêu của thủ tục tìm kiếm ***PVQ(N,K)*** là để tìm vectơ đầu ra được định tỷ lệ và được chuẩn hóa tốt nhất $x_q(n)$. $x_q(n)$ được định nghĩa là:

$$\mathbf{x_q} = \frac{\mathbf{y}}{\sqrt{\mathbf{y^T y}}}$$ (Phương trình 1)

Trong đó $y = y_{N,K}$ là điểm trên bề mặt của siêu hình tháp N chiều và chuẩn L1 của $y_{N,K}$ là K. Nói cách khác, $y_{N,K}$ là vectơ mã hình dạng nguyên được lựa chọn có kích thước N, cũng được biểu thị là chiều N, theo:

$$\mathbf{y}_{N,K} = \left\{ e : \sum_{i=0}^{N-1} |e_i| = K \right\}$$ (Phương trình 2)

Nghĩa là, vectơ x_q là vectơ phụ nguyên được chuẩn hóa năng lượng đơn vị $y_{N,K}$. Vectơ y tốt nhất là một vectơ cực tiểu hóa lỗi hình dạng bình phương trung bình giữa vectơ đích $x(n)$ và vectơ đầu ra được lượng tử hóa được chuẩn hóa đã được định tỷ lệ x_q. Điều này đạt được nhờ cực tiểu hóa độ biến dạng tìm kiếm sau:

$$d_{PVQ} = -\mathbf{x}^T \mathbf{x_q} = -\frac{\left(\mathbf{x}^T \mathbf{y}\right)}{\sqrt{\mathbf{y^T y}}}$$ (Phương trình 3)

Hoặc tương đương, nhờ lấy bình phương tử số và mẫu số, cực đại hóa thương số Q_{PVQ}:

$$Q_{PVQ} = \frac{\left(\mathbf{x}^T \mathbf{y}\right)^2}{\mathbf{y^T y}} = \frac{\left(corr_{xy}\right)^2}{energy_y}$$ (Phương trình 4)

trong đó $corr_{xy}$ là tương quan giữa x và y. Trong tìm kiếm hình dạng vectơ PVQ tối ưu $y(n)$ với chuẩn-L1 K, các cập nhật lặp lại của các biến Q_{PVQ} được thực hiện trong "góc phần tư" dương toàn bộ trong không gian N chiều theo:

$$corr_{xy}(k,n) = corr_{xy}(k-1) + 1 \cdot x(n)$$ (Phương trình 5)

$$energy_y(k,n) = energy_y(k-1) + 2 \cdot 1^2 \cdot y(k-1,n) + 1^2$$ (Phương trình 6)

trong đó $\mathbf{corr_{xy}(k-1)}$ biểu thị tương quan thu được cho đến hiện tại nhờ đặt ***k-1*** xung đơn vị trước, và $\boldsymbol{energy_y(k-1)}$ biểu thị năng lượng được tích lũy thu được cho đến hiện tại nhờ đặt ***k-1*** xung đơn vị trước, và ***y(k-1, n)*** biểu thị biên độ của ***y*** ở vị trí ***n*** từ lần đặt trước của ***k-1*** xung đơn vị. Để tăng tốc thêm nữa đối với việc xử lý lặp lại trong vòng lặp, số hạng năng lượng $\boldsymbol{energy_y(k)}$ được định tỷ lệ xuống bởi 2, theo đó loại bỏ một phép nhân trong vòng lặp bên trong.

$$enloop_y(k,n) = energy_y(k,n)/2, \Rightarrow$$
$$enloop_y(k,n) = enloop_y(k-1) + y(k-1,n) + 0{,}5 \quad \text{(Phương trình 7)}$$

trong đó $\boldsymbol{enloop_y(k,n)}$ là biến năng lượng ưu tiên được sử dụng và được tích lũy bên trong vòng lặp tìm kiếm xung đơn vị trong cùng, do cập nhật lặp lại của nó yêu cầu ít hơn một phép nhân so với $\boldsymbol{energy_y(k,n)}$.

$$Q_{PVQ}(k,n) = \frac{corr_{xy}(k,n)^2}{enloop_y(k,n)} \quad \text{(Phương trình 8)}$$

Vị trí tốt nhất $\boldsymbol{n_{best}}$ cho xung đơn vị thứ ***k***', được cập nhật lặp lại nhờ tăng ***n*** từ ***0*** đến ***N-1***:

$$n_{best} = n \quad , nếu \; Q_{PVQ}(k,n) > Q_{PVQ}(k,n_{best})$$

(Phương trình 9)

Để tránh các phép chia tốn kém, mà đặc biệt quan trọng trong số học dấu phẩy tĩnh, quyết định cập nhật cực đại hóa $\boldsymbol{Q_{PVQ}}$ được thực hiện bằng cách sử dụng sự nhân chéo của tử số tương quan bình phương tốt nhất được lưu ***bestCorrSq*** và mẫu số năng lượng tốt nhất được lưu ***bestEn*** cho đến hiện tại, mà có thể được biểu diễn như:

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq = corr_{xy}(k,n)^2 \\ bestEn = enloop_y(k,n) \end{array}\right\}, \; nếu \; corr_{xy}(k,n)^2 \cdot bestEn > bestCorrSq \cdot enloop_y(k,n)$$

(Phương trình 10)

Sự cực đại hóa lặp lại của $\boldsymbol{Q_{PVQ}(k, n)}$ có thể bắt đầu từ số không (zero) của các xung đơn vị được đặt hoặc từ số đặt trước có phí tổn thấp hơn thích ứng của các xung

đơn vị, dựa trên phép chiếu nguyên thành điểm dưới bề mặt của hình tháp thứ ***K'***, với sự đạt mức dưới chuẩn bảo đảm của các xung đơn vị trong chuẩn L1 đích ***K***.

Phân tích việc chuẩn bị tìm kiếm PVQ

Do bản chất có cấu trúc của vectơ nguyên PVQ $y_{N,K}$, trong đó cho phép tất cả các tổ hợp dấu có thể có và có thể mã hóa tất cả các tổ hợp dấu, miễn là vectơ kết quả báo vào chuẩn L1 của **K** xung đơn vị, sự tìm kiếm được thực hiện trong *"góc phần tư"* thứ nhất dương toàn bộ (nguyên nhân cho các dấu trích dẫn trên "góc phần tư" là góc phần tư đúng chỉ tồn tại khi **N**=2, và **N** ở đây có thể lớn hơn 2). Ngoài ra, như tác giả sáng chế nhận biết, để đạt được độ chính xác cao nhất có thể cho sự thực hiện chính xác giới hạn, trị số tuyệt đối cực đại $xabs_{max}$ của tín hiệu đầu vào ***x(n)*** có thể được phân tích trước để sử dụng trong tương lai trong khi thiết lập thủ tục tích lũy tương quan vòng lặp bên trong.

$$xabs(n) = |\mathbf{x}(n)|, \; cho \;\; n = 0,\ldots,N-1 \qquad \text{(Phương trình 11)}$$

$$xabs_{max} = max(xabs_0, \ldots, xabs_{N-1}) \qquad \text{(Phương trình 12)}$$

Xử lý các đích năng lượng rất thấp và các vectơ phụ năng lượng rất thấp

Trong trường hợp vectơ đích đầu vào (***x*** trong Phương trình 3 hoặc ***t*** trong Phương trình 0) là vectơ tất cả bằng không (zero) và/hoặc độ khuếch đại vectơ (ví dụ ***G*** trong Phương trình 0) là rất thấp, thì sự tìm kiếm PVQ có thể được bỏ qua, và vectơ PVQ hợp lệ ***y*** có thể được tạo ra theo cách tất định nhờ gán một nửa của ***K*** xung đơn vị vào vị trí thứ nhất $\left(y[0] = \left\lfloor \frac{K}{2} \right\rfloor\right)$ và các xung đơn vị còn lại vào vị trí cuối cùng ($y[N-1] = y[N-1] + (K - y[0])$).

Thuật ngữ "các đích năng lượng rất thấp" và "độ khuếch đại vectơ rất thấp" theo một phương án là thấp như không (zero), như được minh họa trong mã C ANSI làm ví dụ được bộc lộ dưới đây, trong đó mã tương ứng là:

```
IF( L_xsum == 0  ||  neg_gain == 0 )
{   /*  zero input  or  zero gain  case */
```

Tuy nhiên, nó cũng có thể nhỏ hơn hoặc bằng epxilon, hoặc EPS, trong đó EPS là trị số nhỏ nhất mà lớn hơn so với không (zero) và nó được coi là đáng để biểu diễn theo độ chính xác được lựa chọn. Ví dụ, trong độ chính xác Q15 trong từ 16 bit có dấu, độ khuếch đại vectơ phụ trở nên nhỏ hơn hoặc bằng EPS 1/2^15= 1/32768 (ví dụ độ khuếch đại vectơ nhỏ hơn hoặc bằng 0,000030517578125), và trong trường hợp của độ chính xác Q12 trong từ 16 bit có dấu cho vectơ đích x(n), thì trị số "rất thấp" trở thành EPS = (1/2^12), ví dụ sum (abs (x(n))) nhỏ hơn hoặc bằng 0,000244140625. Theo một phương án của số học DSP dấu phẩy tĩnh với từ 16 bit, định dạng số nguyên không dấu có thể lấy trị số nguyên bất kỳ từ 0 đến 65546, trong khi đó số nguyên có dấu có thể lấy trị số từ -32768 đến +32767. Khi sử dụng định dạng phân số không dấu, 565536 mức được trải đều giữa 0 và +1, trong khi đó theo phương án định dạng phân số có dấu các mức sẽ được đặt cách đều giữa -1 và +1.

Nhờ áp dụng bước tùy chọn này liên quan đến các vectơ không (zero) và các trị số độ khuếch đại thấp, độ phức tạp tìm kiếm PVQ được giảm và độ phức tạp đánh chỉ số được dàn trải/chia sẻ giữa đánh chỉ số ở bộ mã hóa và giải chỉ số ở bộ giải mã, nghĩa là không có xử lý nào "bị lãng phí" để tìm kiếm vectơ đích không (zero) hoặc vectơ đích rất thấp mà, theo cách bất kỳ, sẽ được định tỷ lệ xuống không (zero).

Phép chiếu tìm kiếm trước PVQ tùy chọn

Nếu tỷ lệ mật độ xung ***K/N*** lớn hơn so với 0,5 xung đơn vị cho mỗi hệ số, ví dụ hệ số biến đổi cosin rời rạc cải biến, thì phép chiếu phí tổn thấp thành hình tháp phụ ***K-1*** được thực hiện và sử dụng là điểm bắt đầu cho ***y***. Mặt khác, nếu tỷ lệ mật độ xung nhỏ hơn so với 0,5 xung đơn vị cho mỗi hệ số, thì sự tìm kiếm PVQ lặp sẽ khởi đầu từ ***0*** xung đơn vị được đặt trước. Phép chiếu phí tổn thấp thành "K-1" thường ít tốn kém về mặt tính toán hơn trong các chu kỳ DSP so với việc lặp lại sự tìm kiếm vòng lặp bên trong xung đơn vị theo ***K-1*** lần. Tuy nhiên, nhược điểm của phép chiếu phí tổn thấp là nó sẽ tạo ra kết quả không chính xác do áp dụng hàm làm tròn xuống (floor) N chiều. Chuẩn-L1 kết quả của phép chiếu phí tổn thấp sử dụng hàm làm tròn xuống (floor) thường có thể là bất kỳ giữa "***K-1***" đến khoảng chừng "***K-5***", nghĩa là kết quả sau phép chiếu cần được tìm kiếm tinh để đạt đến chuẩn đích của **K**.

Phép chiếu phí tổn thấp được thực hiện như:

$$proj_{fac} = \frac{K-1}{\sum_{n=0}^{n=N-1} \mathbf{xabs}(n)}$$

(Phương trình 13)

$$y(n) = y_{start}(n) = \lfloor \mathbf{xabs}(n) \cdot proj_{fac} \rfloor \text{ cho n=0,..,N-1}$$

(Phương trình 14)

Nếu không thực hiện phép chiếu, thì điểm bắt đầu là vectơ ***y(n)*** tất cả bằng không (zero). Phí tổn DSP của phép chiếu trong các chu kỳ DSP là trong lân cận của *N*(tổng tuyệt đối)+***25***(phép chia)+***2N(***phép nhân và làm tròn xuống (floor)***)*** chu kỳ.

Khi chuẩn bị cho sự tìm kiếm tinh để đạt đến bề mặt của hình tháp thứ ***K***' số xung đơn vị được tích lũy $\boldsymbol{pulse_{tot}}$, tương quan được tích lũy $\boldsymbol{corr_{xy}(pulse_{tot})}$ và năng lượng được tích lũy $\boldsymbol{energy_y(pulse_{tot})}$ cho điểm bắt đầu được tính toán như:

$$pulse_{tot} = \sum_{n=0}^{n=N-1} \mathbf{y}(n)$$

(Phương trình 15)

$$corr_{xy}(pulse_{tot}) = \sum_{n=0}^{n=N-1} \mathbf{y}(n) \cdot \mathbf{xabs}(n)$$

(Phương trình 16)

$$energy_y(pulse_{tot}) = \sum_{n=0}^{n=N-1} \mathbf{y}(n) \cdot \mathbf{y}(n) = \|\mathbf{y}\|_{L2}$$

(Phương trình 17)

$$enloop_y(pulse_{tot}) = energy_y(pulse_{tot})/2$$

(Phương trình 18)

Tìm kiếm tinh PVQ

Giải pháp được bộc lộ trong bản mô tả này liên quan đến sự tìm kiếm tinh PVQ (mà tạo thành hoặc là một phần của sự tìm kiếm hình dạng PVQ, như được mô tả trên đây). Những gì đã được mô tả trong các phần trước chủ yếu là PVQ trong tình trạng kỹ thuật, ngoại trừ việc xác định trước của $xabs_{max}$, mà sẽ được mô tả thêm dưới đây. Vectơ hình dạng nguyên cuối cùng ***y(n)*** của chiều N phải bám vào chuẩn L1 của ***K*** xung. Sự

tìm kiếm tinh được giả sử là được tạo kết cấu để bắt đầu từ điểm dưới trong hình tháp, nghĩa là dưới hình tháp thứ K', và tìm lặp lại đường của nó đến bề mặt của siêu hình tháp thứ ***K'*** ***N***-chiều. Trị số ***K*** trong sự tìm kiếm tinh thường có thể nằm trong khoảng từ 1 đến 512 xung đơn vị.

Tác giả đã nhận biết, là để giữ độ phức tạp của sự tìm kiếm và đánh chỉ số PVQ ở mức hợp lý, sự tìm kiếm có thể được tách thành hai nhánh chính, trong đó một nhánh được sử dụng khi đã biết là sự biểu diễn năng lượng trong vòng lặp của ***y(n)*** sẽ ở trong từ 16 bit có dấu hoặc không dấu trong bước lặp của vòng lặp tìm kiếm bên trong tiếp theo, và một nhánh khác được sử dụng khi năng lượng trong vòng lặp có thể vượt quá dải động của từ 16 bit trong bước lặp của vòng lặp tìm kiếm bên trong tiếp theo.

Sự tìm kiếm tinh với độ chính xác cố định cho số xung đơn vị thấp

Khi ***K*** cuối cùng thấp hơn hoặc bằng ngưỡng của t_p=127 xung đơn vị, động lực học của $energy_y(K)$ sẽ luôn ở trong 14 bit, và động lực học của $enloop_y(K)$ được dịch lên 1 bit sẽ luôn ở trong 15 bit. Điều này cho phép sử dụng từ 16 bit có dấu để biểu diễn mọi $enloop_y(k)$ trong tất cả các bước lặp của vòng lặp bên trong tìm kiếm xung tinh lên tới ***k=K***. Nói cách khác, sẽ không cần độ dài bit từ vượt quá 16 bit để biểu diễn $energy_y(K)$ hoặc $enloop_y(K)$ trong bước lặp của vòng lặp bên trong tìm kiếm xung tinh bất kỳ khi K<127.

Trong trường hợp sẵn có các toán tử Multiply (nhân), MultiplyAdd (nhân-cộng) và MultiplySubtract (nhân-trừ) DSP có hiệu quả cho các biến 16 bit không dấu, ngưỡng có thể được tăng thành $t_p = 255$, như vậy thì $enloop_y(K)$ sẽ luôn ở trong từ 16 bit không dấu. MultiplyAdd trong bản mô tả này là, theo một phương án, các lệnh nhân-cộng hoặc các phép toán tương đương để nhân các trị số dữ liệu biểu diễn các tín hiệu âm thanh và viđeo bởi bộ lọc hoặc biến đổi các trị số và tích lũy các tích số để tạo ra kết quả. Các phép toán MultiplySubtract giống như các phép toán MultiplyAdd, ngoại trừ các phép cộng được thay thế bởi các phép trừ.

Khi chuẩn bị cho việc cộng xung đơn vị tiếp theo, độ dịch lên có thể có cực đại gần tối ưu của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo, $corr_{xy}$, trong từ 32 bit có dấu được phân tích trước bằng cách sử dụng trị số đầu vào tuyệt đối cực đại được tính toán trước $xabs_{max}$ như:

$$corr_{upshift} = 31 - \left\lceil \log2\left(corr_{xy}(pulse_{tot}) + 2 \cdot (1 \cdot xabs_{max})\right)\right\rceil$$

(Phương trình 19)

Độ dịch lên được tính toán trong Phương trình 19 này biểu diễn "trường hợp xấu nhất", và bao trùm độ dịch lên có thể có cực đại mà có thể được thực hiện trong vòng lặp bên trong tiếp theo, và theo đó bảo đảm là thông tin có ý nghĩa nhất liên quan đến sự tương quan sẽ không bị mất, hoặc bị dịch ra ngoài, trong bước lặp của vòng lặp bên trong, ngay cả đối với kịch bản cho trường hợp xấu nhất.

Sự phân tích động vòng lặp bên trong trước trong trường hợp xấu nhất có thể được thực hiện trong 2-3 chu kỳ trong hầu hết các kiến trúc DSP sử dụng các lệnh MultiplyAdd và Norm (chuẩn hóa), và sự phân tích luôn có cùng tính độc lập của chiều N. Trong DSP 16/32-bit ảo G.191 ITU-T các phép toán trong Phương trình 19 trở thành: "corr_upshift = norm_l(L_mac(*L_corrxy,1, xabs_max));" với phí tổn là 2 chu kỳ. Cần lưu ý là norm_l(x) trong bản mô tả này tương ứng với "31 - ceil(log2(x))", và theo cách khác có thể được biểu thị là 31 - ceil(log2(x)), trong đó ceil(x) là hàm được gọi là hàm làm tròn trên (ceiling), ánh xạ số thực thành số nguyên nhỏ nhất tiếp theo. Chính xác hơn, ceiling(x) = $\lceil x \rceil$ là số nguyên nhỏ nhất mà không nhỏ hơn x. Đối với $corr_{upshift}$, số hạng nằm trong các dấu ngoặc với thanh nằm ngang phía trên luôn là số dương. $corr_{upshift}$ theo cách khác có thể được tính toán bằng cách sử dụng hàm làm tròn xuống (floor) như:

$$corr_{upshift} = 30 - \left\lfloor \log2\left(corr_{xy}(pulse_{tot}) + 2 \cdot (1 \cdot xabs_{max})\right)\right\rfloor$$

trong đó floor(x) =$\lfloor x \rfloor$ là số nguyên lớn nhất mà không lớn hơn x.

Một lợi ý khác của phương pháp tiếp cận được đề xuất trong bản mô tả này đối với việc định tỷ lệ tương quan tìm kiếm hình dạng gần tối ưu đó là phương pháp được đề xuất không yêu cầu vectơ đích được chuẩn hóa trước x, điều này sẽ tiết kiệm độ phức tạp bổ sung nào đó trước khi bắt đầu tìm kiếm hình dạng.

Để làm cho Phương trình 10 lặp cập nhật hiệu quả nhất có thể, *tử số* $corr_{xy}(k,n)^2$ có thể được biểu diễn bởi từ 16 bit có dấu, ngay cả khi bao gồm nhiều thông tin hơn so với mức phù hợp trong từ 16 bit, nhờ phương pháp tiếp cận sau.

$$corr_{xy16}(k,n)^2 = \left(\text{Round}_{16}\left(\left(corr_{xy}(pulse_{tot}) + 2 \cdot \left(1 \cdot \mathbf{xabs}(n)\right) \right) \cdot 2^{corr_{upshift}} \right) \right)^2$$

(Phương trình 20)

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{16} = corr_{xy16}(k,n)^2 \\ bestEn_{16} = enloop_y(k,n) \end{array}\right\}, \text{nếu } corr_{xy16}(k,n)^2 \cdot bestEn_{16} > bestCorrSq_{16} \cdot enloop_y(k,n)$$

(Phương trình 21)

trong đó hàm "Round_{16}" tách 16 bit trên cùng của biến 32 bit có dấu có làm tròn. Độ dịch lên gần tối ưu này (Phương trình 10) và việc sử dụng biểu diễn 16 bit của bình phương tương quan ***bestCorrSq16*** cho phép sự tìm kiếm vòng lặp bên trong rất nhanh chỉ sử dụng ~9 chu kỳ để thực hiện kiểm tra Phương trình 21 test và ba lần cập nhật biến, khi sử dụng các hàm Multiply, MultiplyAdd, MultiplySubtract được tối ưu DSP.

Vị trí của xung đơn vị tiếp theo trong vectơ y hiện được xác định nhờ lặp qua ***n=0,..., N-1*** vị trí có thể có trong vectơ y, trong khi dùng phương trình 20, phương trình 6 và phương trình 21.

Khi vị trí tốt nhất n_{best} cho xung đơn vị (trong vectơ y đạt được cho đến hiện tại) đã được xác định, tương quan được tích lũy $corr_{xy}(k)$, năng lượng trong vòng lặp được tích lũy $enloop_y(k)$ và số xung đơn vị được tích lũy $pulse_{tot}$ được cập nhật. Nếu có các xung đơn vị thêm nữa để cộng, nghĩa là khi $pulse_{tot} < K$, thì vòng lặp bên trong mới được bắt đầu với phân tích $corr_{upshift}$ gần tối ưu mới (Phương trình 19) để cộng xung đơn vị tiếp theo.

Tổng cộng, phương pháp tiếp cận được đề xuất này có độ phức tạp trong trường hợp xấu nhất cho mỗi xung đơn vị được cộng vào ***y(n)*** là khoảng chừng 5/N+15 chu kỳ cho mỗi hệ số được lượng tử hóa. Nói cách khác, vòng lặp qua vectơ có kích thước N để cộng xung đơn vị có độ phức tạp trong trường hợp xấu nhất là khoảng N*(5/N+15) chu kỳ, nghĩa là 5+15*N chu kỳ.

