

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

**(19) World Intellectual Property Organization
International Bureau**



A standard linear barcode is positioned horizontally across the page, spanning most of the width. It is used for document tracking and identification.

**(43) International Publication Date
30 November 2006 (30.11.2006)**

PCT

(10) International Publication Number
WO 2006/128072 A2

(51) International Patent Classification:

H04N 7/26 (2006.01) **H04N 7/50** (2006.01)
H04N 7/46 (2006.01)

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(21) International Application Number:

PCT/US2006/020633

(22) International Filing Date: 25 May 2006 (25.05.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/685,261 26 May 2005 (26.05.2005) US
Not furnished 24 May 2006 (24.05.2006) US

(71) **Applicant (for all designated states except US): NTT DOCOMO, INC. [US/JP]; Sanno Park Tower, 11-1, Nagata-cho, 2-chome, Chiyoda-ku, Tokyo 100-6150 (JP).**

(72) **Inventors;** and
(75) **Inventors/Applicants (for US only):** **BOSSEN, Frank**
[NL/CH]; 71 Ch. Des Petits-Bois, CH-1228 Plan-les-
Ouates (CH). **TOURAPIS, Alexandros.**

(74) **Agents:** **MALLIE, Michael, J.** et al.; **BLAKELY,**
SOKOLOFF, TAYLOR & ZAFMAN LLP, 12400 Wilshire
Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

- without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2006/128072 A2

(54) Title: METHOD AND APPARATUS FOR CODING MOTION AND PREDICTION WEIGHTING PARAMETERS

WO (57) Abstract: A method and apparatus is disclosed herein for encoding and/or decoding are described. In one embodiment, the encoding method comprises generating weighting parameters for multi-hypothesis partitions, transforming the weighting parameters and coding transformed weighting parameters.

METHOD AND APPARATUS FOR CODING MOTION AND PREDICTION WEIGHTING PARAMETERS

PRIORITY

[0001] The present patent application claims priority to and incorporates by reference the corresponding provisional patent application serial no. 60/685,261, titled, "Method And Apparatus For Coding Motion and Prediction Weighting Parameters" filed on May 26, 2005.

FIELD OF THE INVENTION

[0002] The present invention relates to the field of video coding and decoding; more particularly, the present invention relates to coding motion and prediction weighting parameters.

BACKGROUND OF THE INVENTION

[0003] Motion Compensated (Inter-frame) prediction is a vital component of video coding schemes and standards such as MPEG-1/2, and H.264 (or JVT or MPEG AVC) due to the considerable benefit it could provide in terms of coding efficiency. However, until recently, in most of these standards, motion compensation was performed by primarily, if not only, considering temporal displacement. More specifically, standards such as MPEG-1, 2, and MPEG-4 consider two different picture types for inter-frame prediction, predictive (P) and bi-directionally predicted (B) pictures. Such pictures are partitioned into a set of non-overlapping blocks, each one associated with a set of motion parameters. For P pictures, motion parameters were limited to a horizontal and a vertical displacement of the current block $I(x,y)$ versus a references, which indicated the position of a second block $I(x',y')$ which would be used for predicting $I(x,y)$. Figure 1A illustrates an example of motion compensation in P pictures. For B pictures, however, this could alternatively or additionally involve the consideration of a horizontal and vertical displacement from a second reference. Figure 1B illustrates an example of motion compensation in B pictures. In the later case, essentially prediction is formed by averaging both predictions from these two references, using equal weighting factors of $(\frac{1}{2}, \frac{1}{2})$.

[0004] Unfortunately, the above model was not sufficient for video scenes containing temporal brightness variations, such as fades, crossfades, flashes, camera-iris adjustments etc. Thus, the simple translational only inter-frame technique cannot

sufficiently improve coding efficiency when temporal brightness variation is involved. For this purpose, several methods were previously proposed that also consider luminance variations during motion compensation. See Kamikura, et al., "Global Brightness-Variation Compensation for Video Coding," IEEE Trans on CSVT, vol. 8, pp. 988–1000, Dec. 1998; Rodrigues, et al., "Hierarchical Motion Compensation with Spatial and Luminance Transformations", ICIP, pp. 518-521, Oct. 2001; Rodrigues, et al., "Low-Bit Rate Video Coding with Spatial and Luminance Transformations", ConfTele2001, Figueira da Foz, 22-24 April, 2001; and J. Boyce, "Weighted Prediction in the H.264/MPEG4 AVC Video Coding Standard", ISCAS, pp. 789-792, May 2004.

[0005] More specifically, instead of considering only geometric transformations during motion compensation, the predicted signal is also scaled and/or adjusted using two new parameters w and o . The sample with brightness value $I(x, y, t)$ at position (x, y) at time t , is essentially constructed as $w * I(x + dx, y + dy, t') + o$, where dx and dy are the spatial displacement parameters (motion vectors). However, these new weighting parameters could also considerably increase the overhead bits required for representing motion information, therefore potentially reducing the benefit of such strategies. For this purpose, and by making the assumption that most luminance transformations happen globally, Kamikura, et al., in "Global Brightness-Variation Compensation for Video Coding," IEEE Trans on CSVT, vol. 8, pp. 988–1000, Dec. 1998, suggests the usage of only a single set of global parameters (w, o) for every frame. The use or not of these parameters is also signaled at the block level, thereby providing some additional benefit in the presence of local brightness variations.

[0006] A somewhat similar strategy is also employed in H.264. This standard, however, also supports several other features that could exploit brightness variations even further. In particular, the codec allows for the consideration of multiple references but also reordering, which enabled multiple weights to be associated with each reference, to better handle local brightness variations. Weights for bi-predicted partitions could also optionally be implicitly derived through the consideration of a temporal, or more precisely a picture order count distance associated with each picture. On the other hand, in Rodrigues, et al., "Hierarchical Motion Compensation with Spatial and Luminance Transformations", ICIP, pp. 518-521, Oct. 2001 and Rodrigues, et al., "Low-Bit Rate Video Coding with Spatial and Luminance Transformations", ConfTele2001, Figueira da Foz, 22-24 April, 2001, weights are optionally transmitted, after quantization, using a hierarchical structure to somewhat constrain the bit overhead. However, this method

appears to only consider weighted prediction within P frames, and no consideration of Bi-prediction is discussed.

[0007] With respect to these techniques, although weighting parameters are considered at the block level, which better handles local brightness variations, the method makes no special consideration of bi-predictive motion compensation, and on how these parameters should be efficiently coded. The focus of these papers is mainly on the potential benefits in terms of improved Mean Square Error of the prediction if luminance transformations are additionally used versus translational only prediction, and less on the coding of the parameters.

[0008] The ITU-H.264 (or JVT or ISO MPEG4 AVC) video compression standard has adopted certain weighted prediction tools that could take advantage of temporal brightness variations and could improve performance. The H.264 video coding standard is the first video compression standard to adopt weighted prediction (WP) for motion compensated prediction.

[0009] Motion compensated prediction may consider multiple reference pictures, with a reference picture index coded to indicate which of the multiple reference pictures is used. In P pictures (or P slices), only uni-prediction is used, and the allowable reference pictures are managed in list 0. In B pictures (or B slices), two separate reference picture lists are managed, list 0 and list 1. In B pictures (or B slices), uni-prediction using either list 0 or list 1, or bi-prediction using both list 0 and list 1 is allowed. When bi-prediction is used, the list 0 and the list 1 predictors are averaged together to form a final predictor. The H.264 weighted prediction tool allows arbitrary multiplicative weighting factors and additive offsets to be applied to reference picture predictions in both P and B pictures. Use of weighted prediction is indicated in the sequence parameter set for P and SP slices. There are two weighted prediction modes; explicit mode, which is supported in P, SP, and B slices, and implicit mode, which is supported in B slices only.

[0010] In the explicit mode, weighted prediction parameters are coded in the slice header. A multiplicative weighting factor and an additive offset for each color component may be coded for each of the allowable reference pictures in list 0 for P slices and list 0 and list 1 for B slices. The syntax, however, also allows different blocks in the same picture to make use of different weighting factors even when predicted from the same reference picture store. This can be made possible by using reordering commands to associate more than one reference picture index with a particular reference picture store.

[0011] The same weighting parameters that are used for single prediction are used in combination for bi-prediction. The final inter prediction is formed for the samples of each macroblock or macroblock partition, based on the prediction type used. For a single directional prediction from list 0,

$$\text{SampleP} = \text{Clip1}(((\text{SampleP}_0 \cdot W_0 + 2^{LWD-1}) \gg LWD) + O_0) \quad (1)$$

and for a single directional prediction from list 1,

$$\text{SampleP} = \text{Clip1}(((\text{SampleP}_1 \cdot W_1 + 2^{LWD-1}) \gg LWD) + O_1) \quad (2)$$

and for bi-prediction,

$$\begin{aligned} \text{SampleP} = \text{Clip1} & (((\text{SampleP}_0 \cdot W_0 + \text{SampleP}_1 \cdot W_1 + 2^{LWD}) \\ & \gg (LWD+1)) + (O_0 + O_1 + 1) \gg 1) \end{aligned} \quad (3)$$

where Clip1() is an operator that clips the sample value within the range [0, 1<<SampleBitDepth - 1], with SampleBitDepth being the number of bits associated with the current sample, W_0 and O_0 are the weighting factor and offset associated with the current reference in list 0, W_1 and O_1 are the weighting factor and offset associated with the current reference in list 1, and LWD is the log weight denominator rounding factor, which essentially plays the role of a weighting factor quantizer. SampleP_0 and SampleP_1 are the list 0 and list 1 initial predictor samples, while SampleP is the final weight predicted sample.

