

IMAGE PROCESSING APPARATUS AND METHOD

TECHNICAL FIELD

5 [0001]

The present disclosure relates to an image processing apparatus and a method, and particularly relates to an image processing apparatus and a method that enable suppression of an increase in memory capacity required for orthogonal transform and inverse orthogonal transform.

10

BACKGROUND ART

[0002]

15 Conventionally, adaptive primary transform (adaptive multiple core transforms: AMT) has been disclosed regarding luminance, in which a primary transform is adaptively selected from a plurality of different orthogonal transforms for each horizontal primary transform PThor (also referred to as primary horizontal transform) and vertical primary transform PTver (also referred to as primary vertical transform) for each transform unit (TU) (for example, see Non-Patent Document 1).

20

25 [0003]

In Non-Patent Document 1, there are five one-dimensional orthogonal transforms of DCT-II, DST-VII, DCT-VIII, DST-I, and DCT-VI as candidates for the primary transform. Furthermore, it has been proposed to add two one-dimensional orthogonal transforms of DST-IV and identity transform (IDT: one-dimensional transform skip),

30

and to have a total of seven one-dimensional orthogonal transforms as candidates for the primary transform (for example, see Non-Patent Document 2).

5 CITATION LIST

NON-PATENT DOCUMENT

[0004]

Non-Patent Document 1: Jianle Chen, Elena Alshina, Gary J. Sullivan, Jens-Rainer, Jill Boyce, "Algorithm Description
10 of Joint Exploration Test Model 4", JVET-G1001_v1, Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 7th Meeting: Torino, IT, 13-21 July 2017

Non-Patent Document 2: V. Lorcy, P. Philippe, "Proposed
15 improvements to the Adaptive multiple Core transform", JVET-C0022, Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 3rd Meeting: Geneva, CH, 26 May-1 June 2016

20 SUMMARY OF THE INVENTION

PROBLEMS TO BE SOLVED BY THE INVENTION

[0005]

However, in the case of these methods, there is a possibility that the size of a look up table (LUT)
25 required to hold all of transformation matrices of the primary transformation increases. That is, in a case of considering hardware implementation of the primary transform, there is a possibility of an increase in the memory size required to hold coefficients of the
30 transformation matrices increases.

[0006]

The present disclosure has been made in view of such a situation, and enables suppression of an increase in memory capacity required for orthogonal transform and inverse orthogonal transform.

5

SOLUTIONS TO PROBLEMS

[0007]

An image processing apparatus according to one aspect of the present technology is an image processing
10 apparatus including a derivation unit configured to derive a transformation matrix, using a submatrix configuring a part of the transformation matrix, an orthogonal transform unit configured to orthogonally transform a prediction residual of an image, using the
15 transformation matrix derived by the derivation unit, and an encoding unit configured to encode coefficient data obtained by orthogonally transforming the prediction residual by the orthogonal transform unit to generate a bit stream.

20 [0008]

An image processing method according to one aspect of the present technology is an image processing method including deriving a transformation matrix using a submatrix configuring a part of the transformation matrix,
25 orthogonally transforming a prediction residual of an image, using the derived transformation matrix, and encoding coefficient data obtained by orthogonally transforming the prediction residual to generate a bit stream.

30 [0009]

An image processing apparatus according to another

aspect of the present technology is an image processing apparatus including a decoding unit configured to decode a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image, a derivation unit configured to derive a transformation matrix, using a submatrix configuring a part of the transformation matrix, and an inverse orthogonal transform unit configured to inversely orthogonally transform the coefficient data obtained by the decoding unit, using the transformation matrix derived by the derivation unit.

[0010]

An image processing method according to another aspect of the present technology is an image processing method including decoding a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image, deriving a transformation matrix, using a submatrix configuring a part of the transformation matrix, and inversely orthogonally transforming the obtained coefficient data, using the derived transformation matrix.

[0011]

In the image processing apparatus and the method according to the one aspect of the present technology, a transformation matrix is derived using a submatrix configuring a part of the transformation matrix, a prediction residual of an image is orthogonally transformed using the derived transformation matrix, and coefficient data obtained by orthogonally transforming the prediction residual is encoded to generate a bit stream.

[0012]

In the image processing apparatus and the method according to the another aspect of the present technology, a bit stream is decoded to obtain coefficient data that
5 is an orthogonally transformed prediction residual of an image, a transformation matrix is derived using a submatrix configuring a part of the transformation matrix, and the obtained coefficient data is inversely orthogonally transformed using the derived transformation
10 matrix.

EFFECTS OF THE INVENTION

[0013]

According to the present disclosure, an image can
15 be processed. In particular, an increase in memory capacity required for orthogonal transform and inverse orthogonal transform can be suppressed.

BRIEF DESCRIPTION OF DRAWINGS

20 [0014]

Fig. 1 is a diagram illustrating a correspondence between a transform set and an orthogonal transform to be selected.

Fig. 2 is a diagram illustrating a correspondence
25 between a type of orthogonal transform and a function to be used.

Fig. 3 is a diagram illustrating a correspondence between a transform set and a prediction mode.

Fig. 4 is a diagram illustrating examples of types
30 of orthogonal transform stored in an LUT.

Fig. 5 is a diagram illustrating examples of an LUT

size required to hold a transformation matrix in HEVC.

Fig. 6 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

Fig. 7 is a diagram for describing an example of
5 similarity between transformation matrices.

Fig. 8 is a diagram for describing examples of transformation types substitutable by flip.

Fig. 9 is a diagram for describing examples of transformation types substitutable by transposition.

10 Fig. 10 is a diagram illustrating a list of main specific examples of substitution of a transformation matrix.

Fig. 11 is a block diagram illustrating a main configuration example of an image encoding device.

15 Fig. 12 is a block diagram illustrating a main configuration example of an orthogonal transform unit.

Fig. 13 is a flowchart for describing an example of a flow of image encoding processing.

Fig. 14 is a flowchart for describing an example of
20 a flow of orthogonal transform processing.

Fig. 15 is a block diagram illustrating a main configuration example of an image decoding device.

Fig. 16 is a block diagram illustrating a main configuration example of an inverse orthogonal transform
25 unit.

Fig. 17 is a flowchart for describing an example of a flow of image decoding processing.

Fig. 18 is a flowchart for describing an example of a flow of inverse orthogonal transform processing.

30 Fig. 19 is a diagram illustrating an example of transform type derivation.

Fig. 20 is a diagram illustrating a specific example of transform type derivation.

Fig. 21 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

5 Fig. 22 is a block diagram illustrating a main configuration example of a primary transform unit.

Fig. 23 is a block diagram illustrating a main configuration example of a primary horizontal transform unit.

10 Fig. 24 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 25 is a block diagram illustrating a main configuration example of a primary vertical transform
15 unit.

Fig. 26 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 27 is a flowchart for describing an example of
20 a flow of primary transform processing.

Fig. 28 is a flowchart for describing an example of a flow of primary horizontal transform processing.

Fig. 29 is a diagram illustrating an example of an operation expression for each element.

25 Fig. 30 is a flowchart for describing an example of a flow of transformation matrix derivation processing.

Fig. 31 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

30 Fig. 32 is a flowchart for describing an example of a flow of primary vertical transform processing.

Fig. 33 is a diagram illustrating an example of an operation expression for each element.

Fig. 34 is a block diagram illustrating a main configuration example of an inverse primary transform
5 unit.

Fig. 35 is a block diagram illustrating a main configuration example of an inverse primary vertical transform unit.

Fig. 36 is a block diagram illustrating a main configuration example of a transformation matrix
10 derivation unit.

Fig. 37 is a block diagram illustrating a main configuration example of an inverse primary horizontal transform unit.

Fig. 38 is a block diagram illustrating a main configuration example of a transformation matrix
15 derivation unit.

Fig. 39 is a flowchart for describing an example of a flow of inverse primary transform processing.

Fig. 40 is a flowchart for describing an example of a flow of inverse primary transform selection processing.
20

Fig. 41 is a flowchart for describing an example of a flow of inverse primary vertical transform processing.

Fig. 42 is a flowchart for describing an example of a flow of inverse primary horizontal transform processing.
25

Fig. 43 is a diagram illustrating an example of transform type derivation.

Fig. 44 is a diagram illustrating a specific example of transform type derivation.

Fig. 45 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.
30

Fig. 46 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 47 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 48 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 49 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

Fig. 50 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 51 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 52 is a diagram illustrating an example of transform type derivation.

Fig. 53 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

Fig. 54 is a flowchart for describing an example of a flow of transformation matrix derivation processing.

Fig. 55 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

Fig. 56 is a diagram illustrating an example of transform type derivation.

Fig. 57 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

Fig. 58 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

Fig. 59 is a diagram illustrating an example of
5 transform type derivation.

Fig. 60 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

Fig. 61 is a flowchart for describing an example of a flow of transformation matrix derivation processing.

10 Fig. 62 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

Fig. 63 is a diagram for describing a spatial symmetric property of two-dimensional orthogonal
15 transform.

Fig. 64 is a diagram for describing a horizontal symmetric property.

Fig. 65 is a diagram for describing a vertical symmetric property.

20 Fig. 66 is a diagram for describing horizontal and vertical symmetric properties.

Fig. 67 is a diagram illustrating a list of main specific examples of substitution of a transformation matrix involving transformation of a prediction residual.

25 Fig. 68 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

Fig. 69 is a block diagram illustrating a main configuration example of a primary transform unit.

Fig. 70 is a flowchart for describing an example of
30 a flow of primary transform processing.

Fig. 71 is a flowchart for describing an example of

a flow of prediction residual permutation operation processing.

Fig. 72 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

Fig. 73 is a block diagram illustrating a main configuration example of an inverse primary transform unit.

Fig. 74 is a flowchart for describing an example of a flow of inverse primary transform processing.

Fig. 75 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

Fig. 76 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 77 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 78 is a flowchart for describing an example of a flow of transformation matrix derivation processing.

Fig. 79 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

Fig. 80 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 81 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 82 is a diagram illustrating examples of assignment of a transform type to a transform type

identifier.

Fig. 83 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

5 Fig. 84 is a diagram for describing derivation of a transformation matrix using a submatrix.

Fig. 85 is a diagram illustrating a list of main specific examples of a transformation matrix derived from a submatrix.

10 Fig. 86 is a diagram illustrating an example of a state of transformation matrix derivation.

Fig. 87 is a diagram illustrating an example of a state of transformation matrix derivation.

15 Fig. 88 is a diagram illustrating an example of a state of transformation matrix derivation.

Fig. 89 is a diagram illustrating an example of a state of transformation matrix derivation.

Fig. 90 is a diagram illustrating an example of a state of transformation matrix derivation.

20 Fig. 91 is a diagram illustrating an example of a state of transformation matrix derivation.

Fig. 92 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

25 Fig. 93 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 94 is a flowchart for describing an example of a flow of transformation matrix derivation processing.

30 Fig. 95 is a diagram illustrating examples of assignment of a transform type to a transform type

identifier.

Fig. 96 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

5 Fig. 97 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 98 is a diagram illustrating examples of an LUT size required to hold a transformation matrix.

10 Fig. 99 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit.

Fig. 100 is a flowchart for describing an example of a flow of transformation matrix derivation processing.

15 Fig. 101 is a diagram illustrating examples of assignment of a transform type to a transform type identifier.

Fig. 102 is a block diagram illustrating a main configuration example of a computer.

20

MODE FOR CARRYING OUT THE INVENTION

[0015]

Hereinafter, modes for implementing the present disclosure (hereinafter referred to as embodiments) will
25 be described. Note that the description will be given in the following order.

1. Adaptive Primary Transform [0016]

2. First Embodiment (derivation of transformation matrix from transformation matrix) [0041]

30

2-1. Common Concept

2-2. Example 1-1

2-3. Example 1-2

2-4. Example 1-3

2-5. Example 1-4

2-6. Example 1-5

5 3. Second Embodiment (prediction residual
transform) [0561]

3-1. Common Concept

3-2. Example 2-1

3-3. Example 2-2

10 3-4. Example 2-3

3-5. Example 2-4

4. Third Embodiment (derivation of transformation
matrix from submatrix) [0734]

4-1. Common Concept

15 5. Fourth Embodiment (combination of embodiments)
[0840]

5-1. Common Concept

6. Appendix

[0016]

20 <1. Adaptive Primary Transform>

<Documents that support technical content and
technical terms>

The range disclosed by the present technology
includes not only the content described in the examples
25 but also the content described in the following non-
patent documents that are known at the time of filing.

[0017]

Non-Patent Document 1: (described above)

Non-Patent Document 3: TELECOMMUNICATION

30 STANDARDIZATION SECTOR OF ITU (International
Telecommunication Union), "Advanced video coding for

generic audiovisual services", H.264, 04/2017

Non-Patent Document 4: TELECOMMUNICATION
STANDARDIZATION SECTOR OF ITU (International
Telecommunication Union), "High efficiency video coding",
5 H.265, 12/2016
[0018]

That is, the content described in the above-
mentioned non-patent documents also serves as a basis for
determining the support requirements. For example, the
10 quad-tree block structure described in Non-Patent
Document 4 and the quad tree plus binary tree (QTBT)
block structure described in Non-Patent Document 1 fall
within the disclosure range of the present technology
even in the case where these pieces of content are not
15 directly described in the examples, and satisfy the
support requirements of the claims. Furthermore, for
example, technical terms such as parsing, syntax, and
semantics are similarly fall within the disclosure range
of the present technology even in the case where these
20 technical terms are not directly described in the
examples, and satisfy the support requirements of claims.
[0019]

Furthermore, in the present specification, a "block"
(not a block indicating a processing unit) used for
25 description as a partial region or a unit of processing
of an image (picture) indicates an arbitrary partial
region in a picture unless otherwise specified, and the
size, shape, characteristics, and the like of the block
are not limited. For example, the "block" includes an
30 arbitrary partial region (unit of processing) such as a
transformation block (TB), a transform unit (TU), a

prediction block (PB), a prediction unit (PU), a smallest coding unit (SCU), a coding unit (CU), a largest coding unit (LCU), a coding tree block (CTB), a coding tree unit (CTU), a transformation block, a subblock, a macro block, a tile, or a slice, described in Non-Patent Documents 1, 3, and 4.

[0020]

Furthermore, in specifying the size of such a block, not only the block size is directly specified but also the block size may be indirectly specified. For example, the block size may be specified using identification information for identifying the size. Furthermore, for example, the block size may be specified by a ratio or a difference from the size of a reference block (for example, an LCU, an SCU, or the like). For example, in a case of transmitting information for specifying the block size as a syntax element or the like, information for indirectly specifying the size as described above may be used as the information. With the configuration, the amount of information can be reduced, and the coding efficiency can be improved in some cases. Furthermore, the specification of the block size also includes specification of a range of the block size (for example, specification of a range of allowable block sizes, or the like).

[0021]

<Adaptive Primary Transform>

In the test model (Joint Exploration Test Model 4 (JEM4)) described in Non-Patent Document 1, adaptive primary transform (adaptive multiple core transforms (AMT)) is disclosed, in which a primary transform is

adaptively selected from a plurality of different one-dimensional orthogonal transforms for each horizontal primary transform PThor (also referred to as primary horizontal transform) and vertical primary transform
 5 PTver (also referred to as primary vertical transform) regarding a luminance transformation block.

[0022]

Specifically, regarding the luminance transformation block, in a case where an adaptive primary
 10 transform flag apt_flag indicating whether or not to perform adaptive primary transform is 0 (false), discrete cosine transform (DCT)-II or discrete sine transform (DST)-VII is uniquely determined by mode information as primary transform, as in the table (LUT_TrSetToTrTypIdx)
 15 illustrated in Fig. 1, for example (TrSetIdx = 4).

[0023]

In a case where the adaptive primary transform flag apt_flag is 1 (true) and a current coding unit (CU) including the luminance transformation block to be
 20 processed is an intra CU, a transform set TrSet including orthogonal transform serving as a primary transform candidate is selected for each of a horizontal direction (x direction) and a vertical direction (y direction) from among three transform sets TrSet (TrSetIdx = 0, 1, and 2)
 25 illustrated in Fig. 1, as in the table illustrated in Fig. 1. Note that DST-VII, DCT-VIII, and the like illustrated in Fig. 1 indicate types of orthogonal transform, and functions such as those illustrated in the table in Fig. 2 are used.

30 [0024]

The transform set TrSet is uniquely determined on

the basis of (intra prediction mode information of) a correspondence table of the mode information and the transform set illustrated in Fig. 3. For example, a transform set identifier TrSetIdx for specifying a
 5 corresponding transform set TrSet is set for each of transform sets TrSetH and TrSetV, as in the following expressions (1) and (2).

[0025]

[Math. 1]

TrSetH = LUT_IntraModeToTrSet[IntraMode][0] . . . (1)

10 TrSetV = LUT_IntraModeToTrSet[IntraMode][1] . . . (2)

[0026]

Here, TrSetH represents a transform set of the primary horizontal transform PThor, TrSetV represents a transform set of the primary vertical transform PTver,
 15 and the lookup table LUT_IntraModeToTrSet represents the correspondence table in Fig. 3. The first array of the lookup table LUT_IntraModeToTrSet [][] has an intra prediction mode IntraMode as an argument, and the second array has {H = 0, V = 1} as an argument.

20 [0027]

For example, in a case of the intra prediction mode number 19 (IntraMode == 19), a transform set of the transform set identifier TrSetIdx = 0 illustrated in the table in Fig. 1 is selected as the transform set TrSetH
 25 of the primary horizontal transform PThor (also referred to as primary horizontal transform set), and a transform set of the transform set identifier TrSetIdx = 2 illustrated in the table in Fig. 1 is selected as the transform set TrSetV of the primary vertical transform

PTver (also referred to as primary horizontal transform set).

[0028]

Note that, in a case where the adaptive primary
 5 transform flag `apt_flag` is 1 (true) and the current CU
 including the luminance transformation block to be
 processed is an inter CU, the transform set `InterTrSet`
 (`TrSetIdx = 3`) dedicated to inter CU is assigned to the
 transform set `TrSetH` of primary horizontal transform and
 10 the transform set `TrSetV` of primary vertical transform.

[0029]

Next, which orthogonal transform of the selected
 transform sets `TrSet` is applied is selected according to
 a corresponding specification flag between a primary
 15 horizontal transform specification flag `pt_hor_flag` and a
 primary vertical transform specification flag `pt_ver_flag`,
 for each of the horizontal direction and the vertical
 direction.

[0030]

For example, a transform set is derived from a
 transform set definition table (`LUT_TrSetToTrTypeIdx`)
 illustrated in Fig. 1, using the primary {horizontal,
 vertical} transform set `TrSet {H, V}` and the primary
 {horizontal, vertical} transfer specification flag
 25 `pt_{hor, ver}_flag` as arguments, as in the following
 expressions (3) and (4).

[0031]

[Math. 2]

$$\text{TrTypeIdxH} = \text{LUT_TrSetToTrTypeIdx}[\text{TrSetH}][\text{pt_hor_flag}] \quad \cdot \cdot \cdot \quad (3)$$

$$\text{TrTypeIdxV} = \text{LUT_TrSetToTrTypeIdx}[\text{TrSetV}][\text{pt_ver_flag}] \quad \cdot \cdot \cdot \quad (4)$$

30 [0032]

For example, in a case of the intra prediction mode number 34 (`IntraMode == 34`) (that is, the primary horizontal transform set `TraSetH` is 0) and the primary horizontal transform specification flag `pt_hor_flag` is 0, the value of the transform type identifier `TrTypeIdxH` of the expression (3) is 4 from the transform set definition table (`LUT_TrSetToTrTypeIdx`) in Fig. 1, and the transform type identifier `TrTypeH` corresponding to the value of the transform type identifier `TrTypeIdxH` is DST-VII by reference to Fig. 2. That is, DST-VII of the transform set with the transform set identifier `TrSetIdx` of 0 is selected as the transform type of the primary horizontal transform `PThor`. Furthermore, in the case where the primary horizontal transform specification flag `pt_hor_flag` is 1, DCT-VIII is selected as the transform type. Note that selecting the transform type `TrType` includes selecting a transform type specified with the transform type identifier `TrTypeIdx` via the transform type identifier `TrTypeIdx`.

[0033]

Note that a primary transform identifier `pt_idx` is derived from the primary horizontal transform specification flag `pt_hor_flag` and the primary vertical transform specification flag `pt_ver_flag` on the basis of the following expression (5). That is, an upper 1 bit of the primary transform identifier `pt_idx` corresponds to the value of the primary vertical transform specification flag, and a lower 1 bit corresponds to the value of the primary horizontal transform specification flag.

[0034]

[Math. 3]

$pt_idx = (pt_ver_flag \ll 1) + pt_hor_flag \quad \cdot \cdot \cdot \quad (5)$

[0035]

Encoding is performed by applying arithmetic coding to a derived bin string of the primary transform

5 identifier pt_idx to generate a bit string. Note that the adaptive primary transform flag apt_flag and the primary transform identifier pt_idx are signaled in the luminance transformation block.

[0036]

10 As described above, Non-Patent Document 1 proposes five one-dimensional orthogonal transforms of DCT-II (DCT2), DST-VII (DST7), DCT-VIII (DCT8), DST-I (DST1), and DCT-V (DCT5) as primary transform candidates. Furthermore, Non-Patent Document 2 proposes two one-
15 dimensional orthogonal transforms of DST-IV (DST4) and identity transform (IDT: one-dimensional transform skip) in addition to the above to have a total of seven one-dimensional orthogonal transforms as primary transform candidates.

20 [0037]

That is, in the case of Non-Patent Document 1, as illustrated in Fig. 4, one-dimensional orthogonal transforms are stored in the LUT as the primary transform candidates. Furthermore, in the case of Non-Patent
25 Document 2, DST-IV (DST4) and IDT are further stored in the LUT in addition to the above (see Fig. 4).

[0038]

In a case of high efficiency video coding (HEVC), the size of the lookup table (LUT) required to hold a
30 transformation matrix is as illustrated in the table in

Fig. 5. That is, the size of the LUT is about 1.3 KB in total. In contrast, in the case of the method described in Non-Patent Document 1, DCT2 needs to hold a transformation matrix for each size of 2/4/8/16/32/64/128 points on the LUT, for example. Furthermore, other one-dimensional transforms (DST7/DST1/DCT8) need to hold a transformation matrix for each size of 4/8/16/32/64 points on the LUT. In this case, assuming that bit precision of each coefficient of a transformation matrix is 10 bits, the size of the LUT required to hold all of transformation matrices of the primary transformation is illustrated in A in Fig. 6. That is, the size of the LUT in this case is about 53 KB in total. That is, the size of the LUT in this case increases about 50 times of the case of HEVC.

[0039]

Similarly, in the case of the method described in Non-Patent Document 2, the size of the LUT required to hold all of transformation matrices of the primary transform is as illustrated in the table in B in Fig. 6. That is, the size of the LUT in this case is about 67 KB in total. That is, the size of the LUT in this case increases about 60 times of the case of HEVC.

[0040]

In a case of considering hardware implementation of the primary transform, the size of the LUT is reflected in storage capacity (memory capacity). That is, in the case of the methods described in Non-Patent Documents 1 and 2, there is a possibility of an increase in a circuit scale (memory capacity required to hold coefficients of the transformation matrix) by about 50 to 60 times of the

case of HEVC.

[0041]

<2. First Embodiment>

<2-1. Common Concept>

5 <Derivation of Transformation Matrix>

 Therefore, a second transformation matrix is
derived using a first transformation matrix, a prediction
residual of an image is orthogonally transformed using
the derived second transformation matrix, and coefficient
10 data obtained by orthogonally transforming the prediction
residual is encoded to generate a bit stream.

[0042]

 For example, an image processing apparatus includes
a derivation unit configured to derive a second
15 transformation matrix using a first transformation matrix,
an orthogonal transform unit configured to orthogonally
transform a prediction residual of an image, using the
second transformation matrix derived by the derivation
unit, and an encoding unit configured to encode
20 coefficient data obtained by orthogonally transforming
the prediction residual by the orthogonal transform unit
to generate a bit stream.

[0043]

 With the configuration, the transformation matrix
25 can be derived from another transformation matrix.
Therefore, an increase in the number of transformation
matrices prepared for the orthogonal transform can be
suppressed, and an increase in memory capacity required
for the orthogonal transform can be suppressed.

30 [0044]

 Furthermore, a bit stream is decoded to obtain

coefficient data that is an orthogonally transformed prediction residual of an image, a second transformation matrix is derived using a first transformation matrix, and the obtained coefficient data is inversely
5 orthogonally transformed using the derived second transformation matrix.

[0045]

For example, the image processing apparatus includes a decoding unit configured to decode a bit
10 stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image, a derivation unit configured to derive a second transformation matrix, using a first transformation matrix, and an inverse orthogonal transform unit configured to inversely
15 orthogonally transform the coefficient data obtained by the decoding unit, using the second transformation matrix derived by the derivation unit.

[0046]

With the configuration, the transformation matrix
20 can be derived from another transformation matrix. Therefore, an increase in the number of transformation matrices prepared for the inverse orthogonal transform can be suppressed, and an increase in memory capacity required for the inverse orthogonal transform can be
25 suppressed.

[0047]

<Characteristics of Transformation Matrix>

One of main roles of the transformation matrix is to bias a signal of a low-order (particularly 0-order)
30 frequency component in a DC component direction, and how to collect frequency components is an important

characteristic. A waveform component of a low-order (particularly 0-order) base vector (row vector) is important for how to bias the frequency component. That is, transformation matrices having a similar tendency in the waveform component of the base vector can expect similar performance for orthogonal transform/inverse orthogonal transform (how to bias the frequency component is similar).

[0048]

Therefore, attention is paid to the waveform of the low-order (particularly 0-order) base vector (row vector) of the transformation matrix. For example, in a transformation matrix 30 in Fig. 7, it is assumed that the waveform of a low-order (particularly 0-order) row vector (tendency of values of elements) in a frame 31 is illustrated as a graph 32.

[0049]

The graph 32 illustrates (the tendency that) the value of the element of the frequency component becomes lower toward a left side and (the tendency that) the value of the element of the frequency component becomes higher toward a right side. Furthermore, the graph 32 illustrates that the value becomes larger toward an upper side and the value becomes lower toward a lower side in Fig. 7. Note that the center in the up-down direction in the graph 32 illustrates 0, and an upper side of the center illustrates a positive value and a lower side of the center illustrates a negative value.

[0050]

A waveform 32A in the graph 32 illustrates the waveform of the 0-order row vector of the transformation

matrix 30. As illustrated in the waveform 32A, in this case, the 0-order row vector of the transformation matrix 30 has a tendency that the value becomes larger from low frequency components to high frequency components.

5 [0051]

Furthermore, for example, in the transformation matrix 30 in Fig. 7, it is assumed that the waveform of a low-order (particularly 0-order) column vector (tendency of values of elements) in a frame 33 is illustrated as a graph 34. The graph 34 illustrates (the tendency that) the value of the element of the frequency component becomes lower toward an upper side and (the tendency that) the value of the element of the frequency component becomes higher toward a lower side in Fig. 7.

15 Furthermore, the graph 34 illustrates that the value becomes larger toward a more left side and the value becomes lower toward a more right side in Fig. 7. Note that the center in the right-left direction in the graph 34 illustrates 0, and a left side of the center illustrates a positive value and a right side of the center illustrates a negative value.

[0052]

A waveform 34A in the graph 34 illustrates the waveform of the 0-order column vector of the transformation matrix 30. As illustrated in the waveform 34A, in this case, the 0-order column vector of the transformation matrix 30 has a peak at an intermediate frequency component (that is, has a tendency that the value becomes smaller toward a lower frequency component on the low frequency side, and the value becomes smaller toward a higher frequency component at the high frequency

side.

[0053]

Note that in the present specification, the waveform of the 0-order column vector of the transformation matrix 30 may be illustrated in a transposed state as illustrated as a graph 35. The structure of the graph 35 is similar to that of the graph 32. A waveform 35A is equivalent to the waveform 34A.

[0054]

As described above, transformation matrices having similar waveforms of low-order (particularly 0-order) base vectors (row vectors) have similar performance. In other words, a transformation matrix can be substituted by another transformation matrix having a similar waveform of a low-order (particularly 0-order) base vector (row vector). Therefore, an increase in the number of transformation matrices to be stored in the LUT can be suppressed by using the above characteristic.

[0055]

Here, attention is paid to the transform types described in Non-Patent Document 1 and Non-Patent Document 2, the waveforms of the 0-order row vectors and the 0-order column vectors of the transformation matrices of these transform types can be classified into four types. Fig. 8 illustrates examples.

[0056]

The first type is a flat type. The flat type is a waveform type in which the value is substantially uniform in each frequency component. The second type is an increasing type. The increasing type is a waveform type in which the value tends to increase from a low frequency

component toward a high frequency component. The third type is a decreasing type. The decreasing type is a waveform type in which the value tends to decrease from a low frequency component toward a high frequency component.

5 The fourth type is a chevron type. The chevron type is a waveform type in which the value tends to have a peak (maximum value) in a middle. That is, in the case of the chevron-type, the value of the waveform tends to decrease toward a lower frequency component on the low frequency
10 component side, and the value tends to decrease toward a higher frequency component on the high frequency component side.

[0057]

Note that each of these types exhibits an
15 approximate shape of the waveform, and does not need to completely match. For example, in the case of the increasing type, the waveform does not need to strictly monotonically increase from the low frequency side to the high frequency side as long as the waveform as a whole
20 tends to increase in the value from the low frequency side to the high frequency side.

[0058]

Similarly, in the case of the decreasing type, the waveform does not need to strictly monotonically decrease
25 from the low frequency side to the high frequency side as long as the waveform as a whole tends to decrease in the value from the low frequency side to the high frequency side.

[0059]

30 Similarly, in the case of the chevron type, the value of the waveform does not need to monotonically

decrease in directions away from the peak at both sides of the peak as long as the waveform as a whole has the peak (maximum value) near the center, and tends to decrease in values on the both sides in the directions
5 away from the peak. Furthermore, the peak does not need to be formed by one component. For example, approximate position and value of the peak may be able to be specified from a plurality of components. Furthermore, the position of the peak does not need to be exactly at
10 the center.

[0060]

Similarly, in the case of the flat type, the waveform does not need to be strictly flat as long as the waveform as a whole is substantially uniform in the value.
15 That is, there may be some variation in the value. In other words, a waveform that cannot be classified into the other three types may be classified into the flat type.

[0061]

20 The above waveform classification is an example, and the classification is not limited to the above-described example. That is, waveforms may be classified into types other than those described above, and the number of types to be classified is arbitrary and is not
25 limited to the above four types. Note that this classification is performed for convenience of description of the present technology, and is not performed as actual processing.

[0062]

30 According to this classification, as illustrated in Fig. 8, the waveform of the 0-order row vector of the

transformation matrix of DCT2 is classified into the flat type, and the waveform of the 0-order column vector is classified into the decreasing type. Furthermore, the waveform of the 0-order row vector of the transformation matrix of DST7 is classified into the increasing type, and the waveform of the 0-order column vector is classified into the chevron type. Furthermore, the waveform of the 0-order row vector of the transformation matrix of DCT8 is classified into the decreasing type, and the waveform of the 0-order column vector is classified into the decreasing type. Furthermore, the waveform of the 0-order row vector of the transformation matrix of DCT5 is classified into the flat type, and the waveform of the 0-order column vector is classified into the flat type. Note that the waveform of the 0-order row vector of the transformation matrix of DST4 is classified into the increasing type, and the waveform of the 0-order column vector is classified into the increasing type.

[0063]

As described above, the transformation matrix can be substituted by another transformation matrix having a similar waveform of the 0-order row vector. That is, the transformation matrices can be substituted by each other between the transform types having the same type of waveforms of the 0-order row vectors.

[0064]

That is, when deriving the second transformation matrix, using the above-described first transformation matrix, the derivation unit may derive the second transformation matrix in which the lowest-order row vector has a desired type of waveform. With the

configuration, an increase in the number of transformation matrices prepared for the orthogonal transform/inverse orthogonal transform can be suppressed, and an increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform can be suppressed.

[0065]

For example, the derivation unit may derive the second transformation matrix in which the lowest-order row vector has the flat-type waveform, using the first transformation matrix. With the configuration, a transformation matrix having the flat-type waveform of the lowest-order row vector can be substituted by the derived second transformation matrix. Furthermore, for example, the derivation unit may derive the second transformation matrix in which a lowest-order row vector has an increasing-type waveform, using the first transformation matrix. With the configuration, a transformation matrix having the increasing-type waveform of the lowest-order row vector can be substituted by the derived second transformation matrix.

[0066]

Furthermore, for example, the derivation unit may derive the second transformation matrix in which a lowest-order row vector has a decreasing-type waveform, using the first transformation matrix. With the configuration, a transformation matrix having the decreasing-type waveform of the lowest-order row vector can be substituted by the derived second transformation matrix. Furthermore, for example, the derivation unit may derive the second transformation matrix in which a

lowest-order row vector has a chevron-type waveform,
using the first transformation matrix. With the
configuration, a transformation matrix having the
chevron-type waveform of the lowest-order row vector can
5 be substituted by the derived second transformation
matrix.

[0067]

For example, in Fig. 8, all the DST7, DST4, DST8,
and DST3 have the increasing-type waveforms of the 0th-
10 order row vectors, the transformation matrices can be
substituted by one another. That is, the transformation
matrices can be substituted even if the transformation
matrices have different transformation types.

[0068]

15 That is, when deriving the second transformation
matrix, using the above-described first transformation
matrix, the derivation unit may derive the second
transformation matrix of a transform type different from
that of the first transformation matrix. With the
20 configuration, an increase in the number of
transformation types prepared for the orthogonal
transform/inverse orthogonal transform can be suppressed,
and an increase in the memory capacity required for the
orthogonal transform/inverse orthogonal transform can be
25 suppressed.

[0069]

Note that, in the derivation, the derivation unit
may derive the second transformation matrix having the
same number of rows and the same number of columns as the
30 first transformation matrix. In a case of changing the
number of rows and columns, there is a possibility that

the waveform type unintentionally changes. Therefore, by setting the number of rows and columns to be the same as those of the first transformation matrix, the possibility of an unintended change in the waveform type can be suppressed and the second transformation matrix can be more easily derived.

[0070]

Note that an operation for elements can be easily performed for a matrix. An example of the operation for elements of a matrix includes rearrangement of elements and the like. More specifically, for example, in a matrix, the arrangement order of an element group can be flipped (inverted) in a predetermined direction, or the element group can be transposed to interchange rows and columns. Note that transposition is equivalent to flip (inversion) around a diagonal line connecting an upper left end and a lower right end of the matrix. That is, transposition can be said to be a part of flip. Furthermore, it is also easy to invert the sign of each element (from positive to negative or from negative to positive).

[0071]

By using such an operation, (the type of) the waveform of the 0th-order row vector can be intentionally changed. For example, when a matrix having the increasing-type waveform of the 0-order row vector is flipped in the row direction, the waveform of the 0-order row vector changes to the decreasing type. Conversely, when a matrix having the decreasing-type waveform of the 0-order row vector is flipped in the row direction, the waveform of the 0-order row vector changes to the

increasing type.

[0072]

That is, as illustrated in Fig. 8, by flipping the transformation matrix in which the waveform of the 0-order row vector is the increasing transform type (for example, DST7, DST4, DST8, or DST3) in the row direction, a transformation matrix that can substitute the transformation matrix in which the waveform of the 0-order row vector is the decreasing transform type (for example, DCT7, DCT4, DCT8, or DCT3) can be obtained.

[0073]

Furthermore, for example, when a matrix is transposed, the waveform type of the 0-order row vector and the waveform type of the 0-order column vector are interchanged. That is, by transposition, the waveform of the 0-order row vector of the matrix becomes the same type as the waveform of the 0-order column vector of the matrix before transposition.

[0074]

For example, as illustrated in Fig. 9, by transposing the transformation matrix of DCT2 (DCT6) in which the waveform of the 0-order column vector is the decreasing type, a transformation matrix that can substitute the transformation matrix in which the waveform of the 0-order row vector is the decreasing transform type (DCT3, DCT7, DCT4, or DCT8) can be obtained. Furthermore, as illustrated in Fig. 9, by transposing the transformation matrix of DST7 (DST3) in which the waveform of the 0-order column vector is the chevron type, a transformation matrix that can substitute the transformation matrix in which the waveform of the 0-

order row vector is the chevron transform type (DST2, DST6, DST1, or DST5) can be obtained.

[0075]

That is, the derivation unit may derive the second
5 transformation matrix by an operation for an element of
such a first transformation matrix. Then, the operation
for an element may include the rearrangement of elements
(change of the arrangement order) as described above.
With the configuration, the type of the waveform can be
10 intentionally changed, so that more diverse second
transformation matrices can be derived from the first
transformation matrix. Therefore, an increase in the
number of transformation types prepared for the
orthogonal transform/inverse orthogonal transform can be
15 suppressed, and an increase in the memory capacity
required for the orthogonal transform/inverse orthogonal
transform can be suppressed.

[0076]

Note that, of course, the derivation unit may
20 perform such an operation a plurality of times to derive
the second transformation matrix. For example,
operations such as flip and transposition can be
arbitrarily combined. Furthermore, the same operation
may be repeated a plurality of times. By doing so, more
25 various second transformation matrices can be derived
from the first transformation matrix.

[0077]

Note that, as described above, the transformation
matrix used for the orthogonal transform/inverse
30 orthogonal transform is stored in the LUT. Therefore,
the derivation unit may derive the second transformation

matrix, using the first transformation matrix stored in the lookup table (LUT). By doing so, an increase in the size of the LUT can be suppressed. Therefore, an increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform can be suppressed.

[0078]

<Derivation Example>

Fig. 10 illustrates a list of transformation matrix derivation examples involving the above operations. Note that the transformation matrix (first transformation matrix) used for derivation is also referred to as a base transformation matrix T_{base} . The transform type of the base transformation matrix is also called a base transform type or a first transform type. Furthermore, the transform type of the transformation matrix (second transformation matrix) to be derived is also referred to as a second transform type.

[0079]

In the table illustrated in Fig. 10, the derivation of the first row example from the top except the uppermost row of item names focuses on similarity between the waveform of the lowest-order row vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted.

[0080]

In this case, the derivation unit flips the first transformation matrix to derive the second transformation matrix. More specifically, the derivation unit uses the transformation matrix of DST7 as the base transformation

matrix T_{base} , and flips the transformation matrix in the row direction to derive a transformation matrix of FlipDST7. Since the waveform of the 0-order row vector of the transformation matrix of DST7 is of the increasing type, the waveform of the 0-order row vector of the derived transformation matrix of FlipDST7 is of the decreasing type. Therefore, the transformation matrix of DCT8 having the decreasing-type waveform of the 0-order row vector can be substituted by the transformation matrix of FlipDST7.

[0081]

By applying such derivation, it becomes unnecessary to prepare the transformation matrix of DCT8 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced. That is, an increase in the LUT size can be suppressed. Furthermore, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (FlipDST7), similar coding efficiency to the case of using the transformation matrix of DCT8 for the orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (decreasing-type substitution transformation matrix) can be derived by one-time operation.

[0082]

Furthermore, the derivation of an example one row below the first row example (the second row example from the top) focuses on similarity between the waveform of

the lowest-order column vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted.

5 [0083]

In this case, the derivation unit transposes the first transformation matrix to derive the second transformation matrix. More specifically, the derivation unit uses the transformation matrix of DST7 as the base transformation matrix T_{base} , and transposes the transformation matrix to derive a transformation matrix of TrDST7. Since the waveform of the 0-order column vector of the transformation matrix of DST7 is of the chevron type, the waveform of the 0-order row vector of the derived transformation matrix of TrDST7 is of the chevron type. Therefore, the transformation matrix of DST1 having the chevron-type waveform of the 0-order row vector can be substituted by the transformation matrix of TrDST7.

20 [0084]

By applying such derivation, it becomes unnecessary to prepare the transformation matrix of DST1 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced. That is, an increase in the LUT size can be suppressed. Furthermore, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (TrDST7), similar coding efficiency to the case of using the transformation matrix of DST1 for the

orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (chevron-type substitution transformation matrix) can be derived by one-time operation.

[0085]

Furthermore, the derivation of an example one row below the second row example (the third row example from the top) focuses on characteristics between paired DCT and DST. More specifically, between DCT/DST to be paired (for example, DST7 and DCT8), attention is paid to the point that even-numbered row vectors are axially symmetric and odd-numbered row vectors are point-symmetric.

15 [0086]

In this case, the derivation unit flips the first transformation matrix and inverts the sign of the odd-numbered row vector of the flipped first transformation matrix to derive the second transformation matrix. More specifically, the derivation unit uses the transformation matrix of DST7 as the base transformation matrix T_{base} , and flips the transformation matrix in the row direction and further inverts the sign of the odd-order row vector to derive a transformation matrix of DCT8. Note that row vector sign inversion is simply performed by transforming the most significant bit of each element of the row vector. Naturally, the transformation matrix of DST8 having the decreasing-type waveform of the 0-order row vector can be substituted by the derived transformation matrix of DCT8.

30

[0087]

By applying such derivation, it becomes unnecessary to prepare the transformation matrix of DCT8 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced. That is, an increase in the LUT size can be suppressed. Furthermore, naturally, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (DCT8), the same coding efficiency as the case of using the transformation matrix of DCT8 for the orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (substitution transformation matrix to be paired) can be derived by two-time operation.

[0088]

Furthermore, the derivation of an example one row below the third row example (the fourth row example from the top) focuses on similarity between the waveform of the lowest-order row vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted, similarly to the case of the first row example from the top.

[0089]

In this case, the derivation unit flips the first transformation matrix to derive the second transformation matrix. More specifically, the derivation unit uses the transformation matrix of DCT8 as the base transformation matrix T_{base} , and flips the transformation matrix in the row direction to derive a transformation matrix of

FlipDCT8. Since the waveform of the 0-order row vector of the transformation matrix of DCT8 is of the decreasing type, the waveform of the 0-order row vector of the derived transformation matrix of FlipDCT8 is of the increasing type. Therefore, the transformation matrix of DST7 having the increasing-type waveform of the 0-order row vector can be substituted by the transformation matrix of FlipDCT8.

[0090]

By applying such derivation, it becomes unnecessary to prepare the transformation matrix of DST7 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced. That is, an increase in the LUT size can be suppressed. Furthermore, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (FlipDCT8), similar coding efficiency to the case of using the transformation matrix of DST7 for the orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (decreasing-type substitution transformation matrix) can be derived by one-time operation.

[0091]

Furthermore, the derivation of an example one row below the fourth row example (the fifth row example from the top) focuses on similarity between the waveform of a highest-order column vector of the first transform type and the waveform of the lowest-order row vector of the

transform type of a transformation matrix to be substituted.

[0092]

In this case, the derivation unit flips the first
5 transformation matrix and transposes the flipped first
transformation matrix to derive the second transformation
matrix. More specifically, the derivation unit uses the
transformation matrix of DCT8 as the base transformation
matrix T_{base} , and flips the transformation matrix in the
10 row direction and further transposes the transformation
matrix to derive a transformation matrix of TrFlipDCT8.
Since the waveform of the highest-order column vector of
the transformation matrix of DCT8 is of the chevron type,
the waveform of the 0-order row vector of the derived
15 transformation matrix of TrFlipDCT8 is of the chevron
type. Therefore, the transformation matrix of DST1
having the chevron-type waveform of the 0-order row
vector can be substituted by the transformation matrix of
TrFlipDCT8.

20 [0093]

By applying such derivation, it becomes unnecessary
to prepare the transformation matrix of DST1 as a
candidate for a transformation matrix to be used for
orthogonal transform/inverse orthogonal transform. That
25 is, the number of unique transform types can be reduced.
That is, an increase in the LUT size can be suppressed.
Furthermore, by performing the orthogonal
transform/inverse orthogonal transform using the
transformation matrix of the derived second transform
30 type (TrFlipDCT8), similar coding efficiency to the case
of using the transformation matrix of DST1 for the

orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (chevron-type substitution transformation matrix) can be derived by two-time operation.