<u>Sự tìm kiếm tinh với độ chính xác cố định cho số lượng xung đơn vị cao</u>

Khi ***K*** cao hơn so với ngưỡng t_p, mà theo phương án làm ví dụ này giả sử DSP giới hạn 16/32 bit, là t_p=127 xung đơn vị, động lực học của thông số $energy_y(K)$ có thể

vượt quá 14 bit, và động lực học của $enloop_y(K)$ được dịch lên 1 bit có thể vượt quá 15 bit. Theo đó, để không sử dụng độ chính xác cao không cần thiết, sự tìm kiếm tinh được tạo kết cấu để chọn theo cách thích ứng giữa sự biểu diễn 16 bit và sự biểu diễn 32 bit của cặp { $corr_{xy}(k,n)^2$, $enloop_y(k,n)$ } khi K cao hơn so với t_p. Khi K cho vectơ $y(n)$ được biết trước để kết thúc ở trị số cuối cùng cao hơn so với 127, sự tìm kiếm tinh sẽ theo dõi biên độ xung cực đại $maxamp_y$ trong y đạt được cho đến hiện tại. Việc này cũng có thể được gọi là việc $maxamp_y$ được xác định. Thông tin biên độ xung cực đại này được sử dụng trong bước phân tích trước trước khi vào vòng lặp chiều bên trong được tối ưu hóa. Sự phân tích trước bao gồm xác định sẽ sử dụng độ chính xác nào cho vòng lặp bên trong để cộng xung đơn vị sắp tới. Như được thể hiện trên Fig.12 bởi đầu vào của N, K đối với sự tìm kiếm hình dạng PVQ, sự cấp phát bit được biết/xác định trước khi khởi đầu sự tìm kiếm PVQ. Sự cấp phát bit có thể sử dụng các công thức hoặc các bảng lưu để thu được, xác định và/hoặc tính toán K để được nhập vào sự tìm kiếm hình dạng PVQ, ví dụ K = hàm(bits(band), N) cho băng nhất định với chiều N và số bit (băng) nhất định.

Bit được yêu cầu cho PVQ(N,K)	N=8	N=16	N=32
K=4	11,4594	15,4263	19,4179
K=5	13,2021	18,1210	23,1001
K=6	14,7211	20,5637	26,5222
K=7	16,0631	22,7972	29,7253

Ví dụ, bảng lưu như bảng được thể hiện trên đây có thể được sử dụng để xác định hoặc lựa chọn trị số của K. Nếu chiều N là 8 và các bit sẵn có cho băng ***bits(band)*** là ***14,0***, thì K sẽ được lựa chọn là ***5***, do **PVQ(N=8,K=6)** yêu cầu 14,7211 bit là cao hơn so với số bit sẵn có 14,0.

Nếu sự phân tích trước biểu thị là cần nhiều hơn từ 16 bit có dấu để biểu diễn năng lượng trong vòng lặp mà không mất thông tin năng lượng bất kỳ, thì dùng vòng lặp cộng xung đơn vị có độ chính xác cao hơn và có độ chính xác cao chuyên sâu hơn

về mặt tính toán, trong đó cả số hạng bình phương tương quan tốt nhất được lưu và số hạng năng lượng được tích lũy tốt nhất được lưu được biểu diễn bởi các từ 32 bit.

$$en_{margin} = 31 - \left\lceil \log 2\left(\left(1 + energy_y(pulse_{tot}) + 2 \cdot \left(1 \cdot maxamp_y\right)\right)\right)\right\rceil$$

(Phương trình22)

$$highprecision_{active} = FALSE, \quad nếu\ (en_{margin} \geq 16)$$
$$highprecision_{active} = TRUE, \quad nếu\ (en_{margin} < 16)$$ (Phương trình 23)

Sự phân tích động ở vòng lặp bên trong trước trong trường hợp xấu nhất có thể được thực hiện trong 5-6 chu kỳ bổ sung trong hầu hết của DSP, và phí tổn phân tích là giống nhau cho tất cả các chiều. Trong DSP 16/32bit ảo STL 2009 G.191 ITU-T các phép toán trong Phương trình 22 và Phương trình 23 trở thành:

```
"L_energy_y       = L_add(L_energy_y, 1);  /* 0.5 added */
 en_margin        = norm_l(L_mac(L_energy_y, 1, maxamp_y));
 highprecision_active= 1;                  move16();
 if(sub(16,en_margin <= 0){
   highprecision_active = 0;               move16();
 }",
```

với phí tổn cực đại là 6 chu kỳ.

Mã tương ứng trong ví dụ mã C ANSI dưới đây là:

```
L_yy      = L_add(L_yy,1);   /* .5 added  */
en_margin  = norm_l(L_mac(L_yy,1, max_amp_y)); /*find max "addition", margin,~2 ops */
en_dn_shift = sub(16, en_margin);  /* calc. shift to lower word */
high_prec_active = 1;    move16();
if( en_dn_shift <= 0 ){  /* only use 32 bit energy if actually needed */
high_prec_active = 0;  move16();
}
```

Thay vào đó, số dư năng lượng ***en_margin*** trong Phương trình (22) có thể nối tiếp phép toán của hàm STL G.191 ***norm_l()*** được tính toán bằng cách sử dụng hàm làm tròn xuống (floor) như:

$$en_{margin} = 30 - \left\lfloor \log2\left(\left(1 + energy_y(pulse_{tot}) + 2 \cdot (1 \cdot maxamp_y)\right)\right)\right\rfloor$$

Nếu ***highprecision***$_{active}$ là *FALSE (SAI)*, nghĩa là =0, thì vòng lặp tìm kiếm bên trong có độ chính xác thấp hơn trong Phương trình 20, Phương trình 6 và Phương trình 21 được dùng, mặt khác, khi ***highprecision***$_{active}$ là *TRUE (ĐÚNG),* nghĩa là=1, thì vị trí của xung đơn vị tiếp theo được thực hiện dùng vòng lặp bên trong có độ chính xác cao hơn, biểu diễn enloop$_y$ và corr$_{xy}^2$ với các từ 32 bit theo ví dụ này. Nghĩa là, khi ***highprecision***$_{active}$ là *TRUE* (ĐÚNG), vị trí của xung đơn vị tiếp theo trong y(n) được xác định nhờ lặp qua ***n****=0,..., N-1* vị trí có thể có, sử dụng Phương trình 24, Phương trình 6 và Phương trình 25.

$$corr_{xy32}(k,n)^2 = \left(\left(corr_{xy}(pulse_{tot}) + 2 \cdot (1 \cdot \mathbf{xabs}(n))\right) \cdot 2^{corr_{upshift}}\right)^2$$

(Phương trình 24)

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{32} = corr_{xy32}(k,n)^2 \\ bestEn_{32} = enloop_y(k,n) \end{array}\right\}, nếu\ corr_{xy32}(k,n)^2 \cdot bestEn_{32} > bestCorrSq_{32} \cdot enloop_y(k,n)$$

(Phương trình25)

Nói cách khác, en_margin nhằm biểu thị có bao nhiêu độ dịch lên có thể được sử dụng để chuẩn hóa năng lượng trong vòng lặp tiếp theo. Nếu có thể sử dụng 16 hoặc nhiều hơn 16 lần dịch lên, thì năng lượng nằm ở độ dài từ thấp hơn, giả sử các độ dài từ 16/32 bit, và không cần vòng lặp có độ chính xác cao (sự biểu diễn 32 bit), như vậy ***highprecision***$_{active}$ được thiết lập thành FALSE (SAI). Một nguyên nhân thực hiện để làm theo cách này (cho phép thông tin năng lượng nằm ở phần thấp của từ 32 bit L_energy) là nó rẻ hơn về mặt tính toán: nó chỉ tốn 1 chu kỳ để tính toán extract_l(L_energy) trong khi round_fx(L_shl(L_energy,en_margin)) thay thế tốn hai chu kỳ.

Khi vị trí tốt nhất ***n***$_{best}$ của xung đơn vị đã được xác định, tương quan được tích lũy ***corr***$_{xy}$***(k)***, năng lượng trong vòng lặp được tích lũy ***enloop***$_y$***(k)*** và số xung đơn vị được tích lũy ***pulse***$_{tot}$ được cập nhật. Ngoài ra, biên độ cực đại $maxamp_y$ trong vectơ

nguyên tốt nhất y cho đến hiện tại, được giữ cập nhật, nghĩa là được xác định, cho vòng lặp cộng xung đơn vị tiếp theo.

$$maxamp_y = \max\left(maxamp_y, y[n_{best}]\right) \quad \text{(Phương trình 26)}$$

Nếu có các xung đơn vị thêm nữa để cộng, nghĩa là khi $pulse_{tot} < K$, thì vòng lặp bên trong mới được bắt đầu với sự phân tích $corr_{upshift}$ gần tối ưu mới theo Phương trình 19 và sự phân tích có độ chính xác năng lượng mới theo Phương trình 22 và Phương trình 23, và tiếp theo khởi đầu vòng lặp xung đơn vị tiếp theo với Phương trình 24, Phương trình 6 và Phương trình 26.

Độ phức tạp trong trường hợp xấu nhất theo phương pháp tiếp cận có độ chính xác cao (theo ví dụ này là các từ 32 bit) cho mỗi xung đơn vị được cộng vào $y(n)$ là khoảng chừng 7/N+31 chu kỳ cho mỗi hệ số được lượng tử hóa.

Hiệu quả của việc lựa chọn độ chính xác của vòng lặp bên trong dựa trên năng lượng được tích lũy trong vòng lặp đó là các vectơ phụ đích mà có mức đỉnh cao, hoặc có độ chi tiết rất tinh, nghĩa là K cuối cùng là cao, sẽ sử dụng thường xuyên hơn vòng lặp có độ chính xác cao hơn và nhiều chu kỳ hơn, trong khi các vectơ phụ không có đỉnh hoặc độ chi tiết xung thấp sẽ sử dụng thường xuyên hơn vòng lặp có độ chính xác thấp hơn và ít chu kỳ hơn.

Cần lưu ý là sự phân tích được mô tả trong phần trên cũng có thể được thực hiện khi $K<t_p$. Tuy nhiên, một phương án có thể được thực hiện có hiệu quả hơn nhờ đưa vào ngưỡng t_p để áp dụng sự phân tích trên.

Hoàn thành và chuẩn hóa vectơ PVQ

Sau khi tìm kiếm hình dạng, mỗi thành phần vectơ PVQ không bằng không (non-zero) được gán dấu thích hợp của nó và vectơ được chuẩn hóa L2 (còn được biết đến là chuẩn hóa Ơclit (Euclidean)) thành năng lượng đơn vị. Ngoài ra, nếu băng được tách, thì nó còn được định tỷ lệ với độ khuếch đại vectơ phụ.

$$nếu\left(\mathbf{y}(n) > 0\right) \cap \left(\mathbf{x}(n) < 0\right) \Rightarrow \mathbf{y}(n) = -\mathbf{y}(n), cho \quad n = 0,\ldots,N-1$$

(Phương trình 27)

$$norm_{gain} = \frac{1}{\sqrt{\mathbf{y}^{\mathbf{T}}\mathbf{y}}} \quad \text{(Phương trình 28)}$$

$$\mathbf{x_q}(n) = norm_{gain} \cdot \mathbf{y}(n), cho \quad n = 0,\ldots,N-1 \quad \text{(Phương trình 29)}$$

Trên đây, hai hệ phương pháp về độ chính xác được thể hiện và được chỉ rõ: "***En16 x CorrSq16***", như được định nghĩa trong phần trên đây, (các phương trình từ 19 đến 21) và "***En32 x CorrSq32***", (các phương trình từ 22 đến 26). Sau đây sẽ mô tả hai phương pháp có độ phức tạp trung bình thêm nữa trong đó độ chính xác của số hạng Tương quan bình phương làm tử số và số hạng Năng lượng được thay đổi.

Các phương pháp "En16 x CorrSq32" và "En32 x CorrSq16"

Phương pháp "***En16 x CorrSq32***" tương tự với "***En32 x CorrSq32***", nhưng có khác biệt là sự so sánh và cập nhật xung đơn vị được tìm thấy tốt nhất ở vòng lặp bên trong sử dụng sự biểu diễn 16 bit của ***bestEn16*** năng lượng tốt nhất cho đến hiện tại, theo:

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{32} = corr_{xy32}(k,n)^2 \\ bestEn_{16} = enloop_y(k,n) \end{array}\right\}, nếu\ corr_{xy32}(k,n)^2 \cdot bestEn_{16} > bestCorrSq_{32} \cdot enloop_y(k,n)$$

(Phương trình 30)

Phí tổn xấp xỉ của phương pháp "***En16 x CorrSq32***" cho mỗi xung đơn vị là 5/N + 21 chu kỳ.

Phương pháp "***En32 x CorrSq16***" tương tự với "***En32 x CorrSq32***", nhưng có khác biệt là sự so sánh và cập nhật xung đơn vị được tìm thấy tốt nhất ở vòng lặp bên trong sử dụng sự biểu diễn 16 bit của bình phương tương quan tốt nhất ***bestCorrSq16*** cho đến hiện tại, theo:

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{16} = corr_{xy16}(k,n)^2 \\ bestEn32 = enloop_y(k,n) \end{array}\right\}, nếu\ corr_{xy16}(k,n)^2 \cdot bestEn32 > bestCorrSq16 \cdot enloop_y(k,n)$$

(Phương trình 31)

Phí tổn xấp xỉ của phương pháp "***En32 x CorrSq16***" cho mỗi phép cộng xung đơn vị là 6/N+20 chu kỳ cho mỗi hệ số.

Các khía cạnh và các phương án làm ví dụ

Sau đây, một số phương án làm ví dụ của giải pháp được bộc lộ trong bản mô tả này sẽ được mô tả với tham khảo đến Fig.1-Fig.4.

Fig.1 là biểu đồ tiến trình minh họa phương pháp liên quan đến sự tìm kiếm tinh của tìm kiếm hình dạng PVQ. Phương pháp được dự tính để được thực hiện bởi bộ mã hóa, như bộ mã hóa phương tiện cho các tín hiệu âm thanh và/hoặc viđeo. PVQ lấy vectơ hình dạng đích x làm đầu vào, và suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong. Phương pháp đề cập đến sự phân tích trước, mà được thực hiện trước khi vào vòng lặp bên trong. Vectơ đầu ra x_q sau đó sẽ được suy ra dựa trên vectơ y, như được mô tả trên đây. Tuy nhiên, việc tạo thành x_q không phải là trung tâm của giải pháp được mô tả ở đây, và theo đó sẽ không được mô tả thêm trong bản mô tả này.

Theo phương pháp được minh họa trên Fig.1, có thể giả sử là bộ mã hóa theo dõi trị số $maxamp_y$ của vectơ hiện thời y. "Vectơ hiện thời y" trong bản mô tả này có nghĩa là vectơ y được tạo, được tìm hoặc được cấu tạo cho đến hiện tại, nghĩa là đối với k < K. Như được mô tả trên đây, điểm bắt đầu cho vectơ y có thể là phép chiếu vào bề mặt dưới hình tháp thứ K, hoặc vectơ tất cả bằng không (zero) trống. Phương pháp được minh họa trên Fig.1 bao gồm, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị, xác định 101, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, theo cách không tổn hao trong vòng lặp chiều bên trong sắp tới. Biến $enloop_y$ liên quan đến năng lượng được tích lũy của vectơ y. Việc thực hiện phương pháp cho phép bộ mã hóa giữ độ phức tạp của sự tìm kiếm ở mức hợp lý. Ví dụ, nó cho phép bộ mã hóa áp dụng vòng lặp có độ chính xác tăng (kéo theo độ phức tạp cao hơn) chỉ khi nào việc này có thể là cần thiết, nhờ phân tích liệu "kịch bản cho trường hợp xấu nhất" trong vòng lặp bên trong sắp tới có yêu cầu vòng lặp bên trong với độ chính xác cao hơn so với được sử dụng hiện thời.

Sự phân tích trước được mô tả trên đây được thực hiện trước mỗi lần vào 102 vòng lặp bên trong, nghĩa là trước mỗi phép cộng của xung đơn vị vào vectơ y. Theo một phương án làm ví dụ trong đó chỉ sẵn có hai sự biểu diễn bit khác nhau, nghĩa là các độ dài từ bit như 16 và 32 bit, vòng lặp bên trong sẽ được thực hiện bằng cách sử dụng sự biểu diễn 16 bit của $enloop_y$ cho đến khi xác định được là cần từ bit dài hơn để biểu diễn $enloop_y$, sau đó độ dài từ bit cao hơn, nghĩa là sự biểu diễn 32 bit sẽ được áp dụng cho các tính toán vòng lặp bên trong. Vòng lặp sử dụng sự biểu diễn 16 bit có thể được gọi là "vòng lặp có độ chính xác thấp", và vòng lặp sử dụng sự biểu diễn 32 bit có thể được gọi là "vòng lặp có độ chính xác cao".

Việc xác định 102 là liệu có cần nhiều hơn độ dài từ bit ban đầu hoặc hiện thời theo cách khác có thể được biểu thị như là việc xác định được sẽ yêu cầu độ dài từ bit nào, trong số ít nhất hai độ dài từ bit khác nhau, thay thế, để biểu diễn $enloop_y$ trong "trường hợp xấu nhất" (tăng lớn nhất có thể có) trong vòng lặp bên trong tiếp theo. Ít nhất hai độ dài bit từ khác nhau có thể bao gồm ít nhất, ví dụ, các độ dài từ 16 và 32 bit.

Nói cách khác, khi nhiều hơn độ dài từ bit hiện thời được xác định 102 là cần để biểu diễn $enloop_y$ trong vòng lặp bên trong tiếp theo, các tính toán vòng lặp bên trong được thực hiện 103 với độ dài từ bit dài hơn, so với độ dài từ bit ban đầu hoặc hiện thời, để biểu diễn $enloop_y$ trong vòng lặp bên trong. Mặt khác, khi nhiều hơn độ dài từ bit hiện thời được xác định là không cần để biểu diễn $\boldsymbol{enloop_y}$, các tính toán vòng lặp bên trong có thể được thực hiện nhờ dùng vòng lặp cộng xung đơn vị thứ nhất sử dụng độ dài từ bit thứ nhất hoặc hiện thời để biểu diễn $\boldsymbol{enloop_y}$, nghĩa là độ dài từ bit hiện thời có thể tiếp tục được sử dụng. Điều này cũng được minh họa, ví dụ, trên Fig.4, như việc sử dụng hai vòng lặp khác nhau. Fig.4 thể hiện một vòng lặp bên trong có độ chính xác thấp, mà được chạy 405 khi xác định 403 là độ dài từ bit hiện thời (thấp hơn) là đủ; và một vòng lặp bên trong có độ chính xác cao, mà được chạy khi xác định 403 là cần độ dài từ bit cao hơn để biểu diễn năng lượng trong vòng lặp bên trong, để không mất thông tin.

Phương pháp dựa vào việc thực hiện là sự tăng có thể có cực đại của biến năng lượng, như $enloop_y$, trong vòng lặp bên trong tiếp theo sẽ xảy ra khi xung đơn vị được cộng vào vị trí trong y mà được kết hợp với $maxamp_y$ hiện thời. Sau khi thực hiện điều

này, có thể xác định, trước khi vào vòng lặp bên trong, liệu có rủi ro là vượt quá khả năng biểu diễn của độ dài từ bit được sử dụng hiện thời, ví dụ 16 bit, trong vòng lặp bên trong tiếp theo, hay không. Nói cách khác, việc xác định về liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$ bao gồm xác định các đặc tính của trường hợp khi, trong vòng lặp tìm kiếm bên trong sắp tới, xung đơn vị được cộng vào vị trí trong y mà được kết hợp với $maxamp_y$. Ví dụ, số bit cần để biểu diễn $\boldsymbol{enloop_y}$ trong vòng lặp bên trong sắp tới có thể được xác định, hoặc, theo cách khác là số dư còn lại trong từ bit biểu diễn $enloop_y$ trong vòng lặp bên trong sắp tới.

Đối với các vectơ hình dạng đích được kết hợp với K thấp, có thể nói trước là sẽ không cần độ dài từ bit dài hơn so với độ dài được đề xuất bởi độ dài từ bit ban đầu và được sử dụng hiện thời. Do đó, sẽ có thể áp dụng trị số ngưỡng Tk, sao cho các phép toán nhất định được thực hiện chỉ cho các vectơ hình dạng đích được kết hợp với K mà vượt quá trị số ngưỡng Tk. Đối với các vectơ đích như vậy, bộ mã hóa sẽ theo dõi $maxamp_y$, nhờ cập nhật trị số này sau mỗi phép cộng xung. Đối với các vectơ đích được kết hợp với K mà thấp hơn so với trị số ngưỡng, không cần thiết theo dõi $maxamp_y$. Ví dụ với các từ 16 và 32 bit, Tk có thể có sẽ là 127, như được mô tả trên đây. Nói cách khác, việc theo dõi $maxamp_y$ và việc xác định về liệu có cần nhiều hơn độ dài từ bit hiện thời được thực hiện, ví dụ, chỉ khi trị số cuối cùng của K được kết hợp với vectơ hình dạng đích đầu vào vượt quá trị số ngưỡng Tk.