[0012] In the implicit weighted prediction mode, weighting factors are not explicitly transmitted in the slice header, but instead are derived based on relative distances between the current picture and its reference pictures. This mode is used only for bi-predictively coded macroblocks and macroblock partitions in B slices, including those using direct mode. The same formula for bi-prediction as given in the preceding explicit mode section is used, except that the offset values O_0 and O_1 are equal to zero, and the weighting factors W_0 and W_1 are derived using the formulas:

$$\begin{aligned} X &= (16384 + (TD_D \gg 1)) / TD_D \\ Z &= \text{clip3}(-1024, 1023, (TD_B \cdot X + 32) \gg 6) \\ W_1 &= Z \gg 2, W_0 = 64 - W_1 \end{aligned} \quad (4)$$

This is a division-free, 16-bit safe operation implementation of

$$W_1 = (64 * TD_D) / TD_B$$

where TD_D and TD_B are the clipped within the range [-128,127] temporal distances between the list 1 reference and the current picture versus the list 0 reference picture respectively.

[0013] Although the H.264 video coding standard enables the use of multiple weights for motion compensation, such use is considerably limited by the fact that the standard only allows signaling of up to 16 possible references for each list at the slice level. This implies that only a limited number of weighting factors could be considered. Even if this restriction did not apply, it could be potentially inefficient if not difficult to signal all possible weighting parameters that may be necessary for encoding a picture. Note that in H.264, weighting parameters for each reference are independently coded without the consideration of a prediction mechanism, while the additional overhead for signaling the reference indices may also be significant. Therefore H.264 and the method disclosed in Kamikura, et al., "Global Brightness-Variation Compensation for Video Coding," IEEE Trans on CSVT, vol. 8, pp. 988–1000, Dec. 1998 are more appropriate for global than local brightness variations. In other words, these tools tend to work well for global brightness variations, but due to certain limitations little gain can be achieved in the presence of significant local brightness variation.

SUMMARY OF THE INVENTION

[0014] A method and apparatus for encoding and/or decoding are described. In one embodiment, the encoding method comprises generating weighting parameters for multi-hypothesis partitions, transforming the weighting parameters and coding transformed weighting parameters.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

[0016] **Figure 1A** illustrates an example of motion compensation in P pictures.

[0017] **Figure 1B** illustrates an example of motion compensation in B pictures.

[0018] **Figure 2** is a flow diagram of one embodiment of an encoding process.

[0019] **Figure 3** is a flow diagram of one embodiment of a decoding process.

- [0020] **Figure 4** illustrates a tree with parent-child relationships for encoding of bi-prediction motion information.
- [0021] **Figure 5** illustrates another tree with node information that may be encoded..
- [0022] **Figure 6** is a block diagram of one embodiment of an encoder.
- [0023] **Figure 7** is a block diagram of one embodiment of a decoder.
- [0024] **Figure 8** is an example of the relationship between the representation and the true value of the weighting parameter.
- [0025] **Figure 9** illustrates one embodiment of a flow diagram for a process of using global and local weighting jointly.
- [0026] **Figure 10** illustrates a transform process of the weighting parameters for coding frames using biprediction.
- [0027] **Figure 11** illustrates the process for weighting parameters for the multi-hypothesis case.
- [0028] **Figure 12** illustrates the transform process of the weighting parameters for biprediction with controlled fidelity.
- [0029] **Figure 13** is a block diagram of an exemplary computer system that may perform one or more of the operations described herein.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

[0030] An efficient coding scheme for coding weighting parameters within bi-predicted (or multi-predicted) partitions of a video coding architecture is disclosed. This scheme improves performance of the video coding system and can be used to handle local brightness variations. The coding scheme includes a transform process between pairs of weighting parameters associated with each reference for each block that is being subjected to motion compensated prediction. The transform process causes a transform to be applied to the weighting parameters. In one embodiment, the transformed weighting parameters are signaled for every block using a prediction method and are coded using a zero tree coding structure. This reduces the overhead required for coding these parameters. While the methods described herein are primarily aimed at bi-predictive partitions. Some of the concepts could also apply to uni-predicted partitions.

[0031] In one embodiment, the interactions between uni-predicted and bi-predicted partitions are used to further improve efficiency. Special consideration is also made for bi-

predictive pictures or slices, where a block may be predicted from a different list or both lists.

[0032] In one embodiment, the coding scheme is further extended by combining and considering both global and local weighting parameters. Additional weighting parameters transmitted at the sequence, picture, or slice level are also optionally considered.

[0033] In another embodiment, the variable granularity dynamic range of such parameters is also taken into account. These weighting parameters are finely adjustable, which provides additional benefits in terms of coding efficiency.

[0034] In the following description, numerous details are set forth to provide a more thorough explanation of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

[0035] Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0036] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data

similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0037] The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0038] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

[0039] A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

Overview

[0040] Embodiments of the present invention include a coding scheme that uses motion compensation that includes weighted prediction. The weighting parameters associated with a prediction are transformed and then coded with the offset associated with the prediction. The decoding process is a reverse of the encoding process.

[0041] Figure 2 is a flow diagram of one embodiment of an encoding process. The process is performed by processing logic that may comprise hardware (circuitry, dedicated

logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[0042] Referring to Figure 2, the process begins by processing logic generating weighting parameters for multi-hypothesis partitions (e.g., bi-predictive partitions) as part of performing motion estimation (processing block 201). In one embodiment, the weighting parameters comprise at least one pair of a set of weighting parameters for each partition (block) of a frame being encoded. In one embodiment, the weights for multiple pairs of transformed weighting parameters sum to a constant. In one embodiment, different weighting factors for at least two partitions of the frame are used. In one embodiment, the weighting parameters compensate for local brightness variations.

[0043] In one embodiment, the weighted prediction parameters have variable fidelity levels. The variable fidelity levels are either predefined or signaled as part of a bitstream that includes an encoded representation of the transformed weighting parameters. In one embodiment, one group of weighting parameters have a higher fidelity than those of another group of weighting parameters.

[0044] Next, processing logic applies a transform to the weighting parameters to create transformed weighting parameters (processing block 202). In one embodiment, transforming the weighting parameters comprises applying a transform to at least two weighting parameters associated with each reference. In one embodiment, transforming the weighting parameters comprises applying a transform that satisfies the following:

$$\begin{aligned} B_wQ_0 &= wQ_0 + wQ_1 \\ B_wQ_1 &= wQ_0 - ((wQ_0 + wQ_1) \gg 1) \end{aligned}$$

where wQ_0 and wQ_1 are quantized weight parameters are B_wQ_0 and B_wQ_1 are transformed versions of wQ_0 and wQ_1 .

[0045] After applying a transform to the weighting parameters, processing logic codes transformed weighting parameters and an offset (processing block 203). In one embodiment, the transformed weighting parameters are coded with at least one offset using variable length encoding. The variable length encoding may use a zero tree coding structure. Variable length encoding may comprise Huffman coding or arithmetic coding.

[0046] In one embodiment, coding the transformed weighting parameters comprises differentially coding using predictions based on one of a group consisting of transformed coefficients of neighboring partitions or predetermined weighting parameters. The predetermined weighting parameter are used when no neighboring partition exists for

a prediction. The predetermined weighting parameters may be default weighting parameters or globally transmitted weighting parameters.

[0047] Figure 3 is a flow diagram of one embodiment of a decoding process. In one embodiment, such a decoding process is used to decode information encoded as part of the encoding process of Figure 2. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[0048] Referring to Figure 3, the process begins with processing logic decoding a bitstream using variable length decoding to obtain decoded data that includes transformed weighting parameters for multi-hypothesis partitions (e.g., bi-predictive partitions) (processing block 301). Variable length decoding may comprise Huffman coding or arithmetic coding.

[0049] After variable length decoding, processing logic applies an inverse transform to the transformed weighting parameters (processing block 302). In one embodiment, the inverse transformed weighting parameters comprise at least two weighting parameters associated with a reference.

[0050] Then, processing logic reconstructs frame data using motion compensation in conjunction with inverse transformed weighting parameters (processing block 303).

Weighted Prediction Specifics

[0051] Weighted prediction is useful for encoding video data that comprises fades, cross-fades/dissolves, and simple brightness variations such as flashes, shading changes etc. Bi-prediction (and multi-hypothesis prediction in general) on the other hand makes the assumption that by combining multiple predictions using simple weighting mechanisms, one may generate a better prediction for a given signal. Conventional bi-prediction tends to use equal weights (i.e. $\frac{1}{2}$) for all predictions. In one embodiment, one signal is more correlated than another, for example due to temporal distance or quantization or other noise introduced/existing within each signal. To perform weighted bi-prediction for a sample $I(x, y, t)$ at position (x, y) at time t is performed, the following may be used:

$$I(x, y, t) = weight_0 \cdot I(x + dx_0, y + dy_0, t_0) + weight_1 \cdot I(x + dx_1, y + dy_1, t_1) + offset \quad (5)$$

where $weight_k$, dx_k , dy_k , and t_k are the weights, horizontal and vertical displacement, and time corresponding to prediction from list k , and offset is the offset parameter. In one embodiment, for the implicit weighted prediction, weights are selected such that

$$weight_0 + weight_1 = c \quad (6)$$

where c is a constant. In one embodiment, the constant c is equal to 1. This could basically also be seen as the computation of the weighted average between all prediction samples.

[0052] The above relationship between the two bi-predictive weights holds true in most cases, i.e. for conventional content without the presence of brightness variations or cross fades.