[0094]

Furthermore, the derivation of an example one row below the fifth row example (the sixth row example from the top) focuses on characteristics between paired DCT and DST, similarly to the case of the third row example from the top. More specifically, between DCT/DST to be paired (for example, DCT8 and DST7), attention is paid to the point that even-numbered row vectors are axially symmetric and odd-numbered row vectors are point-symmetric.

[0095]

In this case, the derivation unit flips the first transformation matrix and inverts the sign of the odd-numbered row vector of the flipped first transformation matrix to derive the second transformation matrix. More specifically, the derivation unit uses the transformation matrix of DCT8 as the base transformation matrix T_{base} , and flips the transformation matrix in the row direction and further inverts the sign of the odd-order row vector to derive a transformation matrix of DST7. Naturally, the transformation matrix of DST7 having the increasing-type waveform of the 0-order row vector can be substituted by the derived transformation matrix of DST7.

[0096]

By applying such derivation, it becomes unnecessary to prepare the transformation matrix of DST7 as a

candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced. That is, an increase in the LUT size can be suppressed.

5 Furthermore, of course, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (DST7), the same coding efficiency as the case of using the transformation matrix of DST7 for the
10 orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (substitution transformation matrix to be paired) can be derived by two-time operation.
[0097]

15 Furthermore, the derivation of an example one row below the sixth row example (the seventh row example from the top) focuses on similarity between the waveform of the lowest-order column vector of the first transform type and the waveform of the lowest-order row vector of
20 the transform type of a transformation matrix to be substituted.
[0098]

In this case, the derivation unit transposes the first transformation matrix to derive the second
25 transformation matrix. More specifically, the derivation unit uses the transformation matrix of DCT2 as the base transformation matrix T_{base} , and transposes the transformation matrix to derive a transformation matrix of DCT3. Since the waveform of the 0-order column vector
30 of the transformation matrix of DCT2 is of the decreasing type, the waveform of the 0-order row vector of the

derived transformation matrix of DCT3 is of the decreasing type. Therefore, the transformation matrix of DCT8 having the decreasing-type waveform of the 0-order row vector can be substituted by the transformation matrix of DCT3.

[0099]

By applying such derivation, it becomes unnecessary to prepare the transformation matrix of DCT8 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced. That is, an increase in the LUT size can be suppressed. Furthermore, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (DCT3), similar coding efficiency to the case of using the transformation matrix of DCT8 for the orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (decreasing-type substitution transformation matrix) can be derived by one-time operation.

[0100]

Furthermore, the derivation of an example one row below the seventh row example (the eighth row example from the top) focuses on similarity between the waveform of the highest-order column vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted.

[0101]

In this case, the derivation unit transposes the first transformation matrix and flips the transposed first transformation matrix to derive the second transformation matrix. More specifically, the derivation unit uses the transformation matrix of DCT2 as the base transformation matrix T_{base} , and transposes the transformation matrix and flips the transformation matrix in the row direction to derive a transformation matrix of FlipDCT3. Since the waveform of the highest-order column vector of the transformation matrix of DCT2 is of the chevron type, the waveform of the 0-order row vector of the derived transformation matrix of FlipDCT3 is of the increasing type. Therefore, the transformation matrix of DST7 having the increasing-type waveform of the 0-order row vector can be substituted by the transformation matrix of FlipDCT3.

[0102]

By applying such derivation, it becomes unnecessary to prepare the transformation matrix of DST7 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced. That is, an increase in the LUT size can be suppressed. Furthermore, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (FlipDCT3), similar coding efficiency to the case of using the transformation matrix of DST7 for the orthogonal transform/inverse orthogonal transform can be obtained. Furthermore, in this case, the second transformation matrix (increasing-type substitution

transformation matrix) can be derived by two-time operation.

[0103]

Note that each of the above-described derivation
5 examples may be performed independently or may be performed in combination of a plurality of derivation examples.

[0104]

<Image Encoding Device>

10 Next, a configuration for deriving the above transformation matrix will be described. Fig. 11 is a block diagram illustrating an example of a configuration example of an image encoding device that is one mode of the image processing apparatus to which the present
15 technology is applied. An image encoding device 100 illustrated in Fig. 11 is a device that encodes image data of a moving image. For example, the image encoding device 100 implements the technology described in Non-Patent Document 1, Non-Patent Document 3, or Non-Patent
20 Document 4, and encodes the image data of the moving image by a method conforming to the standard described in any of the aforementioned documents.

[0105]

Note that Fig. 11 illustrates main processing units,
25 data flows, and the like, and those illustrated in Fig. 11 are not necessarily everything. That is, in the image encoding device 100, there may be a processing unit not illustrated as a block in Fig. 11, or processing or data flow not illustrated as an arrow or the like in Fig. 11.
30 This is similar in other drawings for describing a processing unit and the like in the image encoding device

100.

[0106]

As illustrated in Fig. 11, the image encoding device 100 includes a control unit 101, a rearrangement
5 buffer 111, a calculation unit 112, an orthogonal transform unit 113, a quantization unit 114, an encoding unit 115, an accumulation buffer 116, an inverse quantization unit 117, an inverse orthogonal transform unit 118, a calculation unit 119, an in-loop filter unit
10 120, a frame memory 121, a prediction unit 122, and a rate control unit 123.

[0107]

<Control Unit>

The control unit 101 divides moving image data held
15 by the rearrangement buffer 111 into blocks (CUs, PUs, transformation blocks, or the like) in units of processing on the basis of a block size in external or pre-designated units of processing. Furthermore, the control unit 101 determines encoding parameters (header
20 information Hinfo, prediction mode information Pinfo, transform information Tinfo, filter information Finfo, and the like) to be supplied to each block on the basis of, for example, rate-distortion optimization (RDO).

[0108]

25 Details of these encoding parameters will be described below. After determining the above-described encoding parameters, the control unit 101 supplies the encoding parameters to each block. Specifically, the encoding parameters are as follows.

30 [0109]

The header information Hinfo is supplied to each

block.

The prediction mode information Pinfo is supplied to the encoding unit 115 and the prediction unit 122.

The transform information Tinfo is supplied to the
 5 encoding unit 115, the orthogonal transform unit 113, the
 quantization unit 114, the inverse quantization unit 117,
 and the inverse orthogonal transform unit 118.

The filter information Finfo is supplied to the in-
 loop filter unit 120.

10 [0110]

<Rearrangement Buffer>

Each field (input image) of the moving image data
 is input to the image encoding device 100 in reproduction
 order (display order). The rearrangement buffer 111
 15 acquires and holds (stores) each input image in its
 reproduction order (display order). The rearrangement
 buffer 111 rearranges the input images in encoding order
 (decoding order) or divides the input images into blocks
 in units of processing on the basis of the control of the
 20 control unit 101. The rearrangement buffer 111 supplies
 the processed input image to the calculation unit 112.
 Furthermore, the rearrangement buffer 111 also supplies
 the input images (original images) to the prediction unit
 122 and the in-loop filter unit 120.

25 [0111]

<Calculation Unit>

The calculation unit 112 uses an image I
 corresponding to the block in units of processing and a
 predicted image P supplied from the prediction unit 122
 30 as inputs, subtracts the predicted image P from the image
 I as illustrated in the following expression (6) to

derive a prediction residual D, and supplies the prediction residual D to the orthogonal transform unit 113.

[0112]

5 [Math. 4]

$$D = I - P \cdot \cdot \cdot (6)$$

[0113]

<Orthogonal Transform Unit>

The orthogonal transform unit 113 uses the prediction residual D supplied from the calculation unit 112 and the transform information Tinfo supplied from the control unit 101 as inputs, and orthogonally transforms the prediction residual D on the basis of the transform information Tinfo to derive a transform coefficient Coeff.

15 The orthogonal transform unit 113 supplies the obtained transform coefficient Coeff to the quantization unit 114.

[0114]

<Quantization Unit>

The quantization unit 114 uses the transform coefficient Coeff supplied from the orthogonal transform unit 113 and the transform information Tinfo supplied from the control unit 101 as inputs, and scales (quantizes) the transform coefficient Coeff on the basis of the transform information Tinfo. Note that a rate of this quantization is controlled by the rate control unit 123.

25 The quantization unit 114 supplies a quantized transform coefficient obtained by the quantization, that is, a quantized transform coefficient level level to the encoding unit 115 and the inverse quantization unit 117.

30 [0115]

<Encoding Unit>

The encoding unit 115 uses, as inputs, the quantized transform coefficient level level supplied from the quantization unit 114, the various encoding parameters (header information Hinfo, prediction mode information Pinfo, transform information Tinfo, filter information Finfo, and the like) supplied from the control unit 101, information regarding a filter such as a filter coefficient supplied from the in-loop filter unit 120, and information regarding an optimal prediction mode supplied from the prediction unit 122. The encoding unit 115 performs variable-length coding (for example, arithmetic coding) for the quantized transform coefficient level level to generate a bit string (coded data).

[0116]

Furthermore, the encoding unit 115 derives residual information Rinfo from the quantized transform coefficient level level, and encodes the residual information Rinfo to generate a bit string.

[0117]

Moreover, the encoding unit 115 includes the information regarding a filter supplied from the in-loop filter unit 120 to the filter information Finfo, and includes the information regarding an optimal prediction mode supplied from the prediction unit 122 to the prediction mode information Pinfo. Then, the encoding unit 115 encodes the above-described various encoding parameters (header information Hinfo, prediction mode information Pinfo, transform information Tinfo, filter information Finfo, and the like) to generate a bit string.

[0118]

Furthermore, the encoding unit 115 multiplexes the bit string of the various types of information generated as described above to generate coded data. The encoding
5 unit 115 supplies the coded data to the accumulation buffer 116.

[0119]

<Accumulation Buffer>

The accumulation buffer 116 temporarily stores the
10 coded data obtained by the encoding unit 115. The accumulation buffer 116 outputs the stored coded data to an outside of the image encoding device 100 as a bit stream or the like at predetermined timing. For example, the coded data is transmitted to a decoding side via an
15 arbitrary recording medium, an arbitrary transmission medium, an arbitrary information processing device, or the like. That is, the accumulation buffer 116 is also a transmission unit that transmits coded data (bit stream).

[0120]

20 <Inverse Quantization Unit>

The inverse quantization unit 117 performs processing regarding inverse quantization. For example, the inverse quantization unit 117 uses the quantized transform coefficient level level supplied from the
25 quantization unit 114 and the transform information Tinfo supplied from the control unit 101 as inputs, and scales (inversely quantizes) the value of the quantized transform coefficient level level on the basis of the transform information Tinfo. Note that the inverse
30 quantization is inverse processing of the quantization performed in the quantization unit 114. The inverse

quantization unit 117 supplies a transform coefficient
Coeff_IQ obtained by the inverse quantization to the
inverse orthogonal transform unit 118.

[0121]

5 <Inverse Orthogonal Transform Unit>

The inverse orthogonal transform unit 118 performs
processing regarding inverse orthogonal transform. For
example, the inverse orthogonal transform unit 118 uses
the transform coefficient Coeff_IQ supplied from the
10 inverse quantization unit 117 and the transform
information Tinfo supplied from the control unit 101 as
inputs, and inversely orthogonally transforms the
transform coefficient Coeff_IQ on the basis of the
transform information Tinfo to derive a prediction
15 residual D' . Note that the inverse orthogonal transform
is inverse processing of the orthogonal transform
performed in the orthogonal transform unit 113. The
inverse orthogonal transform unit 118 supplies the
prediction residual D' obtained by the inverse orthogonal
20 transform to the calculation unit 119. Note that, since
the inverse orthogonal transform unit 118 is similar to
an inverse orthogonal transform unit on the decoding side
(to be described below), description (to be described
below) to be given for the decoding side can be applied
25 to the inverse orthogonal transform unit 118.

[0122]

<Calculation Unit>

The calculation unit 119 uses the prediction
residual D' supplied from the inverse orthogonal
30 transform unit 118 and the predicted image P supplied
from the prediction unit 122 as inputs. The calculation

unit 119 adds the prediction residual D' and the predicted image P corresponding to the prediction residual D' to derive a locally decoded image R_{local} . The calculation unit 119 supplies the derived locally decoded
 5 image R_{local} to the in-loop filter unit 120 and the frame memory 121.

[0123]

<In-loop Filter Unit>

The in-loop filter unit 120 performs processing
 10 regarding in-loop filter processing. For example, the in-loop filter unit 120 uses the locally decoded image R_{local} supplied from the calculation unit 119, the filter information F_{info} supplied from the control unit 101, and the input image (original image) supplied from the
 15 rearrangement buffer 111 as inputs. Note that the information input to the in-loop filter unit 120 may be information other than the aforementioned information. For example, information such as the prediction mode, motion information, a code amount target value, a
 20 quantization parameter QP , a picture type, a block (a CU, a CTU, or the like) may be input to the in-loop filter unit 120, as necessary.

[0124]

The in-loop filter unit 120 appropriately performs
 25 filtering processing for the locally decoded image R_{local} on the basis of the filter information F_{info} . The in-loop filter unit 120 also uses the input image (original image) and other input information for the filtering processing as necessary.

30 [0125]

For example, the in-loop filter unit 120 applies

four in-loop filters of a bilateral filter, a deblocking filter (DBF), an adaptive offset filter (sample adaptive offset (SAO)), and an adaptive loop filter (adaptive loop filter (ALF)) in this order, as described in Non-Patent

5 Document 1. Note that which filter is applied and in which order the filters are applied are arbitrary and can be selected as appropriate.

[0126]

Of course, the filtering processing performed by
10 the in-loop filter unit 120 is arbitrary, and is not limited to the above example. For example, the in-loop filter unit 120 may apply a Wiener filter or the like.

[0127]

The in-loop filter unit 120 supplies the filtered
15 locally decoded image R_{local} to the frame memory 121. Note that, in a case of transmitting the information regarding filters such as filter coefficients to the decoding side, the in-loop filter unit 120 supplies the information regarding filters to the encoding unit 115.

20 [0128]

<Frame Memory>

The frame memory 121 performs processing regarding storage of data relating to an image. For example, the frame memory 121 uses the locally decoded image R_{local}
25 supplied from the calculation unit 119 and the filtered locally decoded image R_{local} supplied from the in-loop filter unit 120 as inputs, and holds (stores) the inputs. Furthermore, the frame memory 121 reconstructs and holds a decoded image R for each picture unit, using the
30 locally decoded image R_{local} (stores the decoded image R in a buffer in the frame memory 121). The frame memory

121 supplies the decoded image R (or a part thereof) to the prediction unit 122 in response to a request from the prediction unit 122.

[0129]

5 <Prediction Unit>

 The prediction unit 122 performs processing regarding generation of a predicted image. For example, the prediction unit 122 uses, as inputs, the prediction mode information Pinfo supplied from the control unit 101, the input image (original image) supplied from the rearrangement buffer 111, and the decoded image R (or a part thereof) read from the frame memory 121. The prediction unit 122 performs prediction processing such as inter prediction, intra prediction, or the like, using the prediction mode information Pinfo and the input image (original image), performs prediction, using the decoded image R as a reference image, performs motion compensation processing on the basis of a prediction result, and generates a predicted image P. The prediction unit 122 supplies the generated predicted image P to the calculation units 112 and 119. Furthermore, the prediction unit 122 supplies a prediction mode selected by the above processing, that is, the information regarding an optimal prediction mode to the encoding unit 115, as necessary.

[0130]

 <Rate Control Unit>

 The rate control unit 123 performs processing regarding rate control. For example, the rate control unit 123 controls a rate of a quantization operation of the quantization unit 114 so that an overflow or an

underflow does not occur on the basis of the code amount of the coded data accumulated in the accumulation buffer 116.

[0131]

5 In the image encoding device 100 having the above-described configuration, the orthogonal transform unit 113 performs processing to which the above-described present technology is applied, as a derivation unit and an orthogonal transform unit. Furthermore, the encoding
10 unit 115 performs processing to which the above-described present technology is applied, as an encoding unit. Furthermore, the inverse orthogonal transform unit 118 performs processing to which the above-described present technology is applied, as an inverse orthogonal transform
15 unit and a derivation unit. Therefore, the image encoding device 100 can suppress an increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform.

[0132]

20 <Details of Orthogonal Transform Unit>

 Fig. 12 is a block diagram illustrating a main configuration example of the orthogonal transform unit 113 in Fig. 11. As illustrated in Fig. 12, the orthogonal transform unit 113 includes a switch 151, a
25 primary transform unit 152, and a secondary transform unit 153.

[0133]

 The switch 151 uses the prediction residual D and a transform skip flag ts_flag [compID] corresponding to a
30 component identifier compID as inputs, and supplies the prediction residual D to the primary transform unit 152

in a case where a value of the transform skip flag
 ts_flag [compID] is NO_TS (= 0) (in a case where a
 transform skip is not applied). Furthermore, the switch
 151 skips the primary transform unit 152 and the
 5 secondary transform unit 153, and outputs the prediction
 residual D to the outside of the orthogonal transform
 unit 113 (supplies the same to the quantization unit 114)
 as a transform coefficient Coeff, in a case where the
 value of the transform skip flag ts_flag [compID] is
 10 2D_TS (=1) (in a case where the value indicates
 application of a two-dimensional transform skip).
 [0134]

The primary transform unit 152 performs processing
 regarding primary transform that is predetermined
 15 transform processing such as orthogonal transform. For
 example, the primary transform unit 152 uses, as inputs,
 the component identifier compID, an adaptive primary
 transform flag apt_flag[compID] of the component
 identifier compID, a primary transform identifier
 20 pt_idx[compID] of the component identifier compID, the
 prediction mode information PInfo, the size of the
 transformation block (a logarithmic value of a width
 log2TBWSize and a logarithmic value of a height
 log2TBHSize), and the prediction residual D. Note that
 25 the width TBWSize of the transformation block is also
 referred to as TBWidth, and the logarithmic value thereof
 is also referred to as log2TBWidth. Similarly, the
 height TBHSize of the transformation block is also
 referred to as TBHeight, and the logarithmic value
 30 thereof is also referred to as log2TBHeight.
 [0135]

The primary transform unit 152 selects a transform type `TrTypeH` of primary horizontal transform (and a primary horizontal transform type identifier `TrTypeIdxH` indicating the transform type) and a transform type `TrTypeV` of primary vertical transform (and a primary vertical transform type identifier `TrTypeIdxV` indicating the transform type) corresponding to the component identifier `compID` by reference to the prediction mode information `PInfo`, the component identifier `compID`, the adaptive primary transform flag `apt_flag[compID]` of the component identifier `compID`, and the primary transform identifier `pt_idx[compID]` of the component identifier `compID`.

[0136]

Furthermore, the primary transform unit 152 performs, for the prediction residual `D`, primary horizontal transform determined by the primary horizontal transform type identifier `TrTypeIdxH` (or the primary horizontal transform type `TrTypeH`) and the width `log2TBWSize` of the transformation block, and primary vertical transform determined by the primary vertical transform type identifier `TrTypeIdxV` (or the primary vertical transform type `TrTypeV`) and the height `log2TBHSize` of the transformation block, to derive a transform coefficient `Coeff_P`. The primary horizontal transform is horizontal one-dimensional orthogonal transform, and the primary vertical transform is vertical one-dimensional orthogonal transform.

[0137]

The primary transform unit 152 supplies the derived transform coefficient `Coeff_P` to the secondary transform

unit 153.

[0138]

The secondary transform unit 153 performs processing regarding secondary transform that is
5 predetermined transform processing such as orthogonal transform. For example, the secondary transform unit 153 uses a secondary transform identifier `st_idx`, a scan identifier `scanIdx` indicating a method of scanning a transform coefficient, and the transform coefficient
10 `Coeff_P` as inputs. The secondary transform unit 153 performs secondary transform for the transform coefficient `Coeff_P` on the basis of the secondary transform identifier `st_idx` and the scan identifier `scanIdx` to derive a transform coefficient `Coeff_S` after
15 secondary transform.

[0139]

More specifically, in a case where the secondary transform identifier `st_idx` indicates application of secondary transform (`st_idx > 0`), the secondary transform
20 unit 153 executes, for the transform coefficient `Coeff_P`, secondary transform processing corresponding to the secondary transform identifier `st_idx` to derive the transform coefficient `Coeff_S` after secondary transform.

[0140]

25 The secondary transform unit 153 outputs the secondary transform coefficient `Coeff_S` to the outside of the orthogonal transform unit 113 (supplies the same to the quantization unit 114) as the transform coefficient `Coeff`.

30 [0141]

Furthermore, in a case where the secondary

transform identifier `st_idx` indicates non-application of secondary transform (`st_idx == 0`), the secondary transform unit 153 skips the secondary transform and outputs the transform coefficient `Coeff_P` after primary
5 transform to the outside of the orthogonal transform unit 113 (supplies the same to the quantization unit 114) as the transform coefficient `Coeff` (the transform coefficient `Coeff_S` after secondary transform).
[0142]

10 In the orthogonal transform unit 113 having the above configuration, the primary transform unit 152 performs processing to which the above-described present technology is applied, as a derivation unit and an orthogonal transform unit. That is, the derivation unit
15 derives the second transformation matrix, using the first transformation matrix, and the orthogonal transform unit performs the primary transform for the prediction residual, using the second transformation matrix derived by the derivation unit. Therefore, an increase in the
20 memory capacity required for the primary transform can be suppressed.
[0143]

Note that, as described above, the primary transform unit 152 performs the primary horizontal
25 transform and the primary vertical transform as the primary transform. That is, the derivation unit derives the second transformation matrix for horizontal one-dimensional orthogonal transform and the second transformation matrix for vertical one-dimensional
30 orthogonal transform, and the orthogonal transform unit performs, as the primary transform, the horizontal one-

dimensional orthogonal transform, using the second transformation matrix for horizontal one-dimensional orthogonal transform derived by the derivation unit, and further, the vertical one-dimensional orthogonal transform, using the second transformation matrix for vertical one-dimensional orthogonal transform derived by the derivation unit. Therefore, an increase in the memory capacity required for the primary transform where the horizontal one-dimensional orthogonal transform and the vertical one-dimensional orthogonal transform are performed can be suppressed.

[0144]

<Flow of Image Encoding Processing>

Next, a flow of each processing executed by the above image encoding device 100 will be described. First, an example of a flow of image encoding processing will be described with reference to the flowchart in Fig. 13.

[0145]

When the image encoding processing is started, in step S101, the rearrangement buffer 111 is controlled by the control unit 101 and rearranges frames of input moving image data from the display order to the encoding order.

[0146]

In step S102, the control unit 101 sets the unit of processing (performs block division) for an input image held by the rearrangement buffer 111.

[0147]

In step S103, the control unit 101 determines (sets) an encoding parameter for the input image held by the rearrangement buffer 111.

[0148]

In step S104, the prediction unit 122 performs the prediction processing and generates a predicted image or the like in the optimal prediction mode. For example, in
5 the prediction processing, the prediction unit 122 performs the intra prediction to generate a predicted image in an optimal intra prediction mode, performs the inter prediction to generate a predicted image in an optimal inter prediction mode, and selects an optimal
10 prediction mode from among the predicted images on the basis of a cost function value and the like.

[0149]

In step S105, the calculation unit 112 calculates a difference between the input image and the predicted
15 image in the optimal mode selected by the prediction processing in step S104. That is, the calculation unit 112 generates the prediction residual D between the input image and the predicted image. The prediction residual D obtained in this way is reduced in the data amount as
20 compared with the original image data. Therefore, the data amount can be compressed as compared with a case of encoding the image as it is.

[0150]

In step S106, the orthogonal transform unit 113
25 performs orthogonal transform processing for the prediction residual D generated by the processing in step S105 to derive the transform coefficient Coeff.

[0151]

In step S107, the quantization unit 114 quantizes
30 the transform coefficient Coeff obtained by the processing in step S106 by using a quantization parameter

calculated by the control unit 101 or the like to derive the quantized transform coefficient level level.

[0152]

In step S108, the inverse quantization unit 117
5 inversely quantizes the quantized transform coefficient level level generated by the processing in step S107 with characteristics corresponding to the characteristics of the quantization in step S107 to derive the transform coefficient Coeff_IQ.

10 [0153]

In step S109, the inverse orthogonal transform unit 118 inversely orthogonally transforms the transform coefficient Coeff_IQ obtained by the processing in step S108 by a method corresponding to the orthogonal
15 transform processing in step S106 to derive the prediction residual D' . Note that, since the inverse orthogonal transform processing is similar to inverse orthogonal transform processing (to be described below) performed on the decoding side, description (to be given
20 below) for the decoding side can be applied to the inverse orthogonal transform processing in step S109.

[0154]

In step S110, the calculation unit 119 adds the predicted image obtained by the prediction processing in
25 step S104 to the prediction residual D' derived by the processing in step S109 to generate a locally decoded image.

[0155]

In step S111, the in-loop filter unit 120 performs
30 the in-loop filter processing for the locally decoded image derived by the processing in step S110.

[0156]

In step S112, the frame memory 121 stores the locally decoded image derived by the processing in step S110 and the locally decoded image filtered in step S112.

5 [0157]

In step S113, the encoding unit 115 encodes the quantized transform coefficient level level obtained by the processing in step S107. For example, the encoding unit 115 encodes the quantized transform coefficient level level that is information regarding the image by arithmetic coding or the like to generate the coded data. Furthermore, at this time, the encoding unit 115 encodes the various encoding parameters (header information Hinfo, prediction mode information Pinfo, and transform information Tinfo). Moreover, the encoding unit 115 derives the residual information RInfo from the quantized transform coefficient level level and encodes the residual information RInfo.

[0158]

20 In step S114, the accumulation buffer 116 accumulates the coded data thus obtained, and outputs the coded data to the outside of the image encoding device 100, for example, as a bit stream. The bit stream is transmitted to the decoding side via a transmission path or a recording medium, for example. Furthermore, the rate control unit 123 performs rate control as necessary.

[0159]

When the processing in step S114 ends, the image encoding processing ends.

30 [0160]

In the image encoding processing of the above flow,

processing to which the above-described present technology is applied is performed as processing of step S106. Furthermore, processing to which the above-described present technology is applied is performed as processing of step S109. Moreover, processing to which the above-described present technology is applied is performed as processing of step S113. Therefore, by executing the image encoding processing, an increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform can be suppressed.

[0161]

<Flow of Orthogonal Transform processing>

Next, an example of a flow of the orthogonal transform processing executed in step S106 in Fig. 13 will be described with reference to the flowchart in Fig. 14.

[0162]

When the orthogonal transform processing is started, in step S131, the switch 151 determines whether or not the transform skip flag `ts_flag` is 2D_TS (in a case of indicating a two-dimensional transform skip) (for example, 1 (true)) or a transform quantization bypass flag `transquant_bypass_flag` is 1 (true). In a case where it is determined that the transform skip flag `ts_flag` is 2D_TS (for example, 1 (true)) or the transform quantization bypass flag is 1 (true), the orthogonal transform processing ends, and the processing returns to Fig. 13. In this case, the orthogonal transform processing (primary transform and secondary transform) is omitted, and the input prediction residual `D` is used as the transform coefficient `Coeff`.

[0163]

Furthermore, in step S131 in Fig. 14, in a case where it is determined that the transform skip flag `ts_flag` is not 2D_TS (not two-dimensional transform skip) (for example, 0 (false)) and the transform quantization bypass flag `transquant_bypass_flag` is 0 (false), the processing proceeds to step S132. In this case, the primary transform processing and the secondary transform processing are performed.

10 [0164]

In step S132, the primary transform unit 152 performs the primary transform processing for the input prediction residual `D` on the basis of the adaptive primary transform information specified with the component identifier `compID` to derive the transform coefficient `Coeff_P` after primary transform.

[0165]

In step S133, the secondary transform unit 153 performs the secondary transform processing for the transform coefficient `Coeff_P` to derive the transform coefficient `Coeff_S` (transform coefficient `Coeff`) after secondary transform.

[0166]

When the processing in step S133 ends, the orthogonal transform processing ends.

[0167]

In the above orthogonal transform processing, processing to which the above-described present technology is applied is performed as processing of step S132. Therefore, by executing the orthogonal transform processing, an increase in the memory capacity required

for the primary transform can be suppressed.

[0168]

<Image Decoding Device>

Fig. 15 is a block diagram illustrating an example
5 of a configuration of an image decoding device as one
mode of the image processing apparatus to which the
present technology is applied. An image decoding device
200 illustrated in Fig. 15 is a device that decodes coded
data that is a coded prediction residual between an image
10 and a predicted image, such as AVC or HEVC. For example,
the image decoding device 200 implements the technology
described in Non-Patent Document 1, Non-Patent Document 3,
or Non-Patent Document 4, and decodes coded data that is
coded image data of a moving image by a method conforming
15 to the standard described in any of the aforementioned
documents. For example, the image decoding device 200
decodes the coded data (bit stream) generated by the
above-described image encoding device 100.

[0169]

20 Note that Fig. 15 illustrates main processing units,
data flows, and the like, and those illustrated in Fig.
15 are not necessarily everything. That is, in the image
decoding device 200, there may be a processing unit not
illustrated as a block in Fig. 15, or processing or data
25 flow not illustrated as an arrow or the like in Fig. 15.
This is similar in other drawings for describing a
processing unit and the like in the image decoding device
200.

[0170]

30 In Fig. 15, the image decoding device 200 includes
an accumulation buffer 211, a decoding unit 212, an

inverse quantization unit 213, an inverse orthogonal transform unit 214, a calculation unit 215, an in-loop filter unit 216, a rearrangement buffer 217, a frame memory 218, and a prediction unit 219. Note that the prediction unit 219 includes an intra prediction unit and an inter prediction unit (not illustrated). The image decoding device 200 is a device for generating moving image data by decoding coded data (bit stream).

[0171]

10 <Accumulation Buffer>

The accumulation buffer 211 acquires the bit stream input to the image decoding device 200 and holds (stores) the bit stream. The accumulation buffer 211 supplies the accumulated bit stream to the decoding unit 212 at predetermined timing or in a case where a predetermined condition is satisfied, for example.

[0172]

<Decoding Unit>

The decoding unit 212 performs processing regarding image decoding. For example, the decoding unit 212 uses the bit stream supplied from the accumulation buffer 211 as an input, and performs variable length decoding for a syntax value of each syntax element from the bit string according to a definition of a syntax table to derive a parameter.

[0173]

The parameter derived from the syntax element and the syntax value of the syntax element includes, for example, the information such as the header information Hinfo, prediction mode information Pinfo, transform information Tinfo, residual information Rinfo, and filter

information Finfo. That is, the decoding unit 212 parses (analyzes and acquires) such information from the bit stream. These pieces of information will be described below.

5 [0174]

<Header information Hinfo>

The header information Hinfo includes, for example, header information such as a video parameter set (VPS)/a sequence parameter set (SPS)/a picture parameter set (PPS)/a slice header (SH). The header information Hinfo includes, for example, information defining image size (width PicWidth and height PicHeight), bit depth (luminance bitDepthY and chrominance bitDepthC), a chrominance array type ChromaArrayType, CU size maximum value MaxCUSize/minimum value MinCUSize, maximum depth MaxQTDdepth/minimum depth MinQTDdepth of quad-tree division, maximum depth MaxBTDepth/minimum depth MinBTDepth of binary-tree division, a maximum value MaxTSSize of a transform skip block (also called maximum transform skip block size), an on/off flag of each coding tool (also called valid flag), and the like.

[0175]

For example, an example of the on/off flag of the coding tool included in the header information Hinfo includes an on/off flag related to transform and quantization processing below. Note that the on/off flag of the coding tool can also be interpreted as a flag indicating whether or not a syntax related to the coding tool exists in the coded data. Furthermore, in a case where a value of the on/off flag is 1 (true), the value indicates that the coding tool is available. In a case

where the value of the on/off flag is 0 (false), the value indicates that the coding tool is not available. Note that the interpretation of the flag value may be reversed.

5 [0176]

An inter-component prediction enabled flag (ccp_enabled_flag) is flag information indicating whether or not inter-component prediction (cross-component prediction (CCP)) is available. For example, in a case
 10 where the flag information is "1" (true), the flag information indicates that the inter-component prediction is available. In a case where the flag information is "0" (false), the flag information indicates that the inter-component prediction is not available.

15 [0177]

Note that this CCP is also referred to as inter-component linear prediction (CCLM or CCLMP).

[0178]

<Prediction Mode Information Pinfo>

20 The prediction mode information Pinfo includes, for example, information such as size information PBlockSize (prediction block size) of a prediction block (PB) to be processed, intra prediction mode information IPinfo, and motion prediction information MVinfo.

25 [0179]

The intra prediction mode information IPinfo includes, for example, prev_intra_luma_pred_flag, mpm_idx, and rem_intra_pred_mode in JCTVC-W1005, 7.3.8.5 Coding Unit syntax, a luminance intra prediction mode
 30 IntraPredModeY derived from the syntax, and the like.

[0180]

Furthermore, the intra prediction mode information IPinfo includes, for example, an inter-component prediction flag (ccp_flag (cclmp_flag)), a multi-class linear prediction mode flag (mclm_flag), a chrominance sample position type identifier (chroma_sample_loc_type_idx), a chrominance MPM identifier (chroma_mpm_idx), a luminance intra prediction mode (IntraPredModeC) derived from these syntaxes, and the like.

10 [0181]

The inter-component prediction flag (ccp_flag (cclmp_flag)) is flag information indicating whether or not to apply inter-component linear prediction. For example, ccp_flag == 1 indicates that inter-component prediction is applied, and ccp_flag == 0 indicates that the inter-component prediction is not applied.

[0182]

The multi-class linear prediction mode flag (mclm_flag) is information regarding a linear prediction mode (linear prediction mode information). More specifically, the multi-class linear prediction mode flag (mclm_flag) is flag information indicating whether or not to set a multi-class linear prediction mode. For example, "0" indicates one-class mode (single glass mode) (for example, CCLMP), and "1" indicates two-class mode (multiclass mode) (for example, MCLMP).

[0183]

The chrominance sample position type identifier (chroma_sample_loc_type_idx) is an identifier for identifying a type of a pixel position of a chrominance component (also referred to as a chrominance sample

30

position type). For example, in a case where the chrominance array type (ChromaArrayType), which is information regarding a color format, indicates 420 format, the chrominance sample position type identifier
 5 is assigned as in the following expression (7).

[0184]

[Math. 5]

```
chroma_sample_loc_type_idx == 0 : Type2
chroma_sample_loc_type_idx == 1 : Type3
chroma_sample_loc_type_idx == 2 : Type0
chroma_sample_loc_type_idx == 3 : Type1
. . . (7)
```

[0185]

10 Note that the chrominance sample position type identifier (chroma_sample_loc_type_idx) is transmitted as (by being stored in) information (chroma_sample_loc_info ()) regarding the pixel position of the chrominance component.

15 [0186]

The chrominance MPM identifier (chroma_mpm_idx) is an identifier indicating which prediction mode candidate in a chrominance intra prediction mode candidate list (intraPredModeCandListC) is to be specified as a
 20 chrominance intra prediction mode.

[0187]

The motion prediction information MVinfo includes, for example, information such as merge_idx, merge_flag, inter_pred_idc, ref_idx_LX, mvp_LX_flag, X = {0,1}, mvd,
 25 and the like (see, for example, JCTVC-W1005, 7.3.8.6 Prediction Unit Syntax).

[0188]

Of course, the information included in the prediction mode information Pinfo is arbitrary, and information other than the above information may be included.

5 [0189]

<Transform Information Tinfo>

The transform information Tinfo includes, for example, the following information. Of course, the information included in the transform information Tinfo
10 is arbitrary, and information other than the above information may be included:

[0190]

The width TBWSize and the height TBHSize of the transformation block to be processed (or may be
15 logarithmic values $\log_2$ TBWSize and $\log_2$ TBHSize of TBWSize and TBHSize having a base of 2);

a transform skip flag (ts_flag): a flag indicating whether or not to skip (inverse) primary transform and (inverse) secondary transform;

20 a scan identifier (scanIdx);

a quantization parameter (qp); and

a quantization matrix (scaling_matrix (for example, JCTVC-W1005, 7.3.4 Scaling list data syntax)).

[0191]

25 <Residual Information Rinfo>

The residual information Rinfo (for example, see 7.3.8.11 Residual Coding syntax of JCTVC-W1005) includes, for example, the following syntaxes:

[0192]

30 cbf (coded_block_flag): a residual data presence/absence flag;

last_sig_coeff_x_pos: a last nonzero coefficient X coordinate;

last_sig_coeff_y_pos: a last nonzero coefficient Y coordinate;

5 coded_sub_block_flag: a subblock nonzero coefficient presence/absence flag;

sig_coeff_flag: a nonzero coefficient presence/absence flag;

gr1_flag: a flag indicating whether or not the
10 level of the nonzero coefficient is greater than 1 (also called a GR1 flag);

gr2_flag: a flag indicating whether or not the level of the nonzero coefficient is greater than 2 (also called a GR2 flag);

15 sign_flag: a sign indicating positive/negative of the nonzero coefficient (also called a sign code);

coeff_abs_level_remaining: a residual level of the nonzero coefficient (also called a nonzero coefficient residual level);

20 and the like.

[0193]

Of course, the information included in the residual information Rinfo is arbitrary, and information other than the above information may be included.

25 [0194]

<Filter Information Finfo>

The filter information Finfo includes, for example, control information regarding the following filtering processing:

30 [0195]

control information regarding a deblocking filter

(DBF);

control information regarding a pixel adaptive
offset (SAO);

control information regarding an adaptive loop
5 filter (ALF); and

control information regarding other linear and
nonlinear filters.

[0196]

More specifically, the filter information Finfo
10 includes, for example, a picture to which each filter is
applied, information for specifying an area in the
picture, filter on/off control information for each CU,
filter on/off control information for slice and tile
boundaries, and the like. Of course, the information
15 included in the filter information Finfo is arbitrary,
and information other than the above information may be
included.

[0197]

Returning to the description of the decoding unit
20 212, the decoding unit 212 refers to the residual
information Rinfo and derives the quantized transform
coefficient level level at each coefficient position in
each transformation block. The decoding unit 212
supplies the quantized transform coefficient level level
25 to the inverse quantization unit 213.

[0198]

Furthermore, the decoding unit 212 supplies the
parsed header information Hinfo, prediction mode
information Pinfo, quantized transform coefficient level
30 level, transform information Tinfo, and filter
information Finfo to each block. Specific description is

given as follows.

[0199]

The header information Hinfo is supplied to the inverse quantization unit 213, the inverse orthogonal
5 transform unit 214, the prediction unit 219, and the in-loop filter unit 216.

The prediction mode information Pinfo is supplied to the inverse quantization unit 213 and the prediction unit 219.

10 The transform information Tinfo is supplied to the inverse quantization unit 213 and the inverse orthogonal transform unit 214.

The filter information Finfo is supplied to the in-loop filter unit 216.

15 [0200]

Of course, the above example is an example, and the present embodiment is not limited to this example. For example, each encoding parameter may be supplied to an arbitrary processing unit. Furthermore, other
20 information may be supplied to an arbitrary processing unit.

[0201]

<Inverse Quantization Unit>

The inverse quantization unit 213 performs
25 processing regarding inverse quantization. For example, the inverse quantization unit 213 uses the transform information Tinfo and the quantized transform coefficient level level supplied from the decoding unit 212 as inputs, and, on the basis of the transform information Tinfo,
30 scales (inversely quantizes) the value of the quantized transform coefficient level level to derive a transform

coefficient Coeff_IQ after inverse quantization.

[0202]

Note that this inverse quantization is performed as inverse processing of the quantization by the quantization unit 114. Furthermore, the inverse quantization is processing similar to the inverse quantization performed by the inverse quantization unit 117. That is, the inverse quantization unit 117 performs processing (inverse quantization) similar to the inverse quantization unit 213.

[0203]

The inverse quantization unit 213 supplies the derived transform coefficient Coeff_IQ to the inverse orthogonal transform unit 214

15 [0204]

<Inverse Orthogonal Transform Unit>

The inverse orthogonal transform unit 214 performs processing regarding inverse orthogonal transform. For example, the inverse orthogonal transform unit 214 uses the transform coefficient Coeff_IQ supplied from the inverse quantization unit 213 and the transform information Tinfo supplied from the decoding unit 212 as inputs, and performs the inverse orthogonal transform processing for the transform coefficient Coeff_IQ on the basis of the transform information Tinfo to derive the prediction residual D' .

[0205]

Note that this inverse orthogonal transform is performed as inverse processing of the orthogonal transform by the orthogonal transform unit 113. Furthermore, the inverse orthogonal transform is

processing similar to the inverse orthogonal transform performed by the inverse orthogonal transform unit 118. That is, the inverse orthogonal transform unit 118 performs processing (inverse orthogonal transform) similar to the inverse orthogonal transform unit 214.

5 [0206]

The inverse orthogonal transform unit 214 supplies the derived prediction residual D' to the calculation unit 215.

10 [0207]

<Calculation Unit>

The calculation unit 215 performs processing regarding addition of information regarding an image. For example, the calculation unit 215 uses the prediction residual D' supplied from the inverse orthogonal transform unit 214 and the predicted image P supplied from the prediction unit 219 as inputs. The calculation unit 215 adds the prediction residual D' and the predicted image P (prediction signal) corresponding to the prediction residual D' to derive the locally decoded image R_{local} , as illustrated in the following expression (8).

15 20

[0208]

[Math. 6]

25
$$R_{\text{local}} = D' + P \quad \dots \quad (8)$$

[0209]

The calculation unit 215 supplies the derived locally decoded image R_{local} to the in-loop filter unit 216 and the frame memory 218.

30 [0210]

<In-loop Filter Unit>

The in-loop filter unit 216 performs processing regarding in-loop filter processing. For example, the in-loop filter unit 216 uses the locally decoded image R_{local} supplied from the calculation unit 215 and the filter information $Finfo$ supplied from the decoding unit 212 as inputs. Note that the information input to the in-loop filter unit 216 may be information other than the aforementioned information.

10 [0211]

The in-loop filter unit 216 appropriately performs filtering processing for the locally decoded image R_{local} on the basis of the filter information $Finfo$.

[0212]

15 For example, the in-loop filter unit 216 applies four in-loop filters of a bilateral filter, a deblocking filter (DBF), an adaptive offset filter (sample adaptive offset (SAO)), and an adaptive loop filter (adaptive loop filter (ALF)) in this order, as described in Non-Patent Document 1. Note that which filter is applied and in which order the filters are applied are arbitrary and can be selected as appropriate.

[0213]

25 The in-loop filter unit 216 performs filtering processing corresponding to the filtering processing performed on the encoding side (for example, by the in-loop filter unit 120 of the image encoding device 100). Of course, the filtering processing performed by the in-loop filter unit 216 is arbitrary, and is not limited to the above example. For example, the in-loop filter unit 30 216 may apply a Wiener filter or the like.

[0214]

The in-loop filter unit 216 supplies the filtered locally decoded image R_{local} to the rearrangement buffer 217 and the frame memory 218.

5 [0215]

<Rearrangement Buffer>

The rearrangement buffer 217 uses the locally decoded image R_{local} supplied from the in-loop filter unit 216 and holds (stores) the locally decoded image R_{local} .

10 The rearrangement buffer 217 reconstructs the decoded image R for each unit of picture, using the locally decoded image R_{local} , and holds (stores) the decoded image R (in the buffer). The rearrangement buffer 217 rearranges the obtained decoded images R from the
15 decoding order to the reproduction order. The rearrangement buffer 217 outputs a rearranged decoded image R group to the outside of the image decoding device 200 as moving image data.

[0216]

20 <Frame Memory>

The frame memory 218 performs processing regarding storage of data relating to an image. For example, the frame memory 218 uses the locally decoded image R_{local} supplied from the calculation unit 215 as an input,
25 reconstructs the decoded image R for each unit of picture, and stores the decoded image R in the buffer in the frame memory 218.

[0217]

Furthermore, the frame memory 218 uses the in-loop
30 filtered locally decoded image R_{local} supplied from the in-loop filter unit 216 as an input, reconstructs the

decoded image R for each unit of picture, and stores the decoded image R in the buffer in the frame memory 218.