Một phương án được minh họa trên Fig.2 bao gồm theo dõi hoặc xác định 201 maxampy, và xác định 202 xabsmax. Trị số của maxampy có thể được thay đổi khi cộng xung đơn vị mới trong vòng lặp bên trong và theo đó maxampy cần được cập nhật, để được giữ cập nhật sau mỗi vòng lặp. Ví dụ, hoạt động 201 có thể bao gồm theo dõi maxampy cho đến khi trị số của k đã đạt đến trị số ngưỡng trong đó độ dài từ bit ban đầu hoặc hiện thời được sử dụng để biểu diễn enloopy có thể không còn là đủ, và sự phân tích được biểu diễn, ví dụ, bởi hoạt động 204 được khởi đầu. Sự cập nhật của maxampy sau vòng lặp bên trong theo sau sự phân tích của, ví dụ, hoạt động 204 được minh họa như hoạt động 206 trên Fig.2. Tuy nhiên, cần lưu ý là xabsmax không được thay đổi trong quy trình, và do đó chỉ cần được xác định 202 một lần. Như được minh họa trên Fig.2, một phương án của phương pháp cũng có thể bao gồm, trước khi vào 205

vòng lặp chiều bên trong tiếp theo để cộng xung đơn vị, xác định 203, dựa trên trị số tuyệt đối cực đại, $xabs_{max}$, của vectơ hình dạng đầu vào, x, độ dịch lên có thể có, trong từ bit, của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo, $corr_{xy}$, giữa x và vectơ y. Độ dịch lên cũng có thể được biểu thị là định tỷ lệ lên. Phương trình 19 trên đây minh họa việc xác định độ dịch lên có thể có cực đại. Nhờ thực hiện bước này, có thể bảo đảm là duy trì được nhiều bit thông tin tương quan nhất có thể trong ước lượng vòng lặp bên trong, đặc biệt là các bit có ý nghĩa nhất. Trong bản mô tả này cần lưu ý là trị số tương quan $corr_{xy}$, dưới dạng $corr_{xy}^2$, không nhất thiết cần được biểu diễn theo cách không tổn hao. Việc xác định độ dịch lên cực đại có thể được thực hiện trong từ bit "dài hơn", bất kể độ dài từ bit hiện thời được sử dụng trong vòng lặp bên trong. Nghĩa là, độ dịch lên có thể có cực đại có thể được xác định cho từ 32 bit ngay cả khi từ 16 bit sẽ được sử dụng trong vòng lặp bên trong. Khi từ bit ngắn hơn là để được sử dụng trong vòng lặp bên trong, thì độ dịch lên được xác định sẽ được "làm tròn" thành từ bit ngắn hơn, như được minh họa bởi Phương trình 20. Lưu ý là trị số tương quan, $corr_{xy}$, luôn nhỏ hơn hoặc bằng một (1,0) trong độ chính xác DSP được áp dụng cho trị số tương quan, nghĩa là $corr_{xy} \leq 1{,}0$, và do đó, độ dịch lên cực đại được xác định cho $corr_{xy}$ cũng hợp lệ cho $corr_{xy}^2$.

Khi xác định được là cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, các tính toán vòng lặp bên trong có thể được thực hiện sử dụng độ dài từ bit dài hơn (so với độ dài từ bit hiện thời, ví dụ 32 thay vì 16 bit) để biểu diễn $enloop_y$.

Theo một phương án, khi xác định được là cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, các tính toán vòng lặp bên trong được thực hiện với độ dài từ bit dài hơn (so với độ dài từ bit hiện thời), cũng biểu diễn trị số tương quan trong vòng lặp được tích lũy, $corr_{xy}^2$, trong vòng lặp bên trong. Điều này được minh họa, ví dụ, trên Fig.3, trong hoạt động 305. Nghĩa là, độ dài từ bit được xác định cho trị số năng lượng $enloop_y$ cũng có thể được áp dụng cho $corr_{xy}^2$.

Như được đề cập trên đây, tốt hơn là tránh thực hiện phép chia của Phương trình 8 trong vòng lặp tìm kiếm chiều bên trong để cộng xung đơn vị. Do đó, sự nhân chéo có thể được thực hiện, như được minh họa trong Phương trình 10. Nghĩa là, vị trí, n_{best}, trong y để cộng xung đơn vị, có thể được xác định nhờ ước lượng nhân chéo, cho mỗi

vị trí n trong y, của trị số năng lượng và tương quan cho n hiện thời; và tương quan "tốt nhất cho đến hiện tại", BestCorrSq, và trị số năng lượng "tốt nhất cho đến hiện tại" bestEn, được lưu từ các trị số trước của n, như:

$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$

trong đó

$$\left.\begin{array}{c} n_{best} = n \\ bestEn = enloop_y \\ BestCorrSq = corr_{xy}^2 \end{array}\right\} \text{khi } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

Vị trí n_{best} có thể được gọi là vị trí "tốt nhất" trong y để cộng xung đơn vị. Cần lưu ý là "≥" có thể được sử dụng trong các biểu thức trên đây thay cho ">". Tuy nhiên, ">", nghĩa là "lớn hơn so với" có thể được ưu tiên khi cố gắng giữ phí tổn tính toán là thấp nhất có thể, ví dụ liên quan đến số chu kỳ.

Việc thực hiện phương pháp theo phương án bất kỳ trong số các phương án được mô tả trên đây cho phép sự nhân chéo này được thực hiện theo cách có hiệu quả (ví dụ nhờ không sử dụng độ chính xác cao hơn so với cần trên thực tế).

Các phương án thực hiện

Các phương pháp và kỹ thuật được mô tả trên đây có thể được thực hiện trong bộ mã hóa hoặc bộ mã hóa-giải mã (codec), mà có thể được bao gồm trong, ví dụ, thiết bị truyền thông.

Bộ mã hóa, Fig.11a-Fig.11c

Một phương án làm ví dụ của bộ mã hóa được minh họa theo cách tổng quát trên Fig.11a. Bộ mã hóa có thể là bộ mã hóa phương tiện, được tạo kết cấu để mã hóa, ví dụ, các tín hiệu âm thanh và/hoặc viđeo. Bộ mã hóa 1100 được tạo kết cấu để thực hiện ít nhất một trong số các phương án về phương pháp được mô tả trên đây với tham khảo đến hình vẽ bất kỳ trong số Fig.1-Fig.5. Bộ mã hóa 1100 được kết hợp với cùng các dấu hiệu kỹ thuật, mục đích và ưu điểm như các phương án về phương pháp được mô tả trên đây. Theo một số phương án thực hiện, bộ mã hóa được kết hợp với các ràng buộc liên quan đến bộ nhớ và/hoặc độ phức tạp, ví dụ như khi bộ mã hóa được tạo kết cấu với

DSP có độ chính xác cố định. Bộ mã hóa sẽ được mô tả ngắn gọn để tránh việc lặp lại không cần thiết.

Bộ mã hóa có thể được thực hiện và/hoặc được mô tả như sau: Bộ mã hóa 1100 được tạo kết cấu để Lượng tử hóa vectơ hình tháp, bao gồm tìm kiếm tinh hoặc tìm kiếm hình dạng tinh, trong đó PVQ (Pyramid Vector Quantizer - Bộ lượng tử hóa vectơ hình tháp) được tạo kết cấu để lấy vectơ đích x làm đầu vào và suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong. Vectơ đầu vào x có chiều N và chuẩn-L1 K. Bộ mã hóa 1100 bao gồm hệ mạch xử lý, hoặc phương tiện xử lý 1101 và giao diện truyền thông 1102. Hệ mạch xử lý 1101 được tạo kết cấu để làm cho bộ mã hóa 1100, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị: xác định, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn, theo cách không tổn hao, biến, $enloop_y$, liên quan đến năng lượng được tích lũy của y, trong vòng lặp chiều bên trong sắp tới. Giao diện truyền thông 1102, cũng có thể được biểu thị, ví dụ, là giao diện I/O (Input/Output - Đầu vào/Đầu ra), bao gồm giao diện để gửi dữ liệu đến và nhận dữ liệu từ các thực thể hoặc môđun khác.

Hệ mạch xử lý 1101 có thể, như được minh họa trên Fig.11b, bao gồm phương tiện xử lý, như bộ xử lý 1103, ví dụ CPU, và bộ nhớ 1104 để lưu trữ hoặc giữ các lệnh. Tiếp theo, bộ nhớ sẽ bao gồm các lệnh, ví dụ dưới dạng chương trình máy tính 1105, mà khi được chạy bởi phương tiện xử lý 1103 làm cho bộ mã hóa 1100 thực hiện các hoạt động được mô tả trên đây.

Một phương án thực hiện thay thế của hệ mạch xử lý 1101 được thể hiện trên Fig.11c. Hệ mạch xử lý ở đây bao gồm bộ phận xác định 1106, được tạo kết cấu để làm cho bộ mã hóa 1100, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị: xác định, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần độ chính xác cao hơn so với được cho phép với độ dài từ bit hiện thời để biểu diễn, theo cách không tổn hao, biến, $enloop_y$, liên quan đến năng lượng được tích lũy của y, trong vòng lặp chiều bên trong sắp tới. Hệ mạch xử lý 1101 có thể bao gồm nhiều bộ phận hơn, như bộ phận tìm kiếm tinh 1107, được tạo kết cấu để làm cho bộ mã

hóa chạy vòng lặp chiều bên trong với độ dài từ bit nhất định và/hoặc độ chính xác nhất định.

Các bộ mã hóa được mô tả trên đây có thể được tạo kết cấu cho các phương án về phương pháp khác nhau được mô tả ở đây, ví dụ như để thực hiện các tính toán vòng lặp bên trong bằng cách sử dụng từ bit dài hơn biểu diễn $enloop_y$ và $corr_{xy}^2$ có thể có, khi xác định được là cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$. "Dài hơn", trong bản mô tả này chỉ dài hơn so với độ dài từ bit hiện thời hoặc ban đầu.

Bộ mã hóa 1100 có thể được giả sử là bao gồm chức năng thêm nữa, để thực hiện các chức năng của bộ mã hóa thường.

Bộ mã hóa được mô tả trên đây có thể được bao gồm trong thiết bị, như thiết bị truyền thông. Thiết bị truyền thông có thể là UE (user equipment - thiết bị người dùng) dưới dạng điện thoại di động, camera viđeo, máy ghi âm, máy tính bảng, máy tính để bàn, máy tính xách tay, bộ giải mã tín hiệu truyền hình (TV set-top box) hoặc máy chủ gia đình/cổng nối gia đình/điểm truy nhập gia đình/bộ định tuyến gia đình. Thiết bị truyền thông có thể, theo một số phương án, là thiết bị mạng truyền thông được làm thích ứng để mã hóa và/hoặc chuyển mã. Các ví dụ về các thiết bị mạng truyền thông này là các máy chủ, như các máy chủ phương tiện, máy chủ ứng dụng, bộ định tuyến, cổng nối và trạm cơ sở rađio. Thiết bị truyền thông cũng có thể được làm thích ứng để được định vị trong, nghĩa là được gắn vào trong, tàu thuyền, như tàu thủy, máy bay không người lái, máy bay và phương tiện giao thông đường bộ, như xe ô tô, xe buýt hoặc xe tải. Thiết bị được gắn như vậy thông thường sẽ thuộc về bộ phận viễn tin của phương tiện giao thông hoặc hệ thống tin học giải trí của phương tiện giao thông.

Các bước, chức năng, thủ tục, môđun, bộ phận và/hoặc các khối được mô tả ở đây có thể được thực hiện trong phần cứng bằng cách sử dụng kỹ thuật thông thường bất kỳ, như kỹ thuật mạch rời rạc hoặc mạch tích hợp, bao gồm cả hệ mạch điện tử đa năng và hệ mạch chuyên dụng.

Các ví dụ cụ thể bao gồm một hoặc nhiều bộ xử lý tín hiệu số được tạo kết cấu phù hợp và các mạch điện tử đã biết khác, ví dụ các cổng lôgic rời rạc được nối với nhau để thực hiện chức năng chuyên dụng, hoặc các ASIC (Application Specific Integrated Circuit - Mạch tích hợp chuyên dụng).

Theo cách khác, ít nhất một số trong các bước, chức năng, thủ tục, môđun, bộ phận và/hoặc các khối được mô tả trên đây có thể được thực hiện trong phần mềm như chương trình máy tính để chạy bởi hệ mạch xử lý phù hợp bao gồm một hoặc nhiều bộ phận xử lý. Phần mềm có thể được mang bởi vật mang, như tín hiệu điện tử, tín hiệu quang học, tín hiệu rađio, hoặc vật ghi đọc được bởi máy tính trước khi và/hoặc trong khi sử dụng chương trình máy tính trong thiết bị truyền thông.

Một hoặc nhiều biểu đồ tiến trình được thể hiện trong bản mô tả này có thể được coi là một hoặc nhiều biểu đồ tiến trình máy tính, khi được thực hiện bởi một hoặc nhiều bộ xử lý. Thiết bị tương ứng có thể được xác định bởi một nhóm các môđun chức năng, trong đó mỗi bước được thực hiện bởi bộ xử lý tương ứng với môđun chức năng. Trong trường hợp này, các môđun chức năng được thực hiện như chương trình máy tính chạy trên bộ xử lý. Cần hiểu là các môđun chức năng không cần phải tương ứng với các môđun phần mềm thực tế.

Các ví dụ về hệ mạch xử lý bao gồm, nhưng không giới hạn vào, một hoặc nhiều bộ vi xử lý, một hoặc nhiều DSP (Digital Signal Processor - Bộ xử lý tín hiệu số), một hoặc nhiều CPU (Central Processing Unit - Bộ phận xử lý trung tâm), và/hoặc hệ mạch lôgic lập trình được phù hợp bất kỳ như một hoặc nhiều FPGA (Field Programmable Gate Array - Mảng cổng lập trình được theo trường), hoặc một hoặc nhiều PLC (Programmable Logic Controller - Bộ điều khiển lôgic lập trình được). Nghĩa là, các bộ phận hoặc các môđun trong các cách bố trí trong các thiết bị khác nhau được mô tả trên đây có thể được thực hiện bởi kết hợp của các mạch dạng tương tự và số, và/hoặc một hoặc nhiều bộ xử lý được cấu hình với phần mềm và/hoặc phần sụn, ví dụ được lưu trong bộ nhớ. Một hoặc nhiều trong số các bộ xử lý này, cũng như phần cứng dạng số khác, có thể được bao gồm trong ASIC (application-specific integrated circuitry - hệ mạch tích hợp chuyên dụng) đơn, hoặc một số bộ xử lý và phần cứng dạng số khác có thể được phân bố trong một số thành phần riêng rẽ, dù được đóng gói riêng rẽ hay được lắp ráp vào trong SoC (system-on-a-chip - hệ thống trên chip).

Cũng cần hiểu là có thể có khả năng tái sử dụng các khả năng xử lý tổng quát của thiết bị hoặc bộ phận thông thường bất kỳ trong đó thực hiện kỹ thuật được đề xuất.

Cũng có thể có khả năng tái sử dụng phần mềm hiện có, ví dụ nhờ lập trình lại phần mềm hiện có hoặc nhờ bổ sung các thành phần phần mềm mới.

Các phương án làm ví dụ thêm nữa

Được biểu diễn theo cách hơi khác đi, phần bộc lộ này đề cập đến, ví dụ, các khía cạnh và phương án sau.

Một trong số các khía cạnh là bộ mã hóa/bộ mã hóa-giải mã, trong đó bộ mã hóa/bộ mã hóa-giải mã được tạo kết cấu để thực hiện một, nhiều hơn một hoặc thậm chí là tất cả trong số các bước sau, được minh họa, ví dụ, trên Fig.5-Fig.6:

- xác định, tính toán hoặc thu được trị số tuyệt đối cực đại ($xabs_{max}$) của vectơ đích đầu vào (x(n)), ví dụ theo phương trình 11 và phương trình 12 trên đây và như được minh họa, ví dụ, bởi bước S1 trên Fig.5 theo một phương án,
- xác định, tính toán hoặc thu được độ dịch lên có thể có của trị số tương quan dựa ít nhất trên trị số tuyệt đối cực đại ($xabs_{max}$), ví dụ nhờ tính toán độ dịch lên có thể có của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo trong từ 32 bit có dấu thông qua phương trình 19 dưới đây và được minh họa với bước S2 trên Fig.5 theo một phương án,
- nếu số lượng các xung đơn vị cuối cùng (K) sẽ kết thúc cao hơn so với ngưỡng (t_p), mà ví dụ có thể là 127 xung đơn vị, xác định, ví dụ theo dõi/lưu trữ, trị số/thông tin biên độ xung cực đại ($maxamp_y$) được tính toán ví dụ theo phương trình 26 trên đây của vectơ (y(n)), mà có thể được xác định theo phương trình 13 và phương trình 14 trên đây, và
    - xác định/tính toán/quyết định/lựa chọn dựa trên biên độ xung cực đại được lưu trữ, ví dụ thông qua sự tính toán phù hợp với phương trình 22 và phương trình 23 dưới đây và như được minh họa bởi bước S3 trên Fig.6, nếu cần hoặc sẽ sử dụng nhiều hơn độ dài từ nhất định, ví dụ nhiều hơn từ 16 bit có dấu hoặc nhiều hơn từ 32 bit có dấu, để biểu diễn năng lượng trong vòng lặp mà không mất, hoặc về cơ bản là mất, thông tin năng lượng bất kỳ,

- biểu diễn số hạng/thông số/trị số bình phương tương quan tốt nhất và số hạng/thông số/trị số năng lượng được tích lũy tốt nhất bởi nhiều hơn độ dài từ nhất định, ví dụ các từ 32 bit hoặc các từ 64 bit, nếu cần nhiều hơn độ dài từ nhất định, và
- nếu cần ít hơn so với độ dài từ nhất định, thì chạy vòng lặp thứ nhất,
- nếu cần nhiều hơn độ dài từ nhất định, thì chạy vòng lặp thay thế thứ hai với số hạng năng lượng được tích lũy "tốt nhất cho đến hiện tại" (gần tối ưu) và số hạng bình phương tương quan tốt nhất được biểu diễn bởi các từ có độ dài nhiều hơn độ dài từ nhất định.

Vòng lặp thứ hai có thể là vòng lặp xung đơn vị có độ chính xác cao hơn và có độ chính xác cao chuyên sâu hơn về mặt tính toán so với vòng lặp thứ nhất có độ chính xác thấp hơn (nghĩa là so với vòng lặp thứ hai). Sự lựa chọn dựa trên năng lượng được tích lũy trong vòng lặp của độ chính xác của vòng lặp bên trong có hiệu quả là các vectơ phụ đích mà có mức đỉnh cao, hoặc có độ chi tiết rất tinh ($\boldsymbol{K}$ cuối cùng cao) sẽ hoặc có thể đang sử dụng thường xuyên hơn vòng lặp có độ chính xác cao hơn và nhiều chu kỳ hơn, trong khi các vectơ phụ không có đỉnh hoặc độ chi tiết xung thấp sẽ hoặc có thể sử dụng thường xuyên hơn vòng lặp có độ chính xác thấp hơn và ít chu kỳ hơn.

Một khía cạnh đề cập đến thiết bị truyền thông 1, được minh họa trên Fig.9, thiết bị này bao gồm bộ mã hóa hoặc bộ mã hóa-giải mã (codec) 2 để mã hóa viđeo hoặc âm thanh, ví dụ bộ mã hóa EVS.

Bộ mã hóa hoặc bộ mã hóa-giải mã có thể được thực hiện hoàn toàn hoặc một phần như DSP được định vị trong thiết bị truyền thông. Theo một phương án thứ nhất, bộ mã hóa/bộ mã hóa-giải mã được tạo kết cấu để thực hiện sự tìm kiếm hình dạng PVQ dựa trên vectơ phụ đích (x(n)), số xung đơn vị hữu hạn (K), trị số chiều vectơ phụ (N) của vectơ phụ đích và tùy chọn là cả một hoặc nhiều trị số độ khuếch đại (g_{sub}). Bộ mã hóa hoặc bộ mã hóa-giải mã cũng có thể được tạo kết cấu để thực hiện sự tách băng PVQ, và trong trường hợp như vậy sự tìm kiếm hình dạng PVQ cũng sẽ dựa trên số/trị số của các vectơ phụ của băng (N_S) và độ khuếch đại lớn nhất của vectơ khuếch đại $\boldsymbol{G}$, ($\boldsymbol{g}_{\max} = \max(\boldsymbol{G}) = \max(g_o \ldots g_{(Ns-1)})$). Bộ mã hóa hoặc bộ mã hóa-giải mã còn được tạo kết cấu để đưa ra từ sự tìm kiếm hình dạng PVQ vectơ nguyên (y) và/hoặc vectơ phụ hình

dạng $x_q(n)$ để được sử dụng bởi bộ mã hóa để đánh chỉ số PVQ. Vectơ nguyên (y) bao gồm các trị số thành phần và có cùng độ dài với trị số chiều vectơ phụ (N) và tổng tuyệt đối của tất cả trị số thành phần bằng số xung đơn vị (K).

Bộ mã hóa/bộ mã hóa-giải mã/thiết bị truyền thông được tạo kết cấu để thực hiện sự tìm kiếm hình dạng PVQ, trong đó bộ mã hóa/bộ mã hóa-giải mã/thiết bị truyền thông được tạo kết cấu để:

- xác định, tính toán hoặc thu được (S1, S23) trị số tuyệt đối cực đại ($xabs_{max}$) của vectơ (đích) đầu vào (x(n)), ví dụ theo phương trình 11 và phương trình 12 trên đây,
- xác định, tính toán hoặc thu được (S2,S28) độ dịch lên có thể có của trị số tương quan dựa ít nhất trên trị số tuyệt đối cực đại ($xabs_{max}$), ví dụ nhờ tính toán độ dịch lên có thể có của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo trong từ 32 bit có dấu thông qua phương trình 19 trên đây,
- nếu số lượng các xung đơn vị cuối cùng (K) sẽ kết thúc cao hơn so với ngưỡng (t_p), mà ví dụ có thể là 127 xung đơn vị, thì theo dõi/lưu trữ (S30) trị số/thông tin biên độ xung cực đại ($maxamp_y$) được tính toán ví dụ theo phương trình 26 trên đây của vectơ (y(n)), mà có thể được xác định theo phương trình 13 và phương trình 14 trên đây, và
    - xác định/tính toán/quyết định/lựa chọn (S3, S32) dựa trên biên độ xung cực đại được lưu trữ, ví dụ thông qua sự tính toán phù hợp với phương trình 22 và phương trình 23 trên đây, nếu cần hoặc sẽ sử dụng nhiều hơn độ dài từ nhất định, ví dụ nhiều hơn từ 16 bit có dấu hoặc nhiều hơn từ 32 bit có dấu, để biểu diễn năng lượng trong vòng lặp,
    - biểu diễn (S34) số hạng/thông số/trị số bình phương tương quan tốt nhất và số hạng/thông số/trị số năng lượng được tích lũy tốt nhất bởi nhiều hơn độ dài từ nhất định, ví dụ các từ 32 bit hoặc các từ 64 bit, nếu cần nhiều hơn độ dài từ nhất định, và
    - nếu xác định được là ít hơn so với độ dài từ nhất định, thì chạy (S33) vòng lặp thứ nhất,

- nếu xác định được là nhiều hơn độ dài từ nhất định, thì chạy (S35) vòng lặp thay thế thứ hai với số hạng năng lượng được tích lũy tốt nhất và số hạng bình phương tương quan tốt nhất được biểu diễn bởi các từ có độ dài nhiều hơn độ dài từ nhất định.