[0053] In one embodiment, instead of using parameters $weight_0$ and $weight_1$, the N -bit quantized parameters wQ_0 and wQ_1 are used. In one embodiment, N is 6 bits. Using substitution, equation 5 above then becomes equal to:

$$I(x, y, t) = ((wQ_0 \cdot I(x + dx_0, y + dy_0, t_0) + wQ_1 \cdot I(x + dx_1, y + dy_1, t_1) + (1 << (N-1))) >> N) + offset \quad (7)$$

where $<<$ denotes a left bit shift operator and $>>$ denotes a right bit shift operator.

[0054] Then two new parameters, namely the transformed quantized parameters B_wQ_0 and B_wQ_1 , and their relationship to the quantized parameter are as follows:

$$\begin{aligned} wQ_0 &= (B_wQ_0 >> 1) + B_wQ_1 \\ wQ_1 &= ((B_wQ_0 + 1) >> 1) - B_wQ_1 = B_wQ_0 - wQ_0 \end{aligned} \quad (8)$$

$$\begin{aligned} B_wQ_0 &= wQ_0 + wQ_1 \\ \Rightarrow B_wQ_1 &= wQ_0 - ((wQ_0 + wQ_1) >> 1) \end{aligned} \quad (9)$$

[0055] Considering that the relation in equation 6 may hold for a majority of blocks/partitions, in one embodiment, B_wQ_0 may be further modified to

$$B_wQ_0 = wQ_0 + wQ_1 - (1 << N)$$

finally resulting in equations:

$$\begin{aligned} wQ_0 &= ((B_wQ_0 + (1 << N)) >> 1) + B_wQ_1 \\ wQ_1 &= B_wQ_0 + (1 << N) - wQ_0 \end{aligned} \quad (10)$$

In an alternative embodiment, using differentiated coding, $1 << N$ is used as a default predictor for B_wQ_0 .

[0056] In one embodiment, parameters B_wQ_0 and B_wQ_1 are coded as is. In another embodiment, the parameters B_wQ_0 and B_wQ_1 are differentially coded by considering the values of B_wQ_0 and B_wQ_1 from neighboring bi-predicted partitions. If no such partitions exist, then these parameters could be predicted using default values dB_wQ_0 and dB_wQ_1 (e.g., both values could be set to 0).

[0057] Figure 10 illustrates a transform process of the weighting parameters for coding frames using bi-prediction. Referring to Figure 10, transform 1001 receives the

quantized weighting parameters wQ_0 and wQ_1 and produces transformed weighting parameters B_wQ_0 and B_wQ_1 , respectively. These transformed weighting parameters are variable length encoded using variable length encoder 1002 along with the offset.

Thereafter, the encoded weighting parameters are sent as part of the bitstream to a decoder. The variable length decoder 1103 decodes the bitstream and produces the transformed weighting parameters B_wQ_0 and B_wQ_1 , along with the offset. Inverse transform 1004 inverse transforms the transformed weighting parameters to produce the quantized version of the weighting parameters wQ_0 and wQ_1 . Weighting parameter wQ_0 is multiplied by a motion-compensated sample x_0 by multiplier 1006 to produce an output which is input into adder 1007. Weighting parameter wQ_1 is multiplied by a motion-compensated sample x_1 using multiplier 1005. The result from multiplier 1005 is added to the output of multiplier 1006 and to 2^{N-1} using adder 1007. The output of adder 1007 is divided by 2^N by divider 1008. The result of the division is added to the offset using adder 1009 to produce a predicted sample x' .

[0058] Note that the present invention is not limited to the bi-prediction case; it is equally applicable to the multi-hypothesis case where M different hypothesis are used. By again making the assumption that the most likely relationship between the M hypothesis weights is of the form $\sum_{i=0}^{M-1} weight_i = c$, in one embodiment, the original weighting parameters are transformed into the following format:

$$B_wQ_0 = \sum_{i=0}^{M-1} wQ_i \quad (11)$$

$$B_wQ_k = wQ_{k-1} - \left(\frac{B_wQ_0}{M} \right) \quad \text{for } k = 1, \dots, M-1$$

while the final weighting parameters could be generated as:

$$wQ_k = \frac{B_wQ_0}{M} + B_wQ_{k+1} \quad \text{for } k = 0, \dots, M-2 \quad (12)$$

$$wQ_{M-1} = B_wQ_0 - \sum_{i=0}^{M-2} wQ_i$$

[0059] Figure 11 illustrates the process for weighting parameters for the multi-hypothesis case. Referring to Figure 11, the same arrangement as shown in Figure 10 is shown in Figure 11 with the difference that sets of two or more weighting parameters are transformed by transform 1101 and variable length encoder 1102 encodes sets of quantized weighting parameters along with one offset for each set. With respect to decoding, variable length decoder 1103 decodes the bitstream to produce sets of transformed weighting

parameters. Each set of transformed weighting parameters is inverse transformed by an inverse transform 1104, the output of which are combined to produce a predicted sample x' in the same way as is done in Figure 10. Adder 1107 adds the results of all multiplications for all of the quantized weighting parameters.

[0060] In general, the same picture or slice may have a combination of both bi-predicted and uni-predicted partitions. In particular, it is possible for a bi-predicted partition to have a uni-predicted neighbor using either list0 or list1 prediction, or even a uni-predicted partition using listX (where X could be 0 or 1) having neighbors using a different list. Although motion vectors within a bi-predicted partition could still be correlated with motion vectors of a uni-predicted partition, this is less likely to be the case for motion vectors of different lists, but also of weighting parameters using different number of hypotheses (i.e. weights from bi-predicted partitions may have little relationship with weights from uni-predicted partitions, while a weight and motion vector from a uni-predicted partition using list 0, may have little relationship with these parameters from another partition using list 1). In one embodiment, all weights from a given prediction mode (list0, list1, and bi-predicted) are restricted to be only predicted from adjacent partitions in the same mode. In one embodiment, if the current partition is a bi-predicted partition, and all R neighbors ($Neigh_0, Neigh_1, \dots, Neigh_R$) are also bi-predicted, then a weighting parameter B_{wQ_k} is predicted as:

$$B_{wQ_k} = f(Neigh_0, \dots, Neigh_R, k) + dB_{wQ_k} \quad (13)$$

where $f()$ corresponds to the relationship used to determine the prediction for the k -th weighting parameter. In one embodiment, a function used for f , for example, is the median value of all predictors.

[0061] In alternative embodiments, a function such as the mean, consideration of only the top or left neighbors based on the reference frame they use or/and a predefined cost, etc. However, if $Neigh_j$ uses a different prediction type, then it may be excluded from this prediction process. In one embodiment, if no predictions are available, then default values, as previously discussed could be used as a predictor.

[0062] In one embodiment, after prediction, zero tree coding is employed to further efficiently encode the weighting parameters, and in general the motion parameters for a block. Figures 4 and 5 represent two possible trees with the node information that may be encoded. Referring to Figure 4, an example parent-child relationship for encoding of bi-prediction motion information is shown. In this example, nodes are segmented based on lists, with the horizontal and vertical motion vector displacement difference for List0

having a common parent node and the horizontal and vertical motion vector displacement difference for List1 having a common parent node. In Figure 5, motion is segmented based on motion vector component. In this case, the horizontal motion vector displacement difference for both List0 and List1 have a common parent node, while the vertical motion vector displacement difference for both List0 and List1 have a common parent node.

[0063] The trees described in Figure 4 and 5 containing both node type data and leaf indices data may be encoded (and subsequently decoded) using tree coding (e.g., zero tree coding) as described in U.S. Patent Application No. 11/172,052, entitled “Method and Apparatus for Coding Positions of Coefficients,” filed June 29, 2005.

[0064] Default values for a given prediction type, however, do not have to be fixed. In one embodiment, a set of global/default parameters are used and coded such as the ones used by the H.264 explicit mode, or alternatively derive these parameters using a mechanism similar to that of the H.264 implicit mode. The method or parameters could be signaled at the picture or slice level.

[0065] Figure 9 is a flow diagram of one embodiment for a process of using global and local weighting jointly. The process is performed by processing logic that may comprise hardware (circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both.

[0066] Referring to Figure 9, the process begins by setting default weighting parameters (processing block 901). In such a case, there are default scaling and offset parameters for List0, List1, and Bipred. For List0, the default weighting parameters are {DScale_{L0}, DOffset_{L0}}. For List1, the weighting parameters are {DScale_{L1}, DOffset_{L1}}. For bi-prediction, the default weighting parameters are {DScale_{Bi_L0}, DScale_{Bi_L1}, DOffset}.

[0067] Then the process continues for each block in the frame, where processing logic tests whether the block is a bi-prediction block (processing block 903). If so, processing logic transitions to processing block 904 where the processing logic tests whether there are any available bi-prediction predictors. If there are, processing logic computes the weighting parameter predictors using neighbors using a function $f_{bipred}(Neigh_0, \dots, Neigh_1)$ (processing block 905) and processing continues to processing block 914. If there are no available bi-prediction predictors, processing logic sets the weighting parameters to the default {DScale_{Bi_L0}, DScale_{Bi_L1}, DOffset} (processing block 906) and the processing transitions to processing block 914.

[0068] If the block in the frame is not undergoing bi-prediction, processing transitions to processing block 907 where processing logic tests whether the block list is List0. If it is, processing logic transitions to processing block 908 where the processing logic tests whether there are any available List0 prediction predictors. If there are, processing logic computes the weighting parameter predictors using neighbors $f_{L0}(Neigh_0, \dots, Neigh_1)$ (processing block 909) and processing continues to processing block 914. If there are no available List0 predictors, processing logic sets the weighting parameters to the default values $\{DScale_{L0}, DOffset_{L0}\}$ (processing block 910) and the processing transitions to processing block 914.