The frame memory 218 appropriately supplies the stored decoded image R (or a part thereof) to the prediction

5 unit 219 as a reference image.

[0218]

Note that the frame memory 218 may store the header information Hinfo, the prediction mode information Pinfo, the transform information Tinfo, the filter information

10 Finfo, and the like related to generation of the decoded image.

[0219]

<Prediction Unit>

The prediction unit 219 performs processing

15 regarding generation of a predicted image. For example, the prediction unit 219 uses the prediction mode information Pinfo supplied from the decoding unit 212 as an input, and performs prediction by a prediction method specified by the prediction mode information Pinfo to

20 derive the predicted image P. At the time of derivation, the prediction unit 219 uses the decoded image R (or a part thereof) before filtering or after filtering stored in the frame memory 218, the decoded image R being specified by the prediction mode information Pinfo, as

25 the reference image. The prediction unit 219 supplies the derived predicted image P to the calculation unit 215.

[0220]

In the image decoding device 200 having the above configuration, the inverse orthogonal transform unit 214

30 performs processing to which the above-described present technology is applied, as a derivation unit and an

inverse orthogonal transform unit. Furthermore, the decoding unit 212 performs processing to which the above-described present technology is applied, as a decoding unit. Therefore, the image decoding device 200 can suppress an increase in the memory capacity required for the inverse orthogonal transform.

[0221]

<Details of Inverse Orthogonal Transform Unit>

Fig. 16 is a block diagram illustrating a main configuration example of the inverse orthogonal transform unit 214 in Fig. 15. As illustrated in Fig. 16, the inverse orthogonal transform unit 214 includes a switch 251, an inverse secondary transform unit 252, and an inverse primary transform unit 253.

[0222]

The switch 251 uses the transform coefficient Coeff_IQ and the transform skip flag ts_flag [compID] as inputs. In a case where the value of the transform skip flag ts_flag [compID] is NO_TS (= 0), that is, in a case where a transform skip is not applied, the switch 251 supplies the transform coefficient Coeff_IQ to the inverse secondary transform unit 252. Furthermore, in a case where the value of the transform skip flag ts_flag [compID] is 2D_TS (=1), that is, in a case where a two-dimensional transform skip is applied, the switch 251 skips the inverse secondary transform unit 252 and the inverse primary transform unit 253, and outputs the transform coefficient Coeff_IQ to the outside of the inverse orthogonal transform unit 214 (supplies the same to the calculation unit 215) as the prediction residual D' .

[0223]

The inverse secondary transform unit 252 performs processing regarding inverse secondary transform that is inverse processing of the secondary transform performed
5 on the encoding side (for example, by the secondary transform unit 153 of the image encoding device 100). For example, the inverse secondary transform unit 252 uses the secondary transform identifier `st_idx`, the scan identifier `scanIdx` indicating a method of scanning the
10 transform coefficient, and the transform coefficient `Coeff_IQ` supplied from the switch 251 as inputs.

[0224]

The inverse secondary transform unit 252 performs inverse secondary transform for the transform coefficient
15 `Coeff_IQ` on the basis of the secondary transform identifier `st_idx` and the scan identifier `scanIdx` to derive a transform coefficient `Coeff_IS` after inverse secondary transform.

[0225]

20 More specifically, in a case where the secondary transform identifier `st_idx` indicates application of inverse secondary transform (`st_idx > 0`), the inverse secondary transform unit 252 executes, for the transform coefficient `Coeff_IQ`, inverse secondary transform
25 processing corresponding to the secondary transform identifier `st_idx` to derive the transform coefficient `Coeff_IS` after secondary transform. The inverse secondary transform unit 252 supplies the transform coefficient `Coeff_IS` after inverse secondary transform to
30 the inverse primary transform unit 253.

[0226]

Note that in a case where the secondary transform identifier `st_idx` indicates that the inverse secondary transform is not applied (`st_idx == 0`), the inverse secondary transform unit 252 skips the inverse secondary transform, and supplies the transform coefficient `Coeff_IQ` to the inverse primary transform unit 253 as the transform coefficient `Coeff_IS` after inverse secondary transform.

[0227]

10 The inverse primary transform unit 253 performs processing related to inverse primary transform that is inverse processing of the primary transform performed on the encoding side (for example, by the primary transform unit 152 of the image encoding device 100). For example,

15 the inverse primary transform unit 253 uses, as inputs, the component identifier `compID`, an adaptive primary transform flag `apt_flag[compID]` of the component identifier `compID`, a primary transform identifier `pt_idx[compID]` of the component identifier `compID`, the

20 prediction mode information `PInfo`, the size of the transformation block (the logarithmic value of a width `log2TBWSize` and the logarithmic value of a height `log2TBHSize`), and the transform coefficient `Coeff_IS` after inverse secondary transform.

25 [0228]

 The inverse primary transform unit 253 selects a transform type `TrTypeH` of inverse primary horizontal transform (and an inverse primary horizontal transform type identifier `TrTypeIdxH` indicating the transform type)

30 and a transform type `TrTypeV` of inverse primary vertical transform (and an inverse primary vertical transform type

identifier `TrTypeIdxV` indicating the transform type)
 corresponding to the component identifier `compID` by
 reference to the prediction mode information `PInfo`, the
 component identifier `compID`, the adaptive primary
 5 transform flag `apt_flag[compID]` of the component
 identifier `compID`, and the primary transform identifier
`pt_idx[compID]` of the component identifier `compID`.

[0229]

Furthermore, the inverse primary transform unit 253
 10 performs, for the transform coefficient `Coeff_IS` after
 inverse secondary transform, inverse primary vertical
 transform defined by the inverse primary vertical
 transform type identifier `TrTypeIdxV` (or the inverse
 primary vertical transform type `TrTypeV`) and the height
 15 `log2TBHSize` of the transformation block, and inverse
 primary horizontal transform defined by the inverse
 primary horizontal transform type identifier `TrTypeIdxH`
 (or the inverse primary horizontal transform type
 Transform type `TrTypeH`) and the width `log2TBWSize` of the
 20 transformation block, to derive a transform coefficient
`Coeff_IP` after inverse primary vertical transform. The
 inverse primary vertical transform is vertical inverse
 one-dimensional orthogonal transform and the inverse
 primary horizontal transform is horizontal inverse one-
 25 dimensional orthogonal transform.

[0230]

The inverse primary transform unit 253 outputs the
 transform coefficient `Coeff_IP` after inverse primary
 transform to the outside of the inverse orthogonal
 30 transform unit 214 (supplies the same to the calculation
 unit 215) as the prediction residual D' .

[0231]

In the inverse orthogonal transform unit 214 having the above configuration, the inverse primary transform unit 253 performs processing to which the above-described present technology is applied, as a derivation unit and an inverse orthogonal transform unit. That is, the derivation unit derives the second transformation matrix, using the first transformation matrix, and the inverse orthogonal transform unit performs the inverse primary transform for the inverse secondary transform result, using the second transformation matrix derived by the derivation unit. Therefore, an increase in the memory capacity required for the inverse primary transform can be suppressed.

15 [0232]

Note that the above-described inverse primary transform unit 253 performs the inverse primary vertical transform and the inverse primary horizontal transform as the inverse primary transform. That is, the derivation unit derives the second transformation matrix for vertical inverse one-dimensional orthogonal transform and the second transformation matrix for horizontal inverse one-dimensional orthogonal transform, and the inverse orthogonal transform unit performs, as the inverse primary transform, the vertical inverse one-dimensional orthogonal transform, using the second transformation matrix for vertical inverse one-dimensional orthogonal transform derived by the derivation unit, and further, the horizontal inverse one-dimensional orthogonal transform, using the second transformation matrix for horizontal inverse one-dimensional orthogonal transform

inverse one-dimensional orthogonal transform derived by the derivation unit. Therefore, an increase in the memory capacity required for the primary transform in which the vertical inverse one-dimensional orthogonal transform and the horizontal inverse one-dimensional orthogonal transform are performed can be suppressed.

[0233]

<Flow of Image Decoding Processing>

Next, a flow of each processing executed by the above image decoding device 200 will be described. First, an example of a flow of image decoding processing will be described with reference to the flowchart in Fig. 17.

[0234]

When the image decoding processing is started, in step S201, the accumulation buffer 211 acquires and holds (accumulates) the coded data (bit stream) supplied from the outside of the image decoding device 200.

[0235]

In step S202, the decoding unit 212 decodes the coded data (bit stream) to obtain a quantized transform coefficient level level. Furthermore, the decoding unit 212 parses (analyzes and acquires) various encoding parameters from the coded data (bit stream) by this decoding.

[0236]

In step S203, the inverse quantization unit 213 performs inverse quantization that is inverse processing of the quantization performed on the encoding side for the quantized transform coefficient level level obtained by the processing in step S202 to obtain the transform coefficient Coeff_IQ.

[0237]

In step S204, the inverse orthogonal transform unit 214 performs inverse orthogonal transform processing that is inverse processing of the orthogonal transform processing performed on the encoding side for the transform coefficient Coeff_IQ obtained by the processing in step S203 to obtain the prediction residual D' .

[0238]

In step S205, the prediction unit 219 executes prediction processing by a prediction method specified on the encoding side on the basis of the information parsed in step S202, and generates a predicted image P , for example, by reference to the reference image stored in the frame memory 218.

15 [0239]

In step S206, the calculation unit 215 adds the prediction residual D' obtained by the processing in step S204 and the predicted image P obtained by the processing in step S205 to derive a locally decoded image R_{local} .

20 [0240]

In step S207, the in-loop filter unit 216 performs the in-loop filter processing for the locally decoded image R_{local} obtained by the processing in step S206.

[0241]

25 In step S208, the rearrangement buffer 217 derives a decoded image R , using the filtered locally decoded image R_{local} obtained by the processing in step S207, and rearranges a decoded image R group from the decoding order to the reproduction order. The decoded image R group rearranged in the reproduction order is output to the outside of the image decoding device 200 as a moving

30

image.

[0242]

Furthermore, in step S209, the frame memory 218 stores at least one of the locally decoded image R_{local} obtained by the processing in step S206, and the locally decoded image R_{local} after filtering processing obtained by the processing in step S207.

[0243]

When the processing in step S209 ends, the image decoding processing ends.

[0244]

In the image decoding processing of the above flow, processing to which the above-described present technology is applied is performed as processing of step S202. Furthermore, processing to which the above-described present technology is applied is performed as processing of step S204. Therefore, by executing the image decoding processing, an increase in the memory capacity required for the inverse orthogonal transform can be suppressed.

[0245]

<Flow of Inverse Orthogonal Transform Processing>

Next, an example of a flow of the inverse orthogonal transform processing executed in step S204 in Fig. 17 will be described with reference to the flowchart in Fig. 18. When the inverse orthogonal transform processing is started, in step S231, the switch 251 determines whether or not the transform skip flag ts_flag is 2D_TS (in a mode of a two-dimensional transform skip) (for example, 1 (true)) or the transform quantization bypass flag $transquant_bypass_flag$ is 1 (true). In a

case where it is determined that the transform skip identifier `ts_idx` is 2D_TS or the transform quantization bypass flag is 1 (true), the inverse orthogonal transform processing ends, and the processing returns to Fig. 17.

5 In this case, the inverse orthogonal transform processing (the inverse primary transform and the inverse secondary transform) is omitted, and the transform coefficient `Coeff_IQ` is adopted as the prediction residual D' .
[0246]

10 Furthermore, in step S231, in a case where it is determined that the transfer skip identifier `ts_idx` is not 2D_TS (a mode other than the two-dimensional transform skip) (for example, 0 (false)), and the transform quantization bypass flag is 0 (false), the
15 processing proceeds to step S232. In this case, the inverse secondary transform processing and the inverse primary transform processing are performed.
[0247]

In step S232, the inverse secondary transform unit
20 252 performs the inverse secondary transform processing for the transform coefficient `Coeff_IQ` on the basis of the secondary transform identifier `st_idx` to derive a transform coefficient `Coeff_IS`, and outputs the transform coefficient `Coeff_IS`.

25 [0248]

In step S233, the inverse primary transform unit
253 performs the inverse primary transform processing for the transform coefficient `Coeff_IS` to derive a transform coefficient `Coeff_IP` (prediction residual D') after
30 inverse primary transform.
[0249]

When the processing in step S233 ends, the inverse orthogonal transform processing ends.

[0250]

In the above inverse orthogonal transform
 5 processing, processing to which the above-described present technology is applied is performed as processing of step S233. Therefore, by executing the inverse orthogonal transform processing, an increase in the memory capacity required for the inverse primary
 10 transform processing can be suppressed.

[0251]

<2-2. Example 1-1>

<Concept>

Next, each derivation example described with
 15 reference to Fig. 10 will be described in more detail. First, the first row example and the second row example from the top except the uppermost row of item names in the table illustrated in Fig. 10 will be described.

[0252]

20 As described above, the derivation of the first row example from the top focuses on the similarity between the waveform of the lowest-order row vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix
 25 to be substituted. In this case, the derivation unit flips the first transformation matrix to derive the second transformation matrix. That is, the derivation unit uses the transformation matrix of DST7 as the base transformation matrix T_{base} , and flips the transformation
 30 matrix in the row direction to derive the transformation matrix of FlipDST7, as illustrated in Fig. 19. The

(decreasing-type) transformation matrix of DCT8 having a similar waveform of the 0-order row vector can be substituted by the transformation matrix of FlipDST7. [0253]

5 A specific example of this derivation is illustrated in the upper part in Fig. 20. As illustrated in the upper part in Fig. 20, this derivation can be expressed by a matrix product of the base transformation matrix T_{base} (DST7) and a flip matrix J . Here, the flip
10 matrix J (also referred to as cross-identity matrix) is obtained by right-left inverting an $N \times N$ unit matrix I . [0254]

Furthermore, the derivation of the second row example from the top focuses on the similarity between
15 the waveform of the lowest-order column vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted. In this case, the derivation unit transposes the first transformation matrix to derive
20 the second transformation matrix. That is, the derivation unit uses the transformation matrix of DST7 as the base transformation matrix T_{base} , and transposes the transformation matrix to derive the transformation matrix of TrDST7, as illustrated in Fig. 19. The (chevron-type)
25 transformation matrix of DST1 having a similar waveform of the 0-order row vector can be substituted by the transformation matrix of TrDST7. [0255]

A specific example of this derivation is
30 illustrated in the lower part in Fig. 20. As illustrated in the lower part in Fig. 20, this derivation can be

expressed by transposition of the base transformation matrix T_{base} (DST7).

[0256]

That is, in both of the above two derivation examples, the second transformation matrix can be derived by one-time operation (flip or transposition). Furthermore, the operation is easy. That is, the second transformation matrix can be easily derived.

[0257]

Furthermore, by applying the above two derivation examples, it becomes unnecessary to prepare the transformation matrix of DCT8 and the transformation matrix of DST1 as candidates for transformation matrices to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced.

[0258]

In this case, as illustrated in the table in Fig. 21, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to three types. Therefore, the total LUT size can be about 40 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.

[0259]

Note that, as described above, even in this case, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived

second transform type (TrDST7 or FlipDST7), similar coding efficiency to the case of using the transformation matrix of DST1 or the transformation matrix of DCT8 for the orthogonal transform/inverse orthogonal transform can be obtained.

[0260]

<Primary Transform Unit>

Next, configurations, processing, and the like for performing such derivation will be described. Fig. 22 is a block diagram illustrating a main configuration example of the primary transform unit 152 in this case. As illustrated in Fig. 22, the primary transform unit 152 includes a primary transform selection unit 311, a primary horizontal transform unit 312, and a primary vertical transform unit 313.

[0261]

The primary transform selection unit 311 uses the prediction mode information PInfo, the component identifier compID, the adaptive primary transform flag apt_flag[compID], and the primary transform identifier pt_idx[compID] as inputs. The primary transform selection unit 311 derives the transform type identifier TrTypeIdxH of primary horizontal transform and the transform type identifier TrTypeIdxV of primary vertical transform by reference to the above information. The primary transform selection unit 311 supplies the derived transform type identifier TrTypeIdxH of primary horizontal transform to the primary horizontal transform unit 312. Furthermore, the primary transform selection unit 311 supplies the derived transform type identifier TrTypeIdxV of primary vertical transform to the primary

vertical transform unit 313.

[0262]

The primary horizontal transform unit 312 uses the prediction residual D , the transform type identifier $TrTypeIdxH$ of primary horizontal transform, and information regarding the size of the transformation block (not illustrated) as inputs. The information regarding the size of the transformation block may be a natural number N indicating the size of the transformation block in the horizontal direction or the vertical direction (the number of coefficients), or may be $\log_2 TBWSize$ (the logarithmic value of the width) indicating the width of the transformation block ($N = 1 \ll \log_2 TBWSize$). The primary horizontal transform unit 312 executes, for the prediction residual D , primary horizontal transform $Phor$ determined by the transform type identifier $TrTypeIdxH$ and the size of the transformation block to derive a transform coefficient $Coeff_Phor$ after primary horizontal transform. The primary horizontal transform unit 312 supplies the transform coefficient $Coeff_Phor$ after primary horizontal transform to the primary vertical transform unit 313.

[0263]

The primary vertical transform unit 313 uses the transform coefficient $Coeff_Phor$ after primary horizontal transform, the transform type identifier $TrTypeIdxV$ of primary vertical transform, and information regarding the size of the transformation block (not illustrated) as inputs. The information regarding the size of the transformation block may be a natural number N indicating the size of the transformation block in the horizontal

direction or the vertical direction (the number of coefficients), or may be $\log_2 \text{TBHSize}$ (the logarithmic value of the height) indicating the height of the transformation block ($N = 1 \ll \log_2 \text{TBHSize}$). The primary vertical transform unit 313 executes, for the transform coefficient Coeff_Phor after primary horizontal transform, primary vertical transform Pver determined by the transform type identifier TrTypeIdxV and the size of the transformation block to derive a transform coefficient Coeff_Pver after primary vertical transform. The primary vertical transform unit 313 outputs the transform coefficient Coeff_Pver after primary vertical transform to the outside of the primary transform unit 152 (supplies the same to the secondary transform unit 153) as the transform coefficient Coeff_P after primary transform.

[0264]

In the primary transform unit 152 having the above configuration, the primary horizontal transform unit 312 and the primary vertical transform unit 313 perform processing to which the above-described present technology is applied, as a derivation unit and an orthogonal transform unit, respectively.

[0265]

That is, the primary horizontal transform unit 312 derives, as a derivation unit, the second transformation matrix for horizontal one-dimensional orthogonal transform, and further performs, as an orthogonal transform unit, horizontal one-dimensional orthogonal transform, using the second transformation matrix for horizontal one-dimensional orthogonal transform derived

by the derivation unit. Therefore, the primary horizontal transform unit 312 can suppress an increase in the memory capacity required for the horizontal one-dimensional orthogonal transform.

5 [0266]

Furthermore, the primary vertical transform unit 313 derives, as a derivation unit, the second transformation matrix for vertical one-dimensional orthogonal transform, and further performs, as an orthogonal transform unit, the vertical one-dimensional orthogonal transform, using the second transformation matrix for vertical one-dimensional orthogonal transform derived by the derivation unit. Therefore, the primary vertical transform unit 313 can suppress an increase in the memory capacity required for the vertical one-dimensional orthogonal transform.

15

[0267]

<Primary Horizontal Transform Unit>

Fig. 23 is a block diagram illustrating a main configuration example of the primary horizontal transform unit 312 in Fig. 22. As illustrated in Fig. 23, the primary horizontal transform unit 312 includes a transformation matrix derivation unit 321, a matrix calculation unit 322, a scaling unit 323, and a clip unit 324.

25

[0268]

The transformation matrix derivation unit 321 uses the transform type identifier $TrTypeIdxH$ of primary horizontal transform and the information regarding the size of the transformation block as inputs, and derives a transformation matrix T_H for primary horizontal transform

30

(a transformation matrix T_H for horizontal one-dimensional orthogonal transform) having the same size as the transformation block, the transformation matrix T_H corresponding to the transform type identifier $TrTypeIdxH$ of primary horizontal transform. The transformation matrix derivation unit 321 supplies the transformation matrix T_H to the matrix calculation unit 322.

[0269]

The matrix calculation unit 322 performs the horizontal one-dimensional orthogonal transform for input data X_{in} (that is, the transformation block of the prediction residual D), using the transformation matrix T_H supplied from the transformation matrix derivation unit 321, to obtain intermediate data $Y1$. This calculation can be expressed by a determinant as in the following expression (9).

[0270]

[Math. 7]

$$Y1 = X_{in} \times T_H^T \quad \cdot \cdot \cdot \quad (9)$$

20 [0271]

The matrix calculation unit 322 supplies the intermediate data $Y1$ to the scaling unit 323.

[0272]

The scaling unit 323 scales a coefficient $Y1[i, j]$ of each i -row j -column component of the intermediate data $Y1$ with a predetermined shift amount S_H to obtain intermediate data $Y2$. This scaling can be expressed as the following expression (10). Hereinafter, the i -row j -column component ((i, j) component) of a certain two-dimensional matrix (two-dimensional array) X is written

30

as $X[i, j]$.

[0273]

[Math. 8]

$$Y2[i, j] = Y1[i, j] \gg S_H \cdot \cdot \cdot (1\ 0)$$

5 [0274]

The scaling unit 323 supplies the intermediate data $Y2$ to the clip unit 324.

[0275]

The clip unit 324 clips a value of a coefficient $Y2$
 10 $[i, j]$ of each i -row j -column component of the
 intermediate data $Y2$, and derives output data X_{out} (that
 is, the transform coefficient $Coeff_Phor$ after primary
 horizontal transform). This processing can be expressed
 as the following expression (11).

15 [0276]

[Math. 9]

$$X_{out}[i, j] = Clip3(minCoefVal, maxCoefVal, Y2[i, j]) \cdot \cdot \cdot (1\ 1)$$

[0277]

The clip unit 324 outputs the output data X_{out} (the
 20 transform coefficient $Coeff_Phor$ after primary horizontal
 transform) to the outside of the primary horizontal
 transform unit 312 (supplies the same to the primary
 vertical transform unit 313).

[0278]

25 In the primary horizontal transform unit 312 having
 the above configuration, the transformation matrix
 derivation unit 321 performs processing to which the
 above-described present technology is applied, as a
 derivation unit. Furthermore, the matrix calculation

unit 322 performs processing to which the above-described present technology is applied, as an orthogonal transform unit. Therefore, the primary horizontal transform unit 312 can suppress an increase in the memory capacity required for the horizontal one-dimensional orthogonal transform.

[0279]

<Transformation Matrix Derivation Unit>

Fig. 24 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 321 in Fig. 23. As illustrated in Fig. 24, the transformation matrix derivation unit 321 includes a transformation matrix LUT 331, a flip unit 332, and a transposition unit 333. Note that, in Fig. 24, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 321, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0280]

The transformation matrix LUT 331 is a lookup table for holding (storing) a transformation matrix corresponding to the transform type identifier $TrTypeIdxH$ of primary horizontal transform and a size N of the transformation block. When the transform type identifier $TrTypeIdxH$ of primary horizontal transform and the size N of the transformation block are specified, the transformation matrix LUT 331 selects and outputs a transformation matrix corresponding thereto. In the case of this derivation example, the transformation matrix LUT 331 supplies the transformation matrix to both or one of the flip unit 332 and the transposition unit 333 as the

base transformation matrix T_{base} .

[0281]

The flip unit 332 flips an input transformation matrix T of N rows and N columns, and outputs a flipped transformation matrix T_{flip} . In the case of this
 5 derivation example, the flip unit 332 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 331 as an input, flips the base transformation matrix T_{base} in the
 10 row direction (horizontal direction), and outputs the flipped transformation matrix T_{flip} to the outside of the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

15 [0282]

The transposition unit 333 transposes the input transformation matrix T of N rows and N columns, and outputs a transposed transformation matrix $T_{\text{transpose}}$. In the case of this derivation example, the transposition
 20 unit 333 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 331 as an input, transposes the base transformation matrix T_{base} , and outputs the transposed transformation matrix $T_{\text{transpose}}$ to the outside of the
 25 transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

[0283]

As described above, the transformation matrix
 30 derivation unit 321 includes the flip unit 332 and the transposition unit 333. Therefore, the transformation

matrix derivation unit 321 can implement the derivation example of the first row example from the top of the table illustrated in Fig. 10, using the flip unit 332, for example. Furthermore, the transformation matrix derivation unit 321 can implement the derivation example of the second row example from the top of the table illustrated in Fig. 10, using the transposition unit 333, for example.

[0284]

10 <Primary Vertical Transform Unit>

Fig. 25 is a block diagram illustrating a main configuration example of the primary vertical transform unit 313 in Fig. 22. As illustrated in Fig. 25, the primary vertical transform unit 313 includes a transformation matrix derivation unit 351, a matrix calculation unit 352, a scaling unit 353, and a clip unit 354.

[0285]

The transformation matrix derivation unit 351 uses the transform type identifier $TrTypeIdxV$ of primary vertical transform and the information regarding the size of the transformation block as inputs, and derives a transformation matrix T_V for primary vertical transform (a transformation matrix T_V for vertical one-dimensional orthogonal transform) having the same size as the transformation block, the transformation matrix T_V corresponding to the transform type identifier $TrTypeIdxV$ of primary vertical transform. The transformation matrix derivation unit 351 supplies the transformation matrix T_V to the matrix calculation unit 352.

[0286]

The matrix calculation unit 352 performs the vertical one-dimensional orthogonal transform for the input data X_{in} (that is, the transformation block of the transform coefficient $Coeff_Phor$ after primary horizontal transform), using the transformation matrix T_v supplied from the transformation matrix derivation unit 351, to obtain intermediate data $Y1$. This calculation can be expressed by a determinant as in the following expression (12).

10 [0287]

[Math. 10]

$$Y1 = T_v \times X_{in} \quad \cdot \cdot \cdot \quad (12)$$

[0288]

The matrix calculation unit 352 supplies the intermediate data $Y1$ to the scaling unit 353.

15

[0289]

The scaling unit 353 scales a coefficient $Y1[i, j]$ of each i -row j -column component of the intermediate data $Y1$ with a predetermined shift amount S_v to obtain intermediate data $Y2$. This scaling can be expressed as the following expression (13).

20

[0290]

[Math. 11]

$$Y2[i, j] = Y1[i, j] \gg S_v \quad \cdot \cdot \cdot \quad (13)$$

25 [0291]

The scaling unit 353 supplies the intermediate data $Y2$ to the clip unit 354.

[0292]

The clip unit 354 clips a value of a coefficient $Y2$

[i, j] of each i-row j-column component of the intermediate data Y2, and derives output data X_{out} (that is, the transform coefficient Coeff_Pver after primary vertical transform). This processing can be expressed as
 5 the following expression (14).

[0293]

[Math. 12]

$$X_{out}[i, j] = \text{Clip3}(\text{minCoefVal}, \text{maxCoefVal}, Y2[i, j]) \quad \cdot \cdot \cdot \quad (14)$$

[0294]

10 The clip unit 324 outputs the output data X_{out} (transform coefficient Coeff_Pver after primary vertical transform) to the outside of the primary vertical transform unit 313 (supplies the same to the secondary transform unit 153) as the transform coefficient Coeff_P
 15 after primary transform.

[0295]

 In the primary vertical transform unit 313 having the above configuration, the transformation matrix derivation unit 351 performs processing to which the
 20 above-described present technology is applied, as a derivation unit. Furthermore, the matrix calculation unit 352 performs processing to which the above-described present technology is applied, as an orthogonal transform unit. Therefore, the primary vertical transform unit 313
 25 can suppress an increase in the memory capacity required for the vertical one-dimensional orthogonal transform.

[0296]

<Transformation Matrix Derivation Unit>

 Fig. 26 is a block diagram illustrating a main
 30 configuration example of the transformation matrix derivation unit 351 in Fig. 25. As illustrated in Fig.

26, the transformation matrix derivation unit 351 includes a transformation matrix LUT 361, a flip unit 362, and a transposition unit 363. Note that, in Fig. 26, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 351, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0297]

The transformation matrix LUT 361 is a lookup table for holding (storing) a transformation matrix corresponding to the transform type identifier TrTypeIdxV of primary vertical transform and the size N of the transformation block. When the transform type identifier TrTypeIdxV of primary vertical transform and the size N of the transformation block are specified, the transformation matrix LUT 361 selects and outputs a transformation matrix corresponding thereto. In the case of this derivation example, the transformation matrix LUT 361 supplies the transformation matrix to both or one of the flip unit 362 and the transposition unit 363 as the base transformation matrix T_{base} .

[0298]

The flip unit 362 flips an input transformation matrix T of N rows and N columns, and outputs a flipped transformation matrix T_{flip} . In the case of this derivation example, the flip unit 362 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 361 as an input, flips the base transformation matrix T_{base} in the row direction (horizontal direction), and outputs the flipped transformation matrix T_{flip} to the outside of the

transformation matrix derivation unit 351 (supplies the same to the matrix calculation unit 352) as the transformation matrix T_v .

[0299]

5 The transposition unit 363 transposes the input transformation matrix T of N rows and N columns, and outputs a transposed transformation matrix $T_{\text{transpose}}$. In the case of this derivation example, the transposition unit 363 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 361 as an input, transposes the base transformation matrix T_{base} , and outputs the transposed transformation matrix $T_{\text{transpose}}$ to the outside of the transformation matrix derivation unit 351 (supplies the same to the matrix calculation unit 352) as the transformation matrix T_v .

[0300]

As described above, the transformation matrix derivation unit 351 includes the flip unit 362 and the transposition unit 363. Therefore, the transformation matrix derivation unit 351 can implement the derivation example of the first row example from the top of the table illustrated in Fig. 10, using the flip unit 362, for example. Furthermore, the transformation matrix derivation unit 351 can implement the derivation example of the second row example from the top of the table illustrated in Fig. 10, using the transposition unit 363, for example.

[0301]

30 <Flow of Primary Transform Processing>

Next, an example of a flow of processing performed

by the above-described configuration will be described.
An example of a flow of the primary transform processing
executed in step S132 in Fig. 14 in this case will be
described with reference to the flowchart in Fig. 27.

5 [0302]

When the primary transform processing is started,
in step S301, the primary transform selection unit 311
(Fig. 22) of the primary transform unit 152 selects the
transform type identifier TrTypeIdxH of primary
10 horizontal transform (and the transform type TrTypeH
specified with the identifier), and the transform type
identifier TrTypeIdxV of primary vertical transform (the
transform type TrTypeV specified with the identifier), as
described above.

15 [0303]

In step S302, the primary horizontal transform unit
312 performs, for the prediction residual D, the primary
horizontal transform processing corresponding to the
transform type identifier TrTypeIdxH of primary
20 horizontal transform obtained in step S301 to derive the
transform coefficient Coeff_Phori after primary horizontal
transform.

[0304]

In step S303, the primary vertical transform unit
25 313 performs, for a primary horizontal transform result
(the transform coefficient Coeff_Phori after primary
horizontal transform), the primary vertical transform
processing corresponding to the transform type identifier
TrTypeIdxV of primary vertical transform obtained in step
30 S301 to derive the transform coefficient Coeff_Pveri after
primary vertical transform (the transform coefficient

Coeff_P after primary transform).

[0305]

When the processing in step S303 ends, the primary transform processing ends and the processing returns to

5 Fig. 14.

[0306]

In the above primary transform processing, processing to which the above-described present technology is applied is performed as processing of step
10 S302 or S303. Therefore, by executing the primary transform processing, an increase in the memory capacity required for the primary horizontal transform processing and the primary vertical transform processing can be suppressed.

15 [0307]

<Flow of Primary Horizontal Transform Processing>

A flow of the primary horizontal transform processing executed in step S302 in Fig. 27 will be described with reference to the flowchart in Fig. 28.

20 [0308]

When the primary horizontal transform processing is started, in step S321, the transformation matrix derivation unit 321 (Fig. 23) of the primary horizontal transform unit 312 derives the transformation matrix T_H
25 corresponding to the transform type identifier TrTypeIdxH (or the transform type TrTypeH) of primary horizontal transform.

[0309]

In step S322, the matrix calculation unit 322
30 performs the horizontal one-dimensional orthogonal transform for the input data X_{in} (prediction residual D),

using the derived transformation matrix T_H , to obtain the intermediate data $Y1$. When this processing is expressed as a determinant, the processing can be expressed as the above-described expression (9). Furthermore, when this processing is expressed as an operation for each element, the processing can be expressed as the following expression (15).

[0310]

[Math. 13]

$$Y1[i, j] = X_{in}[i, :] \times T_H^T[:, j] = \sum_{k=0}^{N-1} X_{in}[i, k] T_H[j, k] \quad \dots \quad (15)$$

[0311]

That is, as illustrated in Fig. 29, an inner product of an i -th row vector $X_{in}[i, :]$ of the input data X_{in} and a transformation matrix $T_H^T[:, j]$ of a j -th row vector $T_H[j, :]$ of the transformation matrix T_H is set as a coefficient $Y1[i, j]$ of the i -row j -column component of the intermediate data $Y1$ ($j = 0, \dots, M - 1$, and $i = 0, \dots, N - 1$). Here, M represents the size of the input data X_{in} in the x direction, and N represents the size of the input data X_{in} in the y direction. M and N can be expressed as the following expressions (16).

[0312]

[Math. 14]

$$\begin{aligned} M &= 1 \ll \log_2 TBWSize \\ N &= 1 \ll \log_2 TBHSize \\ &\dots \quad (16) \end{aligned}$$

[0313]

Returning to Fig. 28, in step S323, the scaling unit 323 scales, with the shift amount S_H , a coefficient

Y1 [i, j] of each i-row j-column component of the intermediate data Y1 derived by the processing in step S322 to derive intermediate data Y2. This scaling can be expressed as the above-described expression (10).

5 [0314]

In step S324, the clip unit 324 clips the value of the coefficient Y2 [i, j] of each i-row j-column component of the intermediate data Y2 derived by the processing in step S323, and obtains output data X_{out} (that is, the transform coefficient Coeff_Phor after primary horizontal transform). This processing can be expressed as the above-described expression (11).

10 [0315]

When the processing in step S324 ends, the primary horizontal transform processing ends and the processing returns to Fig. 27.

15 [0316]

In the above primary horizontal transform processing, processing to which the above-described present technology is applied is performed as processing of step S321 or S322. Therefore, by executing the primary horizontal transform processing, an increase in the memory capacity required for the horizontal one-dimensional orthogonal transform can be suppressed.

20 [0317]

<Flow of Transformation Matrix Derivation Processing>

Next, an example of a flow of the transformation matrix derivation processing executed in step S321 in Fig. 28 will be described with reference to the flowchart in Fig. 30.

30

[0318]

When the transformation matrix derivation processing is started, in step S341, the transformation matrix derivation unit 321 obtains a base transform type BaseTrType corresponding to the transform type identifier TrTypeIdxH by reference to the correspondence table illustrated in Fig. 31, for example. Note that, when this processing is expressed as a mathematical expression, the processing can be expressed as the expression (17), for example. Moreover, the transformation matrix of N rows and N columns of the obtained base transform type is read from the transformation matrix LUT, and is set as the base transformation matrix T_{base} , as in the following expression (18).

15 [0319]

[Math. 15]

$$\begin{aligned} \text{BaseTrType} &= \text{LUT_TrTypeIdxToBaseTrType}[\text{TrTypeIdxH}] \\ &\quad \cdot \cdot \cdot (17) \\ T_{base} &= T[\text{BaseTrType}][\log_2 N - 1] \\ &\quad \cdot \cdot \cdot (18) \end{aligned}$$

[0320]

Furthermore, the transformation matrix derivation unit 321 sets a value corresponding to the transform type identifier TrTypeIdxH as a flip flag FlipFlag, as in the following expression (19). Furthermore, the transformation matrix derivation unit 321 sets a value corresponding to the transform type identifier TrTypeIdxH as a transposition flag TransposeFlag, as in the following expression (20).

[0321]

[Math. 16]


```

FlipFlag = LUT_TrTypeIdxToFlipFlag[TrTypeIdxH]
. . . (19)
TransposeFlag = LUT_TrTypeIdxToTransposeFlag[TrTypeIdxH]
. . . (20)

```

[0322]

In step S342, the transformation matrix derivation unit 321 determines whether or not the flip flag FlipFlag and the transposition flag TransposeFlag satisfy a condition (ConditionA1) expressed by the following expression (21).

[0323]

[Math. 17]

```

ConditionA1: FlipFlag==F && TransposeFlag==F
. . . (21)

```

10

[0324]

In a case where it is determined that the above-described condition (ConditionA1) is satisfied (in a case where both the flip flag FlipFlag and the transposition flag TransposeFlag are false (0)), the processing proceeds to step S343.

15

[0325]

In step S343, the transformation matrix derivation unit 321 sets the base transformation matrix T_{base} as the transformation matrix T_H as in the following expression (22).

20

[0326]

[Math. 18]

```

 $T_H = T_{\text{base}}$  . . . (22)

```

25

[0327]

When the processing in step S343 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28. Furthermore, in step S342, in a case where it is determined that the above-described condition (ConditionA1) is not satisfied (the flip flag FlipFlag or the transposition flag TransposeFlag is true (1)), the processing proceeds to step S344.

[0328]

In step S344, the transformation matrix derivation unit 321 determines whether or not the flip flag FlipFlag and the transposition flag TransposeFlag satisfy a condition (ConditionA2) expressed by the following expression (23).

[0329]

15 [Math. 19]

ConditionA2: FlipFlag==F && TransposeFlag==T
 . . . (23)

[0330]

In a case where it is determined that the above-described condition (ConditionA2) is satisfied (in a case where the flip flag FlipFlag is false (0) and the transposition flag TransposeFlag is true (1)), the processing proceeds to step S345.

[0331]

In step S345, the transformation matrix derivation unit 321 transposes the base transformation matrix T_{base} via the transposition unit 333 to obtain the transformation matrix T_H . This processing can be expressed as a determinant as in the following expression (24).

[0332]

[Math. 20]

$$T_H = \text{Tr}(T_{\text{base}}) = T_{\text{base}}^T \cdot \cdot \cdot \quad (24)$$

[0333]

5 Furthermore, in a case of expressing the processing as an operation for each element, the transformation matrix derivation unit 321 sets the i-row j-column component ((i, j) component) of the base transformation matrix T_{base} as an (j, i) component of the transformation matrix T_H , as in the following expression (25).

[0334]

[Math. 21]

$$\begin{aligned} T_H[j, i] &= T_{\text{base}}[i, j] \\ \text{for } i, j &= 0, \dots, N-1 \\ &\cdot \cdot \cdot \quad (25) \end{aligned}$$

[0335]

15 Here, the i-row j-column component ((i, j) component) of the transformation matrix T_H of N rows and N columns is written as $T_H[i, j]$. Furthermore, "for i, j = 0, ..., N - 1" on the second row indicates that i and j have values of 0 to N - 1. That is, it means that $T_H[j, i]$ indicates all of elements of the transformation matrix T_H of N rows and N columns.

[0336]

25 By expressing the processing in step S345 as an operation for each element in this way, the transposition operation can be implemented by accessing a simple two-dimensional array. When the processing in step S345 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0337]

Furthermore, in step S344, in a case where it is determined that the above-described condition (ConditionA2) is not satisfied (the flip flag FlipFlag is true (1) or the transposition flag TransposeFlag is false (0)), the processing proceeds to step S346.

[0338]

In step S346, the transformation matrix derivation unit 321 flips the base transformation matrix T_{base} via the flip unit 332 to obtain the transformation matrix T_H . This processing can be expressed as a determinant as in the following expression (26).

[0339]

[Math. 22]

$$T_H = T_{base} \times J \quad \cdot \cdot \cdot \quad (26)$$

[0340]

Here, $\times$ is an operator representing a matrix product. Furthermore, the flip matrix J (cross-identity matrix) is obtained by right-left inverting the $N \times N$ unit matrix I .

[0341]

Furthermore, in a case of expressing the processing as an operation for each element, the transformation matrix derivation unit 321 sets a $(i, N - 1 - j)$ component of the base transformation matrix T_{base} as the i -row j -column component $((i, j)$ component) of the transformation matrix T_H , as in the following expression (27).

[0342]

[Math. 23]

$$T_H[i, j] = T_{base}[i, N-1-j]$$

for $i, j=0, \dots, N-1$
 $\cdot \cdot \cdot (27)$

[0343]

Here, the i -row j -column component ((i, j) component) of the transformation matrix T_H of N rows and N columns is written as $T_H[i, j]$. Furthermore, "for $i, j = 0, \dots, N - 1$ " on the second row indicates that i and j have values of 0 to $N - 1$. That is, it means that $T_H[i, j]$ indicates all of elements of the transformation matrix T_H of N rows and N columns.

10 [0344]

By expressing the processing in step S346 as an operation for each element in this way, the transposition operation can be implemented by accessing a simple two-dimensional array without a matrix calculation of the base transformation matrix T_{base} and the flip matrix J . Furthermore, the flip matrix J becomes unnecessary. When the processing in step S346 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

20 [0345]

In the above transformation matrix derivation processing, processing to which the above-described present technology is applied is performed as processing of step S345 or S346. Therefore, by executing the transformation matrix derivation processing, the derivation example of the first and second row examples from the top of the table in Fig. 10 can be implemented in the horizontal one-dimensional orthogonal transform. Therefore, an increase in the required memory capacity

25

can be suppressed.

[0346]

Note that a branch described below may be inserted between the processing in step S344 and the processing in
 5 step S346. That is, in the step, the transformation matrix derivation unit 321 determines whether or not the flip flag FlipFlag and the transposition flag TransposeFlag satisfy a condition (ConditionA3) expressed by the following expression (28).

10 [0347]

[Math. 24]

ConditionA3 : FlipFlag==T && TransposeFlag==F
 . . . (28)

[0348]

In a case where the transformation matrix
 15 derivation unit 321 determines that the above-described condition (ConditionA3) is satisfied (in a case where the flip flag FlipFlag is true (1) and the transposition flag TransposeFlag is false (0)), the processing proceeds to step S346.

20 [0349]

Furthermore, in a case where it is determined that the above-described condition (ConditionA3) is not satisfied (the flip flag FlipFlag is false (0) or the transposition flag TransposeFlag is true (1)), the
 25 transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0350]

<Flow of Primary Vertical Transform Processing>

Next, a flow of the primary vertical transform
 30 processing executed in step S303 in Fig. 27 will be

described with reference to the flowchart in Fig. 32.

[0351]

When the primary vertical transform processing is started, in step S361, the transformation matrix derivation unit 351 (Fig. 25) of the primary vertical transform unit 313 executes the transformation matrix derivation processing to derive the transformation matrix T_v corresponding to the transform type identifier $TrTypeIdxV$ (or the transform type $TrTypeV$) of primary vertical transform.

[0352]

Since the flow of the transformation matrix derivation processing is similar to the case of primary horizontal transform described with reference to the flowchart in Fig. 30, and description thereof is omitted. For example, the description regarding the horizontal direction, of the description made by reference to Fig. 30, is simply applied to the vertical direction, such as by replacing the transform type identifier $TrTypeIdxH$ of primary horizontal transform with the transform type identifier $TrTypeIdxV$ of primary vertical transform, and replacing the transformation matrix T_H for primary horizontal transform with the transformation matrix T_v for primary vertical transform.

[0353]

In step S362, the matrix calculation unit 352 performs the vertical one-dimensional orthogonal transform for the input data X_{in} (the transform coefficient $Coeff_Phor$ after primary horizontal transform), using the derived transformation matrix T_v , to obtain the intermediate data $Y1$. When this processing

is expressed as a determinant, the processing can be expressed as the above-described expression (12).

Furthermore, when this processing is expressed as an operation for each element, the processing can be

5 expressed as the following expression (29).

[0354]

[Math. 25]

$$Y1[i, j] = T_v[i, :] \times X_m[:, j] = \sum_{k=0}^{N-1} T_v[i, k] X_m[k, j] \quad \cdot \cdot \cdot \quad (29)$$

[0355]

10 That is, in this case, as illustrated in Fig. 33, an inner product of an i-th row vector $T_v[i, :]$ of the transformation matrix T_v and a j-th column vector $X_{in}[:, j]$ of the input data X_{in} as the coefficient $Y1[i, j]$ of the i-row j-column component of the intermediate data $Y1$
15 ($j = 0, \dots, M - 1$, and $i = 0, \dots, N - 1$).