Sự tìm kiếm hình dạng PVQ trên đây, mà có thể là sự tìm kiếm hình dạng PVQ có độ chính xác giới hạn, theo một phương án được thực hiện bởi bộ lượng tử hóa vectơ, mà là một phần của bộ mã hóa/bộ mã hóa-giải mã và có thể được thực hiện ít nhất một phần, mà cũng có thể là hoàn toàn như bộ phận DSP, mà có thể được định vị trong hoặc được làm thích ứng để được định vị trong thiết bị truyền thông. Theo đó bộ mã hóa/bộ mã hóa-giải mã có thể được thực hiện hoàn toàn hoặc một phần như bộ phận phần cứng, ví dụ DSP hoặc FPGA (programmable-field gate array - mảng cổng lập trình được theo trường). Tuy nhiên, theo các phương án thay thế, nó có thể được thực hiện với sự trợ giúp của bộ xử lý đa năng và chương trình máy tính mã hóa-giải mã mà khi được chạy trên bộ xử lý đa năng làm cho thiết bị truyền thông thực hiện một hoặc nhiều trong số các bước được đề cập trong đoạn trên đây. Bộ xử lý cũng có thể là bộ xử lý RISC (Reduced Instruction Set Computing - Tính toán bằng tập lệnh rút gọn).

Một khía cạnh khác của phần bộc lộ ở đây là, như được biểu thị trong đoạn trên đây, chương trình máy tính 6 được minh họa trên Fig.10 và trong đó một phương án được bộc lộ hoàn toàn bởi ví dụ mã C ANSI trong phụ lục 1 dưới đây, như bộ mã hóa chương trình máy tính hoặc chương trình máy tính mã hóa-giải mã, bao gồm mã có thể đọc được bởi máy tính, mà khi được chạy trên bộ xử lý/bộ phận xử lý 4 của thiết bị truyền thông 1 làm cho thiết bị truyền thông thực hiện một hoặc nhiều trong số các bước được đề cập cùng với phương pháp trong đoạn dưới đây hoặc bước bất kỳ trong số các bước được đề cập cùng với Fig.7.

Một khía cạnh khác nữa là phương pháp tìm kiếm hình dạng PVQ được thực hiện bởi thiết bị truyền thông/bộ mã hóa-giải mã/bộ mã hóa, trong đó phương pháp bao gồm một hoặc nhiều trong số các bước sau:

- xác định, tính toán hoặc thu được (S1) trị số tuyệt đối cực đại ($xabs_{max}$) của vectơ (đích) đầu vào (x(n)), ví dụ theo phương trình 11 và phương trình 12 trên đây,

- xác định, tính toán hoặc thu được (S2, S28) độ dịch lên có thể có của trị số tương quan dựa ít nhất trên trị số tuyệt đối cực đại ($xabs_{max}$), ví dụ nhờ tính toán độ dịch lên có thể có của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo trong từ 32 bit có dấu thông qua phương trình 19 trên đây,
- nếu số lượng các xung đơn vị cuối cùng (K) sẽ kết thúc cao hơn so với ngưỡng (t_p), mà ví dụ có thể là 127 xung đơn vị, thì theo dõi/lưu trữ trị số/thông tin biên độ xung cực đại ($maxamp_y$) được tính toán ví dụ theo phương trình 26 trên đây của vectơ (y(n)), mà có thể được xác định theo phương trình 13 và phương trình 14 trên đây, và
    - xác định/tính toán/quyết định/lựa chọn (S3) dựa trên biên độ xung cực đại được lưu trữ, ví dụ thông qua sự tính toán phù hợp với phương trình 22 và phương trình 23 trên đây, nếu cần hoặc sẽ sử dụng nhiều hơn độ dài từ nhất định, ví dụ nhiều hơn từ 16 bit có dấu hoặc nhiều hơn từ 32 bit có dấu, để biểu diễn năng lượng trong vòng lặp,
    - biểu diễn số hạng/thông số/trị số bình phương tương quan tốt nhất và số hạng/thông số/trị số năng lượng được tích lũy tốt nhất bởi nhiều hơn độ dài từ nhất định, ví dụ các từ 32 bit hoặc các từ 64 bit, nếu cần nhiều hơn độ dài từ nhất định, và
    - nếu xác định được là ít hơn so với độ dài từ nhất định, thì chạy vòng lặp thứ nhất,
    - nếu xác định được là nhiều hơn độ dài từ nhất định, thì chạy vòng lặp thay thế thứ hai với số hạng năng lượng được tích lũy tốt nhất và số hạng bình phương tương quan tốt nhất được biểu diễn bởi các từ có độ dài nhiều hơn độ dài từ nhất định.

Thiết bị truyền thông có thể là UE (user equipment - thiết bị người dùng) dưới dạng điện thoại di động, camera viđeo, máy ghi âm, máy tính bảng, máy tính để bàn, máy tính xách tay, bộ giải mã tín hiệu truyền hình hoặc máy chủ gia đình/cổng nối gia đình/điểm truy nhập gia đình/bộ định tuyến gia đình, v.v. như được xác định trên đây.

Một khía cạnh khác nữa là vật ghi đọc được bởi máy tính 5 (xem Fig.10) trên đó lưu trữ chương trình máy tính theo phương án bất kỳ trong số các phương án trên đây. Vật ghi đọc được bởi máy tính có thể ở dưới dạng bộ nhớ khả biến hoặc bất khả biến, ví dụ EEPROM (Electrically Erasable PROM - PROM có thể xóa bằng điện), FPGA, bộ nhớ cực nhanh (flash memory) (bao gồm ổ cứng thể rắn (Solid-state drive)), và ổ cứng.

Một phương án của thiết bị truyền thông 1 được minh họa trên Fig.9. Thiết bị truyền thông bao gồm, để thực hiện sự tìm kiếm hình dạng PVQ, một, nhiều hơn một hoặc tất cả trong số các bộ phận sau:

- bộ phận xác định thứ nhất, U1, để xác định, tính toán hoặc thu được trị số tuyệt đối cực đại ($xabs_{max}$) của vectơ (đích) đầu vào (x(n)), ví dụ theo phương trình 11 và phương trình 12 trên đây,
- bộ phận xác định thứ hai, U2, để xác định, tính toán hoặc thu được độ dịch lên có thể có của trị số tương quan dựa ít nhất trên trị số tuyệt đối cực đại ($xabs_{max}$), ví dụ nhờ tính toán độ dịch lên có thể có của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo trong từ 32 bit có dấu thông qua phương trình 19 trên đây,
- bộ phận lưu trữ, U3, để theo dõi/lưu trữ trị số/thông tin biên độ xung cực đại ($maxamp_y$) được tính toán ví dụ theo phương trình 26 trên đây của vectơ (y(n)), mà có thể được xác định theo phương trình 13 và phương trình 14 trên đây, nếu số lượng các xung đơn vị cuối cùng (K) sẽ kết thúc cao hơn so với ngưỡng (t_p),
- bộ phận lựa chọn, U4 để xác định/tính toán/quyết định/lựa chọn dựa trên biên độ xung cực đại được lưu trữ, ví dụ thông qua sự tính toán phù hợp với phương trình 22 và phương trình 23 trên đây, nếu cần hoặc sẽ sử dụng nhiều hơn độ dài từ nhất định, ví dụ nhiều hơn từ 16 bit có dấu hoặc nhiều hơn từ 32 bit có dấu, để biểu diễn năng lượng trong vòng lặp,
- bộ phận biểu diễn, U5, để tạo ra số hạng/thông số/trị số bình phương tương quan tốt nhất và số hạng/thông số/trị số năng lượng được tích lũy tốt nhất với độ dài

từ, ví dụ các từ 32 bit hoặc các từ 64 bit, nhiều hơn độ dài từ nhất định nếu bộ phận lựa chọn lựa chọn nhiều hơn độ dài từ nhất định, và

- bộ phận xử lý vòng lặp bên trong, U6, để
    - o chạy vòng lặp thứ nhất, nếu bộ phận lựa chọn lựa chọn ít hơn so với độ dài từ nhất định, và
    - o chạy vòng lặp thay thế thứ hai với số hạng năng lượng được tích lũy tốt nhất và số hạng bình phương tương quan tốt nhất được biểu diễn bởi các từ có độ dài nhiều hơn độ dài từ nhất định, nếu xác định được là nhiều hơn độ dài từ nhất định.

Các bộ phận được đề cập trong đoạn trên đây có thể được bao gồm trong bộ mã hóa-giải mã/bộ mã hóa 2 dưới dạng DSP trong bộ phận truyền thông và hơn nữa có thể được bao gồm trong bộ lượng tử hóa vectơ phần cứng của DSP. Theo một phương án thay thế, tất cả các bộ phận trong đoạn trên đây được thực hiện trong thiết bị truyền thông như phần mềm.

Như được minh họa thêm nữa trên Fig.9, thiết bị truyền thông 1 cũng có thể còn bao gồm các bộ phận liên quan đến bộ mã hóa/bộ mã hóa-giải mã và cụ thể là các bộ phận liên quan đến lượng tử hóa vectơ và tìm kiếm hình dạng PVQ. Các bộ phận như vậy được tạo kết cấu để cho phép các tìm kiếm hình dạng theo phần mô tả và các hình vẽ nằm trong bản mô tả này. Các bộ phận làm ví dụ được minh họa trên Fig.9 là:

- bộ phận tách băng PVQ U7 để thực hiện bước S21 tùy chọn được mô tả cùng với Fig.7,

- bộ phận so sánh U8 để thực hiện bước S24 được mô tả cùng với Fig.7 dưới đây,

- bộ phận tạo ra vectơ PVQ U9 để thực hiện bước S25 được mô tả dưới đây,

- bộ phận tạo ra điểm bắt đầu U10 để thực hiện bước S26 được mô tả dưới đây,

- bộ phận tính toán thông số U11 để thực hiện bước S27 được mô tả dưới đây,

- bộ phận cấp phát bit U12 để, ví dụ, cung cấp K và N cho sự tìm kiếm hình dạng, và

- bộ phận đánh chỉ số PVQ U13, mà có thể được thấy như bộ nhận của đầu ra từ sự tìm kiếm hình dạng PVQ được bộc lộ ở đây,

- bộ phận chuẩn hóa U14 để thực hiện bước S36 được mô tả dưới đây, và

- bộ phận đầu ra U15 để thực hiện bước S37 được mô tả dưới đây.

Trong trường hợp của phương án thực hiện phần mềm trong thiết bị truyền thông, một phương án của thiết bị truyền thông 1 có thể được xác định là thiết bị truyền thông bao gồm bộ xử lý 4 và vật lưu trữ chương trình máy tính 5 dưới dạng bộ nhớ, bộ nhớ này chứa các lệnh có thể chạy được bởi bộ xử lý này, nhờ đó thiết bị truyền thông hoạt động để thực hiện một, nhiều hơn một hoặc tất cả trong số các việc sau:

- xác định, tính toán hoặc thu được trị số tuyệt đối cực đại ($xabs_{max}$) của vectơ (đích) đầu vào (x(n)), ví dụ theo phương trình 11 và phương trình 12 trên đây,
- xác định, tính toán hoặc thu được độ dịch lên có thể có của trị số tương quan dựa ít nhất trên trị số tuyệt đối cực đại ($xabs_{max}$), ví dụ nhờ tính toán độ dịch lên có thể có của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo trong từ 32 bit có dấu thông qua phương trình 19 trên đây,
- nếu số lượng các xung đơn vị cuối cùng (K) sẽ kết thúc cao hơn so với ngưỡng (t_p), mà ví dụ có thể là 127 xung đơn vị, thì theo dõi/lưu trữ trị số/thông tin biên độ xung cực đại ($maxamp_y$) được tính toán ví dụ theo phương trình 26 trên đây của vectơ (y(n)), mà có thể được xác định theo phương trình 13 và phương trình 14 trên đây, và
    - xác định/tính toán/quyết định/lựa chọn dựa trên biên độ xung cực đại được lưu trữ, ví dụ thông qua sự tính toán phù hợp với phương trình 22 và phương trình 23 trên đây, nếu cần hoặc sẽ sử dụng nhiều hơn độ dài từ nhất định, ví dụ nhiều hơn từ 16 bit có dấu hoặc nhiều hơn từ 32 bit có dấu, để biểu diễn năng lượng trong vòng lặp,
    - biểu diễn số hạng/thông số/trị số bình phương tương quan tốt nhất và số hạng/thông số/trị số năng lượng được tích lũy tốt nhất bởi nhiều hơn độ dài từ nhất định, ví dụ các từ 32 bit hoặc các từ 64 bit, nếu cần nhiều hơn độ dài từ nhất định, và

- nếu xác định được là ít hơn so với độ dài từ nhất định, thì chạy vòng lặp thứ nhất,
- nếu xác định được là nhiều hơn độ dài từ nhất định, thì chạy vòng lặp thay thế thứ hai với số hạng năng lượng được tích lũy tốt nhất và số hạng bình phương tương quan tốt nhất được biểu diễn bởi các từ có độ dài nhiều hơn độ dài từ nhất định.

Để minh họa thêm các khía cạnh và phương án, một số chúng sẽ được mô tả trong phần sau đây cùng với Fig.7-Fig.8.

Fig.8 đưa ra tổng quan về bên truyền của EVS 3GPP mới nổi, bao gồm bộ mã hóa EVS 3, mà ở đây được bao gồm trong thiết bị truyền thông 1.

Fig.7 minh họa một số bước phương pháp theo cách thay thế để mô tả một số phương án liên quan đến các phương án được minh họa trên Fig.5-Fig.6. Mặc dù một số trong các bước được đề cập liên quan đến Fig.7 có thể nói là được thực hiện cùng với sự tìm kiếm hình dạng PVQ, nhưng cũng rõ ràng là một số trong các bước cũng có thể nói là được thực hiện trước sự tìm kiếm hình dạng PVQ. Trong bước thứ nhất S21 tùy chọn, sự tách băng PVQ được thực hiện.

Các vectơ phụ đích hình dạng, tùy chọn là từ bước S21, được nhận trong bước thứ hai S22, trong đó, phụ thuộc vào phương án, cũng có thể nhận được g_{sub}, g_{max} và N_s.

Trong bước thứ ba S23, mà tương ứng với bước S1 trên Fig.5, trị số tuyệt đối cực đại của vectơ đích được xác định, ví dụ nhờ tính toán trước tiên trị số tuyệt đối của vectơ phụ x(n) của vectơ đích và sau đó lựa chọn trị số tuyệt đối lớn nhất của vectơ phụ.

Trong bước thứ tư S24 tùy chọn, xác định được liệu trị số của vectơ đích thấp hơn hoặc bằng ngưỡng thứ nhất. Ngưỡng được thiết lập thành các vectơ đích "lọc ra" (filter out) mà được coi là có các trị số năng lượng rất thấp. Như được giải thích trên đây, ngưỡng có thể được thiết lập để bằng không (zero) theo một phương án. Trong bước thứ tư này, cũng có thể quyết định được nếu độ khuếch đại vectơ phụ thấp hơn hoặc bằng ngưỡng thứ hai. Theo một phương án ngưỡng thứ hai được thiết lập thành không (zero), nhưng có thể, theo các phương án khác, được thiết lập thành epxilon máy (Machine Epsilon) phụ thuộc vào độ chính xác được sử dụng cho các từ được xử lý.

Nếu, trong bước thứ tư S24, xác định được là vectơ đích thấp hơn hoặc bằng ngưỡng thứ nhất và/hoặc độ khuếch đại vectơ phụ thấp hơn hoặc bằng ngưỡng thứ hai, thì vectơ PVQ được tạo ra trong bước thứ năm S25 tùy chọn. Việc tạo ra, theo một phương án, là được tạo ra theo cách tất định nhờ gán một nửa của K xung đơn vị vào vị trí thứ nhất ($y[0]=\left\lfloor \frac{K}{2} \right\rfloor$), và các xung đơn vị còn lại vào vị trí cuối cùng ($y[N-1]=y[N-1]+(K-y[0])$). Bước này cùng với bước thứ tư S24 có thể được thấy là bỏ qua toàn bộ sự tìm kiếm hình dạng PVQ thực tế, nhưng cũng có thể được thấy như thủ tục con trong ngữ cảnh của thủ tục tìm kiếm hình dạng PVQ tổng quát.

Trong bước thứ sáu S26 tùy chọn, trị số ban đầu (điểm bắt đầu) cho y, y_start, được thiết lập trong sự tìm kiếm hình dạng PVQ để theo sau, trong đó trị số ban đầu phụ thuộc vào tỷ lệ giữa K và N. Nếu tỷ lệ lớn hơn so với trị số ngưỡng thứ ba, mà có thể là 0,5 xung đơn vị cho mỗi hệ số, thì phép chiếu thứ nhất thành hình tháp phụ K-1 được sử dụng làm vector_y_start ban đầu trong bước theo sau. Phép chiếu thứ nhất có thể được tính toán như trong phương trình 13 và phương trình 14 trên đây. Nếu thấp hơn so với ngưỡng thứ ba, thì vectơ ban đầu y_start được quyết định là khởi đầu từ 0 xung đơn vị được đặt trước.

Khi chuẩn bị cho các bước tìm kiếm hình dạng PVQ tiếp theo, tất cả các trị số vectơ ban đầu trong y_start được thiết lập thành không (zero) trong bước thứ bảy S27. Trong bước này thông số thứ nhất, ở đây được gọi là số xung đơn vị được tích lũy, $pulse_{tot}$, và thông số thứ hai, ở đây là tương quan được tích lũy, $corr_{xy}(pulse_{tot})$, và thông số thứ ba, ở đây được gọi là năng lượng được tích lũy $energy_y(pulse_{tot})$ cho điểm bắt đầu được tính toán, ví dụ, tương ứng theo các phương trình 15-17. Thông số thứ tư, ở đây được gọi là $enloop_y(pulse_{tot})$ cũng có thể được tính toán trong bước này theo phương trình 18 trên đây.

Trong bước thứ tám S28, sự tìm kiếm hình dạng PVQ được bắt đầu, hoặc theo cách thay thế để xem xét nó, phần tìm kiếm tinh thứ hai của sự tìm kiếm hình dạng PVQ được bắt đầu cho các xung đơn vị còn lại lên đến K với sự trợ giúp của K, N, X_abs, max_xabs, và y thu được, được xác định hoặc được tính toán trước đó, và theo một số phương án là cả g_{sub}, g_{max} và N_S. Các bước chi tiết của một số phương án của sự tìm

kiếm tinh này được minh họa đầy đủ nhờ ví dụ Fig.6, nhưng có thể nhấn mạnh là, theo một số phương án, sự tìm kiếm tinh bao gồm xác định thông số/trị số thứ năm, ở đây được gọi là độ dịch lên của trị số tương quan, $corr_{upshift,}$ được tính toán cho ít nhất là một số, và theo một số phương án là tất cả, các xung đơn vị mà vòng lặp bên trong hoặc tìm kiếm tinh được thực hiện cho các xung đơn vị này. Theo một số phương án, độ dịch lên có thể có của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tiếp theo trong từ 32 bit có dấu được tính toán dựa trên phương trình 19 trên đây và tiếp theo $corr_{upshift}$ được sử dụng làm đầu vào sự tính toán của trị số tương quan $corr_{xy}$ trong phương trình 20.

Trong bước thứ chín S29, mà có thể nói là một phần sự tìm kiếm hình dạng PVQ tinh, xác định được liệu số lượng các xung đơn vị cuối cùng K sẽ kết thúc cao hơn so với ngưỡng thứ ba, t_p, cho số lượng các xung đơn vị cuối cùng. Nếu là trường hợp này, thì trong bước thứ mười S30, biên độ xung cực đại $maxamp_y$ được lưu trữ.

Trong bước thứ mười một S31, thông số thứ sáu, en_{margin}, được tính toán theo, ví dụ, phương trình 22.

Trong bước thứ mười hai S32, thông số thứ sáu được so sánh với trị số ngưỡng thứ tư, mà tương ứng với độ dài từ nhất định.

Nếu câu trả lời YES (CÓ) (trong S32 trên Fig7) hoặc False (Sai) (trong ví dụ mã Ansi trong Phụ lục 1) là, nghĩa là en_{margin} trong phương trình/quyết định 23 làm ví dụ lớn hơn hoặc bằng ngưỡng thứ tư "16", thì trong bước S33, chạy vòng lặp nhanh hơn và "thô hơn" thứ nhất so với trong vòng lặp thứ hai của sự tìm kiếm tinh. Các phương án của vòng lặp thứ nhất được thể hiện trên, ví dụ, Fig.6.

Nếu câu trả lời là "No" (Không) (trong S32 trên Fig7) hoặc True (Đúng) (trong ví dụ mã Ansi trong Phụ lục 1), nghĩa là en_{margin} trong phương trình/quyết định 23 làm ví dụ nhỏ hơn so với "16", thì trong bước thứ mười bốn S34, thông số thứ bảy, số hạng/thông số/trị số bình phương tương quan tốt nhất, và thông số thứ tám, số hạng/thông số/trị số năng lượng được tích lũy tốt nhất, được tạo ra/biến đổi để trở thành nhiều hơn độ dài từ nhất định, mà sau đó trong bước thứ mười lăm S35 được sử dụng trong vòng lặp bên trong chi tiết hơn thứ hai của sự tìm kiếm tinh. Các phương án của vòng lặp thứ hai được thể hiện chi tiết hơn trên, ví dụ, Fig.6.

Trong bước thứ mười sáu S36, ít nhất là mỗi thành phần vectơ phụ PVQ không bằng không (non-zero) được gán dấu thích hợp của nó và vectơ được chuẩn hóa L2 thành năng lượng đơn vị. Nếu, theo một số phương án, băng đã được tách, thì nó được định tỷ lệ với độ khuếch đại vectơ phụ g_{sub}.

x_q được chuẩn hóa cũng có thể được xác định dựa trên phương trình 28. Thủ tục làm ví dụ cho bước này được mô tả đầy đủ hơn trên đây.

Trong bước thứ mười bảy S37, x_q được chuẩn hóa và y được đưa ra từ quy trình tìm kiếm hình dạng PVQ và được chuyển tiếp đến quy trình đánh chỉ số PVQ được bao gồm trong, ví dụ, bộ mã hóa-giải mã.

Một số ưu điểm của các phương án và khía cạnh

Dưới đây là một số ưu điểm vượt qua tình trạng kỹ thuật mà cho phép ít nhất là một số trong các khía cạnh và phương án được đề xuất trên đây.