[0069] If the block isn't undergoing prediction using List0, processing transitions to processing block 911 where the processing logic tests whether there are any available List1 prediction predictors. If there are, processing logic computes the weighting parameter predictors using neighbors $f_{L0}(Neigh_0, \dots, Neigh_1)$ (processing block 912) and processing continues to processing block 914. If there are no available List1 predictors, processing logic sets the weighting parameters to the default values $\{DScale_{L1}, DOffset_{L1}\}$ (processing block 913) and the processing transitions to processing block 914.

[0070] At processing block 914, processing logic decodes the weighting parameters. Thereafter, the weighting parameters predictors are added to the weighting parameters (processing block 915). The process then repeats for each block in the frame, and when each block in the frame has been processed, the process ends.

[0071] Based on the discussion above, separate default weights for bi-predicted and uni-predicted partitions can be used, while such parameters could be estimated using similar methods as presented in Kamikura, et al., "Global Brightness-Variation Compensation for Video Coding," IEEE Trans on CSVT, vol. 8, pp. 988–1000, Dec. 1998 and J. Boyce, "Weighted prediction in the H.264/MPEG4 AVC video coding standard", ISCAS, pp. 789-792, May 2004. Table 1 shows an example prediction weight table syntax that could be used for this purpose. It could be observed that in this table, global weights could be signaled separately for uni-predicted and bi-predicted regions.

[0072] Furthermore, bi-predicted weights could have different dynamic range, while the parameters `luma_bipred_weight_type` and `chroma_bipred_weight_type` are used to control the method that is to be used for deriving the weights. More specifically for luma, if `luma_bipred_weight_type` is set to 0, then the default $\frac{1}{2}$ weight is used. If 1, then implicit mode is used, while if 2, the uniprediction weights are used instead as is currently

done in H.264 explicit mode. Finally, if 3, weighting parameters are transmitted explicitly within this header.

Table 1 Proposed Prediction Weight Table Syntax

pred_weight_table() {	C	Descriptor
//! Dynamic range of unipred weighted parameters		
luma_log2_weight_denom	2	ue(v)
chroma_log2_weight_denom	2	ue(v)
//! Signal Uni-pred weights for list0		
for(i = 0; i <= num_ref_idx_10_active_minus1; i++) {		
luma_weight_10_flag	2	u(1)
if(luma_weight_10_flag) {		
luma_weight_10[i]	2	se(v)
luma_offset_10[i]	2	se(v)
}		
chroma_weight_10_flag	2	u(1)
if(chroma_weight_10_flag)		
for(j = 0; j < 2; j++) {		
chroma_weight_10[i][j]	2	se(v)
chroma_offset_10[i][j]	2	se(v)
}		
}		
if(slice_type == B) {		
use_separate_bipred_weights	2	u(1)
//! Signal Uni-pred weights for list1		
for(I = 0; i <= num_ref_idx_11_active_minus1;		
i++) {		
luma_weight_11_flag	2	u(1)
if(luma_weight_11_flag) {		
luma_weight_11[i]	2	se(v)
luma_offset_11[i]	2	se(v)
}		
chroma_weight_11_flag	2	u(1)
if(chroma_weight_11_flag)		
for(j = 0; j < 2; j++) {		
	2	se(v)

chroma_weight_l1[i][j]			
	2	se(v)	
chroma_offset_l1[i][j]			
}			
}			
/* Signal bi-predictive weights if required			
if (use_separate_bipred_weights) {			
/* Dynamic range of bipred weighted parameters			
luma_bipred_log2_weight_denom	2	ue(v)	
chroma_bipred_log2_weight_denom	2	ue(v)	
for(j = 0; j <=			
num_ref_idx_l0_active_minus1; j++) {			
luma_bipred_weight_type	2	ue(v)	
chroma_bipred_weight_type	2	ue(v)	
for(i = 0; i <=			
num_ref_idx_l1_active_minus1; i++) {			
if			
(luma_bipred_weight_type == 3) {			
	2	se(v)	
luma_bipred_weight_l0[j][i]			
	2	se(v)	
luma_bipred_weight_l1[j][i]			
	2	se(v)	
luma_bipred_offset[j][i]			
}			
if			
(chroma_bipred_weight_type == 3)			
for(k = 0; k < 2;			
k++) {			
	2	se(v)	
chroma_bipred_weight_l0[j][i][k]			
	chroma	2	se(v)
_bipred_weight_l1[j][i][k]			
	chroma	2	se(v)
_bipred_offset[j][i][k]			
}			

}		
}		
}		
}		
}		

[0073] Although the fidelity of the weighting and offset parameters could be fixed, i.e. a weighting parameter is uniformly quantized to N bits, non uniform quantization may instead be more efficient and appropriate to better handle different variations within a frame. In one embodiment, a finer fidelity is used for smaller weighting or offset parameters, whereas fidelity is increased for larger parameters.

[0074] In general, encoding a parameter q_{w_k} that provides a mapping to the weighting parameters B_{wQ_k, wQ_k} or even possibly to the differential value dB_{wQ_k} is contemplated, using an equation of the form:

$$B_{wQ_k} = g(q_{w_k}) \text{ with } q_{w_k} = -(1 << (N-1)), \dots, (1 << (N-1))-1 \quad (14)$$

Some possible functions that may use are as follows:

$$g(x) = x^2, \quad (15)$$

$$g(x) = \begin{cases} x & \text{if } x < m \\ x^2 & \text{otherwise} \end{cases}, \quad (16)$$

$$g(x) = (a \cdot x)^2 + b \cdot x + c, \text{ where } a, b, c \text{ are constants} \quad \in \Re \quad (17)$$

$$g(x) = \sum_{i=0}^M (c_n \cdot x)^n, \quad \text{with } M \text{ and } c_n \text{ being constants} \quad (18)$$

$$g(x) = x + (x >> 2)^2, \quad (19)$$

$$g(x) = a^{(x+b)} + c, \text{ where } a, b, c \text{ are constants} \quad (20)$$

etc.

[0075] The function represented by equation 17 (or the more generic equation 18) allows for increased flexibility in terms of representing the fidelity of the weighting parameters. Note also that the operation $a \cdot x$ could indicate not only a multiply, but also an integer or floating point divide (i.e. assume that a is less than 1). Figure 8 is an example of the relationship between the representation and the true value of the weighting parameter.

[0076] In one embodiment, the parameters are determined by performing a pre-analysis of the frame or sequence, generating the distribution of the weighting parameters, and approximating the distribution with $g(x)$ (i.e. using a polynomial approximation).

[0077] Figure 12 illustrates the transform process of the weighting parameters for bi-prediction with controlled fidelity. Referring to Figure 12, the process is the same as in Figure 10, with the exception that each of the transformed weighting parameters output from transform 1001 isn't input directly into variable-length encoder 1002, along with the offset. Instead, they are input into functions, the output of which are input to variable length encoder 1002. More specifically, transform weighting parameter B_{wQ_0} is input into function $g_0(x)$ to produce x_0 . More specifically, transform weighting parameter B_{wQ_1} is input into function $g_1(x)$ to produce x_1 (1202). The offset is also subjected to function $g_0(x)$ to produce x_2 (1203).

[0078] Similarly, the output of decoder 1003 which are x_0 , x_1 and x_2 , are input into functions $g_0^{-1}(x)$ (1204), $g_1^{-1}(x)$ (1205), and $g_0^{-1}(x)$ (1206), which produce the transformed weighting parameters in the offset. Thereafter, these are handled in the same manner as shown in Figure 10.

Examples of an Encoder and a Decoder

[0079] Figure 6 is a block diagram of one embodiment of an encoder. In one embodiment the encoder first divides the incoming bitstream into rectangular arrays, referred to as macroblocks. For each macroblock, the encoder then chooses whether to use intra-frame or inter-frame coding. Intra-frame coding uses only the information contained in the current video frame and produces a compressed result referred to as an I-frame. Intra-frame coding can use information of one or more other frames, occurring before or after the current frame. Compressed results that only use data from previous frames are called P-frames, while those that use data from both before and after the current frame are called B-frames.

[0080] Referring to Figure 6, video 601 is input into the encoder. In the case of encoding a frame without motion compensation, frames of video are input into DCT 603. DCT 603 performs a 2D discrete cosign transform (DCT) to create DCT coefficients. The coefficients are quantized by quantizer 604. In one embodiment, the quantization performed by quantizer 604 is weighted by scaler. In one embodiment, the quantizer scaler parameter QP takes values from 1 to 31. The QP value can be modified in both picture and macroblock levels.

[0081] Thereafter, the quantized coefficients undergo variable length encoding at VLC 605, which generates bitstream 620. In one embodiment, VLC 605 performs entropy coding by using Huffman coding or arithmetic coding.

[0082] Note that, in one embodiment, prior to VLC 605, reordering may be performed in which quantized DCT coefficients are zigzag scanned so that a 2D array of coefficients is converted into a 1D array of coefficients in a manner well-known in the art. This may be followed by run length encoding in which the array of reordered quantized coefficients corresponding to each block is encoded to better represent zero coefficients. In such a case, each nonzero coefficient is encoded as a triplet (last, run, level), where “last” indicates whether this is the final nonzero coefficient in the block, “run” signals the preceding 0 coefficients and “level” indicates the coefficients sign and magnitude.