[0356]

In step S363, the scaling unit 353 scales, with the shift amount S_v , the coefficient $Y1[i, j]$ of each i-row j-column component of the intermediate data $Y1$ derived by
20 the processing in step S322 to derive the intermediate data $Y2$. This scaling can be expressed as the above-described expression (13).

[0357]

In step S364, the clip unit 354 clips the value of
25 the coefficient $Y2[i, j]$ of each i-row j-column component of the intermediate data $Y2$ derived by the processing in step S363, and obtains output data X_{out} (that is, the transform coefficient $Coeff_Pver$ after primary vertical transform). This processing can be

expressed as the above-described expression (14).

[0358]

When the processing in step S364 ends, the primary horizontal transform processing ends, and the processing
5 returns to Fig. 27.

[0359]

In the above primary vertical transform processing, processing to which the above-described present technology is applied is performed as processing of step
10 S361 or S362. Therefore, by executing the primary vertical transform processing, an increase in the memory capacity required for the vertical one-dimensional orthogonal transform can be suppressed.

[0360]

15 <Inverse Primary Transform Unit>

Next, a configuration of the image decoding device 200 in the case of the present embodiment will be described. Fig. 34 is a block diagram illustrating a main configuration example of the inverse primary
20 transform unit 253 (Fig. 16) in this case. As illustrated in Fig. 34, the inverse primary transform unit 253 includes an inverse primary transform selection unit 411, an inverse primary vertical transform unit 412, and an inverse primary horizontal transform unit 413.

25 [0361]

The inverse primary transform selection unit 411 uses the prediction mode information PInfo, the component identifier compID, the adaptive primary transform flag apt_flag[compID], and the primary transform identifier
30 pt_idx[compID] as inputs. The inverse primary transform selection unit 411 derives the transform type identifier

TrTypeIdxV of inverse primary vertical transform and the transform type identifier TrTypeIdxH of inverse primary vertical transform by reference to the above information. The inverse primary transform selection unit 411 supplies
 5 the derived transform type identifier TrTypeIdxV of inverse primary transform to the inverse primary vertical transform unit 412. Furthermore, the inverse primary transform selection unit 411 supplies the derived transform type identifier TrTypeIdxH of inverse primary
 10 horizontal transform to the inverse primary horizontal transform unit 413.

[0362]

The inverse primary vertical transform unit 412 uses the transform coefficient Coeff_IS after inverse
 15 secondary transform, the transform type identifier TrTypeIdxV after inverse primary vertical transform, and information regarding the size of the transformation block as inputs. The information regarding the size of the transformation block may be a natural number N
 20 indicating the size of the transformation block in the horizontal direction or the vertical direction (the number of coefficients), or may be $\log_2 \text{TBHSize}$ (the logarithmic value of the height) indicating the height of the transformation block ($N = 1 \ll \log_2 \text{TBHSize}$). The
 25 inverse primary vertical transform unit 412 executes, for the transform coefficient Coeff_IS after inverse secondary transform, inverse primary vertical transform IPver determined by the transform type identifier TrTypeIdxV and the size of the transformation block to
 30 derive a transform coefficient Coeff_IPver after inverse primary vertical transform. The inverse primary vertical

transform unit 412 supplies the transform coefficient
Coeff_IPver after inverse primary vertical transform to
the inverse primary horizontal transform unit 413.

[0363]

5 The inverse primary horizontal transform unit 413
uses the transform coefficient Coeff_IPver after inverse
primary vertical transform, the transform type identifier
TrTypeIdxH of inverse primary horizontal transform, and
information regarding the size of the transformation
10 block as inputs. The information regarding the size of
the transformation block may be a natural number N
indicating the size of the transformation block in the
horizontal direction or the vertical direction (the
number of coefficients), or may be log2TBWSize (the
15 logarithmic value of the width) indicating the width of
the transformation block ($N = 1 \ll \log_2 \text{TBWSize}$). The
inverse primary horizontal transform unit 413 executes,
for the transform coefficient Coeff_IPver after inverse
primary vertical transform supplied from the inverse
20 primary vertical transform unit 412, inverse primary
horizontal transform IPhor determined by the transform
type identifier TrTypeIdxH and the size of the
transformation block to derive the transform coefficient
Coeff_IPhor after inverse primary horizontal transform
25 (that is, the transform coefficient Coeff_IP after
inverse primary transform). The inverse primary
horizontal transform unit 413 outputs the transform
coefficient Coeff_IPhor after inverse primary horizontal
transform to the outside of the inverse primary transform
30 unit 253 (supplies the same to the calculation unit 215)
as the prediction residual D' .

[0364]

In the inverse primary transform unit 253 having the above configuration, the inverse primary vertical transform unit 412 and the inverse primary horizontal transform unit 413 perform processing to which the above-described present technology is applied, as a derivation unit and an orthogonal transform unit, respectively.

[0365]

That is, the inverse primary vertical transform unit 412 derives, as a derivation unit, the second transformation matrix for vertical inverse one-dimensional orthogonal transform, and further performs, as an inverse orthogonal transform unit, the vertical inverse one-dimensional orthogonal transform, using the second transformation matrix for vertical inverse one-dimensional orthogonal transform derived by the derivation unit. Therefore, the inverse primary vertical transform unit 412 can suppress an increase in the memory capacity required for the vertical inverse one-dimensional orthogonal transform.

[0366]

Furthermore, the inverse primary horizontal transform unit 413 derives, as a derivation unit, the second transformation matrix for horizontal inverse one-dimensional orthogonal transform, and further performs, as an orthogonal transform unit, horizontal inverse one-dimensional orthogonal transform, using the second transformation matrix for horizontal inverse one-dimensional orthogonal transform derived by the derivation unit. Therefore, the inverse primary horizontal transform unit 413 can suppress an increase in

the memory capacity required for the horizontal inverse one-dimensional orthogonal transform.

[0367]

<<Inverse Primary Vertical Transform Unit>>

5 Fig. 35 is a block diagram illustrating a main configuration example of the inverse primary vertical transform unit 412 in Fig. 34. As illustrated in Fig. 35, the inverse primary vertical transform unit 412 includes a transformation matrix derivation unit 421, a matrix
10 calculation unit 422, a scaling unit 423, and a clip unit 424.

[0368]

The transformation matrix derivation unit 421 uses the transform type identifier $TrTypeIdxV$ of inverse
15 primary vertical transform and the information regarding the size of the transformation block as inputs, and derives a transformation matrix T_v for inverse primary vertical transform (a transformation matrix T_v for vertical inverse one-dimensional orthogonal transform)
20 having the same size as the transformation block, the transformation matrix T_v corresponding to the transform type identifier $TrTypeIdxV$ of inverse primary vertical transform. The transformation matrix derivation unit 421 supplies the transformation matrix T_v to the matrix
25 calculation unit 422.

[0369]

The matrix calculation unit 422 performs the vertical inverse one-dimensional orthogonal transform for the input data X_{in} (that is, the transformation block of
30 the transform coefficient $Coeff_IS$ after inverse secondary transform), using the transformation matrix T_v

supplied from the transformation matrix derivation unit 421, to obtain intermediate data Y1. This calculation can be expressed by a determinant as in the following expression (30).

5 [0370]

[Math. 26]

$$Y1 = T_V^T \times X_{in} \quad \cdot \cdot \cdot \quad (30)$$

[0371]

The matrix calculation unit 422 supplies the
10 intermediate data Y1 to the scaling unit 423.

[0372]

The scaling unit 423 scales a coefficient Y1 [i, j] of each i-row j-column component of the intermediate data Y1 with a predetermined shift amount S_{IV} to obtain
15 intermediate data Y2. This scaling can be expressed as the following expression (31).

[0373]

[Math. 27]

$$Y2[i, j] = Y1[i, j] \gg S_{IV} \quad \cdot \cdot \cdot \quad (31)$$

20 [0374]

The scaling unit 423 supplies the intermediate data Y2 to the clip unit 424.

[0375]

The clip unit 424 clips a value of a coefficient Y2
25 [i, j] of each i-row j-column component of the intermediate data Y2, and derives output data X_{out} (that is, the transform coefficient Coeff_IPver after inverse primary vertical transform). This processing can be expressed as the above-described expression (11).

[0376]

The clip unit 424 outputs the output data X_{out} (the transform coefficient $Coeff_IPver$ after inverse primary vertical transform) to the outside of the inverse primary vertical transform unit 412 (supplies the same to the inverse primary horizontal transform unit 413).

[0377]

In the inverse primary vertical transform unit 412 having the above configuration, the transformation matrix derivation unit 421 performs processing to which the above-described present technology is applied, as a derivation unit. Furthermore, the matrix calculation unit 422 performs processing to which the above-described present technology is applied, as an orthogonal transform unit. Therefore, the inverse primary vertical transform unit 412 can suppress an increase in the memory capacity required for the vertical inverse one-dimensional orthogonal transform.

[0378]

20 <Transformation Matrix Derivation Unit>

Fig. 36 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 421 in Fig. 35. As illustrated in Fig. 36, the transformation matrix derivation unit 421 includes a transformation matrix LUT 431, a flip unit 432, and a transposition unit 433. Note that, in Fig. 36, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 421, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0379]

The transformation matrix LUT 431 is a lookup table for holding (storing) a transformation matrix corresponding to the transform type identifier TrTypeIdxV of inverse primary vertical transform and the size N of the transformation block. When the transform type identifier TrTypeIdxV of inverse primary vertical transform and the size N of the transformation block are specified, the transformation matrix LUT 431 selects and outputs a transformation matrix corresponding thereto.

In the case of this derivation example, the transformation matrix LUT 431 supplies the transformation matrix to both or one of the flip unit 432 and the transposition unit 433 as the base transformation matrix T_{base} .

[0380]

The flip unit 432 flips an input transformation matrix T of N rows and N columns, and outputs a flipped transformation matrix T_{flip} . In the case of this derivation example, the flip unit 432 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 431 as an input, flips the base transformation matrix T_{base} in the row direction (horizontal direction), and outputs the flipped transformation matrix T_{flip} to the outside of the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 422) as the transformation matrix T_v .

[0381]

The transposition unit 433 transposes the input transformation matrix T of N rows and N columns, and outputs a transposed transformation matrix $T_{\text{transpose}}$. In

the case of this derivation example, the transposition unit 433 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 431 as an input, transposes the base transformation matrix T_{base} , and outputs the transposed transformation matrix $T_{transpose}$ to the outside of the transformation matrix derivation unit 421 (supplies the same to the matrix calculation unit 422) as the transformation matrix T_v .

10 [0382]

As described above, the transformation matrix derivation unit 421 includes the flip unit 432 and the transposition unit 433. Therefore, the transformation matrix derivation unit 421 can implement the derivation example of the first row example from the top of the table illustrated in Fig. 10, using the flip unit 432, for example. Furthermore, the transformation matrix derivation unit 421 can implement the derivation example of the second row example from the top of the table illustrated in Fig. 10, using the transposition unit 433, for example.

[0383]

<Inverse Primary Horizontal Transform Unit>

Fig. 37 is a block diagram illustrating a main configuration example of the inverse primary horizontal transform unit 413 in Fig. 34. As illustrated in Fig. 37, the inverse primary horizontal transform unit 413 includes a transformation matrix derivation unit 451, a matrix calculation unit 452, a scaling unit 453, and a clip unit 454.

[0384]

The transformation matrix derivation unit 451 uses the transform type identifier TrTypeIdxH of inverse primary horizontal transform and the information regarding the size of the transformation block as inputs,
 5 and derives a transformation matrix T_H for inverse primary horizontal transform (a transformation matrix T_H for horizontal inverse one-dimensional orthogonal transform) having the same size as the transformation block, the transformation matrix T_H corresponding to the
 10 transform type identifier TrTypeIdxH of inverse primary horizontal transform. The transformation matrix derivation unit 451 supplies the transformation matrix T_H to the matrix calculation unit 452.

[0385]

15 The matrix calculation unit 452 performs the horizontal inverse one-dimensional orthogonal transform for the input data X_{in} (that is, the transformation block of the transform coefficient Coeff_IPver after inverse primary vertical transform), using the transformation
 20 matrix T_H supplied from the transformation matrix derivation unit 451, to obtain intermediate data Y1. This calculation can be expressed by a determinant as in the following expression (32).

[0386]

25 [Math. 28]

$$Y1 = X_{in} \times T_H \quad \cdot \cdot \cdot \quad (32)$$

[0387]

The matrix calculation unit 452 supplies the intermediate data Y1 to the scaling unit 453.

30 [0388]

The scaling unit 453 scales a coefficient $Y1[i, j]$ of each i -row j -column component of the intermediate data $Y1$ with a predetermined shift amount S_{IH} to obtain intermediate data $Y2$. This scaling can be expressed as
 5 the following expression (33).

[0389]

[Math. 29]

$$Y2[i, j] = Y1[i, j] \gg S_{IH} \quad \cdot \cdot \cdot \quad (33)$$

[0390]

10 The scaling unit 453 supplies the intermediate data $Y2$ to the clip unit 454.

[0391]

The clip unit 454 clips a value of a coefficient $Y2[i, j]$ of each i -row j -column component of the
 15 intermediate data $Y2$, and derives output data X_{out} (that is, the transform coefficient $Coeff_IPhor$ after inverse primary horizontal transform). This processing can be expressed as the above-described expression (11).

[0392]

20 The clip unit 454 outputs the output data X_{out} (the transform coefficient $Coeff_IPhor$ after inverse primary horizontal transform (transform coefficient $Coeff_IP$ after inverse primary transform)) to the outside of the inverse primary horizontal transform unit 413 (supplies
 25 the same to the calculation unit 215) as a prediction residual D' .

[0393]

In the inverse primary horizontal transform unit 413 having the above configuration, the transformation
 30 matrix derivation unit 451 performs processing to which

the above-described present technology is applied, as a derivation unit. Furthermore, the matrix calculation unit 452 performs processing to which the above-described present technology is applied, as an inverse orthogonal transform unit. Therefore, the inverse primary horizontal transform unit 413 can suppress an increase in the memory capacity required for the horizontal inverse one-dimensional orthogonal transform.

[0394]

10 <Transformation Matrix Derivation Unit>

Fig. 38 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 451 in Fig. 37. As illustrated in Fig. 38, the transformation matrix derivation unit 451

15 includes a transformation matrix LUT 461, a flip unit 462, and a transposition unit 463. Note that, in Fig. 38, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 451, arbitrary data can be transferred between arbitrary processing units (processing blocks).

20 [0395]

The transformation matrix LUT 461 is a lookup table for holding (storing) a transformation matrix corresponding to the transform type identifier TrTypeIdxH of inverse primary horizontal transform and the size N of the transformation block. When the transform type identifier TrTypeIdxH of inverse primary horizontal transform and the size N of the transformation block are specified, the transformation matrix LUT 461 selects and outputs a transformation matrix corresponding thereto. In the case of this derivation example, the

transformation matrix LUT 461 supplies the transformation matrix to both or one of the flip unit 462 and the transposition unit 463 as the base transformation matrix T_{base} .

5 [0396]

The flip unit 462 flips an input transformation matrix T of N rows and N columns, and outputs a flipped transformation matrix T_{flip} . In the case of this derivation example, the flip unit 462 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 461 as an input, flips the base transformation matrix T_{base} in the row direction (horizontal direction), and outputs the flipped transformation matrix T_{flip} to the outside of the transformation matrix derivation unit 451 (supplies the same to the matrix calculation unit 452) as the transformation matrix T_H .

[0397]

The transposition unit 463 transposes the input transformation matrix T of N rows and N columns, and outputs a transposed transformation matrix $T_{\text{transpose}}$. In the case of this derivation example, the transposition unit 463 uses the base transformation matrix T_{base} of N rows and N columns supplied from the transformation matrix LUT 461 as an input, transposes the base transformation matrix T_{base} , and outputs the transposed transformation matrix $T_{\text{transpose}}$ to the outside of the transformation matrix derivation unit 451 (supplies the same to the matrix calculation unit 452) as the transformation matrix T_H .

[0398]

As described above, the transformation matrix derivation unit 451 includes the flip unit 462 and the transposition unit 463. Therefore, the transformation matrix derivation unit 451 can implement the derivation example of the first row example from the top of the table illustrated in Fig. 10, using the flip unit 462, for example. Furthermore, the transformation matrix derivation unit 451 can implement the derivation example of the second row example from the top of the table illustrated in Fig. 10, using the transposition unit 463, for example.

[0399]

<Flow of Inverse Primary Transform Processing>

Next, an example of a flow of processing performed by the above-described configuration of the image decoding device 200 will be described. Next, an example of a flow of inverse primary transform processing executed in step S233 in Fig. 18 in this case will be described with reference to the flowchart in Fig. 39.

[0400]

When the inverse primary transform processing is started, in step S401, the inverse primary transform selection unit 411 (Fig. 34) of the inverse primary transform unit 253 performs the inverse primary transform selection processing to select the transform type identifier TrTypeIdxV (or the transform type TrTypeV) of inverse primary vertical transform and the transform type identifier TrTypeIdxH (or the transform type TrTypeH) of inverse primary horizontal transform.

[0401]

In step S402, the inverse primary vertical

transform unit 412 performs, for the transform coefficient `Coeff_IS` after inverse secondary transform, the inverse primary vertical transform processing corresponding to the transform type identifier `TrTypeIdxV` of inverse primary vertical transform obtained in step S401 to derive the transform coefficient `Coeff_IPver` after inverse primary vertical transform.

[0402]

In step S403, the inverse primary horizontal transform unit 413 performs, for the transform coefficient `Coeff_IPver` after inverse primary vertical transform, the inverse primary horizontal transform processing corresponding to the transform type identifier `TrTypeIdxH` of inverse primary horizontal transform obtained in step S401 to derive the transform coefficient `Coeff_IPhor` after inverse primary horizontal transform (that is, transform coefficient `Coeff_IP` after inverse primary transform (prediction residual D')).

[0403]

When the processing in step S403 ends, the inverse primary transform processing ends and the processing returns to Fig. 18.

[0404]

In the above inverse primary transform processing, processing to which the above-described present technology is applied is performed as processing of step S402 or S403. Therefore, by executing the inverse primary transform processing, an increase in the memory capacity required for the inverse primary vertical transform processing or the inverse primary horizontal transform processing can be suppressed.

[0405]

<Flow of Inverse Primary Transform Selection
Processing>

Next, an example of a flow of the inverse primary
5 transform selection processing executed in step S401 in
Fig. 39 will be described with reference to the flowchart
in Fig. 40.

[0406]

When the inverse primary transform selection
10 processing is started, in step S421, the inverse primary
transform selection unit 411 determines whether or not
the adaptive primary transform flag `apt_flag[compID]` of
the component identifier `compID` is true (1). In a case
where the adaptive primary transform flag
15 `apt_flag[compID]` of the component identifier `compID` is
determined to be true (1), the processing proceeds to
step S422.

[0407]

In step S422, the inverse primary transform
20 selection unit 411 selects transform a set `TrSetH` or
`TrSetV` in each direction from a transform set group on
the basis of the prediction mode information `Pinfo`.

[0408]

In step S423, the inverse primary transform
25 selection unit 411 derives the transform type identifier
`TrTypeIdxH` of inverse primary horizontal transform on the
basis of the transform set `TrSetH` and the primary
transform identifier `pt_idx[compID]` This processing can
be expressed as, for example, the following expression
30 (34).

[0409]

[Math. 30]

$$\text{TrTypeIdxH} = \text{TrSetToTrTypeIdx}[\text{TrSetH}][\text{pt_idx}] \quad \dots \quad (34)$$

[0410]

In step S424, the inverse primary transform
 5 selection unit 411 derives the transform type identifier
 TrTypeIdxV of inverse primary vertical transform on the
 basis of the transform set TrSetV and the inverse primary
 transform identifier $\text{pt_idx}[\text{compID}]$. This processing can
 be expressed as, for example, the following expression

10 (35).

[0411]

[Math. 31]

$$\text{TrTypeIdxV} = \text{TrSetToTrTypeIdx}[\text{TrSetV}][\text{pt_idx}] \quad \dots \quad (35)$$

[0412]

15 When the processing in step S424 ends, the inverse
 primary transform selection processing ends and the
 processing returns to Fig. 39.

[0413]

Furthermore, in step S421, in a case where the
 20 adaptive primary transform flag $\text{apt_flag}[\text{compID}]$ of the
 component identifier compID is determined to be false (0),
 the processing proceeds to step S425.

[0414]

In step S425, the inverse primary transform
 25 selection unit 411 sets (selects) predetermined
 orthogonal transform (for example, DCT2) for the
 transform type identifier TrTypeIdxH of inverse primary
 horizontal transform. This processing can be expressed
 as, for example, the following expression (36).

[0415]

[Math. 32]

TrTypeIdxH = Predetermined value . . . (36)

[0416]

5 In step S426, the inverse primary transform selection unit 411 sets (selects) predetermined orthogonal transform (for example, DCT2) for the transform type identifier TrTypeIdxV of inverse primary vertical transform. This processing can be expressed as,
10 for example, the following expression (37).

[0417]

[Math. 33]

TrTypeIdxV = Predetermined value . . . (37)

[0418]

15 When the processing in step S426 ends, the inverse primary transform selection processing ends and the processing returns to Fig. 39.

[0419]

 <Flow of Inverse Primary Vertical Transform
20 Processing>

 Next, a flow of the inverse primary vertical transform processing executed in step S402 in Fig. 39 will be described with reference to the flowchart in Fig. 41.

25 [0420]

 When the inverse primary vertical transform processing is started, in step S441, the transformation matrix derivation unit 421 (Fig. 35) of inverse primary vertical transform unit 412 executes the transformation

matrix derivation processing to derive the transformation matrix T_v corresponding to the transform type identifier $TrTypeIdxV$ of inverse primary vertical transform.

[0421]

5 The transformation matrix derivation processing in this case is performed by a flow similar to the case of the primary horizontal transform described with reference to the flowchart in Fig. 30. Therefore, the description is omitted. For example, the description made with
10 reference to Fig. 30 can be applied as description of the transformation matrix derivation processing of this case, by replacing the transform type identifier $TrTypeIdxH$ of primary horizontal transform with the transform type identifier $TrTypeIdxV$ of inverse primary vertical
15 transform, and replacing the transformation matrix T_H for primary horizontal transform to be derived with the transformation matrix T_v for inverse primary vertical transform.

[0422]

20 In step S442, the matrix calculation unit 422 performs the vertical inverse one-dimensional orthogonal transform for the input data X_{in} (that is, the transform coefficient $Coeff_IS$ after inverse secondary transform), using the derived transformation matrix T_v , to obtain the
25 intermediate data $Y1$. When this processing is expressed as a determinant, the processing can be expressed as the above-described expression (30).

[0423]

30 In step S443, the scaling unit 423 scales, with the shift amount S_{IV} , the coefficient $Y1[i, j]$ of each i -row j -column component of the intermediate data $Y1$ derived by

the processing in step S442 to derive the intermediate data Y2. This scaling can be expressed as the above-described expression (31).

[0424]

5 In step S444, the clip unit 424 clips the value of the coefficient Y2 [i, j] of each i-row j-column component of the intermediate data Y2 derived by the processing in step S443, and obtains output data X_{out} (that is, the transform coefficient Coeff_IPver after
10 inverse primary vertical transform). This processing can be expressed as the above-described expression (11).

[0425]

When the processing in step S444 ends, the inverse primary vertical transform processing ends and the
15 processing returns to Fig. 39.

[0426]

In the above inverse primary vertical transform processing, processing to which the above-described present technology is applied is performed as processing
20 of step S441 or S442. Therefore, by executing the inverse primary vertical transform processing, an increase in the memory capacity required for the vertical inverse one-dimensional orthogonal transform can be suppressed.

25 [0427]

<Flow of Inverse Primary Horizontal Transform Processing>

Next, a flow of the inverse primary horizontal transform processing executed in step S403 in Fig. 39
30 will be described with reference to the flowchart in Fig. 42.

[0428]

When the inverse primary horizontal transform processing is started, in step S461, the transformation matrix derivation unit 451 (Fig. 37) of inverse primary horizontal transform unit 413 executes the transformation matrix derivation processing to derive the transformation matrix T_H corresponding to the transform type identifier $TrTypeIdxH$ of inverse primary horizontal transform.

[0429]

10 The transformation matrix derivation processing in this case is performed by a flow similar to the case of the primary horizontal transform described with reference to the flowchart in Fig. 30. Therefore, the description is omitted. For example, the description made by
15 reference to Fig. 30 can be applied as description of the transformation matrix derivation processing of this case, by replacing the primary horizontal transform with the inverse primary horizontal transform, or the like.

[0430]

20 In step S462, the matrix calculation unit 452 performs the horizontal inverse one-dimensional orthogonal transform for the input data X_{in} (that is, the transform coefficient $Coeff_IPver$ after inverse primary vertical transform), using the derived transformation
25 matrix T_H , to obtain the intermediate data $Y1$. When this processing is expressed as a determinant, the processing can be expressed as the above-described expression (32).

[0431]

In step S463, the scaling unit 453 scales, with the
30 shift amount S_{IH} , the coefficient $Y1[i, j]$ of each i -row j -column component of the intermediate data $Y1$ derived by

the processing in step S462 to derive the intermediate data Y2. This scaling can be expressed as the above-described expression (33).

[0432]

5 In step S464, the clip unit 454 clips the value of the coefficient Y2 [i, j] of each i-row j-column component of the intermediate data Y2 derived by the processing in step S463, and outputs output data X_{out} (that is, the transform coefficient Coeff_IPhor after
10 inverse primary horizontal transform). This processing can be expressed as the above-described expression (14).

[0433]

When the processing in step S464 ends, the inverse primary horizontal transform processing ends and the
15 processing returns to Fig. 39.

[0434]

In the above inverse primary horizontal transform processing, processing to which the above-described present technology is applied is performed as processing
20 of step S461 or S462. Therefore, by executing the inverse primary horizontal transform processing, an increase in the memory capacity required for the horizontal inverse one-dimensional orthogonal transform can be suppressed.

25 [0435]

<2-3. Example 1-2>

<Concept>

Next, the third row example from the top except the uppermost row of item names in the table illustrated in
30 Fig. 10 will be described.

[0436]

As described above, the derivation of the third row example from the top focuses on the characteristics between paired DCT and DST. More specifically, between DCT/DST to be paired (for example, DST7 and DCT8),
 5 attention is paid to the point that even-numbered row vectors are axially symmetric and odd-numbered row vectors are point-symmetric. In this case, the derivation unit flips the first transformation matrix and inverts the sign of the odd-numbered row vector of the
 10 flipped first transformation matrix to derive the second transformation matrix. That is, the derivation unit flips the transformation matrix of DST7 in the row direction and further inverts the sign of an odd-order row vector to losslessly derive the transformation matrix
 15 of DCT8, as illustrated in Fig. 43. Therefore, naturally, the (decreasing-type) transformation matrix of DCT8 having the same waveform of the 0-order row vector can be substituted by the derived DCT8.

[0437]

20 A specific example of this derivation is illustrated in Fig. 44. As illustrated in Fig. 44, this derivation can be expressed by a matrix product of a unit matrix D having a negative element in an odd-order row, the base transformation matrix T_{base} (DST7), and the flip
 25 matrix J .

[0438]

That is, in this derivation example, the second transformation matrix can be derived by two-time operation (flip and sign inversion). Furthermore, each
 30 operation is easy. That is, the second transformation matrix can be easily derived.

[0439]

Furthermore, by applying this derivation example, it becomes unnecessary to prepare the transformation matrix of DCT8 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced.

[0440]

In this case, as illustrated in the table in Fig. 45, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to four types. Therefore, the total LUT size can be about 47 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.

[0441]

Note that, as described above, in this case, the transformation matrix of the transform type DCT8 can be obtained as the second transform type. Therefore, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the second transform type, the same coding efficiency as the case of using the transformation matrix of DCT8 for the orthogonal transform/inverse orthogonal transform can be obtained.

[0442]

<Transformation Matrix Derivation Unit>

Next, configurations, processing, and the like for

performing such derivation will be described. First, a technical configuration of the present embodiment of the image encoding device 100 will be described. Since configurations of the primary transform unit 152, the primary horizontal transform unit 312, the primary vertical transform unit 313, and the like in this case are similar to the case of <2-2. Example 1-1> described with reference to Fig. 22, description thereof is omitted. [0443]

10 Fig. 46 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 321 (the transformation matrix derivation unit 321 (Fig. 23) in the primary horizontal transform unit 312) in this case. As illustrated in Fig. 46, the transformation matrix derivation unit 321 in this case includes a transformation matrix LUT 331, a flip unit 332, and a sign inversion unit 501. Note that, in Fig. 46, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 321, arbitrary data can be transferred between arbitrary processing units (processing blocks). [0444]

 The transformation matrix LUT 331 and the flip unit 332 are similar to the case in Fig. 24.

25 [0445]

 The sign inversion unit 501 uses the transformation matrix T of N rows and N columns as an input, inverts the sign of a predetermined portion of the transformation matrix T , and outputs a transformation matrix T_{InvSign} after sign inversion. In the case of this derivation example (Example 1-2), the transformation matrix

derivation unit 321 horizontally flips the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 331 via the flip unit 332, further inverts the sign of an odd-order row vector via the sign inversion unit 501, and outputs the transformation matrix T_{InvSign} to the outside of the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

10 [0446]

As described above, the transformation matrix derivation unit 321 can implement the derivation example of the third row example from the top of the table illustrated in Fig. 10, using the flip unit 332 and the sign inversion unit 501.

15

[0447]

<Transformation Matrix Derivation Unit>

Fig. 47 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 351 included in the primary vertical transform unit 313 in this case. As illustrated in Fig. 47, the transformation matrix derivation unit 351 in this case includes the transformation matrix LUT 361, the flip unit 362, and a sign inversion unit 502, similarly to the transformation matrix derivation unit 321. Note that, in Fig. 47, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 351, arbitrary data can be transferred between arbitrary processing units (processing blocks).

25
30 [0448]

The transformation matrix LUT 361 and the flip unit

362 are similar to the case in Fig. 26.

[0449]

The sign inversion unit 502 uses a transformation matrix T of N rows and N columns as an input, inverts the sign of a predetermined portion of the transformation matrix T , and performs sign inversion, and outputs a transformation matrix T_{InvSign} after the sign inversion, similarly to the sign inversion unit 501. In the case of this derivation example, the transformation matrix derivation unit 351 horizontally flips the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 361 via the flip unit 362, further inverts the sign of an odd-order row vector via the sign inversion unit 502, and outputs the transformation matrix T_{InvSign} to the outside of the transformation matrix derivation unit 351 (supplies the same to the matrix calculation unit 352) as the transformation matrix T_v .

[0450]

As described above, the transformation matrix derivation unit 351 can implement the derivation example of the third row example from the top of the table illustrated in Fig. 10, using the flip unit 362 and the sign inversion unit 502.

[0451]

<Flow of Transformation Matrix Derivation Processing>

Since the primary transform processing, the primary horizontal transform processing, and the primary vertical transform processing in this case are performed by flows similar to the case of <2-2. Example 1-1>, description of

the processing is omitted.

[0452]

Next, an example of a flow of the transformation matrix derivation processing in this case, which is executed by the transformation matrix derivation unit 321 in the primary horizontal transform unit 312 in step S321 in Fig. 28, will be described with reference to the flowchart in Fig. 48.

[0453]

10 When the transformation matrix derivation processing is started, in step S501, the transformation matrix derivation unit 321 obtains a base transform type BaseTrType corresponding to the transform type identifier TrTypeIdxH by reference to the correspondence table
15 illustrated in Fig. 49, for example. Note that, when this processing is expressed as a mathematical expression, the processing can be expressed as the expression (17), for example. Moreover, the transformation matrix derivation unit 321 reads the transformation matrix of N
20 rows and N columns of the obtained base transform type from the transformation matrix LUT, and sets the transformation matrix as the base transformation matrix T_{base} , as in the following expression (18).

[0454]

25 Furthermore, the transformation matrix derivation unit 321 sets a value corresponding to the transform type identifier TrTypeIdxH as the flip flag FlipFlag, as in the above-described expression (19). Furthermore, the transformation matrix derivation unit 321 sets a value
30 corresponding to the transform type identifier TrTypeIdxH as a sign inversion flag InvSignFlag, as in the above-

described expression (38).

[0455]

[Math. 34]

```
InvSignFlag = LUT_TrTypeIdxToInvSignFlag[TrTypeIdxH]
. . . (38)
```

5 [0456]

In step S502, the transformation matrix derivation unit 321 determines whether or not the flip flag FlipFlag and the transposition flag TransposeFlag satisfy a condition (ConditionB1) expressed by the following

10 expression (39).

[0457]

[Math. 35]

```
ConditionB1: FlipFlag==T && InvSignFlag==T
. . . (39)
```

[0458]

15 In a case where it is determined that the condition illustrated in the above expression (39) is not satisfied (that is, in a case where the flip flag FlipFlag is determined to be false (0) or the sign inversion flag InvSignFlag is determined to be false (0)), the

20 processing proceeds to step S503.

[0459]

In step S503, the transformation matrix derivation unit 321 sets the base transformation matrix T_{base} as the transformation matrix T_H . When the processing in step

25 S503 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0460]

Further, in step S502, in a case where it is

determined that the condition illustrated in the above expression (39) is satisfied (that is, in a case where the flip flag FlipFlag is determined to be true (1) or the sign inversion flag InvSignFlag is determined to be true (1)), the processing proceeds to step S504.
5 [0461]

In step S504, the transformation matrix derivation unit 321 flips the base transformation matrix T_{base} via the flip unit 332, and sets the flipped base
10 transformation matrix T_{base} as the transformation matrix T_H . That is, the transformation matrix derivation unit 321 derives the transformation matrix T_H of Flip (T_{base}).
[0462]

In step S505, the transformation matrix derivation
15 unit 321 inverts the sign of the odd-numbered row vector of the transformation matrix T_H , using the sign inversion unit 501, and resets the transformation matrix T_H . This processing can be expressed as a determinant as in the following expression (40).

20 [0463]

[Math. 36]

$$T_H = D \times T_H \quad \dots \quad (40)$$

[0464]

Here, $\times$ is an operator representing a matrix
25 product, and a sign inversion matrix D is a diagonal matrix including Diag (1, -1, ..., $(-1)^{N-1}$).

[0465]

Furthermore, the processing can also be expressed as an operation for each element, as in the following
30 expression (41). With the expression, the processing can be implemented without the sign inversion matrix D . In

that case, the transformation matrix derivation unit 321 inverts positive/negative signs of the component of an odd-numbered row ($i\%2 == 1$) of the i -row j -column component ((i, j) component) of the transformation matrix

5 T_H .
 [0466]
 [Math. 37]
 $T_H[i, j] = (-1)*T_H[i, j]$
 for $i, j=0, \dots, N-1$ and $i\%2 == 1$
 $\cdot \cdot \cdot (4 \ 1)$
 [0467]

10 Here, the i -row j -column component ((i, j) component) of the transformation matrix T_H of N rows and N columns is written as $T_H[i, j]$. Furthermore, "for $i, j = 0, \dots, N - 1$ and $i\%2 == 1$ " on the second row indicates that j has a value of 0 to $N - 1$, and i has an odd value of in the range of 0 to $N - 1$. That is, it

15 means that $T_H[i, j]$ indicates (each element of) an odd row vector of the transformation matrix T_H of N rows and N columns.

[0468]

20 When the processing in step S505 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0469]

<Flow of Transformation Matrix Derivation Processing>

25

Note that, since the flow of the transformation matrix derivation processing executed in step S361 (Fig. 32) of the primary vertical transform processing is similar to the case of the transformation matrix

derivation processing executed in the primary horizontal transform described with reference to the flowchart in Fig. 48, description thereof is omitted.

[0470]

5 <Transformation Matrix Derivation Unit>

Next, a configuration of the image decoding device 200 in the case will be described. Since configurations of the inverse primary transform unit 253, the inverse primary vertical transform unit 412, the inverse primary horizontal transform unit 413, and the like included in the image decoding device 200 in this case are similar to the case of <2-2. Example 1-1>, description thereof is omitted.

[0471]

15 Fig. 50 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 421 (the transformation matrix derivation unit 421 in the inverse primary vertical transform unit 412) in this case. As illustrated in Fig. 50, the transformation matrix derivation unit 421 in this case includes the transformation matrix LUT 431, the flip unit 432, and a sign inversion unit 511, similarly to the transformation matrix derivation unit 321. Note that, in Fig. 50, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 421, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0472]

The transformation matrix LUT 431 and the flip unit 432 are similar to the case in Fig. 36. The sign inversion unit 511 uses a transformation matrix T of N

rows and N columns as an input, inverts the sign of a predetermined portion of the transformation matrix T , and performs sign inversion, and outputs a transformation matrix T_{InvSign} after the sign inversion, similarly to the
 5 sign inversion unit 501. In the case of this derivation example, the transformation matrix derivation unit 421 horizontally flips the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 431 via the flip unit 432, further inverts the
 10 sign of an odd-order row vector via the sign inversion unit 511, and outputs the transformation matrix T_{InvSign} to the outside of the transformation matrix derivation unit 421 (supplies the same to the matrix calculation unit 422) as the transformation matrix T_v .

15 [0473]

As described above, the transformation matrix derivation unit 421 can implement the derivation example of the third row example from the top of the table illustrated in Fig. 10, using the flip unit 432 and the
 20 sign inversion unit 511.

[0474]

<Transformation Matrix Derivation Unit>

Fig. 51 is a block diagram illustrating a main configuration example of the transformation matrix
 25 derivation unit 451 included in the inverse primary horizontal transform unit 413 in this case. As illustrated in Fig. 51, the transformation matrix derivation unit 451 in this case includes a transformation matrix LUT 461, a flip unit 462, and a
 30 sign inversion unit 512, similarly to the transformation matrix derivation unit 421. Note that, in Fig. 51,

arrows representing data transfer are omitted, but in the transformation matrix derivation unit 451, arbitrary data can be transferred between arbitrary processing units (processing blocks).

5 [0475]

The transformation matrix LUT 461 and the flip unit 462 are similar to the case in Fig. 50. The sign inversion unit 512 uses a transformation matrix T of N rows and N columns as an input, inverts the sign of a
 10 predetermined portion of the transformation matrix T , and performs sign inversion, and outputs a transformation matrix T_{InvSign} after the sign inversion, similarly to the sign inversion unit 501. In the case of this derivation example, the transformation matrix derivation unit 451
 15 horizontally flips the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 461 via the flip unit 462, further inverts the sign of an odd-order row vector via the sign inversion unit 512, and outputs the transformation matrix T_{InvSign} to
 20 the outside of the transformation matrix derivation unit 451 (supplies the same to the matrix calculation unit 452) as the transformation matrix T_v .

[0476]

As described above, the transformation matrix
 25 derivation unit 451 can implement the derivation example of the third row example from the top of the table illustrated in Fig. 10, using the flip unit 462 and the sign inversion unit 512.

[0477]

30 <Flow of Transformation Matrix Derivation Processing>

Note that the transformation matrix derivation unit 421 and the transformation matrix derivation unit 451 perform the transformation matrix derivation processing by a flow similar to the case described with reference to the flowchart in Fig. 48, and thus description of the processing is omitted.

[0478]

<2-4. Example 1-3>

<Concept>

10 Next, the fourth row example and the fifth row example from the top except the uppermost row of item names in the table illustrated in Fig. 10 will be described.

[0479]

15 As described above, the derivation of the fourth row example from the top focuses on the similarity between the waveform of the lowest-order row vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted, similarly to the case of the first row example from the top. In this case, the derivation unit flips the first transformation matrix to derive the second transformation matrix. That is, the derivation unit uses the transformation matrix of DCT8 as the base transformation matrix T_{base} , and flips the transformation matrix in the row direction to derive the transformation matrix of FlipDCT8, as illustrated in Fig. 52. The (increasing-type) transformation matrix of DST7 having a similar waveform of the 0-order row vector can be substituted by the transformation matrix of FlipDCT8.

[0480]

Furthermore, the derivation of the fifth row example from the top focuses on the similarity between the waveform of the highest-order column vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted. In this case, the derivation unit flips the first transformation matrix and transposes the flipped first transformation matrix to derive the second transformation matrix. That is, the derivation unit uses the transformation matrix of DCT8 as the base transformation matrix T_{base} , and flips the transformation matrix in the row direction and further transposes the transformation matrix to derive the transformation matrix of TrFlipDCT8, as illustrated in Fig. 52. The (chevron-type) transformation matrix of DST1 having a similar waveform of the 0-order row vector can be substituted by the transformation matrix of TrFlipDCT8.

[0481]

That is, in both of the above two derivation examples, the second transformation matrix can be derived by one-time or two-time operation (flip, or flip and transposition). Furthermore, the operation is easy. That is, the second transformation matrix can be easily derived.

[0482]

Furthermore, by applying the above two derivation examples, it becomes unnecessary to prepare the transformation matrix of DST7 and the transformation matrix of DST1 as candidates for transformation matrices to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types

can be reduced.

[0483]

In this case, as illustrated in the table in Fig. 53, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to three types. Therefore, the total LUT size can be about 40 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.

[0484]

Note that, as described above, even in this case, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (FlipDCT8 or TrFlipDCT8), similar coding efficiency to the case of using the transformation matrix of DST7 or the transformation matrix of DST1 for the orthogonal transform/inverse orthogonal transform can be obtained.

[0485]

<Configuration>

Since configurations of the image encoding device 100 and the image decoding device 200 in this case are similar to those described in <2-2. Example 1-1>, description thereof is omitted.

[0486]

<Flow of Transformation Matrix Derivation Processing>

Next, a flow of the processing flow will be

described. Since processing other than the transformation matrix derivation processing is performed by a flow similar to the example described in <2-2.

Example 1-1> and the like, description of the processing is omitted.

[0487]

An example of a flow of the transformation matrix derivation processing in this case, which is executed by the transformation matrix derivation unit 321 included in the primary horizontal transform unit 312 in step S321 in Fig. 28, will be described with reference to the flowchart in Fig. 54.

[0488]

When the transformation matrix derivation processing is started, in step S521, the transformation matrix derivation unit 321 obtains a base transform type BaseTrType corresponding to the transform type identifier TrTypeIdxH by reference to the correspondence table illustrated in Fig. 55, for example.

[0489]

Furthermore, the transformation matrix derivation unit 321 sets the value of the flip flag FlipFlag corresponding to the transform type identifier TrTypeIdxH and the value of the transposition flag TransposeFlag corresponding to the transform type identifier TrTypeIdxH, similarly to the case of step S341 of the flowchart in Fig. 30.

[0490]

In step S522, the transformation matrix derivation unit 321 determines whether or not the flip flag FlipFlag satisfies a condition (ConditionC1) expressed by the

following expression (42).

[0491]

[Math. 38]

ConditionC1 : FlipFlag == T . . . (42)

5 [0492]

In a case where it is determined that the above-described condition (ConditionC1) is not satisfied (in a case where the flip flag FlipFlag is false (0)), the processing proceeds to step S523.

10 [0493]

In step S523, the transformation matrix derivation unit 321 sets the base transformation matrix T_{base} as the transformation matrix T_H , similarly to the case of step S343 in Fig. 30. When the processing in step S523 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0494]

Furthermore, in step S522, in a case where it is determined that the above-described condition (ConditionC1) is satisfied (in a case where the flip flag FlipFlag is true (1)), the processing proceeds to step S524.

20

[0495]

In step S524, the transformation matrix derivation unit 321 horizontally flips the base transformation matrix T_{base} , and sets the flipped transformation matrix Flip (T_{base}) as a transformation matrix T_{tmp} . This processing can be expressed as the following expression (43).

30 [0496]

[Math. 39]

$$T_{tmp} = \text{Flip}(T_{base}) = T_{base} \times J \quad \cdot \cdot \cdot \quad (43)$$

[0497]

In step S525, the transformation matrix derivation
 5 unit 321 determines whether or not the transposition flag
 TransposeFlag satisfies a condition (ConditionC2)
 expressed by the following expression (44).