Phương pháp/thuật toán định tỷ lệ tương quan được bộc lộ bằng cách sử dụng sự phân tích trước của tương quan cực đại được tích lũy hiện thời, cải thiện hiệu năng SNR trong trường hợp xấu nhất (cực tiểu) của việc thực hiện tìm kiếm lượng tử hóa hình dạng PVQ có độ chính xác giới hạn. Tiêu chuẩn thích ứng cho sự phân tích số dư tương quan trước yêu cầu độ phức tạp bổ sung rất dư. Hơn nữa, không yêu cầu sự chuẩn hóa trước tốn kém nào của vectơ đích *x* thành, ví dụ, năng lượng đơn vị.

Tiêu chuẩn thích ứng sử dụng việc theo dõi biên độ xung cực đại trong kết quả sơ bộ, theo sau bởi sự phân tích trước của năng lượng được tích lũy trong trường hợp xấu nhất, ví dụ cho quyết định mềm ở vòng lặp bên trong có độ chính xác 16/32 bit yêu cầu độ phức tạp tính toán bổ sung rất nhỏ và đưa ra sự cân bằng tốt trong đó độ phức tạp có thể được giữ ở mức thấp trong khi vẫn sử dụng tương quan có độ chính xác cao và phép đo năng lượng có độ chính xác cao cho các tín hiệu đầu vào có liên quan, và các tín hiệu đỉnh quan trọng theo cách chủ quan hơn nữa sẽ được gán độ chính xác cao hơn. Nói cách khác, ít nhất là một số trong các phương án và khía cạnh cải tiến chức năng của chính máy tính/bộ xử lý.

Trong các Bảng 2/3 trên trong phụ lục 2 dưới đây, người ta có thể thấy là hệ thống dựa trên PVQ làm ví dụ sử dụng phí tổn lôgic có độ chính xác thích ứng sẽ là

6,843 WMOPS, nếu người ta sử dụng độ chính xác bình phương tương quan và năng lượng 32 bit trong tất cả (**K** bất kỳ) vòng lặp tìm kiếm bên trong thì phí tổn được nâng lên thành 10,474 WMOPS.

Các chú ý cuối

Các phương án được mô tả trên đây chỉ được đưa ra làm ví dụ, và cần hiểu là sáng chế không bị giới hạn vào đó. Những người có hiểu biết trung bình trong lĩnh vực sẽ hiểu là các biến đổi, kết hợp và thay đổi khác nhau có thể được thực hiện đối với các phương án mà không lệch khỏi phạm vi bảo hộ của sáng chế. Cụ thể, các giải pháp bộ phận khác nhau theo các phương án khác nhau có thể được kết hợp trong các kết cấu khác, trong đó có thể thực hiện được về mặt kỹ thuật.

Khi sử dụng từ "bao gồm" hoặc "gồm có" cần hiểu là các từ này không có nghĩa giới hạn, nghĩa là các từ này có nghĩa là "gồm có ít nhất là".

Cũng cần lưu ý là, theo một số cách thực hiện thay thế, các chức năng/hoạt động được ghi chú trong các khối có thể xuất hiện khác với thứ tự được ghi chú trên các biểu đồ tiến trình. Ví dụ, hai khối được thể hiện liên tiếp trên thực tế có thể được thực hiện về cơ bản là đồng thời hoặc các khối đôi khi có thể được thực hiện theo thứ tự đảo ngược, phụ thuộc vào chức năng/hoạt động liên quan. Hơn nữa, chức năng của khối đã cho của các biểu đồ tiến trình và/hoặc các sơ đồ khối các thể được tách thành nhiều khối và/hoặc chức năng của hai hoặc nhiều hơn hai khối của các biểu đồ tiến trình và/hoặc các sơ đồ khối có thể được tích hợp ít nhất là một phần. Cuối cùng, các khối khác có thể được bổ sung/chèn vào giữa các khối được minh họa, và/hoặc các khối/hoạt động có thể được bỏ qua mà không lệch khỏi phạm vi bảo hộ của sáng chế.

Cần hiểu rằng sự lựa chọn các khối tương tác, cũng như việc đặt tên cho các bộ phận nằm trong bản mô tả này chỉ nhằm mục đích làm ví dụ, và các nút thích hợp để thực thi phương pháp bất kỳ trong các phương pháp được mô tả ở trên có thể được tạo kết theo nhiều cách thay thế để có thể thực thi các hoạt động của thủ tục được đề xuất.

Cũng cần chú ý rằng các bộ phận được mô tả trong bản mô tả này được coi là các thực thể lôgic và không nhất thiết phải là các thực thể vật lý riêng biệt.

Sự tham khảo đến thành phần ở dạng số ít không được dự tính có nghĩa là "một và chỉ một" trừ khi được tuyên bố rõ ràng như vậy, mà đúng hơn có nghĩa là "một hoặc nhiều". Tất cả các phương án tương đương về kết cấu và chức năng đối với các thành phần của các phương án được mô tả trên đây mà những người có hiểu biết trung bình trong lĩnh vực biết đến được kết hợp rõ ràng vào trong bản mô tả này để tham khảo và dự tính được bao trùm bởi sáng chế. Hơn nữa, không nhất thiết là thiết bị hoặc phương pháp phải giải quyết mỗi và mọi vấn đề được cố gắng giải quyết bởi sáng chế được đề xuất ở đây, nó được bao trùm bởi sáng chế.

Trong một số tình huống trong bản mô tả này, các mô tả chi tiết về các thiết bị, mạch, và phương pháp đã biết rõ được bỏ qua để không làm rối phần mô tả của sáng chế bằng chi tiết không cần thiết. Tất cả các tuyên bố trong bản mô tả này trích dẫn các nguyên lý, khía cạnh, và phương án của sáng chế, cũng như các ví dụ cụ thể của sáng chế, được dự tính để bao trùm các phương án tương đương về cả kết cấu và chức năng của sáng chế. Ngoài ra, dự tính là các phương án tương đương như vậy bao gồm cả phương án tương đương hiện đã biết cũng như phương án tương đương được phát triển trong tương lai, ví dụ các thành phần bất kỳ được phát triển mà thực hiện cùng chức năng, không quan tâm đến kết cấu.

Các thuật ngữ viết tắt

N	chiều vectơ
N_s	chiều vectơ phụ
x	vectơ đích
x_q	Vectơ hình dạng được lượng tử hóa
y_{final}	vectơ nguyên bám vào chuẩn-L1 K
K	Số lượng các xung đơn vị cuối cùng
k	số chỉ số xung đơn vị được tích lũy
n	hệ số hoặc chỉ số mẫu
i	chỉ số vectơ phụ
MDCT	Biến đổi consin rời rạc cải biến (Modified Discrete Cosine Transform)
PVQ	Bộ lượng tử hóa (Lượng tử hóa) vectơ hình tháp
WC	Trường hợp xấu nhất (Worst Case)

WMOPS	Triệu phép toán có trọng số trên mỗi giây (Weighted Million Operations Per Second)
AccEn	Năng lượng được tích lũy
ROM	Bộ nhớ chỉ đọc (Read Only Memory)
PROM	ROM chương trình (Program ROM)
SNR	Tỷ lệ tín hiệu so với nhiễu (Signal-to-Noise Ratio)
EVS	Dịch vụ tiếng nói nâng cao (Enhanced Voice Service)
3GPP	Dự án cộng tác thế hệ thứ 3 (3^{rd} Generation Partnership Project)
DSP	Bộ xử lý tín hiệu số (Digital Signal Processor)
CELT	Biến đổi ghép chồng năng lượng ràng buộc (Constrained Energy Lapped Transform)
IETF	Nhóm đặc trách kỹ thuật Internet (Internet Engineering Task Force)
MAC	Nhân-Tích lũy (Multiply-Accumulate)
ACELP	Dự đoán tuyến tính xuất phát từ mã đại số (Algebraic code-excited linear prediction)
EPS	Epxilon máy (Machine epsilon)

Phụ lục 1: Phương án thực hiện làm ví dụ trong mã C ANSI

Dưới đây là ví dụ về phương án thực hiện làm ví dụ trong mã C ANSI sử dụng (mô phỏng của DSP) 16/32 bit ảo G.191 STL 2009.

```
static

Word16 max_val_fx(              /* o  : maximum value in the input vector          */
            const Word16 *vec,  /* i  : input vector                               */
            const Word16 lvec   /* i  : length of input vector                     */
            )
{
   Word16 j,tmp;
   tmp = vec[0];    move16();
   FOR ( j=1 ; j<lvec ; j++ )
   {
      tmp = s_max(vec[j],tmp);
   }
   return tmp;
}
```

```
/*  The inner search loop for one single additional unit pulse, starting from  pulse_tot  ,
    with information about required energy precision/down scaling for the dim loop in  en_dn_shift,
    and the current max_xabs absolute value to be used for an near optimal  correlation upscaling.
    returns the index of the best positioned unit pulse in imax
*/
static
Word16 one_pulse_search(const Word16 dim,      /* vector dimension      */
                  const Word16* x_abs,   /* absolute vector values */
                  Word16* y,             /* output vector         */
                  Word16 *pulse_tot_ptr,
                  Word32* L_xy_ptr,      /* accumulated correlation */
                  Word32* L_yy_ptr,      /* accumulated energy     */
                  Word16 high_prec_active,
                  Word16 en_dn_shift,
                  Word16 max_xabs)       /* current accumulated max amplitude for pulses */
{
   Word16  i, corr_up_shift, corr_tmp, imax,  corr_sq_tmp, en_max_den, cmax_num, en_tmp;
   Word32  L_tmp_en_lc, L_tmp_corr  ;
   Word32  L_tmp_en, L_en_max_den, L_corr_sq_max, L_tmp_corr_sq;
   Word32  L_left_h, L_right_h;
   UWord32 UL_left_l, UL_right_l, UL_dummy;
```

```
#ifndef NONBE_PVQ_SEARCH_FIX
   Word16  corr_margin;
#endif
#ifdef  BE_CLEAN_PVQ_SEARCH
   Word32  L_tmp;
   UWord32 UL_tmp;
   UWord16 u_sgn;

#endif

   /* maximize correlation precision, prior to every unit pulse addition in the vector */
   corr_up_shift    =  norm_l(L_mac(*L_xy_ptr, 1, max_xabs));
   /* pre analyze worst case L_xy update in the dim  loop       , 2 ops */

   /* clean BE code, with split out low/high precision loops                              */
   /* activate low complexity en/corr search section conditionally if resulting energy is within limits */
   /* typical case for higher dimensions                                                  */

   IF( high_prec_active == 0 )
   {
      en_max_den       =  0; /*move16()*/; /* OPT: move saved by  using high_prec_active as
                                              en_max_den */
      cmax_num         = -1;  move16();    /* req. to force a 1st update for n==0   */
      FOR(i = 0; i < dim; i++)     /* FOR 3 ops  */
      {
         L_tmp_corr    = L_shl(L_mac(*L_xy_ptr,1,x_abs[i]), corr_up_shift ); /* actual in-loop target */
         corr_tmp      = round_fx(L_tmp_corr);
         corr_sq_tmp   = mult(corr_tmp, corr_tmp);  /* CorrSq, is 16bit for low complexity crossmult */

         L_tmp_en_lc   = L_mac(*L_yy_ptr, 1,y[i] );
         /*Q1 result ,  energy may span up to ~14+1(Q1)+1(sign)=16 bits,  1 op */
         /* extract_l without shift can always be used for this section as energy is guaranteed to stay in
           the lower word, 1 op */
         en_tmp     = extract_l(L_tmp_en_lc);
        /* L_shl + round_fx could also be used also but then adds an uphift cost (2-3 ops)*/

         /* 16/32 bit comparison    WC (4 +1+1 + (1+1+1) = 9                  */
         /*  corr_sq_tmp/en_tmp_den   >   cmax_num/en_max_den  */
         /*  corr_sq_tmp * en_max_den  >   cmax_num * en_tmp    */              /* IF       4 ops  */
         IF( L_msu(L_mult(corr_sq_tmp, en_max_den),cmax_num , en_tmp)  > 0)       /*          2 ops
*/
         {
            cmax_num    = corr_sq_tmp;  move16();   /* 1 op */
            en_max_den  = en_tmp;       move16();   /* 1 op */
```

```
            imax      = i;          move16();   /* 1 op */
          }
        } /* dim  */

    }
    ELSE
    {  /* High resolution section activated when vector energy is becoming high  (peaky or many
pulses)
        */
        /* BASOP operator Mpy32_32_ss used to allow higher resolution for both the CorrSq & Energy
           terms  */
        /*Performance:   close to float reference                                          */



        L_en_max_den  = L_deposit_l(0);                                          /* 1 op  */
        L_corr_sq_max = L_deposit_l(-1); /* req. to force a 1st update   */              /* 1 op  */

        FOR(i = 0; i < dim; i++)                                          /* FOR 3 ops */
        {
           L_tmp_corr      = L_shl(L_mac(*L_xy_ptr,1,x_abs[i]), corr_up_shift );/* actual inloop WC
value */
           Mpy_32_32_ss(L_tmp_corr,L_tmp_corr, &L_tmp_corr_sq, &UL_dummy);
           /* CorrSq 32 bits,4 ops  */

           L_tmp_en = L_mac(*L_yy_ptr, 1, y[i]);       /* Q1,energy may span up to sign+19 bits , 1 op
*/
           /* For highest accuracy use pairs of maximum upshifted 32x32 bit signed values          */
           /*  (L_tmp_corr_sq / L_tmp_en)    >  (L_corr_sq_max/L_en_max_den)                    */
           /*  (L_tmp_corr_sq * L_en_max_den) >  (L_corr_sq_max * L_tmp_en)                     */
           Mpy_32_32_ss( L_en_max_den, L_tmp_corr_sq, &L_left_h, &UL_left_l);             /* 4 ops */
           Mpy_32_32_ss( L_tmp_en, L_corr_sq_max, &L_right_h, &UL_right_l);               /* 4 ops */

           /* 32/64 bit comparison  WC (1+1+ 2x2+ 4 +(2+2+1) = 15                              */
           L_tmp  = L_sub(L_left_h, L_right_h);          /* high signed  word check          1 op  */

           UL_tmp = UL_subNs(UL_right_l, UL_left_l, &u_sgn);
           /* low unsigned word check, note switch due to ">=" in UL_subNs, 1 op */
           IF( ( L_tmp > 0 )  ||  ( (L_tmp == 0)  &&  (u_sgn != 0 ) ) )   /* IF  4 ops */
           {
              L_corr_sq_max =  L_tmp_corr_sq;  move32(); /* 2 ops */
              L_en_max_den  =  L_tmp_en;       move32(); /* 2 ops */
              imax          =  i;              move16(); /* 1 op  */
           }
```

```
      } /* dim loop */
   }
   /*  finally add found unit pulse contribution to past L_xy, Lyy,  for next pulse loop    */
   *L_xy_ptr = L_mac(*L_xy_ptr, x_abs[imax], 1);          /* xy_flt += xabs[imax];  Q12+1 */
   *L_yy_ptr = L_mac(*L_yy_ptr, 1, y[imax]);             /* yy_flt += y[imax];          */

   y[imax]          = add(y[imax],1);       move16();      /* Q0 added pulse            */
   (*pulse_tot_ptr)  = add((*pulse_tot_ptr) ,1);          /* increment total pulse sum   */
   return imax;

}
```

```
/*-----------------------------------------------------------------*
 * Function pvq_encode_fx()                                        *
 *                                                                 *
 * Basic PVQ search algorithm:                                     *
 *  Selective L1-norm projection to the lower ~(K-1) pyramid       *
 *  followed by an ACELP-like unit pulse addition fine search      *
 *-----------------------------------------------------------------*/
void pvq_encode_fx(
   Encoder_State_fx *st_fx,
   const Word16 *x,       /* i:   vector to quantize           Q15-3=>Q12      */
   Word16 *y,             /* o:   raw pulses  (non-scaled short) Q0            */
   Word16 *xq,            /* o:   quantized vector             Q15             */

   const short pulses,    /* i:   number of allocated pulses                   */
   const short dim,       /* i:   Length of vector  == N                       */
   const Word16 neg_gain  /* i:  -Subvector Gain   we use -negative gain in  Q15  0..1   */
)
{
   Word16  i;
   Word16 pulse_tot;
   Word16 xabs[PVQ_MAX_BAND_SIZE];
   Word16 max_xabs;
   Word32 L_xsum;
   Word32 L_proj_fac;
   Word32 L_yy, L_xy;
   Word16  max_amp_y, imax;
   Word16  k, en_margin, en_dn_shift, high_prec_active ;
   Word32 L_num, L_tmp;
   Word16 proj_fac, tmp, shift_den,shift_num,shift_delta, num,den;
   UWord16 u16_tmp;
   Word16 dim_m1;
   Word32 L_isqrt;
```

```
Word16 neg_gain_norm, shift_tot;
Word16   high_pulse_density_flag;
PvqEntry_fx entry;
```

```
/* Create absolute vector and calculate sum of abs vector  */
L_xsum  =  L_deposit_h(0);
max_xabs =  -1;              move16();

FOR( i = 0; i < dim; i++)
{
   xabs[i]   = abs_s(x[i]);              move16();         /* Q12 */
   max_xabs  = s_max(max_xabs, xabs[i] );          /* for efficient search correlation scaling */
   L_xsum    = L_mac0(L_xsum, 1, xabs[i] );                /* stay in Q12 */
   y[i]      = 0;                        move16();        /* init */
}
```

```
test();
IF( L_xsum == 0  ||  neg_gain == 0  )
{  /*  zero input  or  zero gain  case */
   /*  put a "half"-pulsesum first and last,  or all pulses in pos 0 */
   pulse_tot = pulses;       move16();
   dim_m1    = sub(dim,1);
   y[dim_m1] = 0;            move16();
   y[0]      = shr(pulses,1); move16();
   y[dim_m1] = add(y[dim_m1], sub(pulses, y[0]));  move16();
   L_yy      = L_mult(y[0],y[0]); /* L_yy needed for normalization */
   if(dim_m1 != 0) {
      L_yy = L_mac(L_yy, y[dim_m1],y[dim_m1]);  /* (single basop) */
   }
}
```

```
ELSE

{  /* regular search */
   /* proj_fac_flt = (pulses - 1)/xsum_flt; */  /* normalize to L1 norm = absolute k-1 pyramid */
  num = sub(pulses,1);
  high_pulse_density_flag = 0; move16();
  if ( sub(pulses, shr(dim, 1)) > 0 )
  {
    high_pulse_density_flag = 1;  move16();
  }

  test();
  IF( (num > 0) && (high_pulse_density_flag != 0) )
  {
```

```
    shift_den = norm_l(L_xsum);                 /* x_sum input  Q12                  */
    den       = extract_h(L_shl(L_xsum, shift_den)); /* now in Q12+shift_den        */

    L_num     = L_deposit_l(num);
    shift_num = sub(norm_l(L_num)   ,1);
    L_num     = L_shl(L_num, shift_num) ;       /* now in Q0 +shift_num -1           */
    proj_fac  = div_l(L_num,  den );            /* L_num always has to be less than den<<16 */


    shift_delta=sub(shift_num,shift_den);
    L_proj_fac = L_shl(L_deposit_l(proj_fac), sub(9, shift_delta));  /* bring  to a fixed  Q12    */
}

  pulse_tot = 0;                move16();
  L_yy = L_deposit_l(0);
  L_xy = L_deposit_l(0);

  /* Find a start position on a lower sub pyramid,  if the pulse density is larger than 0.5 */
  IF((num > 0) &&  (high_pulse_density_flag != 0) )
  {
     FOR( i = 0; i < dim ; i++) /* max 64 */
     {
        /* y_flt[i]  = (short)floor(xabs_flt[i] * proj_fac_flt); */   /* FLOAT  pyramid truncation */
        /* y[i]      =  shr(xabs[i]*L_proj_fac);              */   /* BASOP  pyramid truncation */
        Mpy_32_16_ss(L_proj_fac,xabs[i],&L_tmp,&u16_tmp);          /*Q12 *Q12  +1 */
        y[i]      = extract_l(L_shr( L_tmp, 12+12-16+1 )); move16();  /* the pyramid-"truncation" */
                                                  /* Q12 *Q12  ->  Q0 */


        pulse_tot  = add(pulse_tot, y[i]);        /* Q0                               */
        L_yy       = L_mac(L_yy, y[i], y[i]);     /* Energy, result will scale up by 2 by L_mac */
        L_xy       = L_mac(L_xy, xabs[i], y[i]);  /* Corr, Q0*Q12  +1 --> Q13         */
     }
  }


  /*  PVQ fine search loop  description          */
  /*  Run an ACELP-like full CorrSq/Energy  search */
  /*  the basic search  logic is to test adding one unit pulse to each position n in the output vector y
      and update the correlation and energy terms iteratively.

     Rxy(k,n)   = Rxy(k-1) + 1*abs(x(k,n))       % add target times unit correlation
     Ryy(k,n)   = Ryy(k-1) + 2*y(k-1,n) + 1^2    % added unit energy contribution
     RxySq(k,n) = Rxy(k,n)*Rxy(k,n)
```

minimize (-Rxy(k,n)/Sqrt(Ryy(k,n))), over n and k
(or equiv. maximize (RxySq(k,n)/Ryy(k,n)), over n and k)

RxySq(k,n) / Ryy(k,n) > bestRxySq / bestRyy ? update best candidate

bestRyy * RxySq(k,n) > bestRxySq * Ryy(k,n) ? update best candidate

Misc:
1^2 in Ryy(k,n) can be added before the dimension loop
Ryy(k,n) can optionally be pre down-scaled by 2.0 to avoid one multiplication/shift
Rxy(k,n) can dynamically scaled for each unit pulse loop to optimize RxySq.
CorrSqTerm =(Rxy(k,n))^2, should have maximum precision in a Word16/Word32 variable

Energy term:Ryy(k, n)
should not become zero, i.e. most often have enough precision to not become truncated
i.e. preferably use a 32 bit variable if more than 127 pulses are accumulated in Q1.

k=127=>log2(127^2) = 13.9774 bits, fits in Q1 signed, with Q1 the pre division by 2 works

k=255=>log2(255^2) = 15.9887 bits, i.e 1 bits scaling, fits in Q1 unsigned 16 bit value

Dimension aspects:
dim 6 and higher is always 16 bit energy safe as max pulses is 96
(for 1+31 bits short codewords)
dims 5,4,3,2 may be allocated higher than 127 pulses (or lower)
dim 1 does not really need a search, only the sign is relevant.
Currently we have KMAX=512, or more precisely for DIM 5 kmax is 238
DIM 4,3,2,1 kmax is 512
Unsigned DSP arithmetic could also be used,
but then the "startup" case/ now a negative sentinel has to be handled in a special way
and further currently UL_mac,UL_mac0,UL_msu,UL_msu0 is not available as STL unsigned basops