[0083] A copy of the frames may be saved for use as a reference frame. This is particularly the case of I or P frames. To that end, the quantized coefficients output from quantizer 604 are inverse quantized by inverse quantizer 606. An inverse DCT transform is applied to the inverse quantized coefficients using IDCT 607. The resulting frame data is added to a motion compensated predication from motion compensation (MC) unit 609 in the case of a P frame and then the resulting frame is filtered using loop filter 612 and stored in frame buffer 611 for use as a reference frame. In the case of I frames, the data output from IDCT 607 is not added to a motion compensation prediction from MC unit 609 and is filtered using loop filter 612 and stored in frame buffer 611.

[0084] In the case of a P frame, the P frame is coded with interprediction from previous a I or P frame, which is referred to commonly in the art as the reference frame. In this case, the interprediction is performed by motion estimation (ME) block 610 and motion compensation unit 609. In this case, using the reference frame from frame store 611 and the input video 601, motion estimation unit 610 searches for a location of a region in the reference frame that best matched the current macro block in the current frame. As discussed above, this includes not only determining a displacement, but also determining weighting parameters (WP) and an offset. The displacement between the current macroblock and the compensation region in the reference frame, along with the weighting parameters and the offset is called the motion vector. The motion vectors for motion estimation unit 610 are sent to motion compensation unit 609. At motion compensation unit 609, the prediction is subtracted from the current macroblock to produce a residue macroblock using subtractor 602. The residue is then encoded using DCT 603, quantizer 604 and VLC 605 as described above.

[0085] Motion estimation unit 610 outputs the weighting parameters to VLC 605 for variable length encoding 605. The output of VLC 605 is bitstream 620.

[0086] Figure 7 is a block diagram of one embodiment of a decoder. Referring to Figure 7, bitstream 701 is received by variable length decoder 702 which performs variable length decoding. The output of variable length decoding is sent to inverse quantizer 703 which performs an inverse quantization operation that is the opposite of the quantization performed by quantizer 604. The output of inverse quantizer 703 comprise coefficients that are inverse DCT transformed by IDCT 704 to produce image data. In the case of I frames, the output of IDCT 704 is simply filtered by loop filter 721, stored in frame buffer 722 and eventually output as output 760. In the case of P frames, the image data output from IDCT 704 is added to the prediction from motion compensation unit 710 using adder 705. Motion compensation unit 710 uses the output from variable length decoder 722, which includes the weighting parameters discussed above, as well as reference frames from frame buffer 722. The resulting image data output from adder 705 is filtered using loop filter 721 and stored in frame buffer 722 for eventual output as part of output 760.

[0087] Thus, as set forth above, embodiments of the present invention allows improved representation of weighting parameters for bi-predictive partitions and improved coding efficiency (i.e. in terms of increased PSNR for a given bitrate) in the presence of brightness variations, or/and cross fades.

An Example of a Computer System

[0088] Figure 13 is a block diagram of an exemplary computer system that may perform one or more of the operations described herein. Referring to Figure 13, computer system 1300 may comprise an exemplary client or server computer system. Computer system 1300 comprises a communication mechanism or bus 1311 for communicating information, and a processor 1312 coupled with bus 1311 for processing information. Processor 1312 includes a microprocessor, but is not limited to a microprocessor, such as, for example, Pentium™, PowerPC™, Alpha™, etc.

[0089] System 1300 further comprises a random access memory (RAM), or other dynamic storage device 1304 (referred to as main memory) coupled to bus 1311 for storing information and instructions to be executed by processor 1312. Main memory 1304 also may be used for storing temporary variables or other intermediate information during execution of instructions by processor 1312.

[0090] Computer system 1300 also comprises a read only memory (ROM) and/or other static storage device 1306 coupled to bus 1311 for storing static information and instructions for processor 1312, and a data storage device 1307, such as a magnetic disk or

optical disk and its corresponding disk drive. Data storage device 1307 is coupled to bus 1311 for storing information and instructions.

[0091] Computer system 1300 may further be coupled to a display device 1321, such as a cathode ray tube (CRT) or liquid crystal display (LCD), coupled to bus 1311 for displaying information to a computer user. An alphanumeric input device 1322, including alphanumeric and other keys, may also be coupled to bus 1311 for communicating information and command selections to processor 1312. An additional user input device is cursor control 1323, such as a mouse, trackball, trackpad, stylus, or cursor direction keys, coupled to bus 1311 for communicating direction information and command selections to processor 1312, and for controlling cursor movement on display 1321.

[0092] Another device that may be coupled to bus 1311 is hard copy device 1324, which may be used for marking information on a medium such as paper, film, or similar types of media. Another device that may be coupled to bus 1311 is a wired/wireless communication capability 1325 to communication to a phone or handheld palm device.

[0093] Note that any or all of the components of system 1300 and associated hardware may be used in the present invention. However, it can be appreciated that other configurations of the computer system may include some or all of the devices.

[0094] Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as essential to the invention.

CLAIMS

We claim:

1. An encoding method comprising:
generating weighting parameters for multi-hypothesis partitions;
transforming the weighting parameters; and
coding transformed weighting parameters.

2. An encoder comprising:
a motion estimation unit to generate weighting parameters for multi-hypothesis partitions as part of performing motion compensated prediction;
a transform to transform the weighting parameters into transformed weighting parameters; and
a coder to code transformed weighting parameters.

3. An article of manufacture having one or more recordable media storing instructions thereon which, when executed by a system, cause the system to perform an encoding method comprising:
generating weighting parameters for multi-hypothesis partitions;
transforming the weighting parameters; and
coding transformed weighting parameters.

4. A decoding method comprising:
decoding a bitstream using variable length decoding to obtain decoded data that includes transformed weighting parameters for multi-hypothesis partitions;
applying an inverse transform to the transformed weighting parameters; and
reconstructing frame data using motion compensation in conjunction with inverse transformed weighting parameters.

5. A decoder comprising:

a variable length decoder to decode a bitstream using variable length decoding to obtain decoded data that includes transformed weighting parameters for multi-hypothesis partitions;

an inverse quantizer to perform an inverse quantization operation on a portion of the decoded data to produce coefficients;

a first inverse transform to apply a first inverse transform to the coefficients to produce image data;

a second inverse transform to apply a second inverse transform to the transformed weighting parameters to produce weighting parameters; and

a motion compensation unit to generate a prediction using at least one motion vector and the weighting parameters; and

an adder to add the prediction to the image data to reconstruct frame data.