[0498]

[Math. 40]

10 $\text{ConditionC2} : \text{TransposeFlag} == T \quad \cdot \cdot \cdot \quad (44)$

[0499]

In a case where it is determined that the above-
 described condition (ConditionC2) is not satisfied (in a
 case where the transposition flag TransposeFlag is false
 15 (0)), the processing proceeds to step S526.

[0500]

In step S526, the transformation matrix derivation
 unit 321 sets the transformation matrix T_{tmp} as the
 transformation matrix T_H . This processing can be
 20 expressed as the following expression (45).

[0501]

[Math. 41]

$$T_H = T_{tmp} \quad \cdot \cdot \cdot \quad (45)$$

[0502]

25 When the processing in step S526 ends, the
 transformation matrix derivation processing ends and the
 processing returns to Fig. 28. Furthermore, in step S525,
 in a case where it is determined that the above-described

condition (ConditionC2) is satisfied (in a case where the transposition flag TransposeFlag is true (1)), the processing proceeds to step S527.

[0503]

5 In step S527, the transformation matrix derivation unit 321 sets $\text{Tr}(T_{\text{tmp}})$ obtained by transposing the transformation matrix T_{tmp} as the transformation matrix T_{H} . This processing can be expressed as, for example, the following expression (46).

10 [0504]

[Math. 42]

$$T_{\text{H}} = \text{Tr}(T_{\text{tmp}}) = T_{\text{tmp}}^T \quad \cdot \cdot \cdot \quad (46)$$

[0505]

15 When the processing in step S527 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0506]

20 By executing the transformation matrix derivation processing as described above, the derivation example of the fourth and fifth row examples from the top of the table in Fig. 10 can be implemented in the horizontal one-dimensional orthogonal transform. Therefore, an increase in the required memory capacity can be suppressed.

25 [0507]

30 Note that the above description has been given such that the flip and the transposition are performed in order. However, the flip and the transposition may be collectively performed (by one-time operation). For example, the determinations in steps S522 and S525 in Fig.

54 are collectively performed, and in a case where both of the determinations are true (1), flip and transposition may be collectively performed.

[0508]

5 For example, when the flip and transposition operations are expressed as an operation for each element, an $(i, N - 1 - j)$ component of the base transformation matrix T_{base} is set as a j -row i -column component $((j, i)$ component) of the transformation matrix T_H . That is, the operations can be expressed as the following expression (47).

[0509]

[Math. 43]

$$\begin{aligned} T_H[j, i] &= T_{\text{base}}[i, N-1-j] \\ \text{for } i, j &= 0, \dots, N-1 \\ &\cdot \cdot \cdot \quad (47) \end{aligned}$$

15 [0510]

Here, the i -row j -column component $((i, j)$ component) of the transformation matrix T_H of N rows and N columns is written as $T_H[i, j]$. Furthermore, "for $i, j = 0, \dots, N - 1$ " on the second row indicates that i and j have values of 0 to $N - 1$. That is, it means that $T_H[i, j]$ indicates all of elements of the transformation matrix T_H of N rows and N columns.

[0511]

In this way, the flip and transposition can be implemented by one-time operation, which is by accessing a simple two-dimensional array. Furthermore, since the flip matrix J is unnecessary, an increase in the memory capacity can be suppressed by the capacity of the flip matrix J

[0512]

Note that, in the case of the present example, in step S361 in Fig. 32, the transformation matrix derivation processing executed by the transformation matrix derivation unit 351 included in the primary vertical transform unit 313 is also performed by a flow similar to the flowchart in Fig. 54.

[0513]

Furthermore, in the case of the present example, in step S441 in Fig. 41, the transformation matrix derivation processing executed by the transformation matrix derivation unit 421 included in the inverse primary vertical transform unit 412 is also performed by a flow similar to the flowchart in Fig. 54.

15 [0514]

Moreover, in the case of the present example, in step S461 in Fig. 42, the transformation matrix derivation processing executed by the transformation matrix derivation unit 451 included in the inverse primary horizontal transform unit 413 is also performed by a flow similar to the flowchart in Fig. 54.

[0515]

Therefore, description of the processing is omitted.

[0516]

25 <2-5. Example 1-4>

<Concept>

Next, the sixth row example from the top except the uppermost row of item names in the table illustrated in Fig. 10 will be described.

30 [0517]

As described above, the derivation of the sixth row

example from the top focuses on the characteristics between paired DCT and DST, similarly to the case of the third row example from the top (Example 1-2). More specifically, between DCT/DST to be paired (for example, 5 DCT8 and DST7), attention is paid to the point that even-numbered row vectors are axially symmetric and odd-numbered row vectors are point-symmetric. In this case, the derivation unit flips the first transformation matrix and inverts the sign of the odd-numbered row vector of 10 the flipped first transformation matrix to derive the second transformation matrix. That is, the derivation unit uses the transformation matrix of DCT8 as the base transformation matrix T_{base} , and flips the transformation matrix in the row direction and further inverts the sign 15 of an odd-order row vector to losslessly derive the transformation matrix of DST7, as illustrated in Fig. 56. Therefore, naturally, the (increasing-type) transformation matrix of DST7 having the same waveform of the 0-order row vector can be substituted by the derived 20 transformation matrix of DST7.

[0518]

That is, in this derivation example, the second transformation matrix can be derived by two-time operation (flip and sign inversion). Furthermore, the 25 operation is easy. That is, the second transformation matrix can be easily derived.

[0519]

Furthermore, by applying this derivation example, it becomes unnecessary to prepare the transformation 30 matrix of DST7 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal

transform. That is, the number of unique transform types can be reduced.

[0520]

In this case, as illustrated in the table in Fig. 57, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to four types. Therefore, the total LUT size can be about 47 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.

[0521]

Furthermore, in this derivation example, the transformation matrix of DST7 can be derived as the second transformation matrix. Therefore, by performing the orthogonal transform/inverse orthogonal transform using the second transformation matrix, the same coding efficiency as the case of using the transformation matrix of DST7 for the orthogonal transform/inverse orthogonal transform can be obtained.

[0522]

<Flow of Configuration and Processing>

Since configurations of the image encoding device 100 and the image decoding device 200 in this case are similar to those described in <2-3. Example 1-2>, description thereof is omitted. Furthermore, since processing performed by the image encoding device 100 and the image decoding device 200 is similar to the processing described in <2-3. Example 1-2>, description

thereof is omitted.

[0523]

Note that, in the case of the present example, in obtaining a base transform type corresponding to the transform type identifier TrTypeIdxH in the transformation matrix derivation processing, the base transform type is obtained by reference to the correspondence table illustrated in Fig. 58, for example. Other than the above, the processing is simply performed according to a flow similar to the case described with reference to the flowchart in Fig. 48.

[0524]

<2-6. Example 1-5>

<Concept>

Next, the seventh row example and the eighth row example from the top except the uppermost row of item names in the table illustrated in Fig. 10 will be described.

[0525]

As described above, the derivation of the seventh row example from the top focuses on the similarity between the waveform of the lowest-order row vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted, similarly to the case of the first row example from the top. Note that, in this case, the derivation unit transposes the first transformation matrix to derive the second transformation matrix. That is, the derivation unit uses the transformation matrix of DCT2 as the base transformation matrix T_{base} , and transposes the transformation matrix to

derive the transformation matrix of DCT3, as illustrated in Fig. 59. The (decreasing-type) transformation matrix of DCT8 having a similar waveform of the 0-order row vector can be substituted by the transformation matrix of

DCT3.

[0526]

Furthermore, the derivation of the eighth row example from the top focuses on the similarity between the waveform of the highest-order column vector of the first transform type and the waveform of the lowest-order row vector of the transform type of a transformation matrix to be substituted, similarly to the case of the fifth row example from the top. In this case, the derivation unit transposes the first transformation matrix and flips the transposed first transformation matrix to derive the second transformation matrix. That is, the derivation unit uses the transformation matrix of DCT2 as the base transformation matrix T_{base} , and transposes the transformation matrix and further flips the transformation matrix in the row direction to derive the transformation matrix of FlipDCT3, as illustrated in Fig. 59. The (increasing-type) transformation matrix of DST7 having a similar waveform of the 0-order row vector can be substituted by the transformation matrix of

FlipDCT3.

[0527]

That is, in both of the above two derivation examples, the second transformation matrix can be derived by one-time or two-time operation (transposition, or transposition and flip). Furthermore, the operation is easy. That is, the second transformation matrix can be

easily derived.

[0528]

Furthermore, by applying the above two derivation examples, it becomes unnecessary to prepare the transformation matrix of DCT8 and the transformation matrix of DST7 as candidates for transformation matrices to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced.

10 [0529]

In this case, as illustrated in the table in Fig. 60, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to three types. Therefore, the total LUT size can be about 40 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.

20

[0530]

Note that, as described above, even in this case, by performing the orthogonal transform/inverse orthogonal transform using the transformation matrix of the derived second transform type (DCT3 or FlipDCT3), similar coding efficiency to the case of using the transformation matrix of DCT8 or the transformation matrix of DST7 for the orthogonal transform/inverse orthogonal transform can be obtained.

30 [0531]

<Configuration>

Since configurations of the image encoding device 100 and the image decoding device 200 in this case are similar to those described in <2-2. Example 1-1>, description thereof is omitted.

5 [0532]

<Flow of Transformation Matrix Derivation Processing>

Next, a flow of the processing flow will be described. Since processing other than the transformation matrix derivation processing is performed by a flow similar to the example described in <2-2. Example 1-1> and the like, description of the processing is omitted.

[0533]

15 An example of a flow of the transformation matrix derivation processing in this case, which is executed by the transformation matrix derivation unit 321 included in the primary horizontal transform unit 312 in step S321 in Fig. 28, will be described with reference to the flowchart in Fig. 61.

[0534]

When the transformation matrix derivation processing is started, in step S541, the transformation matrix derivation unit 321 obtains a base transform type BaseTrType corresponding to the transform type identifier TrTypeIdxH by reference to the correspondence table illustrated in Fig. 62, for example.

[0535]

Furthermore, the transformation matrix derivation unit 321 sets the value of the flip flag FlipFlag corresponding to the transform type identifier TrTypeIdxH

and the value of the transposition flag `TransposeFlag` corresponding to the transform type identifier `TrTypeIdxH`, similarly to the case of step S341 of the flowchart in Fig. 30.

5 [0536]

In step S542, the transformation matrix derivation unit 321 determines whether or not the transposition flag `TransposeFlag` satisfies a condition (`ConditionD1`) expressed by the following expression (48).

10 [0537]

[Math. 44]

$\text{ConditionD1} : \text{TransposeFlag} \stackrel{====}{=} \text{T}$
 $\cdot \cdot \cdot (48)$

[0538]

In a case where it is determined that the above-described condition (`ConditionD1`) is not satisfied (in a case where the transposition flag `TransposeFlag` is false (0)), the processing proceeds to step S543.

[0539]

In step S543, the transformation matrix derivation unit 321 sets the base transformation matrix T_{base} as the transformation matrix T_H , similarly to the case of step S343 in Fig. 30. When the processing in step S543 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

25 [0540]

Furthermore, in step S542, in a case where it is determined that the above-described condition (`ConditionD1`) is satisfied (in a case where the transposition flag `TransposeFlag` is true (1)), the processing proceeds to step S544.

30

[0541]

In step S544, the transformation matrix derivation unit 321 transposes the base transformation matrix T_{base} in the horizontal direction, and sets the transposed transformation matrix $Tr(T_{base})$ as the transformation matrix T_{tmp} . This processing can be expressed as the following expression (49).

[0542]

[Math. 45]

$$T_{tmp} = Tr(T_{base}) = T_{base}^T \quad \cdot \cdot \cdot \quad (49)$$

[0543]

In step S545, the transformation matrix derivation unit 321 determines whether or not the flip flag FlipFlag satisfies a condition (ConditionD2) expressed by the following expression (50).

[0544]

[Math. 46]

$$\text{ConditionD2} : \text{FlipFlag} \neq T \quad \cdot \cdot \cdot \quad (50)$$

[0545]

In a case where it is determined that the above-described condition (ConditionD2) is not satisfied (in a case where the flip flag FlipFlag is false (0)), the processing proceeds to step S546.

[0546]

In step S546, the transformation matrix derivation unit 321 sets the transformation matrix T_{tmp} as the transformation matrix T_H , as in the above-described expression (45).

[0547]

When the processing in step S546 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28. Furthermore, in step S545, in a case where it is determined that the above-described condition (ConditionD2) is satisfied (in a case where the flip flag FlipFlag is true (1)), the processing proceeds to step S547.

[0548]

In step S547, the transformation matrix derivation unit 321 sets Flip (T_{tmp}) obtained by flipping the transformation matrix T_{tmp} as the transformation matrix T_H . This processing can be expressed as, for example, the following expression (51).

[0549]

15 [Math. 47]

$$T_H = \text{Flip}(T_{tmp}) = T_{tmp} \times J \quad \dots \quad (51)$$

[0550]

When the processing in step S547 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0551]

By executing the transformation matrix derivation processing as described above, the derivation example of the seventh and eighth row examples from the top of the table in Fig. 10 can be implemented in the horizontal one-dimensional orthogonal transform. Therefore, an increase in the required memory capacity can be suppressed.

[0552]

30 Note that the above description has been given such that the transposition and the flip are performed in

order. However, the transposition and the flip may be collectively performed (by one-time operation). For example, the determinations in steps S542 and S545 in Fig. 61 are collectively performed, and in a case where both of the determinations are true (1), transposition and flip may be collectively performed.

[0553]

For example, when the transposition and flip operations are expressed as an operation for each element, a (j, i) component of the base transformation matrix T_{base} is set as an i-row (N - 1 - j)-column component ((i, (N - 1 - j) component) of the transformation matrix T_H . That is, this processing can be expressed as the following expression (52).

15 [0554]

[Math. 48]

$$T_H[i, N-1-j] = T_{\text{base}}[j, i]$$

for i, j=0, ..., N-1
 . . . (52)

[0555]

Here, the i-row j-column component ((i, j) component) of the transformation matrix T_H of N rows and N columns is written as $T_H[i, j]$. Furthermore, "for i, j = 0, ..., N - 1" on the second row indicates that i and j have values of 0 to N - 1. That is, it means that $T_H[i, j]$ indicates all of elements of the transformation matrix T_H of N rows and N columns.

25

[0556]

In this way, the transposition and flip can be implemented by one-time operation, which is by accessing a simple two-dimensional array. Furthermore, since the

flip matrix J is unnecessary, an increase in the memory capacity can be suppressed by the capacity of the flip matrix J

[0557]

5 Note that, in the case of the present example, in step S361 in Fig. 32, the transformation matrix derivation processing executed by the transformation matrix derivation unit 351 included in the primary vertical transform unit 313 is also performed by a flow similar to the flowchart in Fig. 61. For example, in the flowchart in Fig. 61, interpretation is simply made by replacing the transform type identifier $TrTypeIdxH$ with $TrTypeIdxV$, and replacing the transformation matrix T_H with the transformation matrix T_V .

15 [0558]

 Furthermore, in the case of the present example, in step S441 in Fig. 41, the transformation matrix derivation processing executed by the transformation matrix derivation unit 421 included in the inverse primary vertical transform unit 412 is also performed by a flow similar to the flowchart in Fig. 61. For example, in the flowchart in Fig. 61, interpretation is simply made by replacing the transform type identifier $TrTypeIdxH$ with $TrTypeIdxV$, and replacing the transformation matrix T_H with the transformation matrix T_V .

[0559]

 Moreover, in the case of the present example, in step S461 in Fig. 42, the transformation matrix derivation processing executed by the transformation matrix derivation unit 451 included in the inverse

30

primary horizontal transform unit 413 is also performed by a flow similar to the flowchart in Fig. 61.

[0560]

Therefore, description of the processing is omitted.

5 [0561]

<3. Second Embodiment>

<3-1. Common Concept>

<Prediction Residual Permutation Operation>

In the first embodiment, operating the
10 transformation matrix to generate another transformation matrix has been described. However, not only a transformation matrix but also a prediction residual may be operated. That is, a prediction residual of an image may be operated for permutation, the prediction residual
15 operated for permutation may be orthogonally transformed using a transformation matrix serving as a base, and coefficient data obtained by orthogonally transforming the prediction residual may be encoded to generate a bit stream.

20 [0562]

For example, an image processing apparatus may include an operation unit configured to perform an operation for permutation for a prediction residual of an image, an orthogonal transform unit configured to
25 orthogonally transform the prediction residual operated for permutation by the operation unit, using a transformation matrix serving as a base, and an encoding unit configured to encode coefficient data obtained by orthogonally transforming the prediction residual by the
30 orthogonal transform unit to generate a bit stream.

[0563]

Even in this case, two-dimensional orthogonal transform equivalent to the case of the first embodiment can be implemented. That is, orthogonal transform using another transformation matrix can be substantially
5 implemented by orthogonal transform using a certain transformation matrix. Therefore, an increase in the number of transformation matrices prepared for orthogonal transform can be suppressed, and an increase in memory capacity required for orthogonal transform can be
10 suppressed, similarly to the case of the first embodiment. Furthermore, since the number of operations can be reduced as compared with the case of the first embodiment, an increase in a processing amount of orthogonal transform can be suppressed.

15 [0564]

Furthermore, a bit stream may be decoded to obtain coefficient data that is an orthogonally transformed prediction residual of an image, the obtained coefficient data may be inversely orthogonally transformed, and an
20 inverse orthogonal transform result of the obtained coefficient data may be permuted.

[0565]

For example, the image processing apparatus may include a decoding unit configured to decode a bit stream
25 to obtain coefficient data that is an orthogonally transformed prediction residual of an image, an inverse orthogonal transform unit configured to inversely orthogonally transform the coefficient data obtained by the decoding unit, and an operation unit configured to
30 perform an operation for permutation for an inverse orthogonal transform result of the coefficient data

obtained by the inverse orthogonal transform unit.

[0566]

Even in this case, inverse two-dimensional
orthogonal transform equivalent to the case of the first
5 embodiment can be implemented. That is, inverse
orthogonal transform by another transformation matrix can
be substantially implemented by inverse orthogonal
transform using a certain transformation matrix.
Therefore, an increase in the number of transformation
10 matrices prepared for inverse orthogonal transform can be
suppressed, and an increase in memory capacity required
for inverse orthogonal transform can be suppressed,
similarly to the case of the first embodiment.
Furthermore, since the number of operations can be
15 reduced as compared with the case of the first embodiment,
an increase in a processing amount of inverse orthogonal
transform can be suppressed.

[0567]

More specific description will be given. When
20 focusing on a spatial symmetric property of two different
two-dimensional orthogonal transforms (including a
prediction residual), there are some cases where axial
symmetry exists between one-dimensional orthogonal
transforms in a certain direction (horizontal direction
25 or vertical direction), such as between a transformation
matrix of DST7 and a transformation matrix of FlipDST7
that is a flipped transformation matrix of DST7.
Furthermore, as for the prediction residual, there are
some cases where an axial symmetric property exists
30 between prediction residuals in a certain direction
(horizontal direction or vertical direction), such as

between a prediction residual (X) and a flipped
prediction residual (X·J).

[0568]

In such a case, for example, as illustrated in Fig.
5 63, by orthogonally transforming a flipped original
prediction residual, using a first transformation matrix,
orthogonal transform of the original prediction residual
using a second transformation matrix that is the flipped
first transformation matrix can be substituted.

10 [0569]

In the case of the example in Fig. 63, on the left
side in Fig. 63, vertical one-dimensional orthogonal
transform using the first transformation matrix, and
horizontal one-dimensional orthogonal transform using the
15 flipped first transformation matrix are performed for the
original prediction residual, as in the following
expression (53).

[0570]

[Math. 49]

20
$$Y = I_2 \cdot X \cdot (T_1^T \cdot J) \quad \cdot \cdot \cdot \quad (53)$$

[0571]

Meanwhile, on the right side in Fig. 63, flip of
the prediction residual (the expression (54) below) and
vertical and horizontal one-dimensional orthogonal
25 transforms using the first transformation matrix for the
flipped prediction residual (the expression (55) below)
are illustrated.

[0572]

[Math. 50]

$$X' = (X \cdot J) \quad \cdot \cdot \cdot \quad (5.4)$$

$$Y = T_2 \cdot X' \cdot T_1^T \quad \cdot \cdot \cdot \quad (5.5)$$

[0573]

Such two two-dimensional orthogonal transforms have a spatial symmetric property. Thus, one can be substituted by the other.

[0574]

In other words, in the case where a spatial symmetric property is present between a base transformation matrix (transform type) and a transformation matrix used for orthogonal transform of a prediction residual, which is derived using the base transformation matrix, orthogonal transform using another transformation matrix (orthogonal transform for a prediction residual before flip, using a transformation matrix derived using a base transformation matrix) can be substituted by flipping the prediction residual in the direction of the spatial symmetric property, and orthogonally transforming the flipped prediction residual, using the base transformation matrix.

[0575]

That is, the operation unit may flip the prediction residual in the spatial symmetric direction between one-dimensional orthogonal transforms, and the orthogonal transform unit may orthogonally transform the prediction residual flipped by the operation unit, using the transformation matrix serving as a base. By doing so, orthogonal transform equivalent to orthogonal transform using another transformation matrix can be performed. That is, as described above, orthogonal transform using another transformation matrix can be substantially

implemented. Therefore, an increase in the number of transformation matrices prepared for the orthogonal transform can be suppressed, and an increase in the memory capacity required for the orthogonal transform can
5 be suppressed.

[0576]

This similarly applies to the inverse orthogonal transform. That is, the coefficient data obtained by decoding a bit stream may be inversely orthogonally
10 transformed using the base transformation matrix, and an inverse orthogonal transform result may be flipped in the spatial symmetric direction between one-dimensional orthogonal transforms by the operation unit. With the configuration, the inverse orthogonal transform using
15 another transformation matrix can be substantially implemented. Therefore, an increase in the number of transformation matrices prepared for the inverse orthogonal transform can be suppressed, and an increase in memory capacity required for the inverse orthogonal
20 transform can be suppressed.

[0577]

For example, the left side in Fig. 64 illustrates two-dimensional orthogonal transform in which horizontal one-dimensional orthogonal transform using FlipDST7 that
25 is a flipped transformation matrix of DST7, and vertical one-dimensional orthogonal transform using a transformation matrix of DST7 are performed for a prediction residual X. Furthermore, the right side in Fig. 64 illustrates two-dimensional orthogonal transform
30 in which horizontal and vertical one-dimensional orthogonal transforms using the transformation matrix of

DST7 are performed for a prediction residual X' ($= X \cdot J$) obtained by horizontally flipping the prediction residual X . These are horizontally symmetric with each other and are equivalent. Therefore, for example, the former two-dimensional orthogonal transform can be substituted by the latter two-dimensional orthogonal transform. Note that the former two-dimensional orthogonal transform is expressed by the following expression (56), and the latter two-dimensional orthogonal transform is expressed by the following expression (57).

[0578]

[Math. 51]

$$Y = T_2 \cdot X \cdot (T_1 \cdot J)^T \quad \dots \quad (56)$$

$$Y = T_2 \cdot (X \cdot J) \cdot T_1^T \quad \dots \quad (57)$$

[0579]

That is, for example, in a case where a horizontal symmetric property is present between a base transformation matrix (transform type), and a transformation matrix to be used for prediction residual orthogonal transform, the transformation matrix being derived using the base transformation matrix, the operation unit may horizontally flip the prediction residual, and the orthogonal transform unit may orthogonally transform the prediction residual horizontally flipped by the operation unit, using the base transformation matrix. By doing so, orthogonal transform using another transformation matrix having a horizontal symmetric property with the orthogonal transform using the base transformation matrix can be substantially implemented.

[0580]

This similarly applies to the inverse orthogonal transform. That is, in a case where a horizontal symmetric property is present between a base transformation matrix (transform type), and a transformation matrix to be used for coefficient data inverse orthogonal transform, the transformation matrix being derived using the base transformation matrix, the inverse orthogonal transform unit may inversely orthogonally transform the coefficient data obtained by decoding the bit stream, using the base transformation matrix, and the operation unit may horizontally flip an inverse orthogonal transform result. By doing so, inverse orthogonal transform using another transformation matrix having a horizontal symmetric property with the inverse orthogonal transform using the base transformation matrix can be substantially implemented.

[0581]

Furthermore, for example, the left side in Fig. 65 illustrates two-dimensional orthogonal transform in which horizontal one-dimensional orthogonal transform using the transformation matrix of DST7, and vertical one-dimensional orthogonal transform using FlipDST7 that is the flipped transformation matrix of DST7 are performed for the prediction residual X . Furthermore, the right side in Fig. 65 illustrates two-dimensional orthogonal transform in which horizontal and vertical one-dimensional orthogonal transforms using the transformation matrix of DST7 are performed for the prediction residual X' ($= J \cdot X$) obtained by vertically flipping the prediction residual X . These are vertically symmetric with each other and are equivalent. Therefore,

for example, the former two-dimensional orthogonal transform can be substituted by the latter two-dimensional orthogonal transform. Note that the former two-dimensional orthogonal transform is expressed by the following expression (58), and the latter two-dimensional orthogonal transform is expressed by the following expression (59).

[0582]

[Math. 52]

$$Y = (T_2 \cdot J) \cdot X \cdot T_1^T \quad \cdot \cdot \cdot \quad (58)$$

$$Y = T_2 \cdot (J \cdot X) \cdot T_1^T \quad \cdot \cdot \cdot \quad (59)$$

[0583]

That is, for example, in a case where a vertical symmetric property is present between a base transformation matrix (transform type), and a transformation matrix to be used for prediction residual orthogonal transform, the transformation matrix being derived using the base transformation matrix, the operation unit may vertically flip the prediction residual, and the orthogonal transform unit may orthogonally transform the prediction residual vertically flipped by the operation unit, using the base transformation matrix. By doing so, orthogonal transform using another transformation matrix having a vertical symmetric property with the orthogonal transform using the base transformation matrix can be substantially implemented.

[0584]

This similarly applies to the inverse orthogonal transform. That is, in a case where a vertical symmetric property is present between a base transformation matrix

(transform type), and a transformation matrix to be used for coefficient data inverse orthogonal transform, the transformation matrix being derived using the base transformation matrix, the inverse orthogonal transform unit may inversely orthogonally transform the coefficient data obtained by decoding the bit stream, using the base transformation matrix, and the operation unit may vertically flip an inverse orthogonal transform result. By doing so, inverse orthogonal transform using another transformation matrix having a vertical symmetric property with the inverse orthogonal transform using the base transformation matrix can be substantially implemented.

[0585]

Moreover, for example, the left side in Fig. 66 illustrates two-dimensional orthogonal transform in which horizontal and vertical one-dimensional orthogonal transforms using FlipDST7 that is the flipped transformation matrix of DST7 are performed for the prediction residual X . Furthermore, the right side in Fig. 66 illustrates two-dimensional orthogonal transform in which horizontal and vertical one-dimensional orthogonal transforms using the transformation matrix of DST7 are performed for the prediction residual X' ($= J \cdot X \cdot J$) obtained by horizontally and vertically flipping the prediction residual X . These are horizontally and vertically symmetric with each other and are equivalent. Therefore, for example, the former two-dimensional orthogonal transform can be substituted by the latter two-dimensional orthogonal transform. Note that the former two-dimensional orthogonal transform is expressed

by the following expression (60), and the latter two-dimensional orthogonal transform is expressed by the following expression (61).

[0586]

5 [Math. 53]

$$Y = (T_2 \cdot J) \cdot X \cdot (T_1 \cdot J)^T \quad \dots \quad (60)$$

$$Y = T_2 \cdot (J \cdot X \cdot J) \cdot T_1^T \quad \dots \quad (61)$$

[0587]

That is, for example, in a case where horizontal and vertical symmetric properties are present between a
 10 base transformation matrix (transform type), and a transformation matrix to be used for prediction residual orthogonal transform, the transformation matrix being derived using the base transformation matrix, the operation unit may horizontally and vertically flip the
 15 prediction residual, and the orthogonal transform unit may orthogonally transform the prediction residual horizontally and vertically flipped by the operation unit, using the base transformation matrix. By doing so, orthogonal transform using another transformation matrix
 20 having horizontal and vertical symmetric properties with the orthogonal transform using the base transformation matrix can be substantially implemented.

[0588]

This similarly applies to the inverse orthogonal
 25 transform. That is, in a case where horizontal and vertical symmetric properties are present between a base transformation matrix (transform type), and a transformation matrix to be used for coefficient data inverse orthogonal transform, the transformation matrix
 30 being derived using the base transformation matrix, the

inverse orthogonal transform unit may inversely
orthogonally transform the coefficient data obtained by
decoding the bit stream, using the base transformation
matrix, and the operation unit may horizontally and
5 vertically flip an inverse orthogonal transform result.
By doing so, inverse orthogonal transform using another
transformation matrix having horizontal and vertical
symmetric properties with the inverse orthogonal
transform using the base transformation matrix can be
10 substantially implemented.

[0589]

Note that, in the case of the above expression (60),
transformation matrices T_1 and T_2 are flipped, so that
the operation is required twice. In contrast, in the
15 case of the above expression (61), the flip of the
prediction residual ($J \cdot X \cdot J$) can be permuted with a matrix
product of a permutation matrix P and the prediction
residual X . That is, the operation is required once.
Therefore, the number of operations can be reduced as
20 compared with the case of the above expression (60) (in
the case of flipping the transformation matrix), and an
increase in the processing amount of the orthogonal
transform/inverse orthogonal transform can be suppressed
(the orthogonal transform/inverse orthogonal transforms
25 can be more easily performed). That is, an increase in
the load of the orthogonal transform/inverse orthogonal
transform processing can be suppressed. Furthermore, the
orthogonal transform/inverse orthogonal transform can be
performed at higher speed.

30 [0590]

<Sign Inversion of Transformation Matrix>

An operation of a transformation matrix may be applied in addition to the permutation operation of the prediction residual. For example, the derivation unit may derive the second transformation matrix using the
5 first transformation matrix, the operation unit may flip the prediction residual in a spatial symmetric direction between one-dimensional orthogonal transforms, and the orthogonal transform unit may orthogonally transform the prediction residual flipped by the operation unit, using
10 the second transformation matrix derived by the derivation unit. By doing so, orthogonal transform using another transformation matrix can be substantially implemented. Therefore, an increase in the number of transformation matrices prepared for the orthogonal
15 transform can be suppressed, and an increase in the memory capacity required for the orthogonal transform can be suppressed.

[0591]

This similarly applies to the inverse orthogonal
20 transform. For example, the derivation unit may derive the second transformation matrix using the first transformation matrix, the inverse orthogonal transform unit may inversely orthogonally transform the coefficient data obtained by the decoding unit, using the second
25 transformation matrix derived by the derivation unit, and the operation unit may permute the inverse orthogonal transform result of the coefficient data obtained by the inverse orthogonal transform unit. By doing so, inverse
30 orthogonal transform using another transformation matrix can be substantially implemented. Therefore, an increase in the number of transformation matrices prepared for the

inverse orthogonal transform can be suppressed, and an increase in the memory capacity required for the inverse orthogonal transform can be suppressed.

[0592]

5 Note that, in the derivation in the orthogonal transform or the inverse orthogonal transform, the derivation unit may derive the second transformation matrix having the same number of rows and the same number of columns as the first transformation matrix. In a case
10 of changing the number of rows and columns, there is a possibility that the waveform type unintentionally changes. Therefore, by setting the number of rows and columns to be the same as those of the first transformation matrix, the possibility of an unintended
15 change in the waveform type can be suppressed and the second transformation matrix can be more easily derived.
[0593]

 Furthermore, in the derivation of the second transformation matrix, the derivation unit may invert the
20 sign of an odd-numbered row vector of the first transformation matrix to derive the second transformation matrix. By performing orthogonal transform by permuting the prediction residual, using the second transformation matrix derived as described above, orthogonal transform
25 for a prediction residual, using another transformation matrix where an even-numbered row vector is axially symmetric and an odd-numbered row vector is point-symmetric with respect to the first transformation matrix can be implemented (substituted). Furthermore, by
30 performing inverse orthogonal transform using the second transformation matrix derived as described above, and

permuting an inverse orthogonal transform result, as described above, inverse orthogonal transform using another transformation matrix where an even-numbered row vector is axially symmetric and an odd-numbered row vector is point-symmetric with respect to the first transformation matrix can be implemented (substituted).
[0594]

<Prediction Residual Operation Example>

A list of prediction residual operation examples for substituting the orthogonal transform/inverse orthogonal transform using another transformation matrix is illustrated in Fig. 67.
[0595]

In the table illustrated in Fig. 67, the operation of the first row example from the top except the uppermost row of item names focuses on the fact that an axial symmetric property is present between one-dimensional orthogonal transforms in a certain direction (one or both of horizontal direction and vertical direction) in two different two-dimensional orthogonal transforms.
[0596]

In this case, the operation unit flips the prediction residual in the spatial symmetric direction between one-dimensional orthogonal transforms, and the orthogonal transform unit orthogonally transforms the prediction residual flipped by the operation unit, using the transformation matrix serving as a base. More specifically, the operation unit flips the prediction residual, and the orthogonal transform unit uses the transformation matrix of DST7 as a base transformation

matrix T_{base} and orthogonally transforms the flipped prediction residual using the transformation matrix of DST7. By doing so, orthogonal transform for an unflipped prediction residual using the transformation matrix of FlipDST7 can be substituted. Note that the direction of flipping the prediction residual corresponds to the spatial symmetric direction (one or both of the horizontal direction and vertical direction).

[0597]

10 This similarly applies to the inverse orthogonal transform. The inverse orthogonal transform unit inversely orthogonally transforms the coefficient data obtained by decoding a bit stream, using the base transformation matrix, and the operation unit flips an inverse orthogonal transform result in a spatial symmetric direction between one-dimensional orthogonal transforms. More specifically, the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data, using the transformation matrix of DST7, and the operation unit flips an inverse orthogonal transform result. By doing so, the inverse orthogonal transform of the coefficient data using the transformation matrix of FlipDST7 can be substituted. Note that the direction of flipping the inverse orthogonal transform result corresponds to the spatial symmetric direction (one or both of the horizontal direction and vertical direction).

[0598]

By applying such an operation for the prediction residual (including the inverse orthogonal transform result), it becomes unnecessary to prepare the

transformation matrix of FlipDST7 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced,

5 similarly to the case of the first embodiment. That is, an increase in the LUT size can be suppressed.

Furthermore, by performing the orthogonal transform/inverse orthogonal transform involving such an operation for the prediction residual (including the
10 inverse orthogonal transform result), similar coding efficiency to the case of the orthogonal transform/inverse orthogonal transform using the transformation matrix of FlipDST7 can be obtained.
[0599]

15 Furthermore, since the two-dimensional flip operation for the prediction residual (including the inverse orthogonal transform result) can be performed by a collective operation, the number of flip operations can be reduced, as compared with the case of the orthogonal
20 transform/inverse orthogonal transform using the transformation matrix of FlipDST7. Therefore, an increase in the processing amount of the orthogonal transform/inverse orthogonal transform (the orthogonal transform/inverse orthogonal transform can be more easily
25 performed). That is, an increase in the load of the orthogonal transform/inverse orthogonal transform processing can be suppressed. Furthermore, the orthogonal transform/inverse orthogonal transform can be performed at higher speed.

30 [0600]

Furthermore, the operation one row below the first

row example (the second row from the top) focuses on similarity between one-dimensional orthogonal transforms in a certain direction (one or both of the horizontal direction and vertical direction) in the two different two-dimensional orthogonal transforms. More specifically, attention is paid to an axial symmetric property in even-numbered row vectors and a point-symmetric property in odd-numbered row vectors.

[0601]

10 In this case, the derivation unit derives the second transformation matrix, using the first transformation matrix, the operation unit permutes the prediction residual of an image, and the orthogonal transform unit orthogonally transforms the prediction residual permuted by the operation unit, using the second transformation matrix derived by the derivation unit. More specifically, the derivation unit inverts the sign of an odd-numbered row vector of the transformation matrix of DST7, the operation unit flips the prediction residual, and the orthogonal transform unit orthogonally transforms the flipped prediction residual, using a transformation matrix obtained by inverting the sign of the odd-numbered row vector of the transformation matrix of DST7, which has been derived by the derivation unit.

20 By doing so, the orthogonal transform of an unflipped prediction residual using the transformation matrix of DCT8 can be substituted.

[0602]

30 Note that the direction of flipping the prediction residual corresponds to the spatial symmetric direction (one or both of the horizontal direction and vertical

direction). Further, since the operation for the transformation matrix is only the sign inversion for the odd-numbered row vector, the number of rows and columns of the transformation matrix derived by the derivation unit are the same as those of DST7.

[0603]

This similarly applies to the inverse orthogonal transform. The derivation unit derives the second transformation matrix, using the first transformation matrix, the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data obtained by decoding a bit stream, using the second transformation matrix, and the operation unit flips an inverse orthogonal transform result in the spatial symmetric direction between one-dimensional orthogonal transforms. More specifically, the derivation unit inverts the sign of an odd-numbered row vector of the transformation matrix of DST7, the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data, using the transformation matrix of DST7 with the inverted sign of the odd-numbered row vector, and the operation unit flips an inverse orthogonal transform result. By doing so, the inverse orthogonal transform of the coefficient data using the transformation matrix of DCT8 can be substituted.

[0604]

Note that the direction of flipping the inverse orthogonal transform result corresponds to the spatial symmetric direction (one or both of the horizontal direction and vertical direction). Further, since the operation for the transformation matrix is only the sign

inversion for the odd-numbered row vector, the number of rows and columns of the transformation matrix derived by the derivation unit are the same as those of DST7.

[0605]

5 By applying such an operation for the prediction residual (including the inverse orthogonal transform result), it becomes unnecessary to prepare the transformation matrix of DCT8 as a candidate for a transformation matrix to be used for orthogonal
10 transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced, similarly to the case of the first embodiment. That is, an increase in the LUT size can be suppressed. Furthermore, by performing the orthogonal
15 transform/inverse orthogonal transform involving such an operation for the prediction residual (including the inverse orthogonal transform result), the same coding efficiency as the case of the orthogonal transform/inverse orthogonal transform using the
20 transformation matrix of DCT can be obtained.

[0606]

 Furthermore, the operation one row below the second row example (the third row from the top) focuses on an axial symmetric property between one-dimensional
25 orthogonal transforms in a certain direction (one or both of horizontal direction and vertical direction) in two different two-dimensional orthogonal transforms. This operation is the same as the operation of the first row from the top.

30 [0607]

 Note that, in this case, the transformation matrix

of DCT8 is used as the base transformation matrix T_{base} . Therefore, by doing so, the orthogonal transform of an unflipped prediction residual using the transformation matrix of FlipDCT8 can be substituted. This similarly
 5 applies to the inverse orthogonal transform. That is, by doing so, the inverse orthogonal transform of the coefficient data using the transformation matrix of FlipDCT8 can be substituted.

[0608]

10 Furthermore, the operation one row below the third row example (the fourth row from the top) focuses on similarity between one-dimensional orthogonal transforms in a certain direction (one or both of horizontal direction and vertical direction) (an axial symmetric
 15 property of even-numbered row vectors, and a point-symmetric property of odd-numbered row vectors) in two different two-dimensional orthogonal transforms. This operation is the same as the operation of the second row from the top.

20 [0609]

Note that, in this case, the transformation matrix of DCT8 is used as the base transformation matrix T_{base} . Therefore, by doing so, the orthogonal transform of an unflipped prediction residual using the transformation
 25 matrix of DST7 can be substituted. This similarly applies to the inverse orthogonal transform. That is, by doing so, the inverse orthogonal transform of the coefficient data using the transformation matrix of DST7 can be substituted.

30 [0610]

Note that each of the above-described derivation

examples may be performed independently or may be performed in combination of a plurality of derivation examples. Furthermore, in the above-described example, an example of a base transform type (first transform
5 type) has been described using DST7 or DCT8. However, the derivation can be implemented by replacing the transform type with a transform type having the same type of waveform shape, as illustrated in Fig. 8. For example, DST7 may be replaced with another orthogonal transform
10 having the same type of waveform shape, such as DST4, DST8, or DST3. Similarly, DCT8 may be replaced with another orthogonal transform having the same type of waveform shape, such as DCT3, DCT7, or DCT4.
[0611]

15 <Flow of Configuration and Processing>

A configuration of an image encoding device 100 that performs such a prediction residual permutation operation is similar to the case of the first embodiment. Then, in the image encoding device 100, an orthogonal
20 transform unit 113 performs processing to which the above-described present technology is applied, as an operation unit and an orthogonal transform unit. Furthermore, the encoding unit 115 performs processing to which the above-described present technology is applied,
25 as an encoding unit. Furthermore, an inverse orthogonal transform unit 118 performs processing to which the above-described present technology is applied, as an inverse orthogonal transform unit and an operation unit. Therefore, the image encoding device 100 can suppress an
30 increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform.

[0612]

Furthermore, the configuration of the orthogonal transform unit 113 is similar to the case of the first embodiment. Then, in the orthogonal transform unit 113, a primary transform unit 152 performs processing to which the above-described present technology is applied, as an operation unit and an orthogonal transform unit. That is, the operation unit permutes the prediction residual of an image, and the orthogonal transform unit performs primary transform for the prediction residual permuted by the operation unit, using the transformation matrix serving as a base. Therefore, an increase in the memory capacity required for the primary transform can be suppressed.

[0613]

Note that, as described above, the primary transform unit 152 performs the primary horizontal transform and the primary vertical transform as the primary transform. That is, the operation unit permutes the prediction residual of an image, and the orthogonal transform unit performs, for the permuted prediction residual, horizontal one-dimensional orthogonal transform using a second transformation matrix for horizontal one-dimensional orthogonal transform serving as a base, and further, vertical one-dimensional orthogonal transform using the second transformation matrix for vertical one-dimensional orthogonal transform serving as a base, as the primary transform. Therefore, an increase in the memory capacity required for the primary transform where the horizontal one-dimensional orthogonal transform and the vertical one-dimensional orthogonal transform are performed can be suppressed.

[0614]

Note that a flow of image encoding processing executed by the image encoding device 100 is similar to the case of the first embodiment. That is, in the image encoding processing of the above flow, processing to which the above-described present technology is applied is performed as processing of step S106. Furthermore, processing to which the above-described present technology is applied is performed as processing of step S109. Moreover, processing to which the above-described present technology is applied is performed as processing of step S113. Therefore, by executing the image encoding processing, an increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform can be suppressed.

[0615]

Furthermore, a flow of orthogonal transform processing executed in step S106 of the image encoding processing is similar to the case of the first embodiment. That is, in the orthogonal transform processing of the above flow, processing to which the above-described present technology is applied is performed as processing of step S132. Therefore, by executing the orthogonal transform processing, an increase in the memory capacity required for the primary transform can be suppressed.

[0616]

Furthermore, processing is similar on a decoding side. That is, a configuration of an image decoding device 200 that performs such a permutation operation of the inverse orthogonal transform result is similar to the case of the first embodiment. Then, in the image

decoding device 200, an inverse orthogonal transform unit 214 performs processing to which the above-described present technology is applied, as an inverse orthogonal transform unit and an operation unit. Furthermore, the decoding unit 212 performs processing to which the above-described present technology is applied, as a decoding unit. Therefore, the image decoding device 200 can suppress an increase in the memory capacity required for the inverse orthogonal transform.

10 [0617]

Furthermore, a configuration of an inverse orthogonal transform unit 214 is similar to the case of the first embodiment. Then, in the inverse orthogonal transform unit 214, an inverse primary transform unit 253 performs processing to which the above-described present technology is applied, as an inverse orthogonal transform unit and an operation unit. That is, the inverse orthogonal transform unit performs inverse primary transform for an inverse secondary transform result, using a transformation matrix serving as a base, and the operation unit permutes an obtained inverse primary transform result. Therefore, an increase in the memory capacity required for the inverse primary transform can be suppressed.