To make the PVQ search code work efficiently we do:
1. Always use near optimal norm_l based dynamic up-scaling of the correlation term, by analyzing the accumulated correlation sofar. (Rxy for k-1 pulses) and using a pre-analyzed maximum possible target value max_xabs.
2. Selectively use 16 or 32 bit representation of the energy term, depending on the pre-accumulated energy. further if the energy term Ryy needs more than 16 bits, we switch to a 32 bit

```
                energy representation
                and also a higher 32 bit precision for the correlation Rxy and
                increase the precision in the distortion cross-multiplication calculations
                to near 64 bits.
  */

  /* Rxy_max=(k,*)= max( L_xy + 1*max_x_abs)
    analysis needed for  near optimal L_xy normalization (corr, corrSq)   */
  /* Ryy_max(k,*) = max(L_yy(k-1) + .5  + 1*max_amp_y(k-1))
    analysis needed for energy (L_yy) precision selection when pulses > 127 ,
    ( (127^2)*2 fits in Signed Word16) */

       L_yy=L_shr(L_yy,1);   /* scale down by 2  for search loop  domain  */
       IF (sub(pulses,127)<=0 )
       {  /* LC inner loop, enters here always for dimensions 6 and higher,
             and also sometimes for dimensions 1 .. 5  */
          /* ( if  high energy precision is inactive,  max_amp_y is not needed ,
             no max_amp_y(k-1) update )          */
          FOR (k=pulse_tot; k<pulses;k++){
             L_yy = L_add(L_yy,1);  /* .5 added to energy for pulse_tot+1 */
             imax = one_pulse_search(dim, xabs,y,&pulse_tot,&L_xy,&L_yy,0, 0,max_xabs);
          }
       }
       ELSE
       { /* HC or LC+HC inner loops */
          max_amp_y = max_val_fx(y, dim); /* this loops over max 5 values  */
          /* max_amp_y from projected y is needed when pulses_sum  exceeds 127      */

          /* First section with 32 bit energy inactive,   max_amp_y kept updated though    */
          FOR( k=pulse_tot; k<128; k++){
             L_yy       = L_add(L_yy,1);                    /* .5 added */
             imax       = one_pulse_search(dim, xabs,y,&pulse_tot,&L_xy,&L_yy,0,0,max_xabs);
             max_amp_y   = s_max(max_amp_y, y[imax]);
          }

          /* Second section with higher number of pulses, 32 bit energy precision adaptively
            selected, max_amp_y kept updated              */
          FOR( k=pulse_tot; k<pulses; k++){
             L_yy       = L_add(L_yy,1);    /* .5 added         */
             en_margin   = norm_l(L_mac(L_yy,1, max_amp_y)); /*find max "addition", margin,~2
ops */
             en_dn_shift = sub(16, en_margin);  /* calc. shift to lower word */

             high_prec_active = 1;     move16();
```

```
                if( en_dn_shift <= 0 ){  /* only use 32 bit energy if actually needed */
                  high_prec_active = 0;  move16();
                }
                /* 32 bit energy and corr adaptively active,  max_amp_y kept updated */
                imax       = one_pulse_search(dim,xabs,y,&pulse_tot,&L_xy,&L_yy,high_prec_active,
                                              en_dn_shift, max_xabs);
                max_amp_y  = s_max(max_amp_y, y[imax]);
              }
            }
        L_yy = L_shl(L_yy,1);  /*  yy= yy*2.0  */  /* compensate search loop analysis energy downshift
by
                                                    1,   to make energy right for unit/inverse gain calculation
*/
    }
```

```
/* Apply unit energy (L2) normalization scaling, always at least one pulse so no div-by-zero check is
    needed */
    L_isqrt       = L_deposit_l(0);
    IF( neg_gain != 0  ){
       L_isqrt = lsqrt(L_shr(L_yy,1));      /* Note: one single gain factor  as in flt is  not computed */
    }

    shift_num     = norm_s(pulse_tot);                  /* account for max possible pulse amplitude in y,
                                                          can be used even when max_amp_y is not avail.
*/
    shift_den     = norm_s(neg_gain);                    /* account for gain downscaling shift   */
    neg_gain_norm = shl(neg_gain, shift_den);          /* up to 10 dB loss without this norm   */
    shift_tot     = sub(add(shift_num, shift_den), 15);
    /* DSP OPT : to save average complexity a special loop can be made for the common gain==1.0
case  */
    /* DSP OPT PROM : reuse a common decoder normalization  loop                                  */

    L_isqrt = L_negate(L_isqrt);
    FOR( i = 0; i < dim; i++)
    {
       tmp = shl(y[i],shift_num);  /* upshifted abs(y[i]) used  in scaling */
       if( x[i] < 0 )
       {
          tmp = negate(tmp);                          /* apply sign */
       }
       if( y[i] != 0 )
       {
          y[i]     = shr(tmp, shift_num);   move16();  /* updates sign of y[i},~range -/+ 512),array move */
       }
```

```
    Mpy_32_16_ss( L_isqrt, tmp, &L_tmp, &u16_tmp);          /* Q31*Q(0+x)  +1        */
    Mpy_32_16_ss( L_tmp, neg_gain_norm , &L_tmp, &u16_tmp);  /* Q31*Q(0+x) *Q15 +1    */
    L_tmp   = L_shr(L_tmp, shift_tot);                      /* Q31+x                 */
    xq[i]   = round_fx(L_tmp);        move16();             /* Q15, array move       */
    L_xq[i] = L_tmp;                                        /* Q31 currently  unused */
  }
```

```
/* index the found PVQ vector into short codewords */
 entry = mpvq_encode_vec_fx(y, dim, pulses);
```

```
 /* send the short codeword(s) to the range encoder */
 rc_enc_bits_fx(st_fx,  UL_deposit_l(entry.lead_sign_ind) , 1);  /* 0 or 1 */
 IF( sub( dim, 1) != 0 )
 {
    rc_enc_uniform_fx(st_fx, entry.index, entry.size);
 }
```

```
 return;
```

Tất cả những người có hiểu biết trung bình trong lĩnh vực sẽ dễ dàng đọc được mã trên đây và không phải giải thích chi tiết hơn. Tuy nhiên, đối với những người không ở trong lĩnh vực cần đề cập là toán tử quan hệ "= =" là một toán tử mà theo ví dụ "A= =B" sẽ trả về trị số lôgic được đặt là 1 lôgic (true (đúng)) khi các trị số A và B là bằng nhau; và nếu không thì trả về 0 lôgic (false (sai)). L_mac là phép nhân-tích lũy với nghĩa là L_mac (L_v3, v1, v2) = L_v3+v1*v2.

Phụ lục 2: Các kết quả mô phỏng được lập bảng

Tình trạng mô phỏng

Các phương án của sáng chế đã được mô phỏng. Đối với tất cả các mô phỏng tìm kiếm hình dạng PVQ được thực hiện, tốc độ bit được sử dụng là 64000 bps (bit trên giây), và bộ mã hóa-giải mã (codec) được vận hành trong chế độ MDCT, với các kích thước băng phụ của hệ số MDCT ban đầu là các hệ số [8, 12, 16, 24, 32]. Các băng này có thể được chia rất kỹ thành các đoạn băng nhỏ hơn, mỗi đoạn được biểu diễn bởi vectơ phụ, bởi thuật toán tách băng PVQ. Ví dụ, băng có kích thước 8 có thể được tách thành các đoạn phụ nhỏ hơn, ví dụ "4, 4" hoặc "3,3,2", nếu nó được cấp phát đủ bit. Thông thường, mỗi băng được tách theo cách sao cho cực đại là 32 bit có thể được sử dụng để mã hóa hình dạng của mỗi vectơ phụ cuối cùng.

Theo phương án thực hiện đánh chỉ số PVQ này băng có kích thước 8 có thể có lên tới 36 xung đơn vị, đoạn phụ có kích thước 7 có thể có lên tới 53 xung đơn vị, đoạn có kích thước 6 có thể có lên tới 95 xung đơn vị, đoạn có kích thước 5 có thể có lên tới 238 xung đơn vị, đoạn có kích thước 4,3,2,1 có thể có lên tới 512 xung đơn vị. Do các đoạn ngắn hơn với số xung cao được tạo ra theo cách động nhờ tách băng, nên chúng ít thường xuyên hơn so với các kích thước vectơ phụ dài hơn. Các con số WMOPS trong các bảng kết quả dưới đây bao gồm: (tìm kiếm trước PVQ, tìm kiếm tinh PVQ, chuẩn hóa PVQ và đánh chỉ số PVQ). Các con số "% giống nhau" trong các bảng kết quả dưới đây, là số vectơ giống nhau được tìm thấy theo Thuật toán tìm kiếm hình dạng có độ chính xác giới hạn được ước lượng, so với sự tìm kiếm hình dạng PVQ dấu phẩy động không ràng buộc.

Các bảng kết quả

Xung <= 127, **Thuật toán** En{energy-bits} x CorrSq{corrSq-bits}	**SNR cực tiểu (dB)**	**Seg-SNR (dB)**	**% vectơ giống nhau**	**WMOPS trường hợp xấu nhất**	**WMOPS trung bình**	**Chú ý**
"***En16 x CorrSq16***"/"***En32 x***	4,771	188,803	99,3	6,843	5,496	16x16 luôn được sử dụng,

CorrSq32" trộn, phân tích trước max(x_abs)						WC (trường hợp xấu nhất) trong 16x16
"En16 x CorrSq16" khóa phân tích trước max(x_abs)	4,771	188,803	99,3	6,843		Không có thay đổi do năng lượng không bao giờ vượt quá 16 bit
"En16 x CorrsSq16" sử dụng phương pháp định tỷ lệ tương quan trong tình trạng kỹ thuật "OPUS", sử dụng **số** xung đơn vị được tích lũy.	-6,021	180,556	94,6	6,826	5,476	Thuật toán là kém hơn về bit (minSNR càng thấp thì lần thành công giống nhau càng ít) ở độ phức tạp rất tương tự
"En32 x CorrSq16" khóa, phân tích trước max(x_abs)	4,771	188,803	99,3	8,970	6,961	KHông nhất thiết sử dụng En32 cho các xung <=127, do năng lượng không bao giờ vượt quá độ động 16 bit
"En16 x CorrSq32" khóa, phân tích trước đầu vào max(x_abs)	190,0	190,0	100	9,386	7,248	Yêu cầu thêm 2,5 WMOPS cho 0,7 % các lần thành

						công cuối cùng
"En32 x CorrSq32" khóa, phân tích trước max(x_abs)	190,0	190,0	100	10,474	7,999	0,9 WMOPS không cần thiết tăng lên so với ***"En16xCorrSq32"*** khóa,

Bảng 1: Các kết quả cho K cuối cùng <= 127

Xung > 127 Thuật toán En{energy-bits} x CorrSq{corrSq-bits}	**minSNR (dB)**	**segSNR (dB)**	**% vectơ giống nhau**	**WMOPS trường hợp xấu nhất**	**WMOPS trung bình**	**Chú ý**
"***En16 x CorrSq16***"/ "***En32 x CorrSq32***" được điều khiển ***AccEn*** trộn , phân tích trước đầu vào, độ chính xác được điều khiển năng lượng tích lũy	32,686	160,316	80,4%	6,843 (WC vẫn từ các đoạn 16x16)	5,496	Giải pháp đủ tốt WC vẫn cho 16x16, WC không tăng
"***En16 x CorrSq16***"/ "***En16 x CorrSq32***" được điều khiển	32,686	130,258	59,3%	không áp dụng	không áp dụng	Thông tin năng lượng thỉnh thoảng bị

AccEn trộn phân tích trước đầu vào, độ chính xác được điều khiển năng lượng tích lũy						cắt cụt, gây ra SNR thấp
"***En16 x CorrSq16***"/ "***En32 x CorrSq16***" được điều khiển ***AccEn*** trộn phân tích trước đầu vào, độ chính xác được điều khiển năng lượng tích lũy	32,686	117,634	50,6 %	không áp dụng	không áp dụng	Thông tin tương quan có độ chính xác thấp, gây ra SNR thấp
"***En16 x CorrSq16***" khóa, , phân tích trước đầu vào,	32,686	113,629	47,8 %	không áp dụng	không áp dụng	Thông tin năng lượng thỉnh thoảng bị cắt cụt và tương quan trong thông tin có độ chính xác thấp, gây ra SNR thấp
"***En32x CorrSq16***" khóa, phân tích trước đầu vào	32,686	117,634	50,6 %	không áp dụng	không áp dụng	*Thông tin tương quan có độ chính xác thấp,*

						gây ra SNR thấp
"***En16 x CorrSq32***" khóa, phân tích trước đầu vào	40,994	159,714	78,8 %	không áp dụng	không áp dụng	*Thông tin năng lượng thỉnh thoảng bị cắt cụt, gây ra SNR thấp*
"***En32x CorrSq32***" khóa, phân tích trước đầu vào	49,697	189,773	99,8 %	*7,1*	*5,7*	*WC hiện tại trong đoạn 32x32, WC có độ phức tạp cao hơn*

Bảng 2: Các kết quả cho K > 127

YÊU CẦU BẢO HỘ

1. Phương pháp tìm kiếm hình dạng Bộ lượng tử hóa vectơ hình tháp (Pyramid Vector Quantizer - PVQ), được thực hiện bởi bộ xử lý tín hiệu số, PVQ lấy vectơ đích x làm vectơ đầu vào và suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong, phương pháp này khác biệt bởi bước:

trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị:

- xác định (203), dựa trên trị số tuyệt đối cực đại, $xabs_{max}$, của vectơ đầu vào, x, độ dịch lên có thể có, trong từ bit, của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tìm kiếm chiều bên trong tiếp theo, $corr_{xy}$, giữa vectơ đầu vào x và vectơ y.

2. Phương pháp theo điểm 1, trong đó phương pháp này còn bao gồm bước, trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị:

- xác định (102, 204), dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn, theo cách không tổn hao, biến, $enloop_y$, liên quan đến năng lượng được tích lũy của y, trong vòng lặp tìm kiếm chiều bên trong tiếp theo (103).

3. Phương pháp theo điểm 2, phương pháp này còn bao gồm bước:

khi cần nhiều hơn độ dài từ bit hiện thời (102, 204, 304, 403) để biểu diễn $enloop_y$:

- thực hiện (103, 205, 305, 404) các tính toán vòng lặp bên trong sử dụng độ dài từ bit dài hơn để biểu diễn $enloop_y$.

4. Phương pháp theo một điểm bất kỳ trong các điểm từ 2 đến 3, phương pháp này còn bao gồm bước:

khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$:

- thực hiện (305) các tính toán vòng lặp bên trong sử dụng độ dài từ bit dài hơn để biểu diễn bình phương trị số tương quan trong vòng lặp được tích lũy, $corr_{xy}^2$, giữa x và vectơ y, trong vòng lặp bên trong.

5. Phương pháp theo một điểm bất kỳ trong các điểm từ 2 đến 4, phương pháp này còn bao gồm bước,

khi không cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$:

- thực hiện (405) các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ nhất sử dụng độ dài từ bit thứ nhất để biểu diễn $enloop_y$, và:

khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$:

- thực hiện (404) các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ hai sử dụng độ dài từ bit dài hơn, so với vòng lặp cộng xung đơn vị thứ nhất, để biểu diễn $enloop_y$.

6. Phương pháp theo một điểm bất kỳ trong các điểm nêu trên, phương pháp này còn bao gồm bước,

khi không cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$:

- thực hiện (405) các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ nhất có độ chính xác nhất định, và:

khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$:

- thực hiện (404) các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ hai có độ chính xác cao hơn so với vòng lặp cộng xung đơn vị thứ nhất.

7. Phương pháp theo một điểm bất kỳ trong các điểm từ 2 đến 6, trong đó bước xác định, dựa trên $maxamp_y$, về liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$ bao gồm xác định các đặc tính của trường hợp khi, trong vòng lặp tìm kiếm chiều bên trong tiếp theo, xung đơn vị được cộng vào vị trí trong y mà được kết hợp với $maxamp_y$.

8. Phương pháp theo một điểm bất kỳ trong các điểm từ 2 đến 7, phương pháp này còn bao gồm bước:

trong vòng lặp tìm kiếm chiều bên trong để cộng xung đơn vị:

- xác định vị trí, n_{best}, trong y để cộng xung đơn vị nhờ ước lượng nhân chéo, cho mỗi vị trí n trong y, của trị số năng lượng và tương quan cho n hiện thời; và bình phương tương quan, BestCorrSq và trị số năng lượng, bestEn, được lưu từ các trị số trước của n, như:

$$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

trong đó

$$\left.\begin{array}{c} n_{best} = n \\ bestEn = enloop_y \\ BestCorrSq = corr_{xy}^2 \end{array}\right\} \text{khi } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y.$$

9. Phương pháp theo một điểm bất kỳ trong các điểm từ 2 đến 8, phương pháp này còn bao gồm bước:

- theo dõi $maxamp_y$ khi trị số cuối cùng của K, được kết hợp với vectơ đích x, vượt quá trị số ngưỡng.

10. Bộ xử lý tín hiệu số được tạo kết cấu để tìm kiếm hình dạng Lượng tử hóa vectơ hình tháp (Pyramid Vector Quantization - PVQ); PVQ lấy vectơ đích x làm vectơ đầu vào và suy ra vectơ y nhờ cộng lặp lại các xung đơn vị trong vòng lặp tìm kiếm chiều bên trong, bộ mã hóa khác biệt bởi được tạo kết cấu để:

trước khi vào vòng lặp chiều bên trong tiếp theo để cộng xung đơn vị:

- xác định, dựa trên trị số tuyệt đối cực đại, $xabs_{max}$, của vectơ đầu vào, x, độ dịch lên có thể có, trong từ bit, của trị số tương quan trong vòng lặp được tích lũy của vòng lặp tìm kiếm chiều bên trong tiếp theo, $corr_{xy}$, giữa x và vectơ y.

11. Bộ xử lý tín hiệu số theo điểm 10, bộ xử lý tín hiệu số này còn được tạo kết cấu để,

trước khi vào vòng lặp tìm kiếm chiều bên trong tiếp theo để cộng xung đơn vị:

- xác định, dựa trên biên độ xung cực đại, $maxamp_y$, của vectơ hiện thời y, liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn, theo cách không tổn hao, biến, $enloop_y$, liên quan đến năng lượng được tích lũy của y, trong vòng lặp tìm kiếm chiều bên trong tiếp theo.

12. Bộ xử lý tín hiệu số theo điểm 11, bộ xử lý tín hiệu số này còn được tạo kết cấu để:

- thực hiện các tính toán vòng lặp bên trong sử dụng độ dài từ bit dài hơn để biểu diễn $enloop_y$, khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$.

13. Bộ xử lý tín hiệu số theo điểm bất kỳ trong các điểm từ 11 đến 12, bộ xử lý tín hiệu số này được tạo kết cấu để:

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ nhất sử dụng độ dài từ bit thứ nhất khi không cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$, và:

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ hai sử dụng độ dài từ bit dài hơn so với vòng lặp cộng xung đơn vị thứ nhất khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$.

14. Bộ xử lý tín hiệu số theo điểm bất kỳ trong các điểm từ 11 đến 13, bộ xử lý tín hiệu số này còn được tạo kết cấu để:

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ nhất, có độ chính xác nhất định, khi không cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$; và để

- thực hiện các tính toán vòng lặp bên trong nhờ dùng vòng lặp cộng xung đơn vị thứ hai, có độ chính xác cao hơn so với vòng lặp cộng xung đơn vị thứ nhất, khi cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$.

15. Bộ xử lý tín hiệu số theo một điểm bất kỳ trong các điểm từ 11 đến 14, trong đó việc xác định, dựa trên $maxamp_y$, về liệu có cần nhiều hơn độ dài từ bit hiện thời để biểu diễn $enloop_y$ được tạo kết cấu để bao gồm việc xác định các đặc tính của trường hợp khi, trong vòng lặp tìm kiếm chiều bên trong tiếp theo, xung đơn vị được cộng vào vị trí trong y mà được kết hợp với $maxamp_y$.

16. Bộ xử lý tín hiệu số theo một điểm bất kỳ trong các điểm từ 11 đến 15, bộ xử lý tín hiệu số này còn được tạo kết cấu để:

trong vòng lặp tìm kiếm chiều bên trong để cộng xung đơn vị:

- xác định vị trí, $n_{best,}$ trong y để cộng xung đơn vị nhờ ước lượng nhân chéo, cho mỗi vị trí n trong y, của trị số năng lượng và tương quan cho n hiện thời; và tương quan, BestCorrSq, và trị số năng lượng, bestEn, được lưu từ các trị số trước của n, như:

$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$

trong đó

$$\left.\begin{array}{c} n_{best} = n \\ bestEn = enloop_y \\ BestCorrSq = corr_{xy}^2 \end{array}\right\} \text{khi } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y.$$

17. Bộ xử lý tín hiệu số theo một điểm bất kỳ trong các điểm từ 11 đến 16, bộ xử lý tín hiệu số này còn được tạo kết cấu để theo dõi $maxamp_y$ khi số lượng các xung đơn vị cuối cùng, K, được kết hợp với vectơ đích x, vượt quá trị số ngưỡng.

18. Bộ xử lý tín hiệu số theo một điểm bất kỳ trong các điểm từ 10 đến 17, trong đó bộ xử lý tín hiệu số này là bộ xử lý tín hiệu số có độ chính xác cố định.