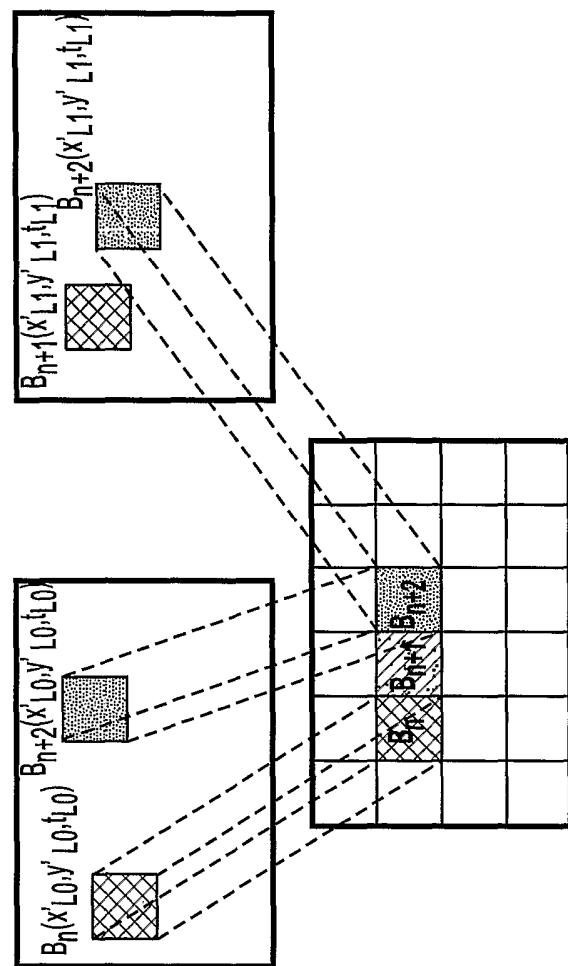


FIG. 1B

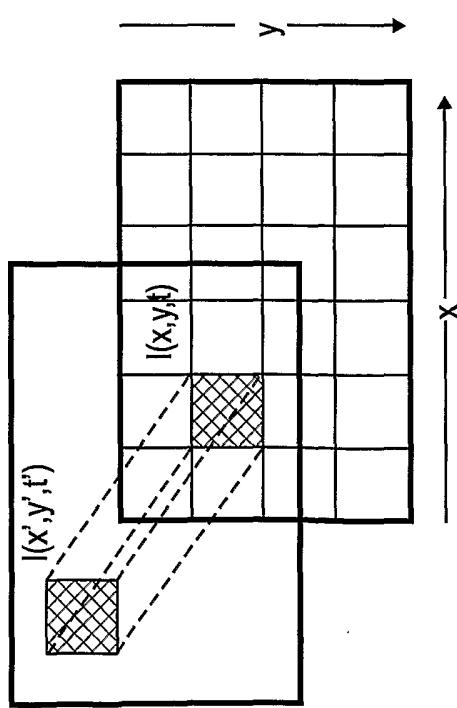


FIG. 1A

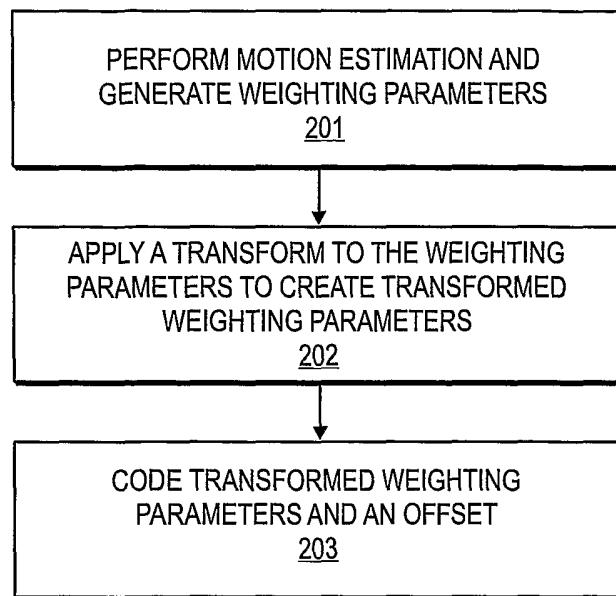


FIG. 2

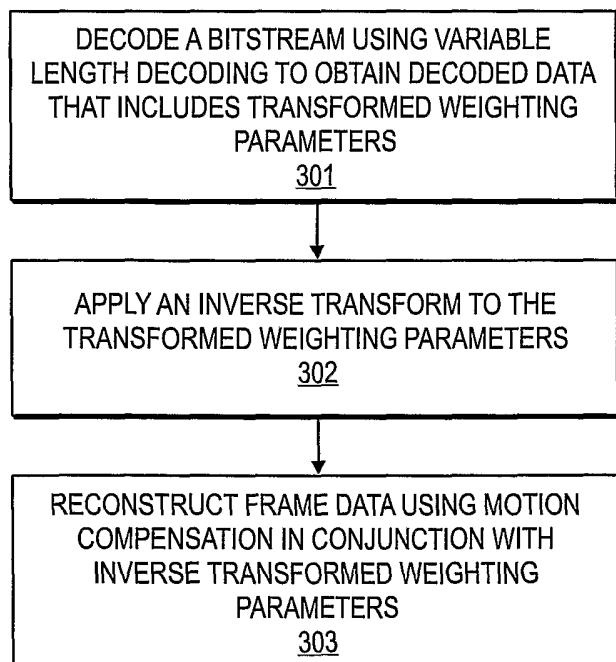
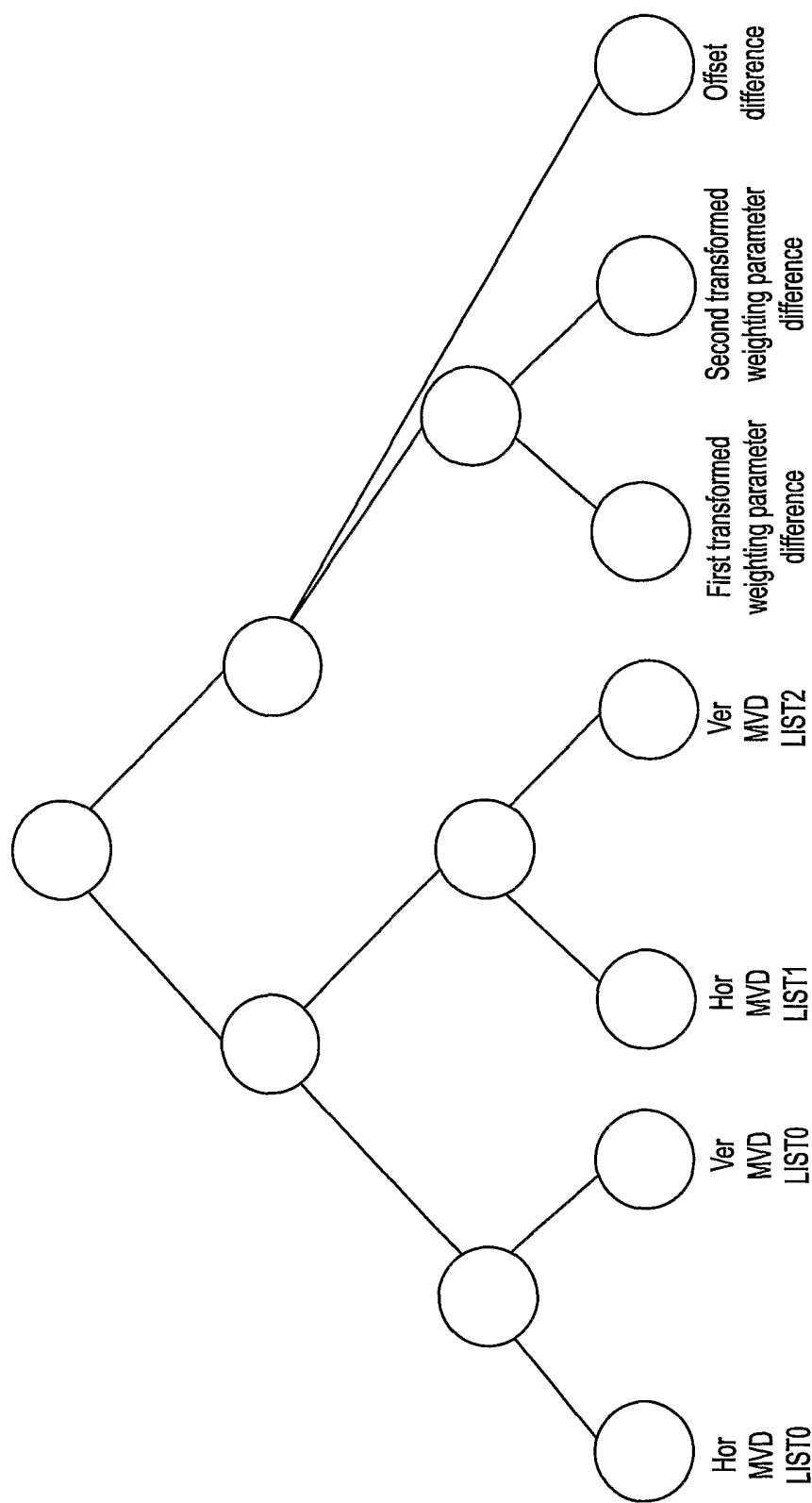