25 [0618]

Note that the above-described inverse primary transform unit 253 performs the inverse primary vertical transform and the inverse primary horizontal transform as the inverse primary transform. That is, the inverse orthogonal transform unit performs, for the inverse secondary transform result, vertical inverse one-

dimensional orthogonal transform using a second transformation matrix for vertical inverse one-dimensional orthogonal transform serving as a base and further, horizontal inverse one-dimensional orthogonal transform using a second transformation matrix for horizontal inverse one-dimensional orthogonal transform serving as a base, as the inverse primary transform. Therefore, an increase in the memory capacity required for the primary transform in which the vertical inverse one-dimensional orthogonal transform and the horizontal inverse one-dimensional orthogonal transform are performed can be suppressed.

[0619]

Note that a flow of image decoding processing executed by the image decoding device 200 is similar to the case of the first embodiment. That is, in the image decoding processing of the above flow, processing to which the above-described present technology is applied is performed as processing of step S202. Furthermore, processing to which the above-described present technology is applied is performed as processing of step S204. Therefore, by executing the image decoding processing, an increase in the memory capacity required for the inverse orthogonal transform can be suppressed.

[0620]

Furthermore, a flow of inverse orthogonal transform processing executed in step S204 of the image decoding processing is similar to the case of the first embodiment. That is, in the inverse orthogonal transform processing of the above flow, processing to which the above-described present technology is applied is performed as

processing of step S233. Therefore, by executing the inverse orthogonal transform processing, an increase in the memory capacity required for the inverse primary transform can be suppressed.

5 [0621]

<3-2. Example 2-1>

<Concept>

Next, each derivation example described with reference to Fig. 67 will be described in more detail.

10 First, in the table illustrated in Fig. 67, the operation of the first row example from the top except the uppermost row of item names focuses on an axial symmetric property between one-dimensional orthogonal transforms in a certain direction (one or both of horizontal direction
15 and vertical direction) in two different two-dimensional orthogonal transforms.

[0622]

In this case, the operation unit flips the prediction residual. The orthogonal transform unit
20 orthogonally transforms the flipped prediction residual, using the base transformation matrix (transformation matrix of DST7). By doing so, the orthogonal transform of the prediction residual using the transformation matrix of FlipDST7 (see <2-2. Example 1-1>) that is the
25 flipped transformation matrix of DST7 can be substituted.

[0623]

This similarly applies to the inverse orthogonal transform. In this case, the inverse orthogonal transform unit inversely orthogonally transforms the
30 coefficient data, using the base transformation matrix (transformation matrix of DST7), and the operation unit

flips the inverse orthogonal transform result. By doing so, the inverse orthogonal transform of the coefficient data (transform coefficient Coeff_IQ) using the transformation matrix of FlipDST7 that is the flipped transformation matrix of DST7 can be substituted.
[0624]

By applying such an operation for the prediction residual (including the inverse orthogonal transform result), it becomes unnecessary to prepare the transformation matrix of FlipDST7 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced, similarly to the case of the first embodiment.
[0625]

In this case, as illustrated in the table in Fig. 68, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to four types. Therefore, the total LUT size can be about 47 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.
[0626]

Furthermore, since the two-dimensional flip operation for the prediction residual (including the inverse orthogonal transform result) can be performed by a collective operation, the number of flip operations can be reduced, as compared with the case of the orthogonal

transform/inverse orthogonal transform using the transformation matrix of FlipDST7. Therefore, an increase in the processing amount of the orthogonal transform/inverse orthogonal transform (the orthogonal transform/inverse orthogonal transform can be more easily performed).

[0627]

Furthermore, by performing the orthogonal transform/inverse orthogonal transform involving such an operation for the prediction residual (including the inverse orthogonal transform result), similar coding efficiency to the case of the orthogonal transform/inverse orthogonal transform using the transformation matrix of FlipDST7 can be obtained.

[0628]

<Primary Transform Unit>

Next, configurations, processing, and the like for performing such processing will be described. Fig. 69 is a block diagram illustrating a main configuration example of the primary transform unit 152 in this case. As illustrated in Fig. 69, the primary transform unit 152 in this case includes a prediction residual permutation operation unit 551, in addition to the configuration illustrated in Fig. 22.

[0629]

A primary transform selection unit 311 supplies a derived transform type identifier TrTypeIdxH of primary horizontal transform and a derived transform type identifier TrTypeIdxV of primary vertical transform to the prediction residual permutation operation unit 551.

[0630]

The prediction residual permutation operation unit 551 uses a prediction residual D supplied from a switch 151, the transform type identifier $TrTypeIdxH$ of primary horizontal transform, and the transform type identifier $TrTypeIdxV$ of primary vertical transform as inputs. The prediction residual permutation operation unit 551 flips the prediction residual D according to the transform type identifier $TrTypeIdxH$ and the transform type identifier $TrTypeIdxV$. The prediction residual permutation operation unit 551 supplies a flipped prediction residual D_{flip} to a primary horizontal transform unit 312.

[0631]

In the primary transform unit 152 having the above configuration, the prediction residual permutation operation unit 551 performs processing to which the above-described present technology is applied, as an operation unit. The primary horizontal transform unit 312 and a primary vertical transform unit 313 perform processing to which the above-described present technology is applied as orthogonal transform units.

[0632]

That is, the primary horizontal transform unit 312 performs horizontal one-dimensional orthogonal transform for the flipped prediction residual D_{flip} , using the transformation matrix for horizontal one-dimensional orthogonal transform serving as a base, as an orthogonal transform unit. Therefore, the primary horizontal transform unit 312 can suppress an increase in the memory capacity required for the horizontal one-dimensional orthogonal transform.

[0633]

Furthermore, the primary vertical transform unit 313 performs vertical one-dimensional orthogonal transform for a transform coefficient Coeff_Ph0r after primary horizontal transform, using the transformation matrix for vertical one-dimensional orthogonal transform serving as a base, as an orthogonal transform unit. Therefore, the primary vertical transform unit 313 can suppress an increase in the memory capacity required for the vertical one-dimensional orthogonal transform.

10 [0634]

<Flow of Primary Transform Processing>

Next, an example of a flow of processing performed by the above-described configuration will be described.

Next, an example of a flow of the primary transform processing executed in step S132 in Fig. 14 in this case will be described with reference to the flowchart in Fig. 70.

[0635]

When the primary transform processing is started, in step S561, the primary transform selection unit 311 of the primary transform unit 152 selects the transform type identifier TrTypeIdxH (or the transform type TrTypeH) of primary horizontal transform and the transform type identifier TrTypeIdxV (or the transform type TrTypeV) of primary vertical transform.

[0636]

In step S562, the prediction residual permutation operation unit 551 executes prediction residual permutation operation processing, and flips the prediction residual D according to the transform type identifier TrTypeIdxH and the transform type identifier

TrTypeIdxV obtained in step S561 to derive the flipped prediction residual D_{flip} .

[0637]

In step S563, the primary horizontal transform unit
5 312 performs primary horizontal transform processing to perform primary horizontal transform for the flipped prediction residual D_{flip} obtained in step S562, using a transformation matrix corresponding to the transform type identifier TrTypeIdxH of primary horizontal transform
10 obtained in step S561.

[0638]

In step S564, the primary vertical transform unit
313 performs primary vertical transform processing to perform primary vertical transform for the transform
15 coefficient Coeff_Phor after primary horizontal transform obtained in the processing in step S563, using a transformation matrix corresponding to the transform type identifier TrTypeIdxV of primary vertical transform obtained in step S561.

20 [0639]

When the processing in step S564 ends, the primary transform processing ends and the processing returns to Fig. 14.

[0640]

25 In the above primary transform processing, processing to which the above-described present technology is applied is performed as each processing of step S562 to S564. Therefore, by executing the primary transform processing, an increase in the memory capacity
30 required for the primary horizontal transform processing and the primary vertical transform processing can be

suppressed.

[0641]

<Flow of Prediction Residual Permutation Operation Processing>

5 A flow of the prediction residual permutation operation processing executed in step S562 in Fig. 70 will be described with reference to the flowchart in Fig. 71.

[0642]

10 When the prediction residual permutation operation processing is started, in step S581, the prediction residual permutation operation unit 551 derives a flip flag FlipFlagH corresponding to the transform type identifier TrTypeIdxH (or the transform type TrTypeH) of
15 primary horizontal transform and a flip flag FlipFlagV corresponding to the transform type identifier TrTypeIdxV (or the transform type TrTypeV) of primary vertical transform. At that time, the prediction residual permutation operation unit 551 obtains the flip flag
20 FlipFlagH and the flip flag FlipFlagV by reference to a correspondence table (LUT_TrTypeIdxToFlipFlag) as illustrated in Fig. 72, for example. Derivation of these flip flags can be expressed as, for example, the following expression (62) and expression (63).

25 [0643]

[Math. 54]

$$\begin{aligned} \text{FlipFlagH} &= \text{LUT_TrTypeIdxToFlipFlag}[\text{TrTypeIdxH}] \quad \cdot \cdot \cdot (62) \\ \text{FlipFlagV} &= \text{LUT_TrTypeIdxToFlipFlag}[\text{TrTypeIdxV}] \quad \cdot \cdot \cdot (63) \end{aligned}$$

[0644]

30 In step S582, the prediction residual permutation operation unit 551 determines whether or not the flip

flag FlipFlagH and the flip flag FlipFlagV derived in step S581 satisfy a condition expressed in the following expression (64).

[0645]

5 [Math. 55]

$$\text{FlipFlagH} \stackrel{\text{true}}{=} \text{T} \ \&\& \ \text{FlipFlagV} \stackrel{\text{false}}{=} \text{F} \quad \cdot \cdot \cdot \quad (64)$$

[0646]

In a case where it is determined that the condition is satisfied, that is, in a case where FlipFlagH is determined to be true (1) and FlipFlagV is determined to be false (0), the processing proceeds to step S583.

[0647]

In step S583, the prediction residual permutation operation unit 551 horizontally flips the prediction residual D to obtain the flipped prediction residual D_{flip} . When this operation is expressed as a matrix, the operation can be expressed as the following expression (65).

[0648]

20 [Math. 56]

$$D_{\text{flip}} = \text{FlipH}(D) = D \times J \quad \cdot \cdot \cdot \quad (65)$$

[0649]

Here, $\times$ is an operator representing a matrix product, FlipH (X) is an operator representing a flip operation for a matrix X in the horizontal direction, and a flip matrix J corresponding to FlipV (·) is a matrix obtained by right-left inverting a unit matrix I of N rows and N columns. Furthermore, in a case of expressing this operation as an operation for each element, the prediction residual permutation operation unit 551 sets

an $(i, N - 1 - j)$ component of the prediction residual D as the i -row j -column component $((i, j)$ component) of the flipped prediction residual D_{flip} , as in the following expression (66).

```

5  [0650]
    [Math. 57]
    
$$D_{\text{flip}}[i, j] = D[i, N-1-j]$$

    for  $i=0, \dots, M-1, j=0, \dots, N-1$ 
    . . . (66)
    [0651]
```

Note that the size of the prediction residual D is
 10 a width N and a height M (M rows and N columns). N and M satisfy the following expressions (67) and (68).

```

    [0652]
    [Math. 58]
    
$$N = 1 \ll \log_2 \text{TBWSize} \quad \dots \quad (67)$$

    
$$M = 1 \ll \log_2 \text{TBHSize} \quad \dots \quad (68)$$

15  [0653]
```

When the processing in step S583 ends, the processing proceeds to step S588. Furthermore, in step S582, in a case where it is determined that the condition is not satisfied, that is, in a case where FlipFlagH is
 20 determined to be false (0) or FlipFlagV is determined to be true (1), the processing proceeds to step S584.

```
[0654]
```

In step S584, the prediction residual permutation operation unit 551 determines whether or not the flip
 25 flag FlipFlagH and the flip flag FlipFlagV derived in step S581 satisfy a condition expressed in the following expression (69).

```
[0655]
```

[Math. 59]

$\text{FlipFlagH} == \text{F} \ \&\& \ \text{FlipFlagV} == \text{T} \quad \cdot \cdot \cdot \quad (69)$

[0656]

In a case where it is determined that the condition
 5 is satisfied, that is, in a case where FlipFlagH is
 determined to be false (0) and FlipFlagV is determined to
 be true (1), the processing proceeds to step S585.

[0657]

In step S585, the prediction residual permutation
 10 operation unit 551 vertically flips the prediction
 residual D to obtain the flipped prediction residual D_{flip} .
 When this operation is expressed as a matrix, the
 operation can be expressed as the following expression
 (70).

15 [0658]

[Math. 60]

$D_{\text{flip}} = \text{FlipV}(D) = J \times D \quad \cdot \cdot \cdot \quad (70)$

[0659]

Here, $\times$ is an operator representing a matrix
 20 product, FlipV (X) is an operator representing a flip
 operation for a matrix X in the vertical direction, and a
 flip matrix J corresponding to FlipV (·) is a matrix
 obtained by right-left inverting a unit matrix I of M
 rows and M columns. Furthermore, in a case of expressing
 25 this operation as an operation for each element, the
 prediction residual permutation operation unit 551 sets
 an (M - 1 - i, j) component of the prediction residual D
 as the i-row j-column component ((i, j) component) of the
 flipped prediction residual D_{flip} , as in the following

expression (71). Note that the size of the prediction residual D is a width N and a height M (M rows and N columns), and N and M satisfy the above-described expressions (67) and (68).

5 [0660]

[Math. 61]

$D_{flip}[i, j] = D[M-1-i, j]$
 for $i=0, \dots, N-1, j=0, \dots, M-1$
 . . . (71)

[0661]

When the processing in step S585 ends, the
 10 processing proceeds to step S588. Furthermore, in step S584, in a case where it is determined that the condition is not satisfied, that is, in a case where FlipFlagH is determined to be true (1) or FlipFlagV is determined to be false (0), the processing proceeds to step S586.

15 [0662]

In step S586, the prediction residual permutation operation unit 551 determines whether or not the flip flag FlipFlagH and the flip flag FlipFlagV derived in step S581 satisfy a condition expressed in the following
 20 expression (72).

[0663]

[Math. 62]

$FlipFlagH \neq T \ \&\& \ FlipFlagV \neq T \ \dots \ (72)$

[0664]

25 In a case where it is determined that the condition is satisfied, that is, in a case where FlipFlagH is determined to be true (1) and FlipFlagV is determined to be true (1), the processing proceeds to step S587.

[0665]

In step S587, the prediction residual permutation operation unit 551 horizontally and vertically flips the prediction residual D to obtain the flipped prediction residual D_{flip} . When this operation is expressed as a matrix, the operation can be expressed as the following expression (73).

[0666]

[Math. 63]

$$D_{\text{flip}} = \text{FlipH}(\text{FlipV}(D)) = J_2 \times D \times J_1 \quad \cdot \cdot \cdot \quad (73)$$

[0667]

Here, $\times$ is an operator representing a matrix product, a flip matrix J_1 is a matrix obtained by inverting a unit matrix I of N rows and N columns, and a flip matrix J_2 is a matrix obtained by inverting a unit matrix I of M rows and M columns. Note that the flip matrix J is equivalent to the flip matrix J even when transposed ($J^T = J$). Furthermore, in a case of expressing this operation as an operation for each element, the prediction residual permutation operation unit 551 sets an $(M - i - 1, N - j - 1)$ component of the prediction residual D as the i-row j-column component $((i, j)$ component) of the flipped prediction residual D_{flip} , as in the following expression (74). Note that the size of the prediction residual D is a width N and a height M (M rows and N columns), and N and M satisfy the above-described expressions (67) and (68).

[0668]

[Math. 64]

```

Dflip[i, j] = D[M-1-i, N-1-j]
for i=0, ..., M-1, j=0, ..., N-1
  * * * (74)

```

[0669]

When the processing in step S587 ends, the processing proceeds to step S588.

5 [0670]

In step S588, the prediction residual permutation operation unit 551 sets the flipped prediction residual D_{flip} as the prediction residual D , as in the following expression (75). That is, the primary horizontal

10 transform processing (step S563 in Fig. 70) is performed for the prediction residual D (flipped prediction residual D_{flip}).

[0671]

[Math. 65]

15 $D = D_{\text{flip}} \quad * * * (75)$

[0672]

When the processing in step S588 ends, the prediction residual permutation operation processing ends and the processing returns to Fig. 70. Furthermore, in

20 step S586, in a case where it is determined that the condition is not satisfied, that is, in a case where FlipFlagH is determined to be false (0) or FlipFlagV is determined to be false (0), the prediction residual permutation operation processing ends and the processing

25 returns to Fig. 70.

[0673]

By performing the prediction residual permutation operation processing as described above, the prediction

residual permutation operation unit 551 can flip the prediction residual in the spatial symmetric direction of the two-dimensional orthogonal transform (including the prediction residual) (that is, the direction

5 corresponding to the transform type identifier TrTypeIdxH (or the transform type TrTypeH) of primary horizontal transform and the transform type identifier TrTypeIdxV (or the transform type TrTypeV) of primary vertical transform.

10 [0674]

<Inverse Primary Transform Unit>

Next, a configuration of the image decoding device 200 in the case of the present embodiment will be described. Fig. 73 is a block diagram illustrating a
15 main configuration example of the inverse primary transform unit 253 (Fig. 16) in this case. As illustrated in Fig. 73, the inverse primary transform unit 253 in this case includes a prediction residual permutation operation unit 552, in addition to the
20 configuration illustrated in Fig. 34.

[0675]

An inverse primary transform selection unit 411 supplies the derived transform type identifier TrTypeIdxV of inverse primary vertical transform and the derived
25 transform type identifier TrTypeIdxH of inverse primary vertical transform to the prediction residual permutation operation unit 551. Furthermore, an inverse primary horizontal transform unit 413 supplies the derived transform coefficient Coeff_IPhor after inverse primary
30 horizontal transform to the prediction residual permutation operation unit 552.

[0676]

The prediction residual permutation operation unit 552 uses the transform coefficient `Coeff_IPhor` after inverse primary horizontal transform supplied from the inverse primary horizontal transform unit 413, and the transform type identifier `TrTypeIdxH` (or the transform type `TrTypeH`) of primary horizontal transform and the transform type identifier `TrTypeIdxV` (or the transform type `TrTypeV`) of primary vertical transform as inputs.

The prediction residual permutation operation unit 552 flips the transform coefficient `Coeff_IPhor` after inverse primary horizontal transform according to the transform type identifier `TrTypeIdxH` (or the transform type `TrTypeH`) and the transform type identifier `TrTypeIdxV` (or the transform type `TrTypeV`). The prediction residual permutation operation unit 552 outputs a flipped transform coefficient `Coeff_IPhor` (transform coefficient `Coeff_IP` after inverse primary transform) to the outside of the inverse primary transform unit 253 (supplies the same to a calculation unit 215) as the prediction residual D' .

[0677]

In the inverse primary transform unit 253 having the above configuration, the prediction residual permutation operation unit 552 performs processing to which the above-described present technology is applied, as an operation unit. An inverse primary vertical transform unit 412 and the inverse primary horizontal transform unit 413 perform processing to which the above-described present technology is applied, as inverse orthogonal transform units, respectively.

[0678]

That is, the inverse primary vertical transform unit 412 performs vertical inverse one-dimensional orthogonal transform for a transform coefficient Coeff_IS
 5 after inverse secondary transform, using a transformation matrix for vertical inverse one-dimensional orthogonal transform serving as a base, as an inverse orthogonal transform unit. By flipping the transform coefficient by the prediction residual permutation operation unit 552
 10 afterward, as described above, vertical inverse one-dimensional orthogonal transform using another transformation matrix can be substituted. Therefore, the inverse primary vertical transform unit 412 can suppress an increase in the memory capacity required for the
 15 vertical inverse one-dimensional orthogonal transform.

[0679]

Furthermore, the inverse primary horizontal transform unit 413 performs horizontal inverse one-dimensional orthogonal transform for a transform
 20 coefficient Coeff_IPver after inverse primary vertical transform, using a transformation matrix for horizontal inverse one-dimensional orthogonal transform serving as a base, as an inverse orthogonal transform unit. By flipping the transform coefficient by the prediction
 25 residual permutation operation unit 552 afterward, as described above, vertical horizontal inverse one-dimensional orthogonal transform using another transformation matrix can be substituted. Therefore, the inverse primary horizontal transform unit 413 can
 30 suppress an increase in the memory capacity required for the horizontal inverse one-dimensional orthogonal

transform.

[0680]

<Flow of Inverse Primary Transform Processing>

Next, an example of a flow of processing performed
5 by the above-described configuration will be described.
Next, an example of a flow of inverse primary transform
processing executed in step S233 in Fig. 18 in this case
will be described with reference to the flowchart in Fig.
74.

10 [0681]

When the inverse primary transform processing is
started, in step S601, the inverse primary transform
selection unit 411 of the inverse primary transform unit
253 performs the inverse primary transform selection
15 processing to select the transform type identifier
TrTypeIdxV (or the transform type TrTypeV) of inverse
primary vertical transform and the transform type
identifier TrTypeIdxH (or the transform type TrTypeH) of
inverse primary horizontal transform.

20 [0682]

In step S602, the inverse primary vertical
transform unit 412 performs the inverse primary vertical
transform processing to perform vertical inverse one-
dimensional orthogonal transform for the transform
25 coefficient Coeff_IS after inverse secondary transform,
using the transformation matrix corresponding to the
transform type identifier TrTypeIdxV of inverse primary
vertical transform obtained in step S601.

[0683]

30 In step S603, the inverse primary horizontal
transform unit 413 performs the inverse primary

horizontal transform processing to perform horizontal
inverse one-dimensional orthogonal transform for the
transform coefficient `Coeff_IPver` after inverse primary
vertical transform derived in step S602, using the
5 transformation matrix corresponding to the transform type
identifier `TrTypeIdxH` of inverse primary horizontal
transform obtained in step S601.

[0684]

10 In step S604, the prediction residual permutation
operation unit 552 performs the prediction residual
permutation operation processing, and flips the transform
coefficient `Coeff_IPhor` after inverse primary horizontal
transform derived in step S603, using the transform type
identifier `TrTypeIdxV` of inverse primary vertical
15 transform and the transform type identifier `TrTypeIdxH` of
inverse primary horizontal transform obtained in step
S601, to obtain the transform coefficient `Coeff_IP`
(prediction residual D') after inverse primary transform.

[0685]

20 The prediction residual permutation operation
processing is performed by a flow similar to the example
described with reference to the flowchart in Fig. 71.
Therefore, the description regarding Fig. 71 can be
applied to the processing in step S604, for example, by
25 appropriately replacing the prediction residual D with
the transform coefficient `Coeff_IPhor`, and thus
description thereof is omitted.

[0686]

30 When the processing in step S604 ends, the inverse
primary transform processing ends and the processing
returns to Fig. 18.

[0687]

In the above inverse primary transform processing, processing to which the above-described present technology is applied is performed as each processing of step S602 or S604. Therefore, by executing the inverse primary transform processing, an increase in the memory capacity required for the inverse primary vertical transform processing or the inverse primary horizontal transform processing can be suppressed.

10 [0688]

<3-3. Example 2-2>

<Concept>

Next, the second row example from the top except the uppermost row of item names in the table illustrated in Fig. 67 will be described. This operation focuses on the similarity between one-dimensional orthogonal transforms in a certain direction (one or both of the horizontal direction and vertical direction) in the two different two-dimensional orthogonal transforms. More specifically, attention is paid to an axial symmetric property in even-numbered row vectors and a point-symmetric property in odd-numbered row vectors.

[0689]

In this case, the derivation unit derives the second transformation matrix, using the first transformation matrix, the operation unit permutes the prediction residual, and the orthogonal transform unit orthogonally transforms the permuted prediction residual, using the second transformation matrix. For example, the transformation matrix of DST7 is used as the base transformation matrix, the derivation unit inverts the

sign of an odd-numbered row vector of the transformation matrix of DST7, the operation unit flips the prediction residual, and the orthogonal transform unit orthogonally transforms the flipped prediction residual, using the transformation matrix of DST7 with the inverted sign of the odd-numbered row vector. By doing so, the orthogonal transform of the prediction residual using the transformation matrix (that is, DCT8 (see <2-3. Example 1-2>)) that is obtained by flipping the transformation matrix of DST7 and inverting the sign of an odd-numbered row vector of the flipped transformation matrix can be substituted.

[0690]

This similarly applies to the inverse orthogonal transform. In this case, the derivation unit derives the second transformation matrix, using the first transformation matrix, the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data, using the second transformation matrix, and the operation unit permutes the inverse orthogonal transform result. For example, the transformation matrix of DST7 is used as the base transformation matrix, the derivation unit inverts the sign of an odd-numbered row vector of the transformation matrix of DST7, the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data, using the transformation matrix of DST7 with the inverted sign of the odd-numbered row vector, and the operation unit flips an inverse orthogonal transform result. By doing so, the inverse orthogonal transform of the coefficient data (transform coefficient Coeff_IQ) using the transformation matrix (that is, DCT8

(see <2-3. Example 1-2>)) that is obtained by flipping the transformation matrix of DST7 and inverting the sign of an odd-numbered row vector of the flipped transformation matrix can be substituted.

5 [0691]

By applying such an operation for the prediction residual (including the inverse orthogonal transform result), it becomes unnecessary to prepare the transformation matrix of DCT8 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced, similarly to the case of the first embodiment.

[0692]

15 In this case, as illustrated in the table in Fig. 75, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to four types. Therefore, the total LUT size can be about 47 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.

25 [0693]

<Transformation Matrix Derivation Unit>

In this case, the primary transform unit 152 has a similar configuration to the case of <3-2. Example 2-1>. Furthermore, the primary horizontal transform unit 312 has a similar configuration to the case of <3-2. Example 2-1>. Furthermore, the primary vertical transform unit

313 has a similar configuration to the case of <3-2. Example 2-1>. Note that an inner configuration of a transformation matrix derivation unit 321 of the primary horizontal transform unit 312 and an inner configuration of a transformation matrix derivation unit 351 of the primary vertical transform unit 313 are different from the case of <3-2. Example 2-1>.

[0694]

Fig. 76 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 321 included in the primary horizontal transform unit 312 in this case. As illustrated in Fig. 76, the transformation matrix derivation unit 321 in this case includes a transformation matrix LUT 331 and a sign inversion unit 561. Note that, in Fig. 76, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 321, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0695]

The sign inversion unit 561 is a processing unit similar to the sign inversion unit 501 (Fig. 46), and uses a transformation matrix T of N rows and N columns as an input, inverts the sign of a predetermined portion of the transformation matrix T , and performs sign inversion, and outputs a transformation matrix T_{InvSign} after the sign inversion. In this derivation example (Example 2-2), the transformation matrix derivation unit 321 inverts the sign of an odd-order row vector of the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 331 via the

sign inversion unit 561 to derive the transformation matrix T_{InvSign} . The transformation matrix derivation unit 321 outputs the transformation matrix T_{InvSign} to the outside of the transformation matrix derivation unit 321 (supplies the same to a matrix calculation unit 322) as a transformation matrix T_H .

[0696]

As described above, the transformation matrix derivation unit 321 can implement the "operation of the base transformation matrix T_{base} " in primary horizontal transform, which is the second row example from the top of the table illustrated in Fig. 67, using the sign inversion unit 561.

[0697]

15 <Transformation Matrix Derivation Unit>

Fig. 77 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 351 included in the primary vertical transform unit 313 in this case. As illustrated in Fig. 77, the transformation matrix derivation unit 351 in this case includes a transformation matrix LUT 361 and a sign inversion unit 562, similarly to the case of the transformation matrix derivation unit 321. Note that, in Fig. 77, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 351, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0698]

The sign inversion unit 562 is a processing unit similar to the sign inversion unit 502 (Fig. 47), and uses a transformation matrix T of N rows and N columns as

an input, inverts the sign of a predetermined portion of the transformation matrix T , and performs sign inversion, and outputs a transformation matrix T_{InvSign} after the sign inversion. In this derivation example (Example 2-2), the
 5 transformation matrix derivation unit 351 inverts the sign of an odd-order row vector of the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 361 via the sign inversion unit 562 to derive the transformation
 10 matrix T_{InvSign} . The sign inversion unit 562 outputs the transformation matrix T_{InvSign} to the outside of the transformation matrix derivation unit 351 (supplies the same to a matrix calculation unit 352) as a transformation matrix T_v .

15 [0699]

As described above, the transformation matrix derivation unit 351 can implement the "operation of the base transformation matrix T_{base} " in primary vertical transform, which is the second row example from the top
 20 of the table illustrated in Fig. 67, using the sign inversion unit 562.

[0700]

<Flow of Transformation Matrix Derivation Processing>

25 Since the primary transform processing in this case is performed by a flow similar to the case of <3-2. Example 2-1> (performed by a flow similar to the flowchart in Fig. 70), description thereof is basically omitted.

30 [0701]

Note that, in this case, in step S581 of the

prediction residual permutation operation processing (Fig. 71) executed in step S562 in Fig. 70, the prediction residual permutation operation unit 551 obtains a flip flag FlipFlagH and a flip flag FlipFlagV by reference to a correspondence table (LUT_TrTypeIdxToFlipFlag) illustrated in Fig. 79, instead of the correspondence table illustrated in Fig. 72.

[0702]

Furthermore, in this case, the primary horizontal transform processing executed in step S563 in Fig. 70 is performed by a flow similar to the case of <2-2. Example 1-1> (performed by a flow similar to the flowchart in Fig. 28) but the transformation matrix derivation processing executed in step S321 in Fig. 28 is performed by a flow different from the case in Fig. 30.

[0703]

An example of a flow of the transformation matrix derivation processing executed by the transformation matrix derivation unit 321 in the primary horizontal transform unit 312 in step S321 in Fig. 28 in this case will be described with reference to the flowchart in Fig. 78.

[0704]

When the transformation matrix derivation processing is started, in step S621, the transformation matrix derivation unit 321 derives a base transformation matrix T_{base} of the base transform type corresponding to the transform type identifier TrTypeIdxH and the sign inversion flag InvSignFlag by reference to the correspondence table (LUT_TrTypeIdxToFlipFlag) illustrated in Fig. 79, for example.

[0705]

In step S622, the transformation matrix derivation unit 321 determines whether or not the sign inversion flag InvSignFlag derived in step S621 is true (1). In a case where the sign inversion flag InvSignFlag is determined to be false (0), the processing proceeds to step S623.

[0706]

In step S623, the transformation matrix derivation unit 321 reads the base transformation matrix T_{base} (DST7) derived in step S621 from the transformation matrix LUT 331, and sets the base transformation matrix T_{base} as the transformation matrix T_H . This can be expressed as in the following expression (76).

15 [0707]

[Math. 66]

$$T_H = T_{\text{base}} \cdot \cdot \cdot (76)$$

[0708]

When the processing in step S623 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28. Furthermore, in step S622, in a case where the sign inversion flag InvSignFlag is determined to be true (1), the processing proceeds to step S624.

25 [0709]

In step S624, the transformation matrix derivation unit 321 reads the base transformation matrix T_{base} (DST7) derived in step S621 from the transformation matrix LUT 331, inverts the sign of an odd-numbered row vector of the base transformation matrix T_{base} (DST7) via the sign

inversion unit 561, and sets the base transformation matrix T_{base} with the inverted sign as the transformation matrix T_H . This can be expressed as in the following expression (77).

5 [0710]

[Math. 67]

$$T_H = D \times T_{\text{base}} \quad \cdot \cdot \cdot \quad (77)$$

[0711]

Here, x is an operator representing a matrix product, and a sign inversion matrix D is a diagonal matrix including $\text{Diag}(1, -1, \dots, (-1)^{N-1})$.

10

[0712]

When the processing in step S624 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

15

[0713]

<Flow of Transformation Matrix Derivation Processing>

Since the flow of the transformation matrix derivation processing executed in step S361 (Fig. 32) of the primary vertical transform processing is similar to the case of the transformation matrix derivation processing executed in the primary horizontal transform described with reference to the flowchart in Fig. 78, description thereof is omitted. In the description in Fig. 78, the transform type identifier TrTypeIdxH is simply replaced with the transform type identifier TrTypeIdxV , and the transformation matrix T_H is simply replaced with the transformation matrix T_V .

25

30 [0714]

<Transformation Matrix Derivation Unit>

Next, a configuration of the image decoding device 200 in the case will be described. Since configurations of the inverse primary transform unit 253, the inverse primary vertical transform unit 412, the inverse primary horizontal transform unit 413, and the like included in the image decoding device 200 in this case are similar to the case of <2-2. Example 1-1>, description thereof is omitted.

10 [0715]

Fig. 80 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit 421 (a transformation matrix derivation unit 421 included in the inverse primary vertical transform unit 412) in this case. As illustrated in Fig. 80, the transformation matrix derivation unit 421 in this case includes a transformation matrix LUT 431 and a sign inversion unit 571, similarly to the transformation matrix derivation unit 321. Note that, in Fig. 80, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 421, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0716]

25 The transformation matrix LUT 431 is similar to that in the case of Fig. 36. The sign inversion unit 571 is a processing unit similar to the sign inversion unit 501, and uses a transformation matrix T of N rows and N columns as an input, inverts the sign of a predetermined portion of the transformation matrix T , and outputs the transformation matrix T_{InvSign} after sign inversion. In

30

this derivation example (Example 2-2), the transformation matrix derivation unit 421 inverts the sign of an odd-order row vector of the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 431 via the sign inversion unit 571 to derive the transformation matrix $T_{InvSign}$. The sign inversion unit 571 outputs the transformation matrix $T_{InvSign}$ to the outside of the transformation matrix derivation unit 421 (supplies the same to the matrix calculation unit 422) as the transformation matrix T_v .

[0717]

As described above, the transformation matrix derivation unit 421 can implement the "operation of the base transformation matrix T_{base} " in inverse primary vertical transform, which is the second row example from the top of the table illustrated in Fig. 67, using the sign inversion unit 571.

[0718]

<Transformation Matrix Derivation Unit>

Fig. 81 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 451 included in the inverse primary horizontal transform unit 413 in this case. As illustrated in Fig. 81, the transformation matrix derivation unit 451 in this case includes a transformation matrix LUT 461 and a sign inversion unit 572, similarly to the transformation matrix derivation unit 421. Note that, in Fig. 81, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 451, arbitrary data can be transferred between arbitrary processing units

(processing blocks).

[0719]

The transformation matrix LUT 461 is similar to that in the case of Fig. 80. The sign inversion unit 572 is a processing unit similar to the sign inversion unit 571, and uses a transformation matrix T of N rows and N columns as an input, inverts the sign of a predetermined portion of the transformation matrix T , and performs sign inversion, and outputs a transformation matrix T_{InvSign} after the sign inversion. In this derivation example (Example 2-2), the transformation matrix derivation unit 451 inverts the sign of an odd-order row vector of the base transformation matrix T_{base} of N rows and N columns selected in the transformation matrix LUT 461 via the sign inversion unit 572 to derive the transformation matrix T_{InvSign} . The sign inversion unit 572 outputs the transformation matrix T_{InvSign} to the outside of the transformation matrix derivation unit 451 (supplies the same to a matrix calculation unit 452) as the transformation matrix T_H .

[0720]

As described above, the transformation matrix derivation unit 451 can implement the "operation of the base transformation matrix T_{base} " in inverse primary vertical transform, which is the second row example from the top of the table illustrated in Fig. 67, using the sign inversion unit 512.

[0721]

<Flow of Transformation Matrix Derivation Processing>

Note that the transformation matrix derivation unit

421 and the transformation matrix derivation unit 451 perform the transformation matrix derivation processing by a flow similar to the case described with reference to the flowchart in Fig. 78, and thus description of the processing is omitted.

[0722]

<3-4. Example 2-3>

<Concept>

Next, the third row example from the top except the uppermost row of item names in the table illustrated in Fig. 67 will be described. This example focuses on axial symmetry between one-dimensional orthogonal transforms in a certain direction (one or both of the horizontal direction and vertical direction) in two different two-dimensional orthogonal transform, similarly to the first row example from the top.

[0723]

In this case, the operation unit flips the prediction residual. The orthogonal transform unit orthogonally transforms the flipped prediction residual, using the base transformation matrix (transformation matrix of DCT8). By doing so, the orthogonal transform of the prediction residual using the transformation matrix of FlipDCT8 (see <2-4. Example 1-3>) that is the flipped transformation matrix of DCT8 can be substituted.

[0724]

This similarly applies to the inverse orthogonal transform. In this case, the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data, using the base transformation matrix (transformation matrix of DCT8), and the operation unit

flips the inverse orthogonal transform result. By doing so, the inverse orthogonal transform of the coefficient data (transform coefficient Coeff_IQ) using the transformation matrix of FlipDCT8 that is the flipped transformation matrix of DCT8 can be substituted.
[0725]

By applying such an operation for the prediction residual (including the inverse orthogonal transform result), it becomes unnecessary to prepare the transformation matrix of FlipDCT8 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced, similarly to the case of the first embodiment.
[0726]

Note that, in this case, in the prediction residual permutation operation processing (Fig. 71), the correspondence table (LUT_TrTypeIdxToFlipFlag) illustrated in Fig. 82 is simply referred to instead of the correspondence table illustrated in Fig. 72.
[0727]

Furthermore, processing is similar even in the case of inverse orthogonal transform, and processing similar to the case of orthogonal transform is only required to be performed with a configuration similar to the case of orthogonal transform.
[0728]

<3-5. Example 2-4>

<Concept>

Next, the fourth row example from the top except the uppermost row of item names in the table illustrated

in Fig. 67 will be described. This example focuses on similarity between one-dimensional orthogonal transforms in a certain direction (one or both of horizontal direction and vertical direction) (an axial symmetric property of even-numbered row vectors, and a point-symmetric property of odd-numbered row vectors) in two different two-dimensional orthogonal transforms, similarly to the second row example from the top. This operation is the same as the operation of the second row from the top.

[0729]

In this case, the derivation unit derives the second transformation matrix from the first transformation matrix, and the operation unit flips the prediction residual D. The orthogonal transform unit orthogonally transforms the flipped prediction residual, using the transformation matrix of DST7 that is obtained by inverting the sign of the base transformation matrix (transformation matrix of DCT8). By doing so, the orthogonal transform of the prediction residual using the transformation matrix of DST7 (see <2-5. Example 1-4>) that is obtained by flipping the transformation matrix of DCT8 and inverting the sign can be substituted.

[0730]

This similarly applies to the inverse orthogonal transform. In this case, the derivation unit inverts the sign of an odd-numbered row vector of the transformation matrix of DCT8, the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data, using the transformation matrix of DCT8 with the inverted sign of the odd-numbered row vector (that is, DST7), and

the operation unit flips an inverse orthogonal transform result. By doing so, the inverse orthogonal transform of the coefficient data (transform coefficient Coeff_IQ) using the transformation matrix of DST7 that is obtained
 5 by flipping the transformation matrix of DCT8 and inverting the sign.

[0731]

By applying such an operation for the prediction residual (including the inverse orthogonal transform
 10 result), it becomes unnecessary to prepare the transformation matrix of DST7 as a candidate for a transformation matrix to be used for orthogonal transform/inverse orthogonal transform. That is, the number of unique transform types can be reduced,
 15 similarly to the case of the first embodiment.

[0732]

Note that, in this case, in the prediction residual permutation operation processing (Fig. 71), the correspondence table (LUT_TrTypeIdxToFlipFlag)
 20 illustrated in Fig. 83 is simply referred to instead of the correspondence table illustrated in Fig. 79.

[0733]

Furthermore, processing is similar even in the case of inverse orthogonal transform, and processing similar
 25 to the case of orthogonal transform is only required to be performed with a configuration similar to the case of orthogonal transform.

[0734]

<4. Third Embodiment>

30 <4-1. Common Concept>

<Derivation of Transformation Matrix Using

Submatrix>

In the first embodiment, derivation of another transformation matrix using a transformation matrix has been description. However, the embodiment is not limited thereto, and a submatrix that is a part of a transformation matrix is prepared, and the transformation matrix may be derived using the submatrix.

[0735]

That is, a transformation matrix may be derived using a submatrix configuring a part of the transformation matrix, a prediction residual of an image may be orthogonally transformed using the derived transformation matrix, and coefficient data obtained by orthogonally transforming the prediction residual may be encoded to generate a bit stream.

[0736]

For example, an image processing apparatus may include a derivation unit configured to derive a transformation matrix, using a submatrix configuring a part of the transformation matrix, an orthogonal transform unit configured to orthogonally transform a prediction residual of an image, using the transformation matrix derived by the derivation unit, and an encoding unit configured to encode coefficient data obtained by orthogonally transforming the prediction residual by the orthogonal transform unit to generate a bit stream.

[0737]

With the configuration, it is only required to hold the submatrix (it is not necessary to hold all the transformation matrices). Therefore, an increase in memory capacity required for the orthogonal transform can

be suppressed.

[0738]

Furthermore, a bit stream may be decoded to obtain coefficient data that is an orthogonally transformed prediction residual of an image, a transformation matrix may be derived using a submatrix configuring a part of the transformation matrix, and the obtained coefficient data may be inversely orthogonally transformed using the derived transformation matrix.

10 [0739]

For example, the image processing apparatus may include a decoding unit configured to decode a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image, a derivation unit configured to derive a transformation matrix, using a submatrix configuring a part of the transformation matrix, and an inverse orthogonal transform unit configured to inversely orthogonally transform the coefficient data obtained by the decoding unit, using the transformation matrix derived by the derivation unit.

20 transformation matrix derived by the derivation unit.

[0740]

With the configuration, it is only required to hold the submatrix (it is not necessary to hold all the transformation matrices). Therefore, an increase in the memory capacity required for the inverse orthogonal transform can be suppressed.

25 memory capacity required for the inverse orthogonal transform can be suppressed.

[0741]

<Example of Relationship Between Submatrix and Whole Transformation Matrix>

30 More specific description will be given. For example, as illustrated in Fig. 84, in a case of a

transformation matrix 601 of DCT2, a waveform of an even-numbered row vector in a left portion and a waveform of an even-numbered row vector in a right portion are axially symmetric with each other. Furthermore,
5 waveforms of odd-numbered row vectors in the left portion of the transformation matrix 601 are point-symmetric with each other.

[0742]

Therefore, a submatrix constituting the right
10 portion of the transformation matrix 601 of DCT2 can be derived by flipping a submatrix constituting the left portion of the transformation matrix 601 of DCT2 in a horizontal direction (row direction) around a column direction passing through a center of the transformation
15 matrix 601, and inverting a sign of an odd-numbered row vector. That is, the transformation matrix 601 of DCT2 can be derived from the submatrix constituting the left portion.

[0743]

20 Furthermore, for example, in a case of a transformation matrix 602 of DST7, a waveform of an even-numbered row vector of an upper left triangular portion and a waveform of an even-numbered column vector of a lower right triangular portion are axially symmetric with
25 each other. Furthermore, a waveform of an odd-numbered row vector of the upper left triangular portion and a waveform of an odd-numbered column vector of the lower right triangular portion of the transformation matrix 602 are point-symmetric with each other.

30 [0744]

Therefore, a submatrix constituting the upper left

triangular portion of the transformation matrix 602 of DST7 can be derived by flipping a submatrix constituting the lower right triangular portion of the transformation matrix 602 of DST7 in an oblique direction from the upper left to lower right around a diagonal line connecting an upper right end and a lower left end of the transformation matrix 602, and inverting a sign of the odd-numbered column vector. That is, the transformation matrix 602 of DST7 can be derived from the submatrix constituting the upper left triangular portion.

[0745]

That is, the transformation matrix LUT is only required to store a submatrix of a whole transformation matrix, the submatrix being able to derive the whole transformation matrix, instead of storing the whole transformation matrix. For example, the derivation unit may derive the (whole) transformation matrix, using the submatrix stored in the lookup table (transformation matrix LUT). With the configuration, an increase in the UT size can be suppressed as compared with the case of storing the whole transformation matrices of DCT2 and DST7. That is, an increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform can be suppressed.