19. Thiết bị truyền thông bao gồm bộ xử lý tín hiệu số theo điểm bất kỳ trong các điểm từ 10 đến 18.

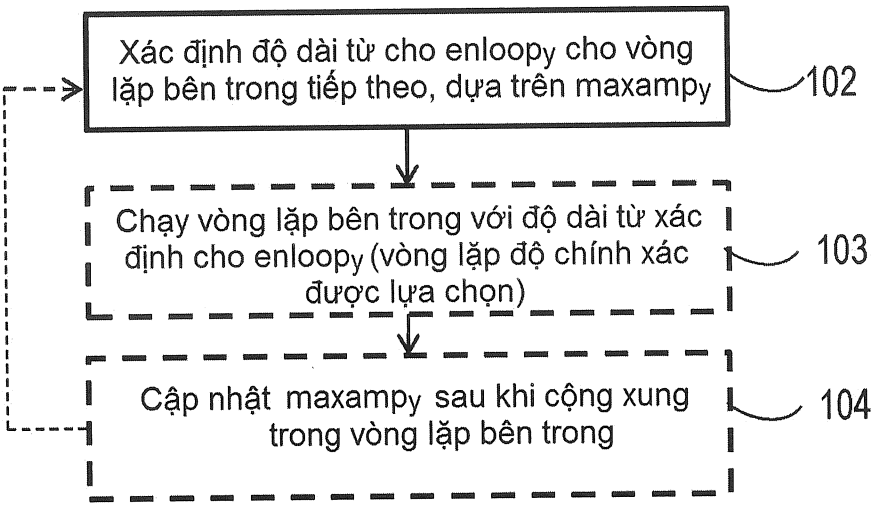

Xác định độ dài từ cho enloopy cho vòng lặp bên trong tiếp theo, dựa trên maxampy
102
Chạy vòng lặp bên trong với độ dài từ xác định cho enloopy (vòng lặp độ chính xác được lựa chọn)
103
Cập nhật maxampy sau khi cộng xung trong vòng lặp bên trong
104

Fig.1

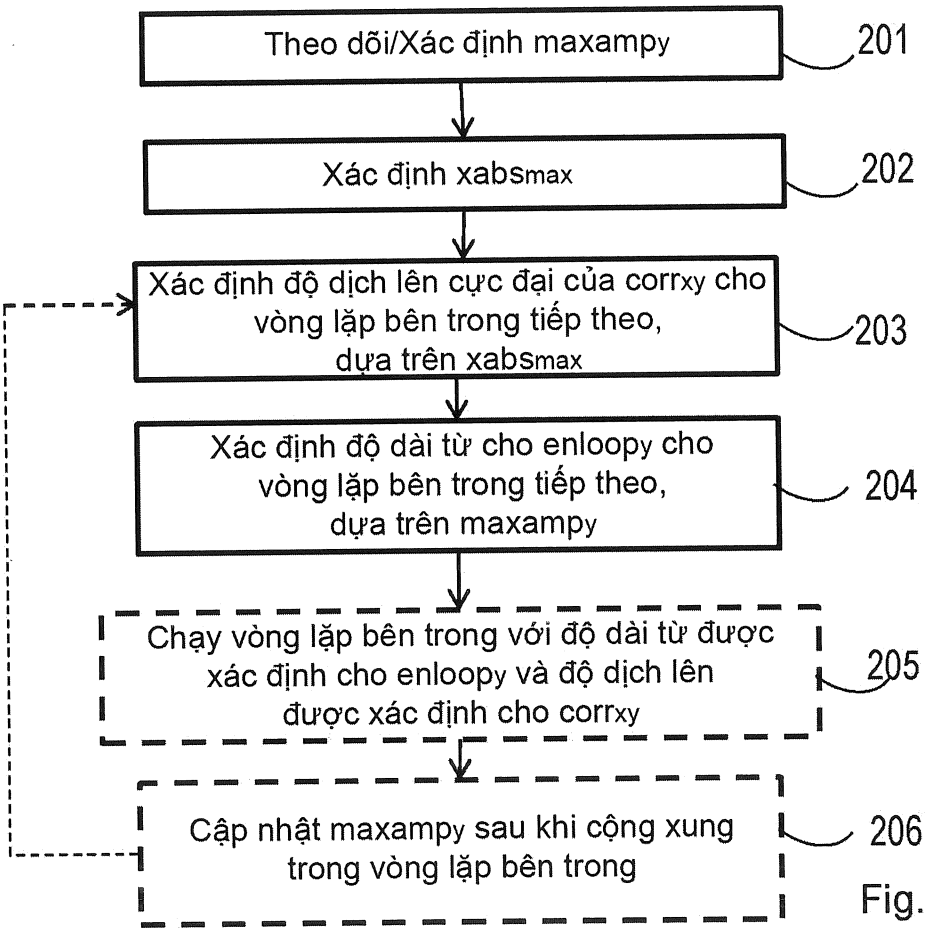

Theo dõi/Xác định maxampy
201
Xác định xabsmax
202
Xác định độ dịch lên cực đại của corrxy cho vòng lặp bên trong tiếp theo, dựa trên xabsmax
203
Xác định độ dài từ cho enloopy cho vòng lặp bên trong tiếp theo, dựa trên maxampy
204
Chạy vòng lặp bên trong với độ dài từ được xác định cho enloopy và độ dịch lên được xác định cho corrxy
205
Cập nhật maxampy sau khi cộng xung trong vòng lặp bên trong
206

Fig.2

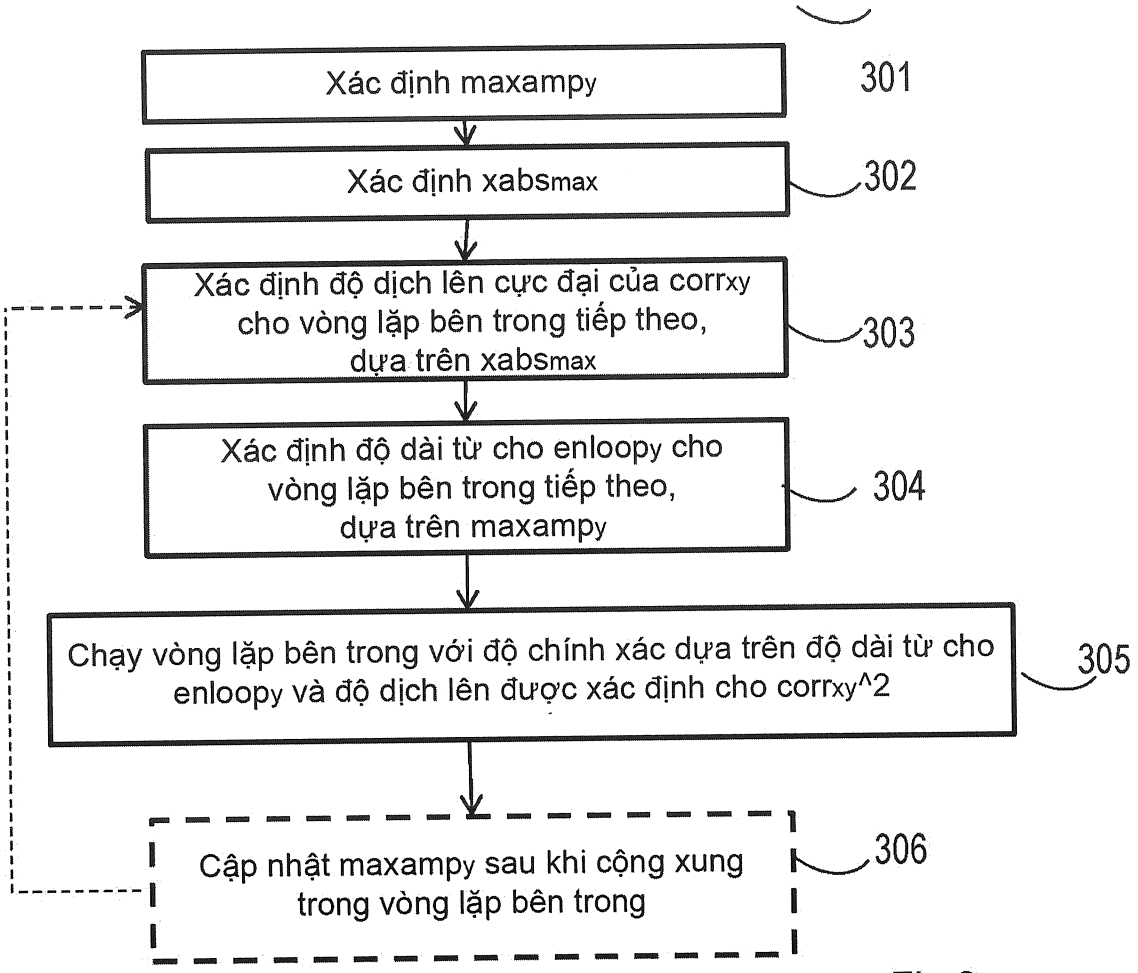

Xác định maxampy
301
Xác định xabsmax
302
Xác định độ dịch lên cực đại của corrxy cho vòng lặp bên trong tiếp theo, dựa trên xabsmax
303
Xác định độ dài từ cho enloopy cho vòng lặp bên trong tiếp theo, dựa trên maxampy
304
Chạy vòng lặp bên trong với độ chính xác dựa trên độ dài từ cho enloopy và độ dịch lên được xác định cho corrxy^2
305
Cập nhật maxampy sau khi cộng xung trong vòng lặp bên trong
306

Fig.3

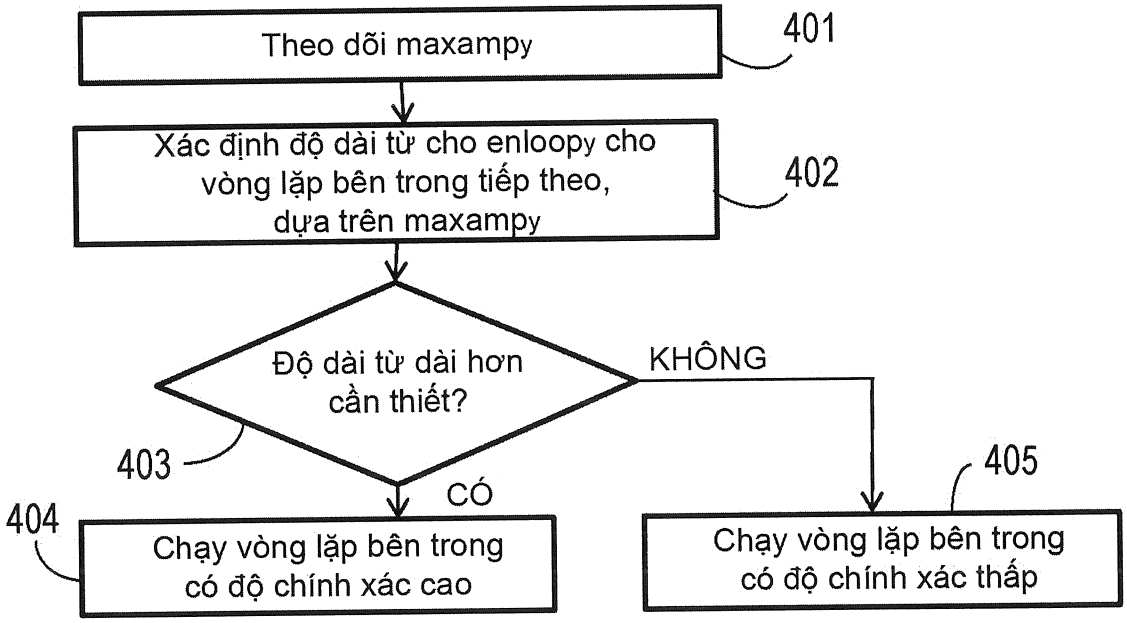

Theo dõi maxampy
401
Xác định độ dài từ cho enloopy cho vòng lặp bên trong tiếp theo, dựa trên maxampy
402
Độ dài từ dài hơn cần thiết?
403
KHÔNG
CÓ
404
Chạy vòng lặp bên trong có độ chính xác cao
405
Chạy vòng lặp bên trong có độ chính xác thấp

Fig.4

3/10

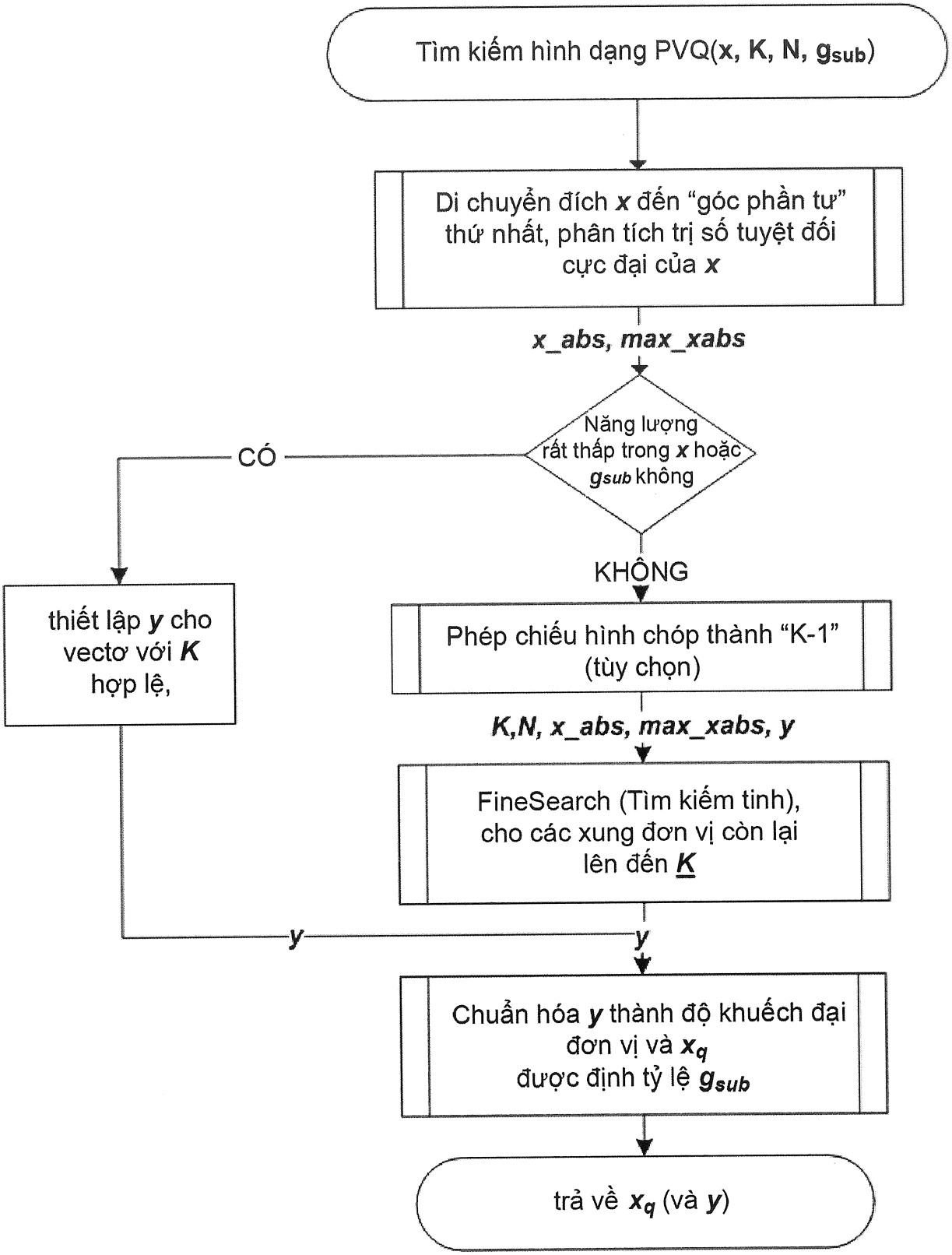

Tìm kiếm hình dạng PVQ(x, K, N, gsub)
Di chuyển đích x đến "góc phần tư" thứ nhất, phân tích trị số tuyệt đối cực đại của x
x_abs, max_xabs
Năng lượng rất thấp trong x hoặc gsub không
CÓ
KHÔNG
thiết lập y cho vectơ với K hợp lệ,
Phép chiếu hình chóp thành "K-1" (tùy chọn)
K,N, x_abs, max_xabs, y
FineSearch (Tìm kiếm tinh), cho các xung đơn vị còn lại lên đến K
y
y
Chuẩn hóa y thành độ khuếch đại đơn vị và xq được định tỷ lệ gsub
trả về xq (và y)

Fig.5

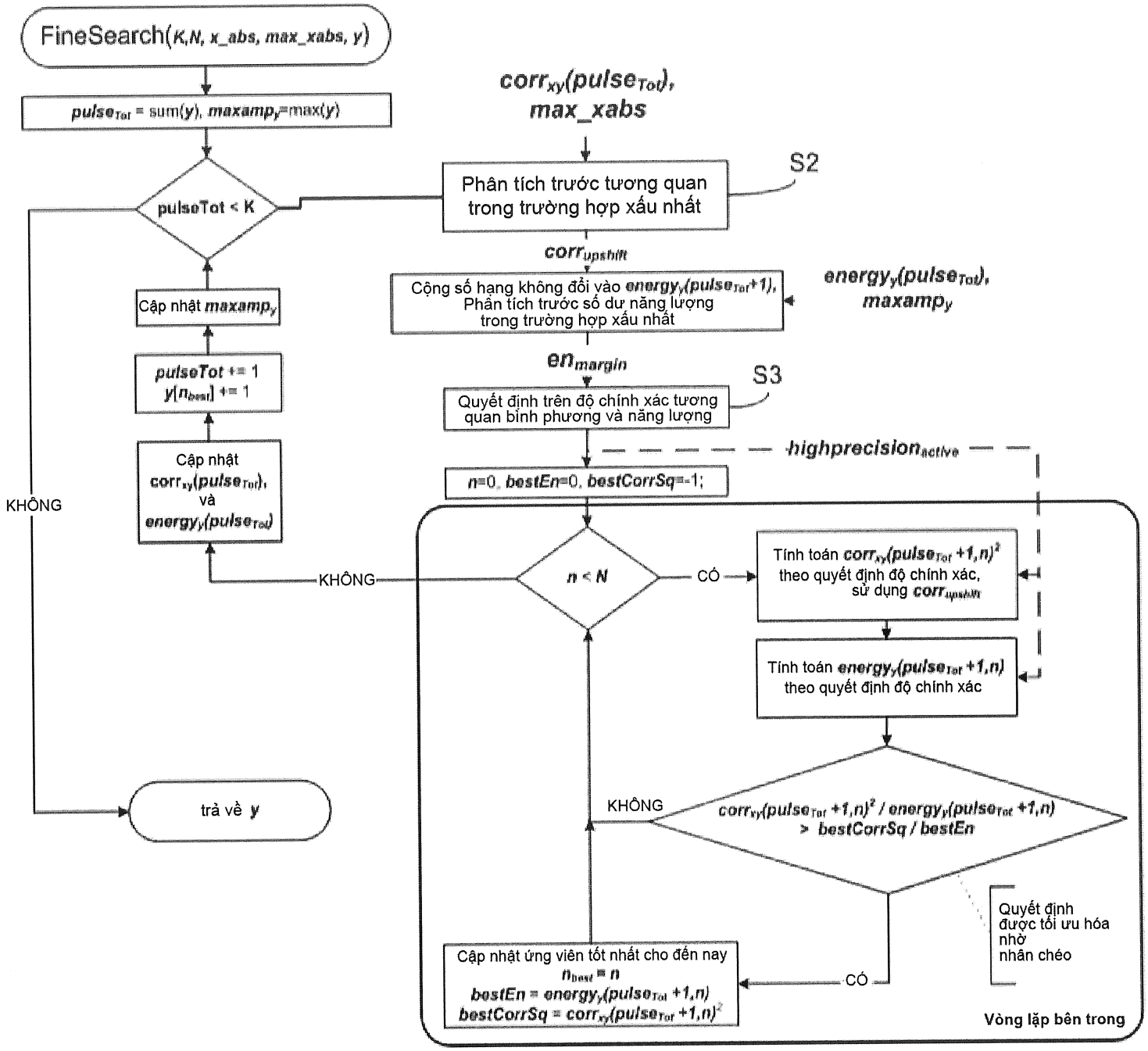

FineSearch(K,N, x_abs, max_xabs, y)
pulseTot = sum(y), maxampy=max(y)
pulseTot < K
Cập nhật maxampy
pulseTot += 1
y[nbest] += 1
Cập nhật corrxy(pulseTot), và energyy(pulseTot)
KHÔNG
trả về y
corrxy(pulseTot), max_xabs
Phân tích trước tương quan trong trường hợp xấu nhất
S2
corrupshift
Cộng số hạng không đổi vào energyy(pulseTot+1), Phân tích trước số dư năng lượng trong trường hợp xấu nhất
energyy(pulseTot), maxampy
enmargin
Quyết định trên độ chính xác tương quan bình phương và năng lượng
S3
highprecisionactive
n=0, bestEn=0, bestCorrSq=-1;
n < N
KHÔNG
CÓ
Tính toán corrxy(pulseTot +1,n)² theo quyết định độ chính xác, sử dụng corrupshift
Tính toán energyy(pulseTot +1,n) theo quyết định độ chính xác
KHÔNG
corrxy(pulseTot +1,n)² / energyy(pulseTot +1,n) > bestCorrSq / bestEn
Quyết định được tối ưu hóa nhờ nhân chéo
CÓ
Cập nhật ứng viên tốt nhất cho đến nay
nbest = n
bestEn = energyy(pulseTot +1,n)
bestCorrSq = corrxy(pulseTot +1,n)²
Vòng lặp bên trong