FIG. 3



MVD = displacement difference

FIG. 4

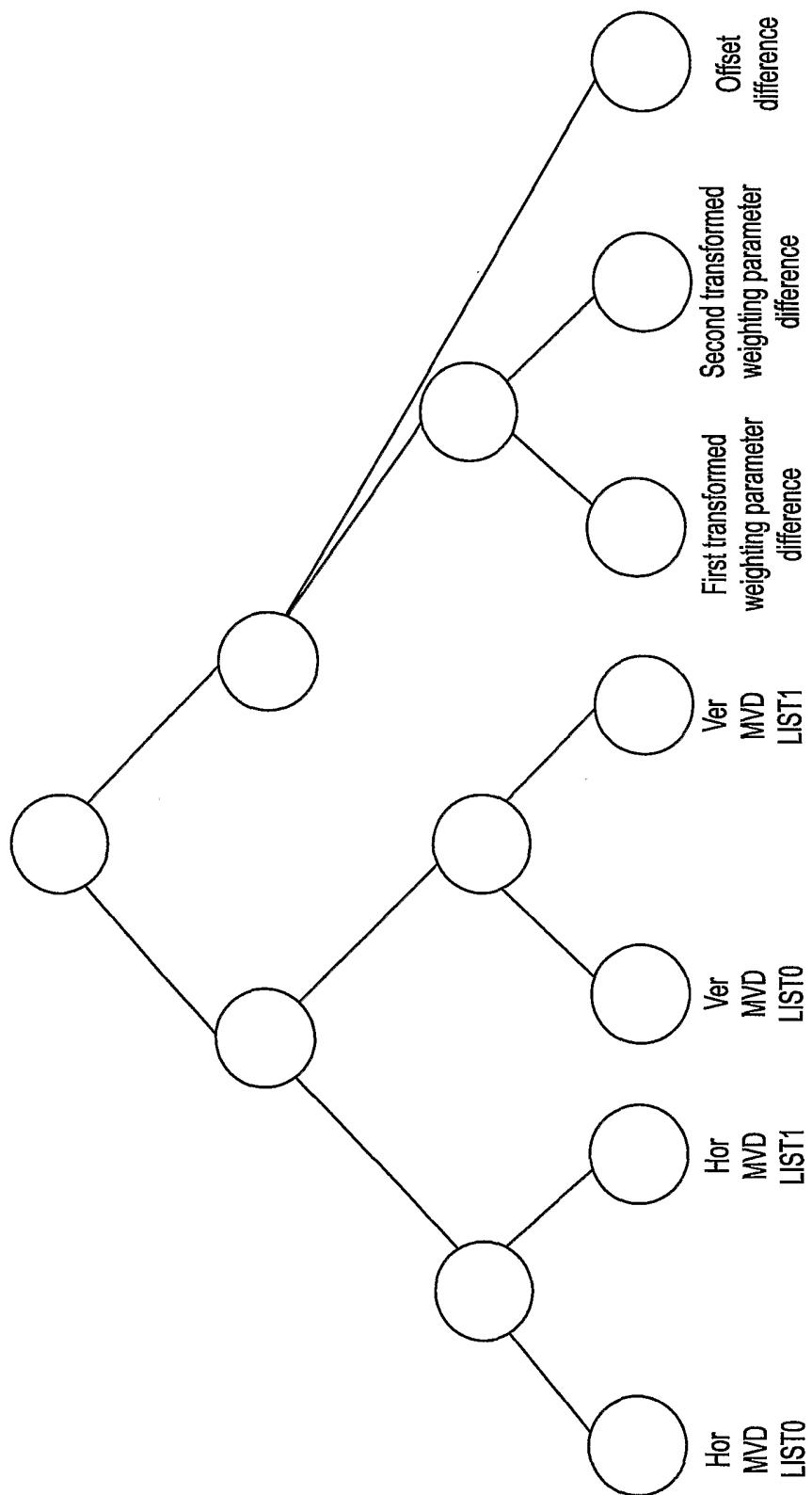


FIG. 5

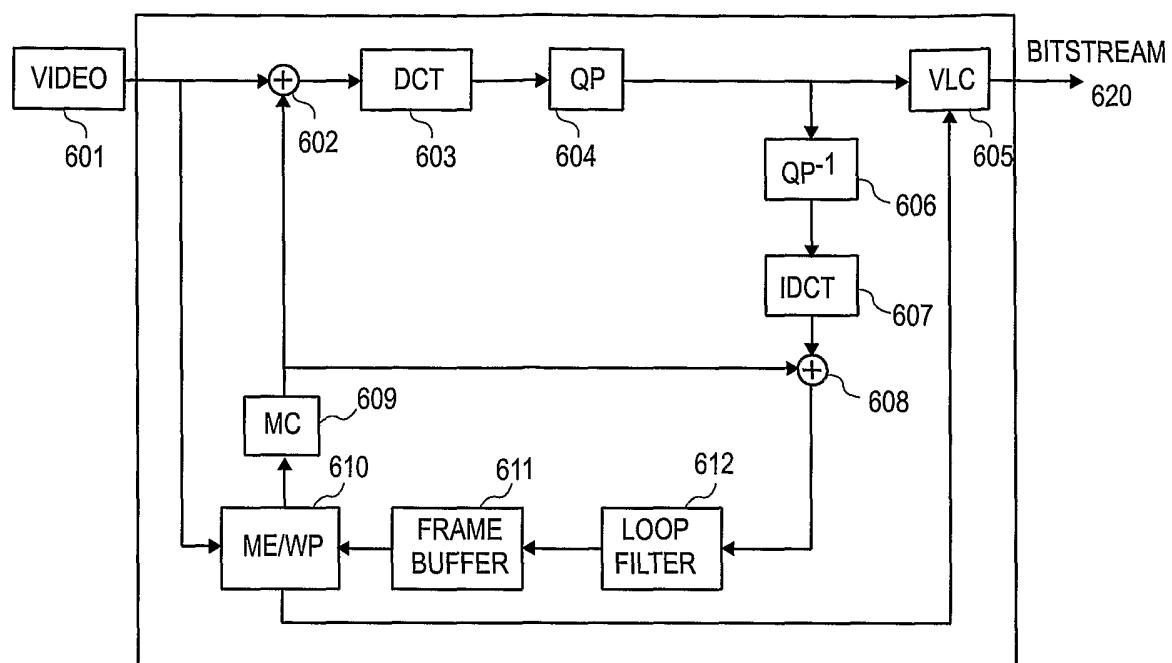


FIG. 6

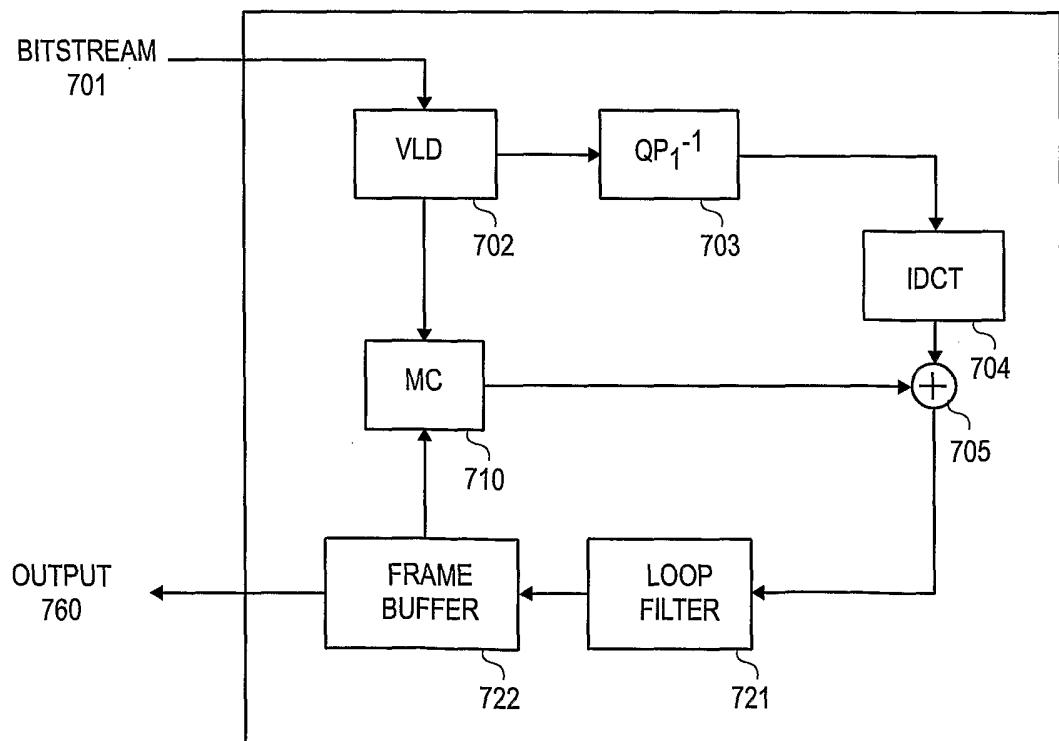


FIG. 7

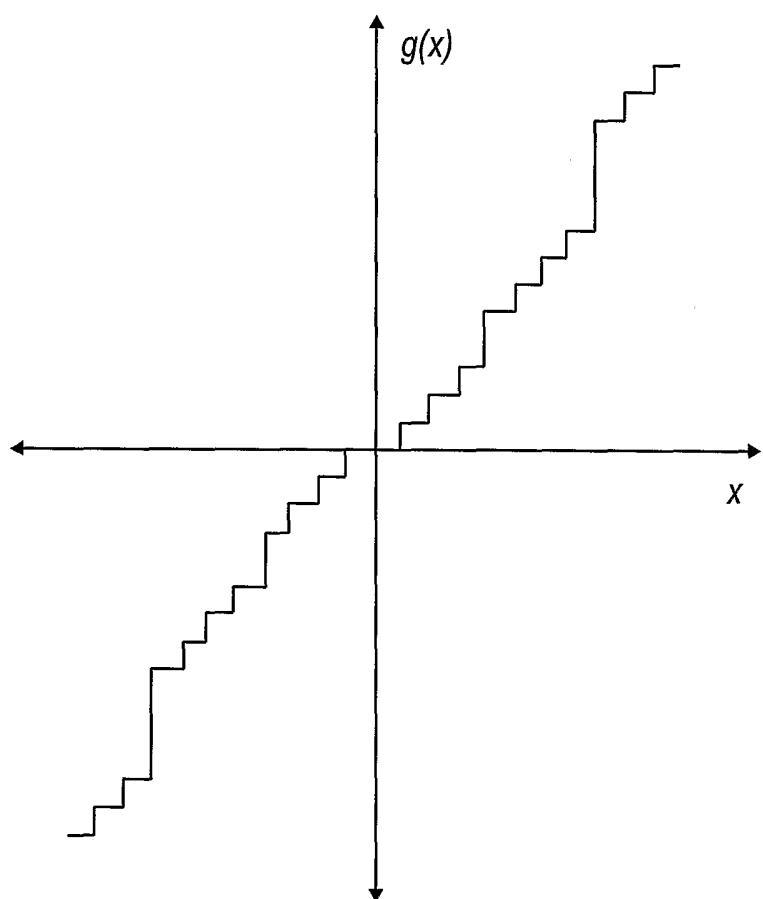
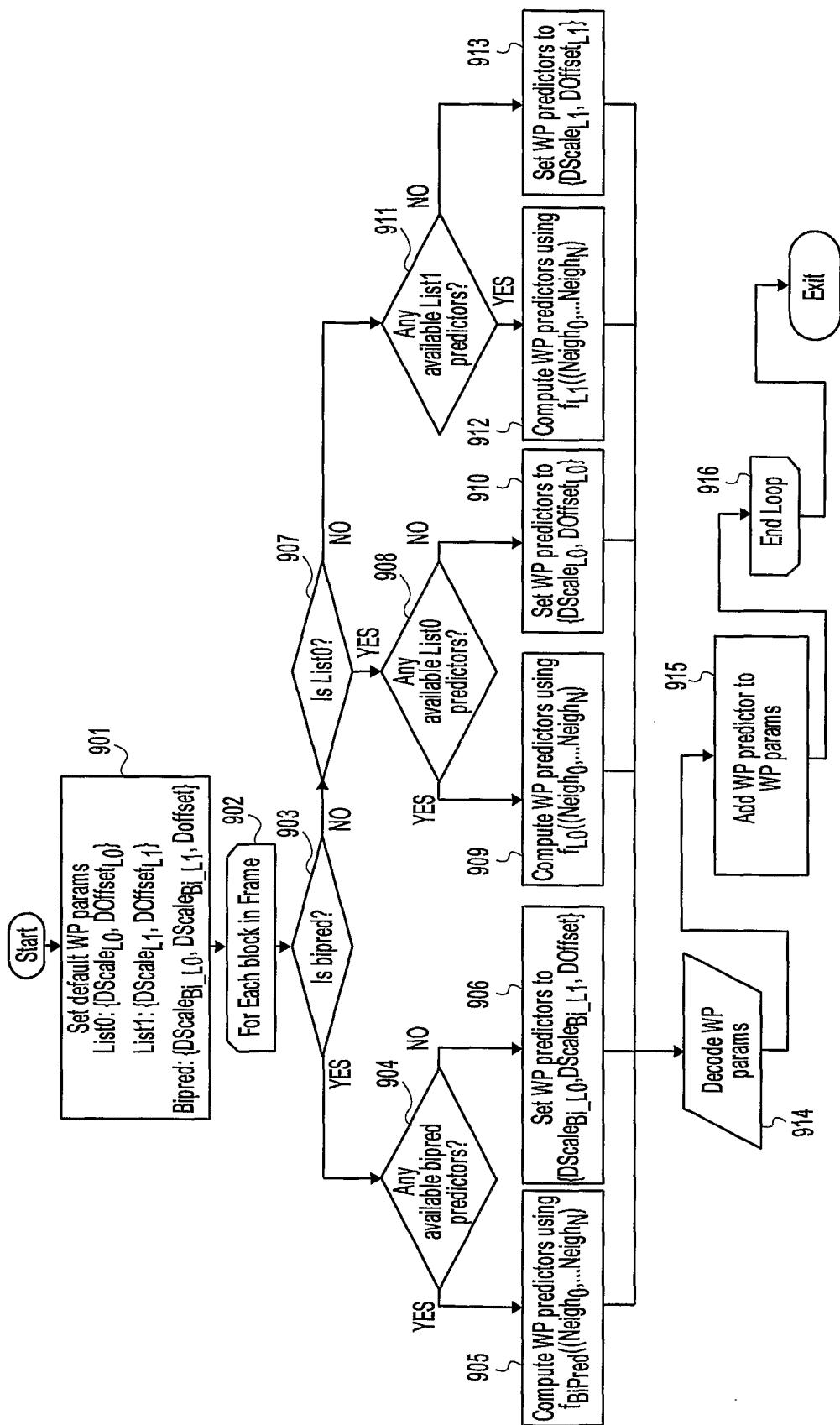