[0746]

Furthermore, flip may be performed, as described above, as the operation for deriving the whole transformation matrix from the submatrix. Furthermore, the flipping direction is arbitrary. For example, the derivation unit may derive the transformation matrix by flipping the submatrix around an axis of a predetermined

direction passing through a center of the transformation matrix and deriving a remaining submatrix of the transformation matrix. With the configuration, the whole transformation matrix can be derived from the submatrix
 5 by the simple operation.
 [0747]

Note that the flipping direction is arbitrary. For example, the derivation unit may flip the submatrix in a direction parallel to a row (the direction is also
 10 referred to as a row direction or a horizontal direction). Furthermore, for example, the derivation unit may flip the submatrix in a direction parallel to a column (the direction is also referred to as a column direction or a vertical direction). Moreover, for example, the
 15 derivation unit may flip the submatrix in a rotation direction around the center of the transformation matrix. Furthermore, for example, the derivation unit may flip the submatrix in an oblique direction around a diagonal line of the transformation matrix.

20 [0748]

With the configurations, another submatrix having a symmetric property in various directions with respect to a submatrix can be derived from the submatrix by a simple operation, for example.

25 [0749]

Furthermore, sign inversion may be performed, as described above, as the operation for deriving the whole transformation matrix from the submatrix. For example, the derivation unit may further invert the sign of an
 30 element of the flipped submatrix to derive the transformation matrix. With the configuration, various

whole transformation matrices can be derived from the submatrix by the simple operation.

[0750]

<Derivation Example>

5 Fig. 85 illustrates a list of derivation examples of a whole transformation matrix from a submatrix. Note that the submatrix to be used for derivation is also referred to as a base submatrix.

[0751]

10 In the table illustrated in Fig. 85, derivation of the first row example from the top except the uppermost row of item names focuses on an axial symmetric property with respect to a vertical axis.

[0752]

15 In this case, the submatrix is a left-half submatrix or a right-half submatrix of a transformation matrix, and the derivation unit derives the whole transformation matrix by flipping the submatrix in a row direction of the transformation matrix and further
20 inverting a sign of an odd-numbered row vector of the flipped submatrix to derive the right-half submatrix or the left-half submatrix of the transformation matrix (that is, a remaining submatrix of the transformation matrix).

25 [0753]

 For example, as illustrated in Fig. 86, the derivation unit flips a left-half submatrix 601A of the transformation matrix 601 having the transform type of DCT2 around an axis 601X in a direction (column
30 direction) parallel to a column passing through the transformation matrix 601 (that is, flip is made in the

horizontal direction). Moreover, the derivation unit inverts the sign of an odd-numbered row vector. By doing so, a right-half submatrix 601B of the transformation matrix 601 is derived. That is, the whole transformation matrix 601 is derived. When such derivation is expressed as an operation expression for each element, where a transformation matrix to be derived is a transformation matrix T of N rows and N columns, and a submatrix to be used for the derivation is C, the derivation can be expressed as the following expression (78).

[0754]

[Math. 68]

```

sign = (y%2==0) ? 1:-1
T[y,N-1-x] = sign * C[y,x]
for x=0,...,N/2-1, y=0,...,N-1
  . . . (78)

```

[0755]

Since the whole transformation matrix can be derived from the submatrix by applying such derivation, it becomes unnecessary to prepare the whole transformation matrix 601 of DCT2 as a candidate of a transformation matrix to be used for orthogonal transform/inverse orthogonal transform (it is only required to prepare the submatrix 601A). That is, the information amount of this candidate can be reduced to half. That is, an increase in the LUT size can be suppressed. Furthermore, by performing orthogonal transform/inverse orthogonal transform using the derived transformation matrix, the same coding efficiency as a case of using the transformation matrix 601 of DCT2 for the orthogonal transform/inverse orthogonal transform can

be obtained.

[0756]

Note that the transform type is not limited to the above-described example as long as the transformation
5 matrix to which such derivation is applied has an element having the above-described symmetric property. For example, the transform type may be DST2.

[0757]

Furthermore, in the above description, derivation
10 of the right-half submatrix (the whole transformation matrix) from the left-half submatrix has been described. However, the embodiment is not limited to the example, and for example, the left-half submatrix (the whole transformation matrix) may be derived from the right-half
15 submatrix.

[0758]

Furthermore, derivation of an example one row below the first row example (the second row example from the top) focuses on an axial symmetric property with respect
20 to a horizontal axis.

[0759]

In this case, the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation matrix, and the derivation unit derives the whole
25 transformation matrix by flipping the submatrix in a column direction of the transformation matrix, further inverting a sign of an odd-numbered column vector of the flipped submatrix, and deriving the lower-half submatrix or the upper-half submatrix of the transformation matrix
30 (that is, a remaining submatrix of the transformation matrix).

[0760]

For example, as illustrated in Fig. 87, the derivation unit flips an upper-half submatrix 611A of a transformation matrix 611 having the transform type of DST1 around an axis 611X in a direction (row direction) parallel to a row passing through a center of the transformation matrix 611 (that is, flip is made in the vertical direction). Moreover, the derivation unit inverts the sign of an odd-numbered column vector. By doing so, a lower-half submatrix 611B of the transformation matrix 611 is derived. That is, the whole transformation matrix 611 is derived. When such derivation is expressed as an operation expression for each element, where a transformation matrix to be derived is a transformation matrix T of N rows and N columns, and a submatrix to be used for the derivation is C, the derivation can be expressed as the following expression (79).

[0761]

20 [Math. 69]

```

sign = (x%2==0 ? 1:-1)
T[N-1-y,x] = sign * C[y,x]
for x=0,...,N-1 y=0,...,N/2-1
  . . . (79)

```

[0762]

Since the whole transformation matrix can be derived from the submatrix by applying such derivation, it becomes unnecessary to prepare the whole transformation matrix 611 of DST1 as a candidate of a transformation matrix to be used for orthogonal transform/inverse orthogonal transform (it is only

required to prepare the submatrix 611A). That is, the information amount of this candidate can be reduced to half. That is, an increase in the LUT size can be suppressed. Furthermore, by performing orthogonal
 5 transform/inverse orthogonal transform using the derived transformation matrix, the same coding efficiency as a case of using the transformation matrix 611 of DST1 for the orthogonal transform/inverse orthogonal transform can be obtained.

10 [0763]

Note that the transform type is not limited to the above-described example as long as the transformation matrix to which such derivation is applied has an element having the above-described symmetric property. For
 15 example, the transform type may be DCT3, DCT1, or DST3.
 [0764]

Furthermore, in the above description, the lower-half submatrix (the whole transformation matrix) has been derived from the upper-half submatrix. However, the
 20 embodiment is not limited to the example, and for example, the upper-half submatrix (the whole transformation matrix) may be derived from the lower-half submatrix.
 [0765]

Furthermore, derivation of an example one row below
 25 the second row example (the third row example from the top) focuses on a point-symmetric property.
 [0766]

In this case, the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation
 30 matrix, and the derivation unit derives the whole transformation matrix by flipping the submatrix in a

rotation direction around a center of the transformation matrix and further inverting signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number of the flipped submatrix to derive the lower-half submatrix or the upper-half submatrix of the transformation matrix (that is, a remaining submatrix of the transformation matrix).

[0767]

10 For example, as illustrated in Fig. 88, the derivation unit flips an upper-half submatrix 612A of a transformation matrix 612 having the transform type of DST4 in a rotation direction around a center 612X of the transformation matrix 612. Moreover, the derivation unit
15 inverts signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number. By doing so, a lower-half submatrix 612B of the transformation matrix 612 is derived. That is, the whole transformation matrix 612 is
20 derived. When such derivation is expressed as an operation expression for each element, where a transformation matrix to be derived is a transformation matrix T of N rows and N columns, and a submatrix to be used for the derivation is C, the derivation can be
25 expressed as the following expression (80).

[0768]

[Math. 70]

$$\begin{aligned} \text{sign} &= (x\%2==1 \parallel y\%2==1) ? 1:-1 \\ T[N-1-y, N-1-x] &= \text{sign} * C[y, x] \\ \text{for } x &= 0, \dots, N-1, \quad y = 0, \dots, N/2-1 \\ &\dots (80) \end{aligned}$$

[0769]

Since the whole transformation matrix can be derived from the submatrix by applying such derivation, it becomes unnecessary to prepare the whole transformation matrix 612 of DST4 as a candidate of a transformation matrix to be used for orthogonal transform/inverse orthogonal transform (it is only required to prepare the submatrix 612A). That is, the information amount of this candidate can be reduced to half. That is, an increase in the LUT size can be suppressed. Furthermore, by performing orthogonal transform/inverse orthogonal transform using the derived transformation matrix, the same coding efficiency as a case of using the transformation matrix 612 of DST4 for the orthogonal transform/inverse orthogonal transform can be obtained.

[0770]

Note that the transform type is not limited to the above-described example as long as the transformation matrix to which such derivation is applied has an element having the above-described symmetric property. For example, the transform type may be DCT4.

[0771]

Furthermore, in the above description, the lower-half submatrix (the whole transformation matrix) has been derived from the upper-half submatrix. However, the embodiment is not limited to the example, and for example, the upper-half submatrix (the whole transformation matrix) may be derived from the lower-half submatrix.

30 [0772]

Furthermore, derivation of an example one row below

the third row example (the fourth row example from the top) focuses on a symmetric property with respect to a diagonal axis.

[0773]

5 In this case, the submatrix is a submatrix of an upper right triangular portion or a submatrix of a lower left triangular portion of the transformation matrix, and the derivation unit derives the transformation matrix by transposing the submatrix to derive the submatrix of a
10 lower left triangular portion or the submatrix of an upper right triangular portion of the transformation matrix.

[0774]

For example, as illustrated in Fig. 89, the
15 derivation unit flips a submatrix 613A of an upper right triangular portion of a transformation matrix 613 having the transform type of DCT5 around a diagonal line connecting an upper left end and a lower right end of the transformation matrix 613. By doing so, a submatrix 613B
20 of a lower left triangular portion of the transformation matrix 613 is derived. That is, the whole transformation matrix 613 is derived. When such derivation is expressed as an operation expression for each element, where a transformation matrix to be derived is a transformation
25 matrix T of N rows and N columns, and a submatrix to be used for the derivation is C, the derivation can be expressed as the following expression (81).

[0775]

[Math. 71]

$$T[x, y] = C[y, x]$$

for $y=1, \dots, N-1$, $x=y, \dots, N-1$
 $\dots (81)$

[0776]

Since the whole transformation matrix can be derived from the submatrix by applying such derivation,
 5 it becomes unnecessary to prepare the whole transformation matrix 613 of DCT5 as a candidate of a transformation matrix to be used for orthogonal transform/inverse orthogonal transform (it is only required to prepare the submatrix 613A). That is, the
 10 information amount of this candidate can be reduced to half. That is, an increase in the LUT size can be suppressed. Furthermore, by performing orthogonal transform/inverse orthogonal transform using the derived transformation matrix, the same coding efficiency as a
 15 case of using the transformation matrix 613 of DCT5 for the orthogonal transform/inverse orthogonal transform can be obtained.

[0777]

Note that, as illustrated in Fig. 89, transposition
 20 is equivalent to flip around a diagonal line connecting an upper left end and a lower right end of a transformation matrix. That is, the transposition is included in the flip (one of flip operations).

[0778]

25 Furthermore, the transform type is not limited to the above-described example as long as the transformation matrix to which such derivation is applied has an element having the above-described symmetric property. For

example, the transform type may be DST5, DCT4, or DCT8.
[0779]

Furthermore, in the above description, derivation
of the submatrix (whole transformation matrix) of a lower
5 left triangular portion from the submatrix of an upper
right triangular portion has been described. However,
the embodiment is not limited thereto, and a submatrix
(whole transformation matrix) of an upper right
triangular portion may be derived from the submatrix of a
10 lower left triangular portion, for example.
[0780]

Furthermore, derivation of an example one row below
the fourth row example (the fifth row example from the
top) focuses on a symmetric property with respect to a
15 diagonal axis at a cross position.
[0781]

In this case, the submatrix is a submatrix of an
upper left triangular portion or a submatrix of a lower
right triangular portion of the transformation matrix,
20 and the derivation unit is configured to derive the
transformation matrix by flipping the submatrix in an
oblique direction having, as an axis, a diagonal line
connecting an upper right end and a lower left end of the
transformation matrix, further inverting signs of an
25 element having an even row number and an even column
number and an element having an odd row number and an odd
column number of the flipped submatrix, and deriving the
submatrix of a lower right triangular portion or the
submatrix of an upper left triangular portion of the
30 transformation matrix.
[0782]

For example, as illustrated in Fig. 90, the derivation unit flips a submatrix 602A of an upper left triangular portion of a transformation matrix 602 having the transform type of DST7 around a diagonal line
 5 connecting an upper right end and a lower left end of the transformation matrix 602. Moreover, the derivation unit inverts signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number. By doing so, a
 10 submatrix 602B of a lower right triangular portion of the transformation matrix 602 is derived. That is, the whole transformation matrix 602 is derived. When such derivation is expressed as an operation expression for each element, where a transformation matrix to be derived
 15 is a transformation matrix T of N rows and N columns, and a submatrix to be used for the derivation is C , the derivation can be expressed as the following expression
 (82).

[0783]

20 [Math. 72]

$$\begin{aligned} T[x, y] &= \text{sign} * C[y, x] \\ \text{for } y &= 1, \dots, N-1, \quad x = 0, \dots, N-1-y \\ &\dots \quad (82) \end{aligned}$$

[0784]

Since the whole transformation matrix can be derived from the submatrix by applying such derivation,
 25 it becomes unnecessary to prepare the whole transformation matrix 602 of DST7 as a candidate of a transformation matrix to be used for orthogonal transform/inverse orthogonal transform (it is only required to prepare the submatrix 602A). That is, the

information amount of this candidate can be reduced to half. That is, an increase in the LUT size can be suppressed. Furthermore, by performing orthogonal transform/inverse orthogonal transform using the derived transformation matrix, the same coding efficiency as a case of using the transformation matrix 602 of DST7 for the orthogonal transform/inverse orthogonal transform can be obtained.

[0785]

10 Note that the flip in the oblique direction having, as an axis, a diagonal line connecting an upper right end and a lower left end of the transformation matrix in this case is also included in the "transposition" (one of transposition operations) because rows and columns are
15 interchanged.

[0786]

 Note that the transform type is not limited to the above-described example as long as the transformation matrix to which such derivation is applied has an element
20 having the above-described symmetric property. For example, the transform type may be DST6.

[0787]

 Furthermore, in the above description, derivation of the submatrix (whole transformation matrix) of the lower right triangular portion from the submatrix of the
25 upper left triangular portion has been described. However, the embodiment is not limited thereto, and the submatrix (whole transformation matrix) of the upper left triangular portion may be derived from the submatrix of
30 the lower right triangular portion, for example.

[0788]

Furthermore, derivation of examples one row below the fifth row example (the sixth to eighth row examples) focuses on axial symmetric property with respect to a vertical axis, an axial symmetric property with respect to a horizontal axis, and a point-symmetric property.
[0789]

In this case, the submatrix is the upper left quarter submatrix of the transformation matrix, and the derivation unit flips the submatrix in the row direction of the transformation matrix and further inverts a sign of an odd-numbered row vector of the flipped submatrix to derive the upper right quarter submatrix of the transformation matrix, flips the submatrix in the column direction of the transformation matrix and further inverts a sign of an odd-numbered column vector of the flipped submatrix to derive the lower left quarter submatrix of the transformation matrix, and flips the submatrix in the rotation direction around the center of the transformation matrix and further inverts signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number of the flipped submatrix to derive the lower right quarter submatrix of the transformation matrix.
[0790]

For example, as illustrated in Fig. 91, the derivation unit flips an upper left quarter submatrix 614A of a transformation matrix 614 having the transform type of DST1 around an axis 614X-1 in a direction (column direction) parallel to a column passing through a center of the transformation matrix 614 (that is, flip is made

in the horizontal direction). Moreover, the derivation unit inverts the sign of an odd-numbered row vector. By doing so, an upper right quarter submatrix 614B of the transformation matrix 614 is derived.

5 [0791]

Furthermore, the derivation unit flips an upper left quarter submatrix 614A of a transformation matrix 614 having the transform type of DST1 around an axis 614X-2 in a direction (row direction) parallel to a row
 10 passing through a center of the transformation matrix 614 (that is, flip is made in the vertical direction). Moreover, the derivation unit inverts the sign of an odd-numbered column vector. By doing so, a lower left
 15 quarter submatrix 614C of the transformation matrix 614 is derived.

[0792]

Moreover, the derivation unit flips an upper left quarter submatrix 614A of a transformation matrix 614 having the transform type of DST1 in a rotation direction
 20 around a center 614X-3 of the transformation matrix 614. Moreover, the derivation unit inverts signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number. By doing so, a lower right quarter submatrix
 25 614D of the transformation matrix 614 is derived.

[0793]

That is, the whole transformation matrix 614 is derived. When such derivation is expressed as an operation expression for each element, where a
 30 transformation matrix to be derived is a transformation matrix T of N rows and N columns, and a submatrix to be

used for the derivation is C , the derivation can be expressed as the following expressions (83) to (85).

[0794]

[Math. 73]

[Upper right portion matrix]

```
sign = (y%2==0) ? 1:-1
T[y,N-1-x] = sign * C[y,x]
for x=0,...,N/2-1, y=0,...,N/2-1
    . . . (8 3)
```

[Lower left portion matrix]

```
sign = (x%2==0) ? 1:-1
T[N-1-y,x] = sign * C[y,x]
for x=0,...,N/2-1, y=0,...,N/2-1
    . . . (8 4)
```

[Lower right portion matrix]

```
sign = (x%2==1 || y%2==1) ? 1:-1
T[N-1-y,N-1-x] = sign * C[y,x]
for x=0,...,N/2-1, y=0,...,N/2-1
    . . . (8 5)
```

5

[0795]

Since the whole transformation matrix can be derived from the submatrix by applying such derivation, it becomes unnecessary to prepare the whole transformation matrix 614 of DST1 as a candidate of a transformation matrix to be used for orthogonal transform/inverse orthogonal transform (it is only required to prepare the submatrix 614A). That is, the information amount of this candidate can be reduced to one fourth. That is, an increase in the LUT size can be suppressed. Furthermore, by performing orthogonal transform/inverse orthogonal transform using the derived transformation matrix, the same coding efficiency as a case of using the transformation matrix 614 of DST1 for

10

15

the orthogonal transform/inverse orthogonal transform can be obtained.

[0796]

Note that the transform type is not limited to the
5 above-described example as long as the transformation
matrix to which such derivation is applied has an element
having the above-described symmetric property. For
example, the transform type may be DCT1.

[0797]

10 Furthermore, in the above-description, derivation
of the remaining submatrix from the upper left quarter
submatrix has been described. However, the embodiment is
not limited thereto, and for example, the remaining
submatrix may be derived from the upper right quarter
15 submatrix, from the lower left quarter submatrix, or from
the lower right quarter submatrix.

[0798]

<Transformation Matrix Derivation Unit>

A configuration of an image encoding device 100
20 that derives a transformation matrix from such a
submatrix is similar to the case of the first embodiment.
Then, in the image encoding device 100, an orthogonal
transform unit 113 performs processing to which the
above-described present technology is applied, as a
25 derivation unit and an orthogonal transform unit.
Furthermore, the encoding unit 115 performs processing to
which the above-described present technology is applied,
as an encoding unit. Furthermore, the inverse orthogonal
transform unit 118 performs processing to which the
30 above-described present technology is applied, as an
inverse orthogonal transform unit and a derivation unit.

Therefore, the image encoding device 100 can suppress an increase in the memory capacity required for the orthogonal transform/inverse orthogonal transform.

[0799]

5 Furthermore, the configuration of the orthogonal transform unit 113 is similar to the case of the first embodiment. Then, in the orthogonal transform unit 113, a primary transform unit 152 performs processing to which the above-described present technology is applied, as a
10 derivation unit and an orthogonal transform unit. That is, the derivation unit derives the whole transformation matrix, using the submatrix constituting a part of the transformation matrix, and the orthogonal transform unit performs primary transform for the prediction residual,
15 using the transformation matrix derived by the derivation unit. Therefore, an increase in the memory capacity required for the primary transform can be suppressed.

[0800]

 Note that, as described above, the primary
20 transform unit 152 performs the primary horizontal transform and the primary vertical transform as the primary transform. That is, the derivation unit derives the transformation matrix for horizontal one-dimensional orthogonal transform and the transformation matrix for
25 vertical one-dimensional orthogonal transform, and the orthogonal transform unit performs, as the primary transform, horizontal one-dimensional orthogonal transform, using the transformation matrix for horizontal one-dimensional orthogonal transform derived by the
30 derivation unit, and further, vertical one-dimensional orthogonal transform, using the transformation matrix for

vertical one-dimensional orthogonal transform derived by the derivation unit. Therefore, an increase in the memory capacity required for the primary transform where the horizontal one-dimensional orthogonal transform and
 5 the vertical one-dimensional orthogonal transform are performed can be suppressed.

[0801]

Note that, in this case, the primary transform unit 152 has a similar configuration to the case of <2-2.

10 Example 1-1>. Furthermore, a primary horizontal transform unit 312 has a similar configuration to the case of <2-2. Example 1-1>. Furthermore, a primary vertical transform unit 313 has a similar configuration to the case of <2-2. Example 1-1>. Note that an inner
 15 configuration of a transformation matrix derivation unit 321 of the primary horizontal transform unit 312 and an inner configuration of a transformation matrix derivation unit 351 of the primary vertical transform unit 313 are different from the case of <2-2. Example 1-1>.

20 [0802]

Fig. 92 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 321 included in the primary horizontal transform unit 312 in this case. As illustrated in Fig.

25 92, the transformation matrix derivation unit 321 in this case includes a transformation matrix LUT 331, a horizontal flip unit 632, a vertical flip unit 633, a transposition unit 634, and a sign inversion unit 635. Note that, in Fig. 92, arrows representing data transfer
 30 are omitted, but in the transformation matrix derivation unit 321, arbitrary data can be transferred between

arbitrary processing units (processing blocks).

[0803]

The horizontal flip unit 632 flips an input submatrix in the horizontal direction (row direction) around a straight line in the column direction passing through a center of a transformation matrix and outputs the flipped submatrix. The vertical flip unit 633 flips an input submatrix in the vertical direction (column direction) around a straight line in the row direction passing through the center of the transformation matrix and outputs the flipped submatrix. The transposition unit 634 transposes (obliquely flips) an input submatrix and outputs the transposed submatrix. The sign inversion unit 635 inverts a sign of a part of an input submatrix and outputs the submatrix after sign inversion.

[0804]

For example, in the case of the first row derivation example from the top except the uppermost row of item names in the table illustrated in Fig. 85, the transformation matrix derivation unit 321 reads the left-half (or right-half) submatrix of the transformation matrix from the transformation matrix LUT 331, and flips the submatrix in the row direction (horizontal direction) via the horizontal flip unit 632 and further inverts the sign of an odd-numbered row vector of the flipped submatrix via the sign inversion unit 635 to derive the remaining submatrix of the transformation matrix. By the processing, all of submatrices of the transformation matrix can be obtained. That is, the whole transformation matrix is derived. The transformation matrix derivation unit 321 outputs the derived (whole)

transformation matrix to the outside of the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

5 [0805]

Furthermore, for example, in the case of the second row derivation example from the top of the table illustrated in Fig. 85, the transformation matrix derivation unit 321 reads the upper-half (or lower-half) submatrix of the transformation matrix from the transformation matrix LUT 331, and flips the submatrix in the column direction (vertical direction) via the vertical flip unit 633 and further inverts the sign of an odd-numbered column vector of the flipped submatrix via the sign inversion unit 635 to derive the remaining submatrix of the transformation matrix. By the processing, all of submatrices of the transformation matrix can be obtained. That is, the whole transformation matrix is derived. The transformation matrix derivation unit 321 outputs the derived (whole) transformation matrix to the outside of the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

25 [0806]

Furthermore, for example, in the case of the third derivation example from the top of the table illustrated in Fig. 85, the transformation matrix derivation unit 321 reads the upper-half (or lower-half) submatrix of the transformation matrix from the transformation matrix LUT 331, flips the submatrix in the row direction (horizontal

direction) via the horizontal flip unit 632, further
 flips the horizontally flipped submatrix in the column
 direction (vertical direction) via the vertical flip unit
 633, and inverts the signs of an element having an even
 5 row number and an even column number and an element
 having an odd row number and an odd column number of the
 derived submatrix via the sign inversion unit 635 to
 derive the remaining submatrix of the transformation
 matrix. By the processing, all of submatrices of the
 10 transformation matrix can be obtained. That is, the
 whole transformation matrix is derived. The
 transformation matrix derivation unit 321 outputs the
 derived (whole) transformation matrix to the outside of
 the transformation matrix derivation unit 321 (supplies
 15 the same to the matrix calculation unit 322) as the
 transformation matrix T_H .

[0807]

Furthermore, for example, in the case of the fourth
 derivation example from the top of the table illustrated
 20 in Fig. 85, the transformation matrix derivation unit 321
 reads a submatrix of the upper right triangular portion
 (or the lower left triangular portion) of the
 transformation matrix from the transformation matrix LUT
 331 and transposes the submatrix around the straight line
 25 connecting the upper left end and the lower right end of
 the transformation matrix via the transposition unit 634
 to derive the remaining submatrix of the transformation
 matrix. By the processing, all of submatrices of the
 transformation matrix can be obtained. That is, the
 30 whole transformation matrix is derived. The
 transformation matrix derivation unit 321 outputs the

derived (whole) transformation matrix to the outside of the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

5 [0808]

Furthermore, for example, in the case of the fifth row derivation example from the top of the table illustrated in Fig. 85, the transformation matrix derivation unit 321 reads the submatrix of the upper left
 10 triangular portion (or lower right triangular portion) of the transformation matrix from the transformation matrix LUT 331, transposes the submatrix around the straight line connecting the upper right end and the lower left end of the transformation matrix via the transposition
 15 unit 634, and inverts the signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number of the derive submatrix via the sign inversion unit 635 to derive the remaining submatrix of the transformation
 20 matrix. By the processing, all of submatrices of the transformation matrix can be obtained. That is, the whole transformation matrix is derived. The transformation matrix derivation unit 321 outputs the derived (whole) transformation matrix to the outside of
 25 the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

[0809]

Moreover, for example, in the case of the sixth
 30 derivation example from the top illustrated in Fig. 85, the transformation matrix derivation unit 321 reads the

upper left quarter submatrix of the transformation matrix from the transformation matrix LUT 331. The transformation matrix derivation unit 321 flips the submatrix in the row direction (horizontal direction) via the horizontal flip unit 632 and further inverts the sign of an odd-numbered row vector of the flipped submatrix via the sign inversion unit 635 to derive the upper right quarter submatrix of the transformation matrix.

[0810]

Furthermore, the transformation matrix derivation unit 321 flips the upper left quarter submatrix in the column direction (vertical direction) via the vertical flip unit 633 and further inverts the sign of an odd-numbered column vector of the flipped submatrix via the sign inversion unit 635 to derive the lower left quarter submatrix of the transformation matrix.

[0811]

Furthermore, the transformation matrix derivation unit 321 flips the upper left quarter submatrix in the row direction (horizontal direction) via the horizontal flip unit 632, further flips the horizontally flipped submatrix in the column direction (vertical direction) via the vertical flip unit 633, and inverts the signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number of the derived submatrix via the sign inversion unit 635 to derive the lower right quarter submatrix of the transformation matrix.

[0812]

That is, by the processing, all of submatrices of the transformation matrix can be obtained. That is, the

whole transformation matrix is derived. The transformation matrix derivation unit 321 outputs the derived (whole) transformation matrix to the outside of the transformation matrix derivation unit 321 (supplies the same to the matrix calculation unit 322) as the transformation matrix T_H .

[0813]

With the above-described configuration, the transformation matrix derivation unit 321 can implement each derivation example of the table illustrated in Fig. 85.

[0814]

<Transformation Matrix Derivation Unit>

Fig. 93 is a block diagram illustrating a main configuration example of the transformation matrix derivation unit 351 included in the primary vertical transform unit 313 in this case. As illustrated in Fig. 93, the transformation matrix derivation unit 351 in this case includes a transformation matrix LUT 361, a horizontal flip unit 642, a vertical flip unit 643, a transposition unit 644, and a sign inversion unit 645, similarly to the case of the transformation matrix derivation unit 321. Note that, in Fig. 93, arrows representing data transfer are omitted, but in the transformation matrix derivation unit 351, arbitrary data can be transferred between arbitrary processing units (processing blocks).

[0815]

The horizontal flip unit 642 is a processing unit similar to the horizontal flip unit 632. The vertical flip unit 643 is a processing unit similar to the

vertical flip unit 633. The transposition unit 644 is a processing unit similar to the transposition unit 634. The sign inversion unit 645 is a processing unit similar to the sign inversion unit 635.

5 [0816]

The transformation matrix derivation unit 351 reads a predetermined submatrix of the transformation matrix from the transformation matrix LUT 361, and derives a remaining submatrix from the submatrix by any of the methods of the derivation examples illustrated in the table in Fig. 85, appropriately via the horizontal flip unit 632 to the sign inversion unit 635. That is, the whole transformation matrix is derived. The transformation matrix derivation unit 351 outputs the derived (whole) transformation matrix to the outside of the transformation matrix derivation unit 351 (supplies the same to a matrix calculation unit 352) as the transformation matrix T_v .

[0817]

20 With the above-described configuration, the transformation matrix derivation unit 351 can implement each derivation example of the table illustrated in Fig. 85.

[0818]

25 <Flow of Transformation Matrix Derivation Processing>

Since the primary transform processing in this case is performed by a flow similar to the case of <2-2. Example 1-1> (performed by a flow similar to the flowchart in Fig. 27), description thereof is basically omitted.

30

[0819]

Furthermore, in this case, the primary horizontal transform processing executed in step S302 in Fig. 27 is performed by a flow similar to the case of <2-2. Example 5 1-1> (performed by a flow similar to the flowchart in Fig. 28) but the transformation matrix derivation processing executed in step S321 in Fig. 28 is performed by a flow different from the case in Fig. 30.

[0820]

10 An example of a flow of the transformation matrix derivation processing executed by the transformation matrix derivation unit 321 in the primary horizontal transform unit 312 in step S321 in Fig. 28 in this case will be described with reference to the flowchart in Fig. 15 94.

[0821]

When the transformation matrix derivation processing is started, in step S701, the transformation matrix derivation unit 321 reads a submatrix C 20 corresponding to a transform type identifier TrTypeIdxH from the transformation matrix LUT 331 by reference to the correspondence table illustrated in Fig. 95, for example.

[0822]

25 In step S702, the transformation matrix derivation unit 321 derives a transformation matrix T_H of the transform type corresponding to the transform type identifier TrTypeIdxH from the submatrix C on the basis of a predetermined derivation method (any of the 30 derivation methods illustrated in the table in Fig. 85) appropriately using the horizontal flip unit 632 to the

sign inversion unit 635.

[0823]

When the processing in step S702 ends, the transformation matrix derivation processing ends and the processing returns to Fig. 28.

[0824]

<Flow of Transformation Matrix Derivation Processing>

Note that, in this case, the primary vertical transform processing executed in step S303 in Fig. 27 is performed by a flow similar to the case of <2-2. Example 1-1> (performed by a flow similar to the flowchart in Fig. 32). Note that the transformation matrix derivation processing executed in step S361 in Fig. 32 is performed by a flow similar to the case of the flowchart in Fig. 94. Therefore, the description is omitted. In the description in Fig. 94, the transform type identifier TrTypeIdxH is simply replaced with a transform type identifier TrTypeIdxV, and the transformation matrix T_H is simply replaced with a transformation matrix T_V .

[0825]

<Transformation Matrix Derivation Unit>

Next, a configuration of the image decoding device 200 in the case will be described. Since configurations of the inverse primary transform unit 253, the inverse primary vertical transform unit 412, the inverse primary horizontal transform unit 413, and the like included in the image decoding device 200 in this case are similar to the case of <2-2. Example 1-1>, description thereof is omitted.

[0826]

That is, the inverse orthogonal transform unit may perform inverse secondary transform for the coefficient data obtained by the decoding unit, and further perform inverse primary transform for an inverse secondary
 5 transform result, using the transformation matrix derived by the derivation unit.

[0827]

Furthermore, the derivation unit may derive the transformation matrix for horizontal inverse one-
 10 dimensional orthogonal transform and the transformation matrix for vertical inverse one-dimensional orthogonal transform, and the inverse orthogonal transform unit may perform the horizontal inverse one-dimensional orthogonal transform using the transformation matrix for horizontal
 15 inverse one-dimensional orthogonal transform derived by the derivation unit and further perform the vertical inverse one-dimensional orthogonal transform using the transformation matrix for vertical inverse one-dimensional orthogonal transform derived by the
 20 derivation unit, as the inverse primary transform.

[0828]

Fig. 96 is a block diagram illustrating a main configuration example of a transformation matrix derivation unit 421 included in the inverse primary
 25 vertical transform unit 412 in this case. As illustrated in Fig. 96, the transformation matrix derivation unit 421 in this case includes a transformation matrix LUT 431, a horizontal flip unit 652, a vertical flip unit 653, a transposition unit 654, and a sign inversion unit 655,
 30 similarly to the transformation matrix derivation unit 321. Note that, in Fig. 96, arrows representing data

transfer are omitted, but in the transformation matrix derivation unit 421, arbitrary data can be transferred between arbitrary processing units (processing blocks).
[0829]

5 The horizontal flip unit 652 is a processing unit similar to the horizontal flip unit 632. The vertical flip unit 653 is a processing unit similar to the vertical flip unit 633. The transposition unit 654 is a processing unit similar to the transposition unit 634.
10 The sign inversion unit 655 is a processing unit similar to the sign inversion unit 635.
[0830]

 In the inverse orthogonal transform (inverse primary vertical transform), the transformation matrix
15 derivation unit 421 reads a predetermined submatrix of the transformation matrix from the transformation matrix LUT 431, and derives a remaining submatrix from the submatrix by any of the methods of the derivation examples illustrated in the table in Fig. 85,
20 appropriately via the horizontal flip unit 652 to the sign inversion unit 655. That is, the whole transformation matrix is derived. The transformation matrix derivation unit 421 outputs the derived (whole) transformation matrix to the outside of the
25 transformation matrix derivation unit 421 (supplies the same to the matrix calculation unit 422) as the transformation matrix T_v .
[0831]

 With the above-described configuration, the
30 transformation matrix derivation unit 421 can implement each derivation example of the table illustrated in Fig.

85.

[0832]

<Transformation Matrix Derivation Unit>

Fig. 97 is a block diagram illustrating a main
 5 configuration example of a transformation matrix
 derivation unit 451 included in the inverse primary
 horizontal transform unit 413 in this case. As
 illustrated in Fig. 97, the transformation matrix
 derivation unit 451 in this case includes a
 10 transformation matrix LUT 461, a horizontal flip unit 662,
 a vertical flip unit 663, a transposition unit 664, and a
 sign inversion unit 665, similarly to the transformation
 matrix derivation unit 421. Note that, in Fig. 97,
 arrows representing data transfer are omitted, but in the
 15 transformation matrix derivation unit 451, arbitrary data
 can be transferred between arbitrary processing units
 (processing blocks).

[0833]

The horizontal flip unit 662 is a processing unit
 20 similar to the horizontal flip unit 632. The vertical
 flip unit 663 is a processing unit similar to the
 vertical flip unit 633. The transposition unit 664 is a
 processing unit similar to the transposition unit 634.
 The sign inversion unit 665 is a processing unit similar
 25 to the sign inversion unit 635.

[0834]

In the inverse orthogonal transform (inverse
 primary horizontal transform), the transformation matrix
 derivation unit 451 reads a predetermined submatrix of
 30 the transformation matrix from the transformation matrix
 LUT 431, and derives a remaining submatrix from the

submatrix by any of the methods of the derivation
 examples illustrated in the table in Fig. 85,
 appropriately via the horizontal flip unit 662 to the
 sign inversion unit 665. That is, the whole
 5 transformation matrix is derived. The transformation
 matrix derivation unit 451 outputs the derived (whole)
 transformation matrix to the outside of the
 transformation matrix derivation unit 451 (supplies the
 same to the matrix calculation unit 452) as the
 10 transformation matrix T_v .

[0835]

With the above-described configuration, the
 transformation matrix derivation unit 451 can implement
 each derivation example of the table illustrated in Fig.
 15 85.

[0836]

<Flow of Transformation Matrix Derivation Processing>

The inverse primary transform processing in this
 20 case is performed by a flow similar to the case of <2-
 Example 1-1> (performed by a flow similar to the
 flowchart in Fig. 39). Furthermore, the inverse primary
 transform selection processing executed in step S401 in
 Fig. 39 is performed by a flow similar to the case of <2-
 25 2. Example 1-1> (performed by a flow similar to the
 flowchart in Fig. 40). Furthermore, the inverse primary
 vertical transform processing executed in step S402 in
 Fig. 39 is performed by a flow similar to the case of <2-
 2. Example 1-1> (performed by a flow similar to the
 30 flowchart in Fig. 41). Therefore, description of the
 above processing is basically omitted.

[0837]

Note that the transformation matrix derivation processing (step S441 in Fig. 41) executed in the inverse primary vertical transform processing is performed by a flow similar to the case of the flowchart in Fig. 94. Therefore, the description is omitted.

[0838]

<Flow of Transformation Matrix Derivation Processing>

10 Note that, in this case, the inverse primary horizontal transform processing executed in step S403 in Fig. 39 is performed by a flow similar to the case of <2-2. Example 1-1> (performed by a flow similar to the flowchart in Fig. 42). Therefore, the description is
15 omitted.

[0839]

Note that the transformation matrix derivation processing (step S461 in Fig. 42) executed in the inverse primary horizontal transform processing is performed by a flow similar to the case of the flowchart in Fig. 94. Therefore, the description is omitted.

[0840]

<5. Fourth Embodiment>

<5-1. Common Concept>

25 <Combination of Embodiments>

The first to third embodiments have been described. These embodiments may be combined and applied. For example, the method described in Example 1-1 and the method described in the third embodiment may be applied
30 in combination to the method described in Non-Patent Document 1.

[0841]

That is, a derivation unit may derive a first transformation matrix, using a submatrix, and further derive a second transformation matrix, using the derived first transformation matrix, and an orthogonal transform unit may orthogonally transform a prediction residual, using the second transformation matrix derived by the derivation unit.

[0842]

10 At that time, the derivation unit may derive the second transformation matrix of a different transform type from the first transformation matrix, using the first transformation matrix. Furthermore, the derivation unit may flip or transpose the first transformation matrix to derive the second transformation matrix.

[0843]

In this case, as illustrated in the table in Fig. 98, the five types of transformation matrices required in the case of the technology described in Non-Patent Document 1 (see the table A in Fig. 6) can be reduced to three types, and moreover, it is only required to prepare submatrices for the three types of transformation matrices. Therefore, the total LUT size can be about 20 KB. That is, the size of the LUT can be reduced (by about 53 KB (the table A in Fig. 6)) as compared with the case of the technology described in Non-Patent Document 1. That is, an increase in the size of the LUT can be suppressed.

[0844]

30 <Transformation Matrix Derivation Unit>

Also in this case, configurations of an image

encoding device 100 and an image decoding device 200 are basically similar to those of the first embodiment.

However, only configurations of transformation matrix derivation units are different from those of the first
5 embodiment.

[0845]

For example, Fig. 99 illustrated a main configuration example of a transformation matrix derivation unit 321 in this case. As illustrated in Fig.
10 99, the transformation matrix derivation unit 321 in this case is only required to have both the configuration (Fig. 24) of Example 1-1 and the configuration (Fig. 92) of the third embodiment. That is, the transformation matrix derivation unit 321 in this case includes a
15 transformation matrix LUT 331, a flip unit 332, a transposition unit 333, a horizontal flip unit 632, a vertical flip unit 633, a transposition unit 634, and a sign inversion unit 635.

[0846]

20 <Flow of Transformation Matrix Derivation Processing>

An example of a flow of transformation matrix derivation processing executed in primary horizontal transform processing by the transformation matrix
25 derivation processing will be described with reference to the flowchart in Fig. 100.

[0847]

When the transformation matrix derivation processing is started, in step S721, the transformation
30 matrix derivation unit 321 reads a base submatrix C corresponding to a transform type identifier TrTypeIdxH

from the transformation matrix LUT 331 by reference to the correspondence table illustrated in Fig. 101.

[0848]

In step S722, the transformation matrix derivation
5 unit 321 derives a flip flag FlipFlag and a transposition
flag TransposeFlag corresponding to the transform type
identifier TrTypeIdxH (sets a value) on the basis of the
correspondence table.

[0849]

10 In step S723, the transformation matrix derivation
unit 321 generates a base transformation matrix T_{base}
corresponding to the transform type identifier TrTypeIdxH
from the submatrix C on the basis of a predetermined
derivation method, appropriately using the horizontal
15 flip unit 632 to the sign inversion unit 635.

[0850]

Then, the transformation matrix derivation unit 321
executes each processing from steps S724 to S728,
appropriately using the flip unit 332 and the
20 transposition unit 333, similarly to each processing from
step S342 to S346 in Fig. 30. That is, the first
transformation matrix is appropriately transformed to the
second transformation matrix.

[0851]

25 When the processing in step S728 ends, the
transformation matrix derivation processing ends and the
processing returns to Fig. 28.

[0852]

Note that all of a transformation matrix derivation
30 unit 351 included in a primary vertical transform unit
313, a transformation matrix derivation unit 421 included

in an inverse primary vertical transform unit 412, and a transformation matrix derivation unit 451 included in an inverse primary horizontal transform unit 413 have similar configurations to the case described with reference to Fig. 99. Therefore, description of the processing is omitted.

[0853]

Furthermore, all of transformation matrix derivation processing executed in primary vertical transform processing, transformation matrix derivation processing executed in inverse primary vertical transform processing, and transformation matrix derivation processing executed in inverse primary horizontal transform processing are executed by flows similar to the case described with reference to the flowchart in Fig. 100. Therefore, description of the processing is omitted.

[0854]

By doing so, effects of both Example 1-1 and the third embodiment can be obtained.

[0855]

Note that the combination of the embodiments (examples) is arbitrary, and is not limited to this example. For example, the method described in the third embodiment can be combined with each example of the first embodiment. Furthermore, the method described in the third embodiment can be combined with the method described in the second embodiment.

[0856]

<6. Appendix>

<Computer>

The above-described series of processing can be

executed by hardware or by software. In the case of executing the series of processing by software, a program that configures the software is installed in a computer. Here, the computer includes a computer incorporated in
5 dedicated hardware, a computer, for example, general-purpose personal computer, capable of executing various functions by installing various programs, and the like.

[0857]

Fig. 102 is a block diagram illustrating a
10 configuration example of hardware of a computer that executes the above-described series of processing by a program.

[0858]

In a computer 800 illustrated in Fig. 102, a
15 central processing unit (CPU) 801, a read only memory (ROM) 802, and a random access memory (RAM) 803 are mutually connected by a bus 804.

[0859]

An input/output interface 810 is also connected to
20 the bus 804. An input unit 811, an output unit 812, a storage unit 813, a communication unit 814, and a drive 815 are connected to the input/output interface 810.

[0860]

The input unit 811 includes, for example, a
25 keyboard, a mouse, a microphone, a touch panel, an input terminal, and the like. The output unit 812 includes, for example, a display, a speaker, an output terminal, and the like. The storage unit 813 includes, for example, a hard disk, a RAM disk, a nonvolatile memory, and the
30 like. The communication unit 814 includes, for example, a network interface. The drive 815 drives a removable

medium 821 such as a magnetic disk, an optical disk, a magneto-optical disk, or a semiconductor memory.

[0861]

In the computer configured as described above, the
5 CPU 801 loads, for example, a program stored in the
storage unit 813 into the RAM 803 and executes the
program via the input/output interface 810 and the bus
804, so that the above-described series of processing is
performed. Furthermore, the RAM 803 appropriately stores
10 data and the like necessary for the CPU 801 to execute
the various types of processing.