Fig.6

5/10

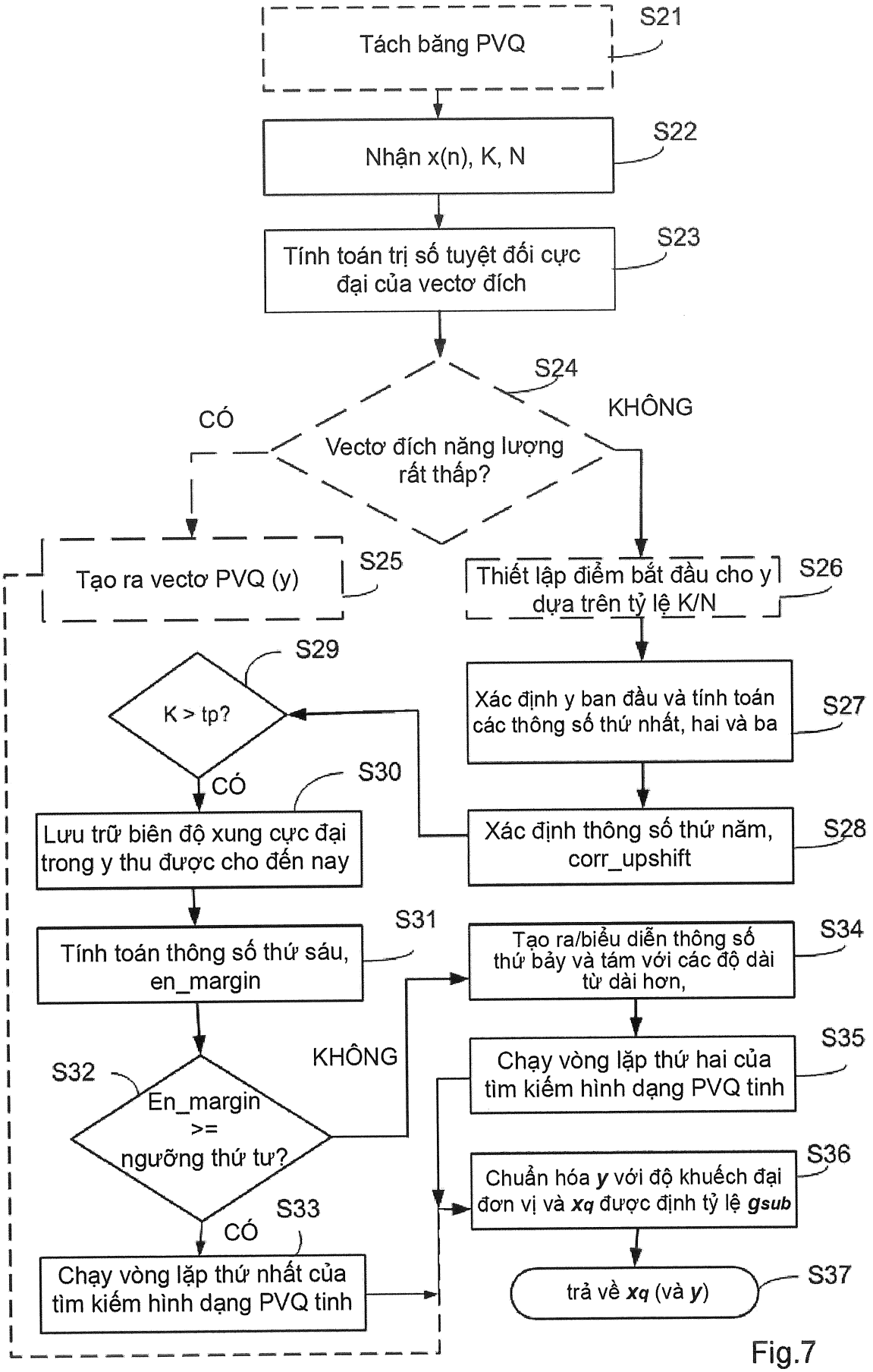
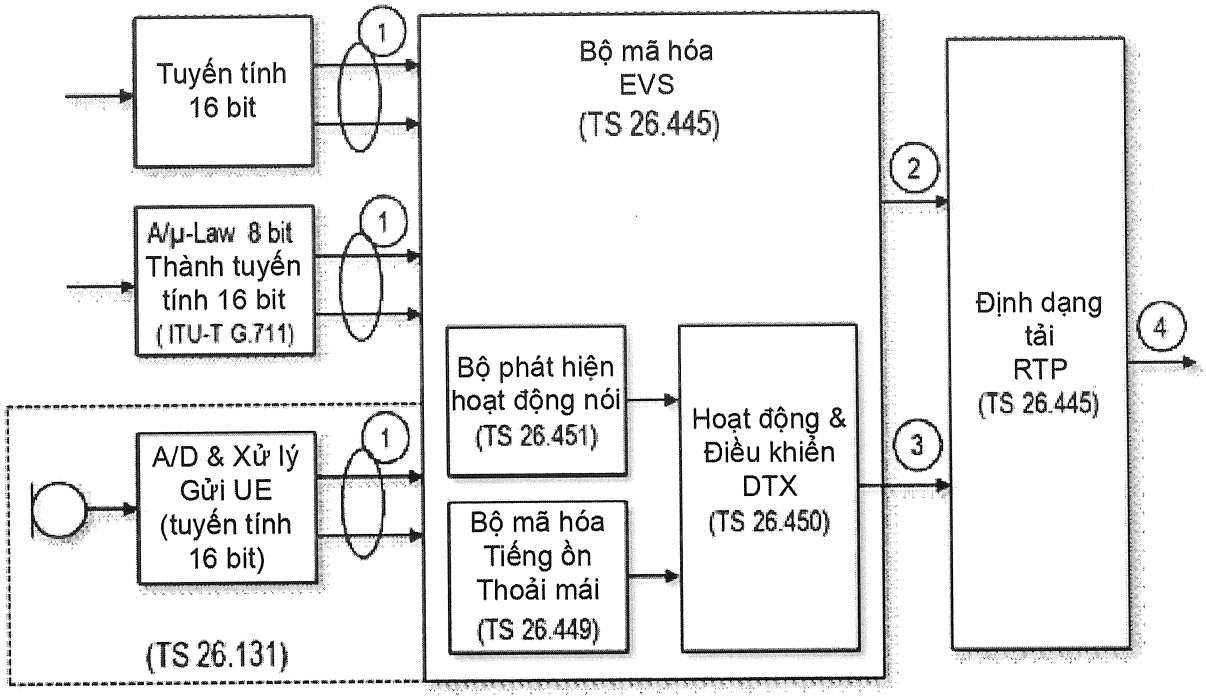

Tách băng PVQ
S21
Nhận x(n), K, N
S22
Tính toán trị số tuyệt đối cực đại của vectơ đích
S23
S24
Vectơ đích năng lượng rất thấp?
CÓ
KHÔNG
Tạo ra vectơ PVQ (y)
S25
Thiết lập điểm bắt đầu cho y dựa trên tỷ lệ K/N
S26
Xác định y ban đầu và tính toán các thông số thứ nhất, hai và ba
S27
Xác định thông số thứ năm, corr_upshift
S28
K > tp?
S29
CÓ
Lưu trữ biên độ xung cực đại trong y thu được cho đến nay
S30
Tính toán thông số thứ sáu, en_margin
S31
En_margin >= ngưỡng thứ tư?
S32
KHÔNG
CÓ
Chạy vòng lặp thứ nhất của tìm kiếm hình dạng PVQ tinh
S33
Tạo ra/biểu diễn thông số thứ bảy và tám với các độ dài từ dài hơn,
S34
Chạy vòng lặp thứ hai của tìm kiếm hình dạng PVQ tinh
S35
Chuẩn hóa y với độ khuếch đại đơn vị và xq được định tỷ lệ gsub
S36
trả về xq (và y)
S37

Fig.7

6/10



(1) Các mẫu PCM tuyến tính 16 bit và Tỷ lệ mẫu (8, 16, 32 hoặc 48 kHz)

(2) Khung âm thanh được mã hóa, 50 khung/giây, số bit/khung phụ thuộc vào chế độ mã hóa-giải mã EVS

(3) Các khung phần tử mô tả im lặng được mã hóa (tỷ lệ khung biến đổi được)

(4) Các gói tải RTP

Fig.8

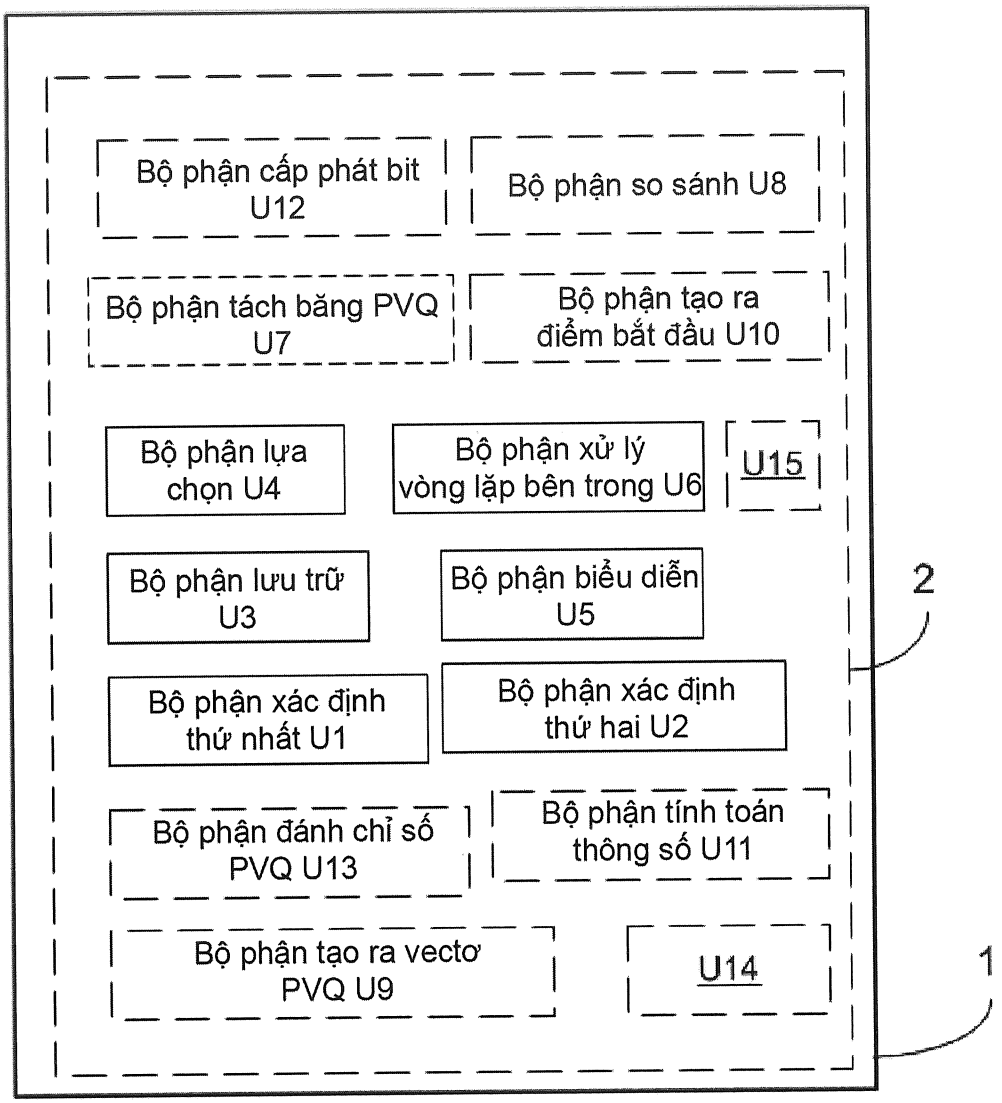

Bộ phận cấp phát bit U12
Bộ phận so sánh U8
Bộ phận tách băng PVQ U7
Bộ phận tạo ra điểm bắt đầu U10
Bộ phận lựa chọn U4
Bộ phận xử lý vòng lặp bên trong U6
U15
Bộ phận lưu trữ U3
Bộ phận biểu diễn U5
Bộ phận xác định thứ nhất U1
Bộ phận xác định thứ hai U2
Bộ phận đánh chỉ số PVQ U13
Bộ phận tính toán thông số U11
Bộ phận tạo ra vectơ PVQ U9
U14
2
1

Fig.9

8/10

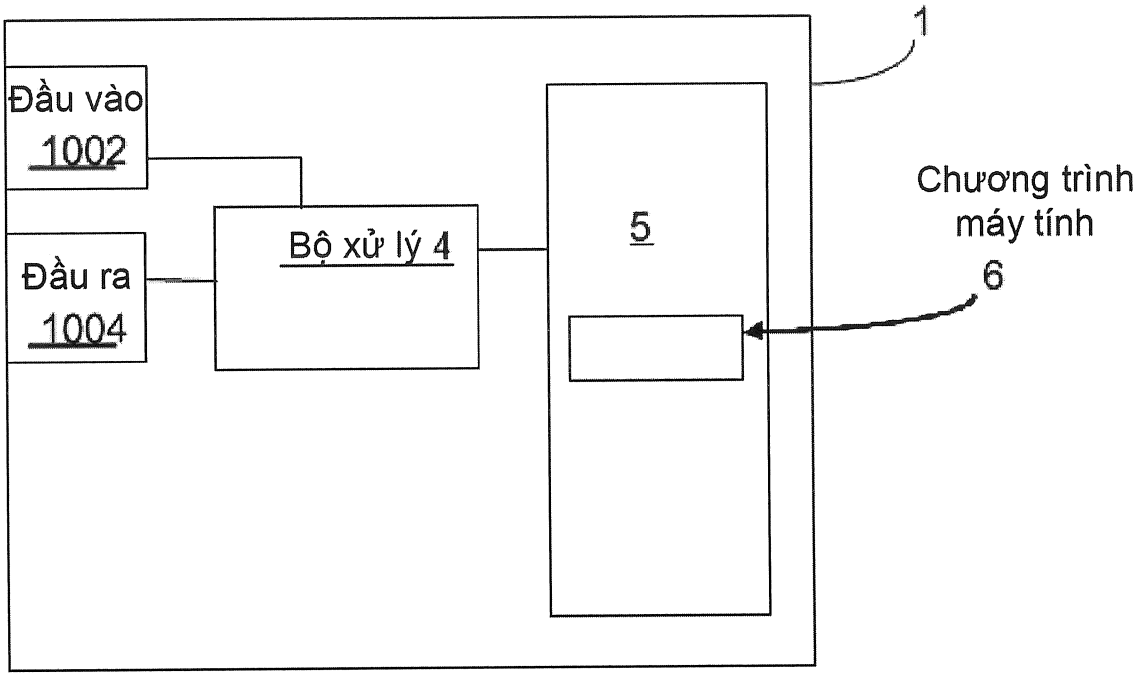


Fig.10

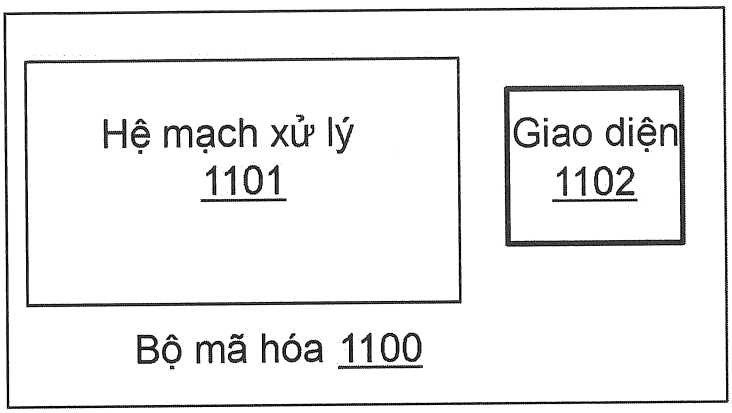


Fig.11a

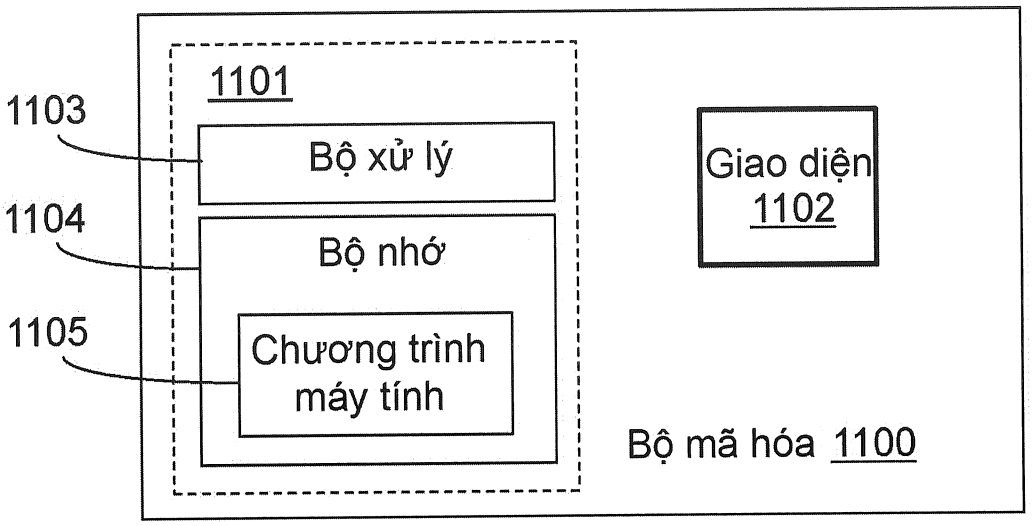


Fig.11b

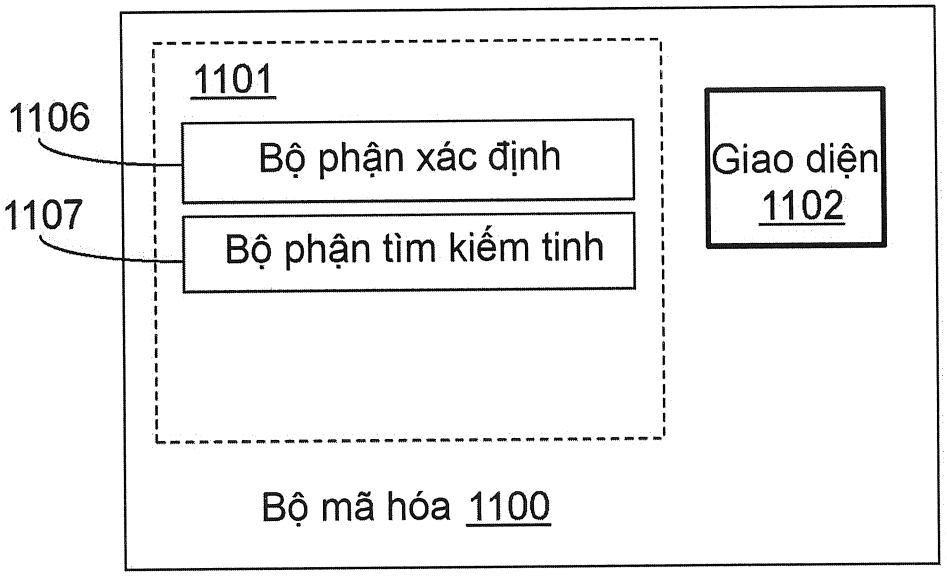


Fig.11c

10/10

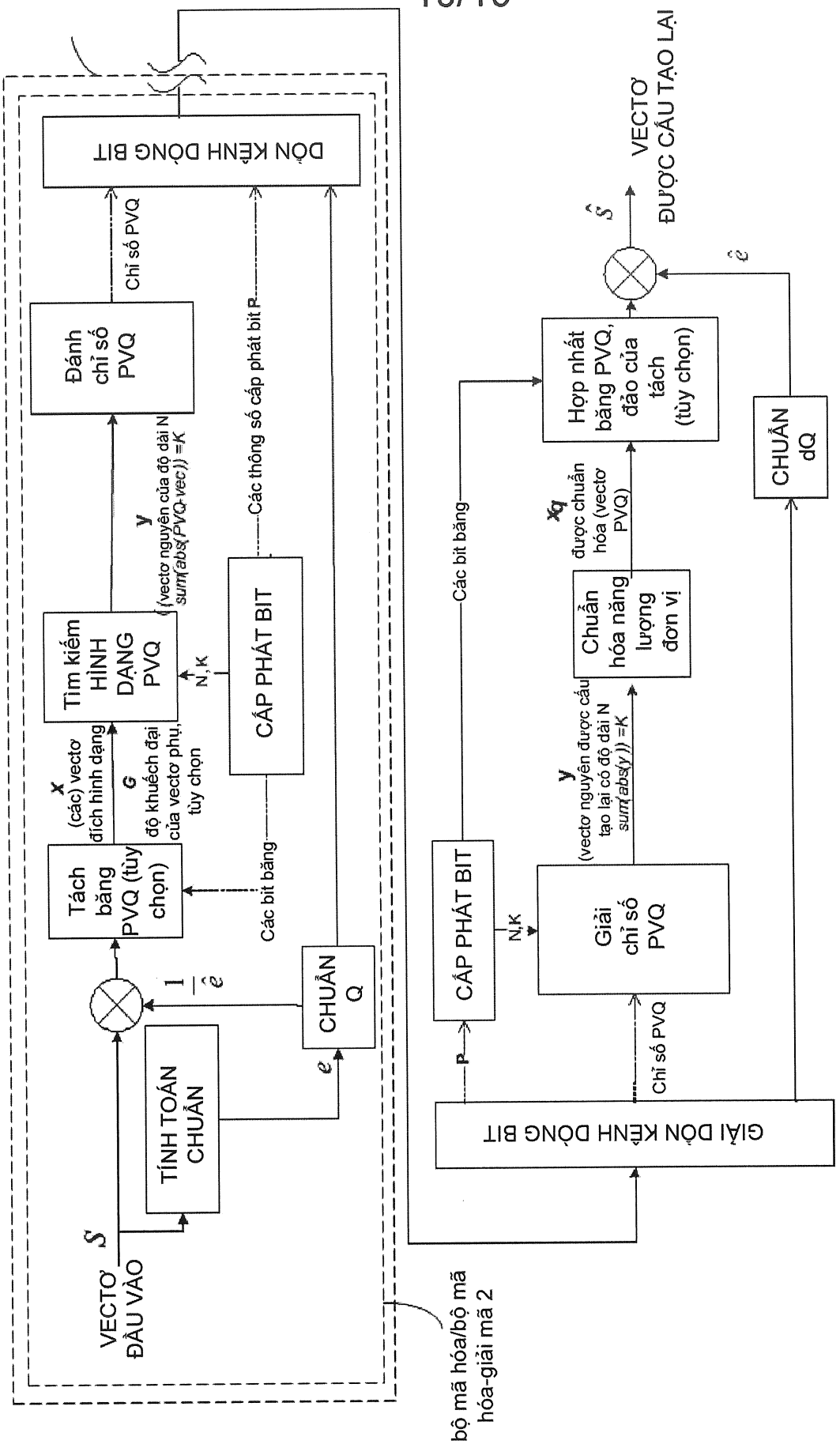

VECTƠ ĐẦU VÀO
S
TÍNH TOÁN CHUẨN
CHUẨN Q
e
1/ê
Tách băng PVQ (tùy chọn)
x (các) vectơ đích hình dạng
g độ khuếch đại của vectơ phụ, tùy chọn
Các bit băng
Tìm kiếm HÌNH DẠNG PVQ
N,K
CẤP PHÁT BIT
y (vectơ nguyên của độ dài N sum(abs(PVQ-vec)) =K
Các thông số cấp phát bit R
Đánh chỉ số PVQ
Chỉ số PVQ
DỒN KÊNH DỒNG BIT
bộ mã hóa/bộ mã hóa-giải mã 2
GIẢI DỒN KÊNH DỒNG BIT
P
CẤP PHÁT BIT
N,K
Các bit băng
Giải chỉ số PVQ
Chỉ số PVQ
y (vectơ nguyên được cấu tạo lại có độ dài N sum(abs(y)) =K
Chuẩn hóa năng lượng đơn vị
xq được chuẩn hóa (vectơ PVQ)
Hợp nhất băng PVQ, đảo của tách (tùy chọn)
ê
CHUẨN dQ
Ŝ
VECTƠ ĐƯỢC CẤU TẠO LẠI

Fig.12