FIG. 8



6
FIG.

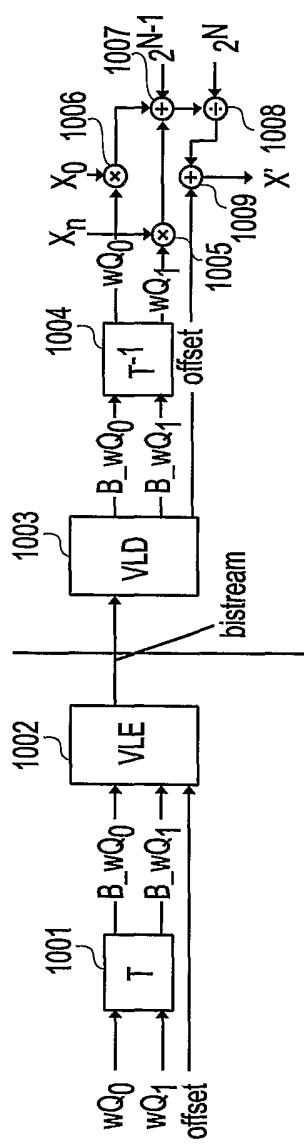


FIG. 10

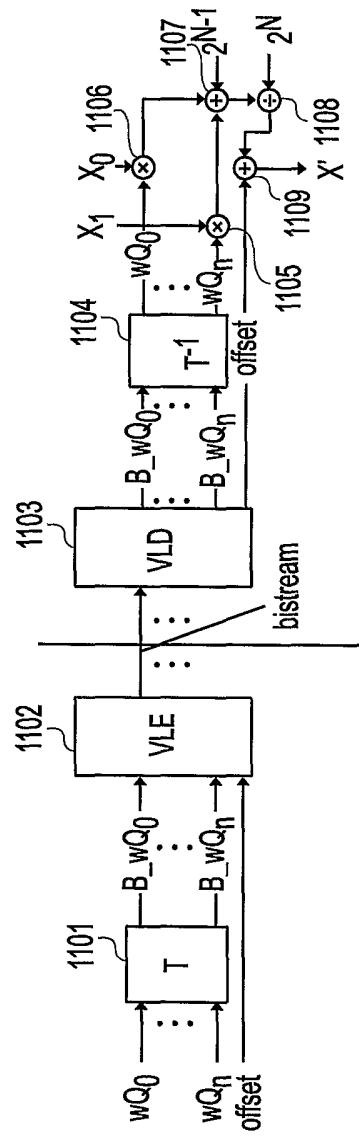


FIG. 11

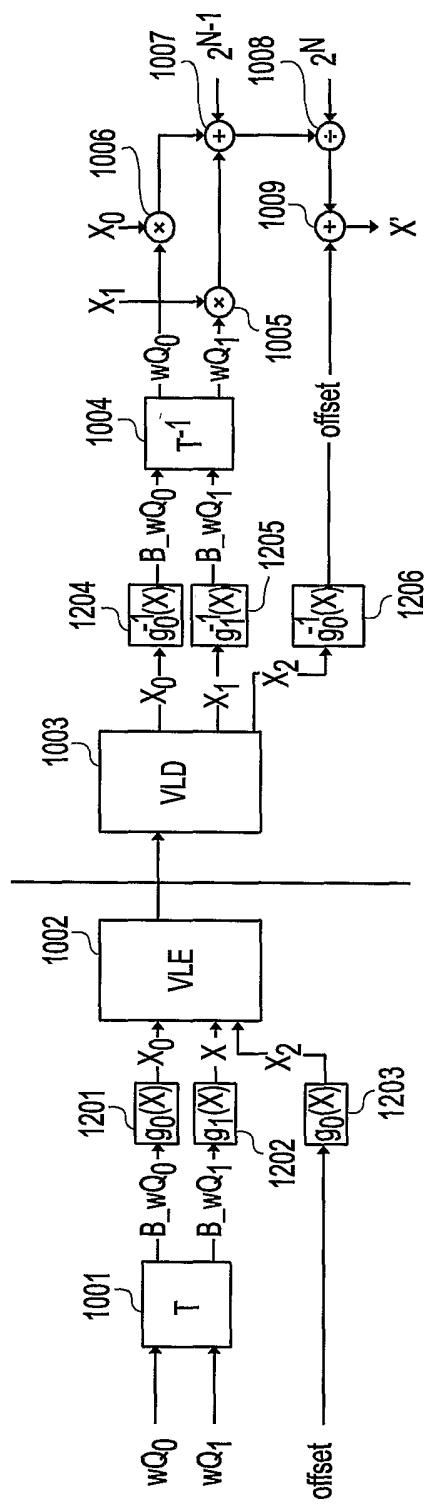


FIG. 12

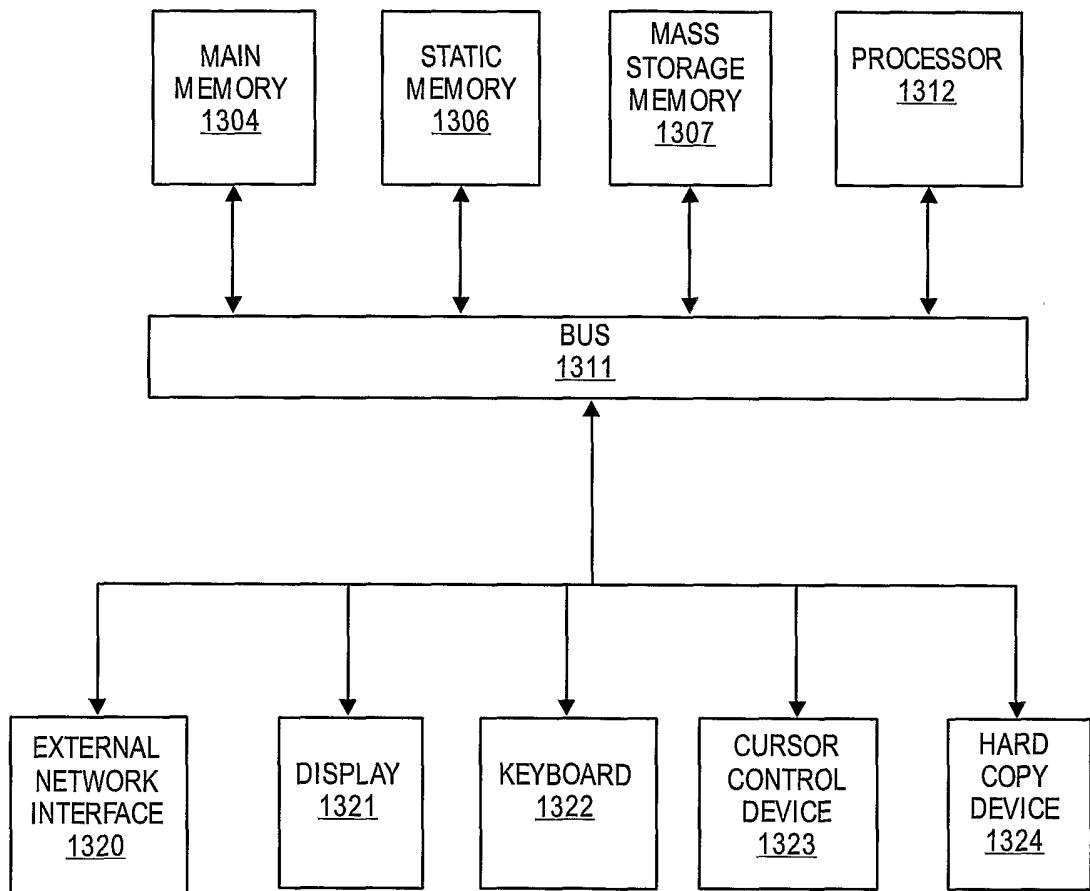
1300

FIG. 13