[0862]

The program executed by the computer (CPU 801) can
be recorded on the removable medium 821 as a package
15 medium or the like, for example, and applied. In that
case, the program can be installed to the storage unit
813 via the input/output interface 810 by attaching the
removable medium 821 to the drive 815.

[0863]

20 Furthermore, this program can be provided via a
wired or wireless transmission medium such as a local
area network, the Internet, or digital broadcast. In
that case, the program can be received by the
communication unit 814 and installed in the storage unit
25 813.

[0864]

Other than the above method, the program can be
installed in the ROM 802 or the storage unit 813 in
advance.

30 [0865]

<Units of Information and Processing>

The data unit in which various types of information described above are set and the data unit to be processed by various types of processing are arbitrary, and are not limited to the above-described examples. For example,
 5 these pieces of information and processing may be set for each transform unit (TU), transformation block (TB), prediction unit (PU), prediction block (PB), coding unit (CU), largest coding unit (LCU), subblock, block, tile, slice, picture, sequence, or component, or data in these
 10 data units may be used. Of course, this data unit can be set for each information and processing, and the data units of all pieces of information and processing need not to be unified. Note that the storage location of these pieces of information is arbitrary, and may be
 15 stored in a header, a parameter, or the like of the above-described data unit. Furthermore, the information may be stored in a plurality of locations.
 [0866]

<Control Information>

20 Control information regarding the present technology described in the above embodiments may be transmitted from the encoding side to the decoding side. For example, control information (for example, enabled_flag) for controlling whether or not application
 25 of the above-described present technology is to be permitted (or prohibited) may be transmitted. Furthermore, for example, control information indicating an object to which the above-described present technology is applied (or an object to which the present technology
 30 is not applied) may be transmitted. For example, control information for specifying a block size (upper limit,

lower limit, or both) to which the present technology is applied (or application is permitted or prohibited), a frame, a component, a layer, or the like may be transmitted.

5 [0867]

<Applicable Object of Present Technology>

The present technology can be applied to any image encoding/decoding method. That is, specifications of various types of processing regarding image
10 encoding/decoding such as transform (inverse transform), quantization (inverse quantization), encoding (decoding), and prediction are arbitrary and are not limited to the above-described examples as long as no contradiction occurs with the above-described present technology.
15 Furthermore, part of the processing may be omitted as long as no contradiction occurs with the above-described present technology.

[0868]

Furthermore, the present technology can be applied
20 to a multi-view image encoding/decoding system that performs encoding/decoding of a multi-view image including images of a plurality of viewpoints (views). In this case, the present technology is simply applied to encoding/decoding of each viewpoint (view).

25 [0869]

Furthermore, the present technology can be applied to a hierarchical image encoding (scalable encoding)/decoding system that encodes/decodes a hierarchical image that is multi-layered (hierarchized)
30 so as to have a scalability function for a predetermined parameter. In this case, the present technology is

simply applied to encoding/decoding of each layer (layer).
[0870]

The image encoding devices 100 and the image decoding devices 200 according to the above-described
5 embodiments can be applied to, for example, transmitters and receivers (such as television receivers and mobile phones) in satellite broadcasting, cable broadcasting such as cable TV, distribution on the Internet, and distribution to terminals by cellular communication, or
10 various electronic devices such as devices (for example, hard disk recorders and cameras) that record images on media such as optical disks, magnetic disks, and flash memories, and reproduce images from these storage media.
[0871]

15 Furthermore, the present technology can be implemented as any configuration to be mounted on a device that configures arbitrary device or system, such as a processor (for example, a video processor) as a system large scale integration (LSI) or the like, a
20 module (for example, a video module) using a plurality of processors or the like, a unit (for example, a video unit) using a plurality of modules or the like, or a set (for example, a video set) in which other functions are added to the unit (that is, a configuration of a part of
25 the device), for example.
[0872]

Moreover, the present technology can also be applied to a network system including a plurality of devices. For example, the present technology can be
30 applied to a cloud service that provides a service regarding an image (moving image) to an arbitrary

terminal such as a computer, an audio visual (AV) device,
a portable information processing terminal, or an
internet of things (IoT) device.

[0873]

5 Note that the systems, apparatuses, processing
units, and the like to which the present technology is
applied can be used in arbitrary fields such as traffic,
medical care, crime prevention, agriculture, livestock
industry, mining, beauty, factory, household appliance,
10 weather, and natural surveillance, for example.
Furthermore, uses in the arbitrary fields are also
arbitrary.

[0874]

 For example, the present technology can be applied
15 to systems and devices provided for providing content for
appreciation and the like. Furthermore, for example, the
present technology can also be applied to systems and
devices used for traffic, such as traffic condition
monitoring and automatic driving control. Moreover, for
20 example, the present technology can also be applied to
systems and devices provided for security. Furthermore,
for example, the present technology can be applied to
systems and devices provided for automatic control of
machines and the like. Moreover, for example, the
25 present technology can also be applied to systems and
devices provided for agriculture or livestock industry.
Furthermore, the present technology can also be applied
to systems and devices that monitor nature states such as
volcanos, forests, and ocean, wildlife, and the like.
30 Moreover, for example, the present technology can also be
applied to systems and devices provided for sports.

[0875]

<Others>

Note that the "flag" in the present specification is information for identifying a plurality of states, and includes not only information used for identifying two states of true (1) and false (0) but also information capable of identifying three or more states. Therefore, the value that the "flag" can take may be, for example, a binary value of 1/0 or may be a ternary value or more. That is, the number of bits constituting the "flag" is arbitrary, and may be 1 bit or a plurality of bits. Furthermore, the identification information (including flag) is assumed to be in not only a form of including the identification information in a bit stream but also a form of including difference information of the identification information from certain reference information in a bit stream. Therefore, in the present specification, the "flag" and "identification information" include not only the information itself but also the difference information for the reference information.

[0876]

Furthermore, various types of information (metadata and the like) regarding coded data (bit stream) may be transmitted or recorded in any form as long as the various types of information are associated with the coded data. Here, the term "associate" means that, for example, one data can be used (linked) when the other data is processed. That is, data associated with each other may be collected as one data or may be individual data. For example, information associated with coded data (image) may be transmitted on a transmission path

different from that of the coded data (image).

Furthermore, for example, information associated with

coded data (image) may be recorded on a different

recording medium (or another recording area of the same

5 recording medium) from the coded data (image). Note that

this "association" may be a part of data instead of

entire data. For example, an image and information

corresponding to the image may be associated with each

other in an arbitrary unit such as a plurality of frames,

10 one frame, or a part in a frame.

[0877]

Note that, in the present specification, terms such

as "combining", "multiplexing", "adding", "integrating",

"including", "storing", and "inserting" mean putting a

15 plurality of things into one, such as putting coded data

and metadata into one data, and means one method of the

above-described "association".

[0878]

Furthermore, embodiments of the present technology

20 are not limited to the above-described embodiments, and

various modifications can be made without departing from

the gist of the present technology.

[0879]

Furthermore, for example, the present technology

25 can be implemented as any configuration constituting a

device or a system, such as a processor as a system large

scale integration (LSI) or the like, a module using a

plurality of processors or the like, a unit using a

plurality of modules or the like, or a set in which other

30 functions are added to the unit (that is, a configuration

of a part of the device), for example.

[0880]

Note that, in this specification, the term "system" means a group of a plurality of configuration elements (devices, modules (parts), and the like), and whether or
5 not all the configuration elements are in the same casing is irrelevant. Therefore, a plurality of devices housed in separate housings and connected via a network, and one device that houses a plurality of modules in one casing are both systems.

10 [0881]

Further, for example, the configuration described as one device (or processing unit) may be divided into and configured as a plurality of devices (or processing units). On the contrary, the configuration described a
15 plurality of devices (or processing units) may be collectively configured as one device (or processing unit). Furthermore, a configuration other than the above-described configuration may be added to the configuration of each device (or each processing unit).
20 Moreover, a part of the configuration of a certain device (or processing unit) may be included in the configuration of another device (or another processing unit) as long as the configuration and operation of the system as a whole are substantially the same.

25 [0882]

Further, for example, in the present technology, a configuration of cloud computing in which one function is shared and processed in cooperation by a plurality of devices via a network can be adopted.

30 [0883]

Furthermore, for example, the above-described

program can be executed by an arbitrary device. In that case, the device is only required to have necessary functions (functional blocks and the like) and obtain necessary information.

5 [0884]

Further, for example, the steps described in the above-described flowcharts can be executed by one device or can be executed by a plurality of devices in a shared manner. Furthermore, in the case where a plurality of
10 processes is included in one step, the plurality of processes included in the one step can be executed by one device or can be shared and executed by a plurality of devices. In other words, the plurality of processes included in one step can be executed as processes of a
15 plurality of steps. Conversely, the processing described as a plurality of steps can be collectively executed as one step.

[0885]

Note that, in the program executed by the computer,
20 the processing of the steps describing the program may be executed in chronological order according to the order described in the present specification, or may be individually executed in parallel or at necessary timing when a call is made, for example. That is, the
25 processing of each step may be executed in an order different from the above-described order as long as no contradiction occurs. Further, the processing of the steps describing the program may be executed in parallel with the processing of another program, or may be
30 executed in combination with the processing of another program.

[0886]

Note that the plurality of present technologies described in the present specification can be implemented independently of one another as a single unit as long as
5 there is no inconsistency. Of course, an arbitrary number of the present technologies can be implemented together. For example, part or whole of the present technology described in any of the embodiments can be implemented in combination with part or whole of the
10 present technology described in another embodiment. Further, part or whole of the above-described arbitrary present technology can be implemented in combination with another technology not described above.

[0887]

15 Note that the present technology can also have the following configurations.

(1) An image processing apparatus including:

a derivation unit configured to derive a second transformation matrix using a first transformation
20 matrix;

an orthogonal transform unit configured to orthogonally transform a prediction residual of an image, using the second transformation matrix derived by the derivation unit; and

25 an encoding unit configured to encode coefficient data obtained by orthogonally transforming the prediction residual by the orthogonal transform unit to generate a bit stream.

(2) The image processing apparatus according to (1),
30 in which

the derivation unit derives the second

transformation matrix of a transform type different from that of the first transformation matrix, using the first transformation matrix.

(3) The image processing apparatus according to (1)
5 or (2), in which

the derivation unit derives the second transformation matrix having a same number of rows and a same number of columns as the first transformation matrix, using the first transformation matrix.

10 (4) The image processing apparatus according to any one of (1) to (3), in which

the derivation unit derives the second transformation matrix by an operation for an element of the first transformation matrix.

15 (5) The image processing apparatus according to (4), in which

the operation includes rearrangement of elements.

(6) The image processing apparatus according to (4) or (5), in which

20 the derivation unit derives the second transformation matrix by performing the operation a plurality of times.

(7) The image processing apparatus according to any one of (1) to (6), in which

25 the derivation unit derives the second transformation matrix, using the first transformation matrix to be stored in a lookup table.

(8) The image processing apparatus according to any one of (1) to (7), in which

30 the orthogonal transform unit performs primary transform for the prediction residual, using the second

transformation matrix derived by the derivation unit, and further performs secondary transform for a result of the primary transform.

(9) The image processing apparatus according to (8),
5 in which

the derivation unit derives the second transformation matrix for horizontal one-dimensional orthogonal transform and the second transformation matrix for vertical one-dimensional orthogonal transform, and
10 the orthogonal transform unit performs, as the primary transform,

the horizontal one-dimensional orthogonal transform, using the second transformation matrix for horizontal one-dimensional orthogonal transform derived
15 by the derivation unit, and further,

the vertical one-dimensional orthogonal transform, using the second transformation matrix for vertical one-dimensional orthogonal transform derived by the derivation unit.

(10) The image processing apparatus according to
20 any one of (1) to (9), in which

the derivation unit flips the first transformation matrix and derives the second transformation matrix.

(11) The image processing apparatus according to
25 any one of (1) to (10), in which

the derivation unit transposes the first transformation matrix and derives the second transformation matrix.

(12) The image processing apparatus according to
30 any one of (1) to (11), in which

the derivation unit flips the first transformation

matrix, inverts a sign of an odd-numbered row vector, of the flipped first transformation matrix, and derives the second transformation matrix.

(13) The image processing apparatus according to
5 any one of (1) to (12), in which

the derivation unit flips the first transformation matrix, transposes the flipped first transformation matrix, and derives the second transformation matrix.

(14) The image processing apparatus according to
10 any one of (1) to (13), in which

the derivation unit transposes the first transformation matrix, flips the transposed first transformation matrix, and derives the second transformation matrix.

(15) The image processing apparatus according to
15 any one of (1) to (14), in which

the derivation unit derives the second transformation matrix in which a lowest-order row vector has a waveform of a desired type, using the first
20 transformation matrix.

(16) The image processing apparatus according to
any one of (1) to (15), in which

the derivation unit derives the second transformation matrix in which the lowest-order row
25 vector has a flat-type waveform, using the first transformation matrix.

(17) The image processing apparatus according to
any one of (1) to (16), in which

the derivation unit derives the second
30 transformation matrix in which the lowest-order row vector has an increasing-type waveform, using the first

transformation matrix.

(18) The image processing apparatus according to any one of (1) to (17), in which

the derivation unit derives the second
5 transformation matrix in which the lowest-order row vector has a decreasing-type waveform, using the first transformation matrix.

(19) The image processing apparatus according to any one of (1) to (18), in which

10 the derivation unit derives the second transformation matrix in which the lowest-order row vector has a chevron-type waveform, using the first transformation matrix.

(20) An image processing method including:
15 deriving a second transformation matrix using a first transformation matrix;
orthogonally transforming a prediction residual of an image, using the derived second transformation matrix;
and
20 encoding coefficient data obtained by orthogonally transforming the prediction residual to generate a bit stream.

[0888]

(21) An image processing apparatus including:
25 a decoding unit configured to decode a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image;

a derivation unit configured to derive a second transformation matrix, using a first transformation
30 matrix; and

an inverse orthogonal transform unit configured to

inversely orthogonally transform the coefficient data obtained by the decoding unit, using the second transformation matrix derived by the derivation unit.

(22) The image processing apparatus according to
5 (21), in which

the derivation unit derives the second transformation matrix of a transform type different from that of the first transformation matrix, using the first transformation matrix.

10 (23) The image processing apparatus according to (21) or (22), in which

the derivation unit derives the second transformation matrix having a same number of rows and a same number of columns as the first transformation matrix,
15 using the first transformation matrix.

(24) The image processing apparatus according to any one of (21) to (23), in which

the derivation unit derives the second transformation matrix by an operation for an element of
20 the first transformation matrix.

(25) The image processing apparatus according to (24), in which

the operation includes rearrangement of elements.

(26) The image processing apparatus according to
25 (24) or (25), in which

the derivation unit derives the second transformation matrix by performing the operation a plurality of times.

(27) The image processing apparatus according to
30 any one of (21) to (26), in which

the derivation unit derives the second

transformation matrix, using the first transformation matrix to be stored in a lookup table.

(28) The image processing apparatus according to any one of (21) to (27), in which

5 the inverse orthogonal transform unit performs inverse secondary transform for the coefficient data, and further performs inverse primary transform for a result of the inverse secondary transform, using the second transformation matrix derived by the derivation unit.

10 (29) The image processing apparatus according to (28), in which

 the derivation unit derives the second transformation matrix for horizontal inverse one-dimensional orthogonal transform and the second transformation matrix for vertical inverse one-dimensional orthogonal transform, and

 the inverse orthogonal transform unit performs, as the inverse primary transform,

 the horizontal inverse one-dimensional orthogonal transform, using the second transformation matrix for horizontal inverse one-dimensional orthogonal transform derived by the derivation unit, and further,

 the vertical inverse one-dimensional orthogonal transform, using the second transformation matrix for vertical inverse one-dimensional orthogonal transform derived by the derivation unit.

(30) The image processing apparatus according to any one of (21) to (29), in which

 the derivation unit flips the first transformation matrix and derives the second transformation matrix.

(31) The image processing apparatus according to

any one of (21) to (30), in which

the derivation unit transposes the first transformation matrix and derives the second transformation matrix.

5 (32) The image processing apparatus according to any one of (21) to (31), in which

the derivation unit flips the first transformation matrix, inverts a sign of an odd-numbered row vector, of the flipped first transformation matrix, and derives the
10 second transformation matrix.

(33) The image processing apparatus according to any one of (21) to (32), in which

the derivation unit flips the first transformation matrix, transposes the flipped first transformation
15 matrix, and derives the second transformation matrix.

(34) The image processing apparatus according to any one of (21) to (33), in which

the derivation unit transposes the first transformation matrix, flips the transposed first
20 transformation matrix, and derives the second transformation matrix.

(35) The image processing apparatus according to any one of (21) to (34), in which

the derivation unit derives the second
25 transformation matrix in which a lowest-order row vector has a waveform of a desired type, using the first transformation matrix.

(36) The image processing apparatus according to any one of (21) to (35), in which

30 the derivation unit derives the second transformation matrix in which the lowest-order row

vector has a flat-type waveform, using the first transformation matrix.

(37) The image processing apparatus according to any one of (21) to (36), in which

5 the derivation unit derives the second transformation matrix in which the lowest-order row vector has an increasing-type waveform, using the first transformation matrix.

(38) The image processing apparatus according to
10 any one of (21) to (37), in which

 the derivation unit derives the second transformation matrix in which the lowest-order row vector has a decreasing-type waveform, using the first transformation matrix.

15 (39) The image processing apparatus according to any one of (21) to (38), in which

 the derivation unit derives the second transformation matrix in which the lowest-order row vector has a chevron-type waveform, using the first
20 transformation matrix.

(40) An image processing method including:

 decoding a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image;

25 deriving a second transformation matrix, using a first transformation matrix; and

 inversely orthogonally transforming the obtained coefficient data, using the derived second transformation matrix.

30 [0889]

(41) An image processing apparatus including:

an operation unit configured to perform an operation for permutation for a prediction residual of an image;

an orthogonal transform unit configured to
5 orthogonally transform the prediction residual operated for permutation by the operation unit, using a transformation matrix serving as a base; and

an encoding unit configured to encode coefficient data obtained by orthogonally transforming the prediction
10 residual by the orthogonal transform unit to generate a bit stream.

(42) The image processing apparatus according to (41), in which

the operation unit flips the prediction residual in
15 a spatial symmetric direction between one-dimensional orthogonal transforms, and

the orthogonal transform unit orthogonally transforms the prediction residual flipped by the operation unit, using the transformation matrix.

20 (43) The image processing apparatus according to (41) or (42), in which

the operation unit horizontally flips the prediction residual, and

the orthogonal transform unit orthogonally
25 transforms the prediction residual horizontally flipped by the operation unit, using the transformation matrix.

(44) The image processing apparatus according to any one of (41) to (43), in which

the operation unit vertically flips the prediction
30 residual, and

the orthogonal transform unit orthogonally

transforms the prediction residual vertically flipped by the operation unit, using the transformation matrix.

(45) The image processing apparatus according to any one of (41) to (44), in which

5 the operation unit horizontally and vertically flips the prediction residual, and

 the orthogonal transform unit orthogonally transforms the prediction residual horizontally and vertically flipped by the operation unit, using the transformation matrix.

(46) The image processing apparatus according to any one of (41) to (45), further including:

 a derivation unit configured to derive a second transformation matrix using a first transformation matrix,

15 in which

 the orthogonal transform unit is configured to orthogonally transform the prediction residual flipped by the operation unit, using the second transformation matrix derived by the derivation unit.

20 (47) The image processing apparatus according to (46), in which

 the derivation unit derives the second transformation matrix having a same number of rows and a same number of columns as the first transformation matrix,

25 using the first transformation matrix.

(48) The image processing apparatus according to (46) or (47), in which

 the derivation unit inverts a sign of an odd-numbered row vector, of the first transformation matrix,

30 and derives the second transformation matrix.

(49) The image processing apparatus according to

any one of (41) to (48), in which

the orthogonal transform unit performs primary transform for the prediction residual operated for permutation by the operation unit, using a transformation matrix serving as a base, and further performs secondary transform for a result of the primary transform.

(50) An image processing method including:

performing an operation for permutation for a prediction residual of an image;

10 orthogonally transforming the prediction residual operated for permutation, using a transformation matrix serving as a base; and

encoding coefficient data obtained by orthogonally transforming the prediction residual to generate a bit stream.

[0890]

(51) An image processing apparatus including:

a decoding unit configured to decode a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image;

20 an inverse orthogonal transform unit configured to inversely orthogonally transform the coefficient data obtained by the decoding unit; and

an operation unit configured to perform an operation for permutation for an inverse orthogonal transform result of the coefficient data obtained by the inverse orthogonal transform unit.

(52) The image processing apparatus according to

(51), in which

30 the operation unit flips the inverse orthogonal transform result in a spatial symmetric direction between

one-dimensional orthogonal transforms.

(53) The image processing apparatus according to (51) or (52), in which

the operation unit horizontally flips the inverse
5 orthogonal transform result.

(54) The image processing apparatus according to any one of (51) to (53), in which

the operation unit vertically flips the inverse
orthogonal transform result.

10 (55) The image processing apparatus according to any one of (51) to (54), in which

the operation unit horizontally and vertically
flips the inverse orthogonal transform result.

(56) The image processing apparatus according to
15 any one of (51) to (55), further including:

a derivation unit configured to derive a second
transformation matrix using a first transformation matrix,
in which

the inverse orthogonal transform unit is configured
20 to inversely orthogonally transform the coefficient data
obtained by the decoding unit, using the second
transformation matrix derived by the derivation unit, and

the operation unit is configured to perform an
operation for permutation for the inverse orthogonal
25 transform result of the coefficient data obtained by the
inverse orthogonal transform unit.

(57) The image processing apparatus according to (56), in which

the derivation unit derives the second
30 transformation matrix having a same number of rows and a
same number of columns as the first transformation matrix,

using the first transformation matrix.

(58) The image processing apparatus according to (56) or (57), in which

the derivation unit inverts a sign of an odd-
5 numbered row vector, of the first transformation matrix,
and derives the second transformation matrix.

(59) The image processing apparatus according to any one of (51) to (58), in which

the inverse orthogonal transform unit performs
10 inverse secondary transform for the coefficient data
obtained by the decoding unit, and further performs
inverse primary transform for an inverse secondary
transform result, using a transformation matrix serving
as a base, and

15 the operation unit performs an operation for
permutation for an inverse primary transform result
obtained by the inverse orthogonal transform unit.

(60) An image processing method including:

decoding a bit stream to obtain coefficient data
20 that is an orthogonally transformed prediction residual
of an image;

inversely orthogonally transforming the obtained
coefficient data; and

performing an operation for permutation for an
25 inverse orthogonal transform result of the obtained
coefficient data.

[0891]

(61) An image processing apparatus including:

a derivation unit configured to derive a
30 transformation matrix, using a submatrix configuring a
part of the transformation matrix;

an orthogonal transform unit configured to orthogonally transform a prediction residual of an image, using the transformation matrix derived by the derivation unit; and

5 an encoding unit configured to encode coefficient data obtained by orthogonally transforming the prediction residual by the orthogonal transform unit to generate a bit stream.

(62) The image processing apparatus according to
10 (61), in which

the derivation unit derives the transformation matrix, using the submatrix to be stored in a lookup table.

(63) The image processing apparatus according to
15 (61) or (62), in which

the orthogonal transform unit performs primary transform for the prediction residual, using the transformation matrix derived by the derivation unit, and further performs secondary transform for a result of the
20 primary transform.

(64) The image processing apparatus according to (63), in which

the derivation unit derives the transformation matrix for horizontal one-dimensional orthogonal
25 transform and the transformation matrix for vertical one-dimensional orthogonal transform, and

the orthogonal transform unit performs, as the primary transform,

the horizontal one-dimensional orthogonal
30 transform, using the transformation matrix for horizontal one-dimensional orthogonal transform derived by the

derivation unit, and further,

the vertical one-dimensional orthogonal transform, using the transformation matrix for vertical one-dimensional orthogonal transform derived by the derivation unit.

(65) The image processing apparatus according to any one of (61) to (64), in which

the derivation unit derives the transformation matrix by flipping the submatrix around an axis of a predetermined direction passing through a center of the transformation matrix and deriving a remaining submatrix of the transformation matrix.

(66) The image processing apparatus according to (65), in which

the derivation unit flips the submatrix in a direction parallel to a row.

(67) The image processing apparatus according to (65) or (66), in which

the derivation unit flips the submatrix in a direction parallel to a column.

(68) The image processing apparatus according to any one of (65) to (67), in which

the derivation unit flips the submatrix in a rotation direction around the center of the transformation matrix.

(69) The image processing apparatus according to any one of (65) to (68), in which

the derivation unit flips the submatrix in an oblique direction around a diagonal line of the transformation matrix.

(70) The image processing apparatus according to

any one of (65) to (69), in which

the derivation unit further inverts a sign of an element of the flipped submatrix, and derives the transformation matrix.

5 (71) The image processing apparatus according to any one of (61) to (70), in which

the submatrix is a left-half submatrix or a right-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the
10 transformation matrix by flipping the submatrix in a row direction of the transformation matrix, further inverting a sign of an odd-numbered row vector of the flipped submatrix, and deriving the right-half submatrix or the left-half submatrix of the transformation matrix.

15 (72) The image processing apparatus according to any one of (61) to (71), in which

the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the
20 transformation matrix by flipping the submatrix in a column direction of the transformation matrix, further inverting a sign of an odd-numbered column vector of the flipped submatrix, and deriving the lower-half submatrix or the upper-half submatrix of the transformation matrix.

25 (73) The image processing apparatus according to any one of (61) to (72), in which

the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the
30 transformation matrix by flipping the submatrix in a rotation direction around a center of the transformation

matrix, further inverting signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number of the flipped submatrix, and deriving the lower-half submatrix or the upper-half submatrix of the transformation matrix.

(74) The image processing apparatus according to any one of (61) to (73), in which

the submatrix is a submatrix of an upper right triangular portion or a submatrix of a lower left triangular portion of the transformation matrix, and

the derivation unit is configured to derive the transformation matrix by transposing the submatrix to derive the submatrix of a lower left triangular portion or the submatrix of an upper right triangular portion of the transformation matrix.

(75) The image processing apparatus according to any one of (61) to (74), in which

the submatrix is a submatrix of an upper left triangular portion or a submatrix of a lower right triangular portion of the transformation matrix, and

the derivation unit is configured to derive the transformation matrix by flipping the submatrix in an oblique direction having, as an axis, a diagonal line connecting an upper right end and a lower left end of the transformation matrix, further inverting signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number of the flipped submatrix, and deriving the submatrix of a lower right triangular portion or the submatrix of an upper left triangular portion of the transformation matrix.

(76) The image processing apparatus according to any one of (61) to (75), in which

the submatrix is an upper left quarter submatrix of the transformation matrix, and

5 the derivation unit is configured to derive the transformation matrix by

flipping the submatrix in a row direction of the transformation matrix, further inverting a sign of an odd-numbered row vector of the flipped submatrix, and
10 deriving an upper right quarter submatrix of the transformation matrix,

flipping the submatrix in a column direction of the transformation matrix, further inverting a sign of an odd-numbered column vector of the flipped submatrix,
15 and deriving a lower left quarter submatrix of the transformation matrix, and

flipping the submatrix in a rotation direction around a center of the transformation matrix, further inverting signs of an element having an even row
20 number and an even column number and an element having an odd row number and an odd column number of the flipped submatrix, and deriving a lower right quarter submatrix of the transformation matrix.

(77) The image processing apparatus according to any one of (61) to (76), in which

the derivation unit derives a first transformation matrix, using the submatrix, and further derives a second transformation matrix, using the derived first transformation matrix, and

30 the orthogonal transform unit orthogonally transforms the prediction residual, using the second

transformation matrix derived by the derivation unit.

(78) The image processing apparatus according to (77), in which

the derivation unit derives the second
5 transformation matrix of a transform type different from that of the first transformation matrix, using the first transformation matrix.

(79) The image processing apparatus according to (77) or (78), in which

10 the derivation unit flips or transposes the first transformation matrix and derives the second transformation matrix.

(80) An image processing method including:

deriving a transformation matrix using a submatrix
15 configuring a part of the transformation matrix;
orthogonally transforming a prediction residual of an image, using the derived transformation matrix; and
encoding coefficient data obtained by orthogonally transforming the prediction residual to generate a bit
20 stream.

[0892]

(81) An image processing apparatus including:

a decoding unit configured to decode a bit stream to obtain coefficient data that is an orthogonally
25 transformed prediction residual of an image;

a derivation unit configured to derive a transformation matrix, using a submatrix configuring a part of the transformation matrix; and

an inverse orthogonal transform unit configured to
30 inversely orthogonally transform the coefficient data obtained by the decoding unit, using the transformation

matrix derived by the derivation unit.

(82) The image processing apparatus according to (81), in which

the derivation unit derives the transformation
5 matrix, using the submatrix to be stored in a lookup table.

(83) The image processing apparatus according to (81) or (82), in which

the inverse orthogonal transform unit performs
10 inverse secondary transform for the coefficient data obtained by the decoding unit, and further performs inverse primary transform for an inverse secondary transform result, using the transformation matrix derived by the derivation unit.

15 (84) The image processing apparatus according to (83), in which

the derivation unit derives the transformation matrix for horizontal inverse one-dimensional orthogonal transform and the transformation matrix for vertical
20 inverse one-dimensional orthogonal transform, and

the inverse orthogonal transform unit performs, as the inverse primary transform,

the horizontal inverse one-dimensional orthogonal transform, using the transformation matrix for
25 horizontal inverse one-dimensional orthogonal transform derived by the derivation unit, and further,

the vertical inverse one-dimensional orthogonal transform, using the transformation matrix for vertical inverse one-dimensional orthogonal transform
30 derived by the derivation unit.

(85) The image processing apparatus according to

any one of (81) to (84), in which

the derivation unit derives the transformation matrix by flipping the submatrix around an axis of a predetermined direction passing through a center of the transformation matrix and deriving a remaining submatrix
5 of the transformation matrix.

(86) The image processing apparatus according to (85), in which

the derivation unit flips the submatrix in a
10 direction parallel to a row.

(87) The image processing apparatus according to (85) or (86), in which

the derivation unit flips the submatrix in a direction parallel to a column.

15 (88) The image processing apparatus according to any one of (85) to (87), in which

the derivation unit flips the submatrix in a rotation direction around the center of the transformation matrix.

20 (89) The image processing apparatus according to any one of (85) to (88), in which

the derivation unit flips the submatrix in an oblique direction around a diagonal line of the transformation matrix.

25 (90) The image processing apparatus according to any one of (85) to (89), in which

the derivation unit further inverts a sign of an element of the flipped submatrix, and derives the transformation matrix.

30 (91) The image processing apparatus according to any one of (81) to (90), in which

the submatrix is a left-half submatrix or a right-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the transformation matrix by flipping the submatrix in a row
5 direction of the transformation matrix, further inverting a sign of an odd-numbered row vector of the flipped submatrix, and deriving the right-half submatrix or the left-half submatrix of the transformation matrix.

(92) The image processing apparatus according to
10 any one of (81) to (91), in which

the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the transformation matrix by flipping the submatrix in a
15 column direction of the transformation matrix, further inverting a sign of an odd-numbered column vector of the flipped submatrix, and deriving the lower-half submatrix or the upper-half submatrix of the transformation matrix.

(93) The image processing apparatus according to
20 any one of (81) to (92), in which

the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the transformation matrix by flipping the submatrix in a
25 rotation direction around a center of the transformation matrix, further inverting signs of an element having an even row number and an even column number and an element having an odd row number and an odd column number of the flipped submatrix, and deriving the lower-half submatrix
30 or the upper-half submatrix of the transformation matrix.

(94) The image processing apparatus according to

any one of (81) to (93), in which

the submatrix is a submatrix of an upper right triangular portion or a submatrix of a lower left triangular portion of the transformation matrix, and

5 the derivation unit is configured to derive the transformation matrix by transposing the submatrix to derive the submatrix of a lower left triangular portion or the submatrix of an upper right triangular portion of the transformation matrix.

10 (95) The image processing apparatus according to any one of (81) to (94), in which

the submatrix is a submatrix of an upper left triangular portion or a submatrix of a lower right triangular portion of the transformation matrix, and

15 the derivation unit is configured to derive the transformation matrix by flipping the submatrix in an oblique direction having, as an axis, a diagonal line connecting an upper right end and a lower left end of the transformation matrix, further inverting a sign of an
20 odd-numbered row vector, and deriving the submatrix of a lower right triangular portion or the submatrix of an upper left triangular portion of the transformation matrix.

(96) The image processing apparatus according to
25 any one of (81) to (95), in which

the submatrix is an upper left quarter submatrix of the transformation matrix, and

the derivation unit is configured to derive the transformation matrix by

30 flipping the submatrix in a row direction of the transformation matrix, further inverting a sign of an

odd-numbered row vector of the flipped submatrix, and
 deriving an upper right quarter submatrix of the
 transformation matrix,

flipping the submatrix in a column direction
 5 of the transformation matrix, further inverting a sign of
 an odd-numbered column vector of the flipped submatrix,
 and deriving a lower left quarter submatrix of the
 transformation matrix, and

flipping the submatrix in a rotation
 10 direction around a center of the transformation matrix,
 further inverting signs of an element having an even row
 number and an even column number and an element having an
 odd row number and an odd column number of the flipped
 submatrix, and deriving a lower right quarter submatrix
 15 of the transformation matrix.

(97) The image processing apparatus according to
 any one of (81) to (96), in which

the derivation unit derives a first transformation
 matrix, using the submatrix, and further derives a second
 20 transformation matrix, using the derived first
 transformation matrix, and

the inverse orthogonal transform unit inversely
 orthogonally transforms the coefficient data, using the
 second transformation matrix derived by the derivation
 25 unit.

(98) The image processing apparatus according to
 (97), in which

the derivation unit derives the second
 transformation matrix of a transform type different from
 30 that of the first transformation matrix, using the first
 transformation matrix.

(99) The image processing apparatus according to (97) or (98), in which

the derivation unit flips or transposes the first transformation matrix and derives the second

5 transformation matrix.

(100) An image processing method including:

decoding a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image;

10 deriving a transformation matrix, using a submatrix configuring a part of the transformation matrix; and

inversely orthogonally transforming the obtained coefficient data, using the derived transformation matrix.

15 REFERENCE SIGNS LIST

[0893]

100 Image encoding device

101 Control unit

113 Orthogonal transform unit

20 115 Encoding unit

118 Inverse orthogonal transform unit

152 Primary transform unit

200 Image decoding device

212 Decoding unit

25 214 Inverse orthogonal transform unit

253 Inverse primary transform unit

311 Primary transform selection unit

312 Primary horizontal transform unit

313 Primary vertical transform unit

30 321 Transformation matrix derivation unit

331 Transformation matrix LUT

	332	Flip unit
	333	Transposition unit
	351	Transformation matrix derivation unit
	361	Transformation matrix LUT
5	362	Flip unit
	363	Transposition unit
	411	Inverse primary transform selection unit
	412	Inverse primary vertical transform unit
	413	Inverse primary horizontal transform unit
10	421	Transformation matrix derivation unit
	431	Transformation matrix LUT
	432	Flip unit
	433	Transposition unit
	451	Transformation matrix derivation unit
15	461	Transformation matrix LUT
	462	Flip unit
	463	Transposition unit
	501, 502, 511, and 512	Sign inversion unit
	551 and 552	Prediction residual permutation operation
20		unit
	561, 562, 571, and 572	Sign inversion unit
	632	Horizontal flip unit
	633	Vertical flip unit
	634	Transposition unit
25	635	Sign inversion unit
	642	Horizontal flip unit
	643	Vertical flip unit
	644	Transposition unit
	645	Sign inversion unit
30	652	Horizontal flip unit
	653	Vertical flip unit

- 654 Transposition unit
- 655 Sign inversion unit
- 662 Horizontal flip unit
- 663 Vertical flip unit
- 5 664 Transposition unit
- 665 Sign inversion unit

CLAIMS

1. An image processing apparatus comprising:
a derivation unit configured to derive a
5 transformation matrix, using a submatrix configuring a
part of the transformation matrix;
an orthogonal transform unit configured to
orthogonally transform a prediction residual of an image,
using the transformation matrix derived by the derivation
10 unit; and
an encoding unit configured to encode coefficient
data obtained by orthogonally transforming the prediction
residual by the orthogonal transform unit to generate a
bit stream.
15
2. The image processing apparatus according to claim 1,
wherein
the derivation unit derives the transformation
matrix, using the submatrix to be stored in a lookup
20 table.
3. The image processing apparatus according to claim 1,
wherein
the orthogonal transform unit performs primary
25 transform for the prediction residual, using the
transformation matrix derived by the derivation unit, and
further performs secondary transform for a result of the
primary transform.
- 30 4. The image processing apparatus according to claim 3,
wherein

the derivation unit derives the transformation matrix for horizontal one-dimensional orthogonal transform and the transformation matrix for vertical one-dimensional orthogonal transform, and

5 the orthogonal transform unit performs, as the primary transform,

the horizontal one-dimensional orthogonal transform, using the transformation matrix for horizontal one-dimensional orthogonal transform derived by the derivation unit, and further,

10

the vertical one-dimensional orthogonal transform, using the transformation matrix for vertical one-dimensional orthogonal transform derived by the derivation unit.

15

5. The image processing apparatus according to claim 1, wherein

the derivation unit derives the transformation matrix by flipping the submatrix around an axis of a predetermined direction passing through a center of the transformation matrix and deriving a remaining submatrix of the transformation matrix.

20

6. The image processing apparatus according to claim 5, wherein

25

the derivation unit further inverts a sign of an element of the flipped submatrix, and derives the transformation matrix.

30 7. The image processing apparatus according to claim 1, wherein

the submatrix is a left-half submatrix or a right-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the transformation matrix by flipping the submatrix in a row
5 direction of the transformation matrix, further inverting a sign of an odd-numbered row vector of the flipped submatrix, and deriving the right-half submatrix or the left-half submatrix of the transformation matrix.

10 8. The image processing apparatus according to claim 1, wherein

the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation matrix, and

the derivation unit is configured to derive the
15 transformation matrix by flipping the submatrix in a column direction of the transformation matrix, further inverting a sign of an odd-numbered column vector of the flipped submatrix, and deriving the lower-half submatrix or the upper-half submatrix of the transformation matrix.

20

9. The image processing apparatus according to claim 1, wherein

the submatrix is an upper-half submatrix or a lower-half submatrix of the transformation matrix, and

25 the derivation unit is configured to derive the transformation matrix by flipping the submatrix in a rotation direction around a center of the transformation matrix, further inverting signs of an element having an even row number and an even column number and an element
30 having an odd row number and an odd column number of the flipped submatrix, and deriving the lower-half submatrix

or the upper-half submatrix of the transformation matrix.

10. The image processing apparatus according to claim 1,
wherein

5 the submatrix is a submatrix of an upper right
triangular portion or a submatrix of a lower left
triangular portion of the transformation matrix, and
the derivation unit is configured to derive the
transformation matrix by transposing the submatrix to
10 derive the submatrix of a lower left triangular portion
or the submatrix of an upper right triangular portion of
the transformation matrix.

11. The image processing apparatus according to claim 1,
15 wherein

the submatrix is a submatrix of an upper left
triangular portion or a submatrix of a lower right
triangular portion of the transformation matrix, and
the derivation unit is configured to derive the
20 transformation matrix by flipping the submatrix in an
oblique direction having, as an axis, a diagonal line
connecting an upper right end and a lower left end of the
transformation matrix, further inverting signs of an
element having an even row number and an even column
25 number and an element having an odd row number and an odd
column number of the flipped submatrix, and deriving the
submatrix of a lower right triangular portion or the
submatrix of an upper left triangular portion of the
transformation matrix.

30

12. The image processing apparatus according to claim 1,

wherein

the submatrix is an upper left quarter submatrix of the transformation matrix, and

the derivation unit is configured to derive the
5 transformation matrix by

flipping the submatrix in a row direction of the transformation matrix, further inverting a sign of an odd-numbered row vector of the flipped submatrix, and deriving an upper right quarter submatrix of the
10 transformation matrix,

flipping the submatrix in a column direction of the transformation matrix, further inverting a sign of an odd-numbered column vector of the flipped submatrix, and deriving a lower left quarter submatrix of the
15 transformation matrix, and

flipping the submatrix in a rotation direction around a center of the transformation matrix, further inverting signs of an element having an even row number and an even column number and an element having an
20 odd row number and an odd column number of the flipped submatrix, and deriving a lower right quarter submatrix of the transformation matrix.

13. The image processing apparatus according to claim 1,
25 wherein

the derivation unit derives a first transformation matrix, using the submatrix, and further derives a second transformation matrix, using the derived first transformation matrix, and

30 the orthogonal transform unit orthogonally transforms the prediction residual, using the second

transformation matrix derived by the derivation unit.

14. The image processing apparatus according to claim 1,
wherein

5 the orthogonal transform unit orthogonally
transforms the prediction residual, using a transform
unit (TU) of a quad-tree block structure or a quad tree
plus binary tree (QTBT) block structure as a unit of
processing.

10

15. The image processing apparatus according to claim 1,
wherein

 the encoding unit encodes the coefficient data,
using a coding unit (CU) of a quad-tree block structure
15 or a quad tree plus binary tree (QTBT) block structure as
a unit of processing.

16. An image processing method comprising:

 deriving a transformation matrix using a submatrix
20 configuring a part of the transformation matrix;

 orthogonally transforming a prediction residual of
an image, using the derived transformation matrix; and

 encoding coefficient data obtained by orthogonally
transforming the prediction residual to generate a bit
25 stream.

17. An image processing apparatus comprising:

 a decoding unit configured to decode a bit stream
to obtain coefficient data that is an orthogonally
30 transformed prediction residual of an image;

 a derivation unit configured to derive a

transformation matrix, using a submatrix configuring a part of the transformation matrix; and

an inverse orthogonal transform unit configured to inversely orthogonally transform the coefficient data
5 obtained by the decoding unit, using the transformation matrix derived by the derivation unit.

18. The image processing apparatus according to claim 17, wherein

10 the inverse orthogonal transform unit performs inverse secondary transform for the coefficient data obtained by the decoding unit, and further performs inverse primary transform for a result of the inverse secondary transform, using the transformation matrix
15 derived by the derivation unit.

19. The image processing apparatus according to claim 18, wherein

the derivation unit derives the transformation
20 matrix for horizontal inverse one-dimensional orthogonal transform and the transformation matrix for vertical inverse one-dimensional orthogonal transform, and

the inverse orthogonal transform unit performs, as the inverse primary transform,

25 the horizontal inverse one-dimensional orthogonal transform, using the transformation matrix for horizontal inverse one-dimensional orthogonal transform derived by the derivation unit, and further,

the vertical inverse one-dimensional
30 orthogonal transform, using the transformation matrix for vertical inverse one-dimensional orthogonal transform

derived by the derivation unit.

20. The image processing apparatus according to claim 17, wherein

5 the derivation unit derives the transformation matrix by flipping the submatrix around an axis of a predetermined direction passing through a center of the transformation matrix and deriving a remaining submatrix of the transformation matrix.

10

21. The image processing apparatus according to claim 17, wherein

the derivation unit derives a first transformation matrix, using the submatrix, and further derives a second transformation matrix, using the derived first transformation matrix, and

15

the inverse orthogonal transform unit inversely orthogonally transforms the coefficient data, using the second transformation matrix derived by the derivation unit.

20

22. The image processing apparatus according to claim 17, wherein

the decoding unit decodes the bit stream, using a coding unit (CU) of a quad-tree block structure or a quad tree plus binary tree (QTBT) block structure as a unit of processing.

25

23. The image processing apparatus according to claim 17, wherein

30

the inverse orthogonal transform unit inversely

orthogonally transforms the coefficient data, using a transform unit (TU) of a quad-tree block structure or a quad tree plus binary tree (QTBT) block structure as a unit of processing.

5

24. An image processing method comprising:

decoding a bit stream to obtain coefficient data that is an orthogonally transformed prediction residual of an image;

10 deriving a transformation matrix, using a submatrix configuring a part of the transformation matrix; and
inversely orthogonally transforming the obtained coefficient data, using the derived transformation matrix.