PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H03M 13/03 — A1

(11) International Publication Number: WO 00/48321

(43) International Publication Date: 17 August 2000 (17.08.00)

(21) International Application Number: PCT/US00/03599

(22) International Filing Date: 10 February 2000 (10.02.00)

(30) Priority Data:
09/249,418 12 February 1999 (12.02.99) US

(71) Applicant: SONY ELECTRONICS, INC. [US/US]; 1 Sony Drive, Park Ridge, NJ 07656 (US).

(72) Inventors: KONDO, Tetsujiro; 12–1255 1–14 Tsumadakita, Atsugi–shi, Kanagawa–Prefecture 243 (JP). CARRIG, James, J.; 1652 Hicks Avenue, San Jose, CA 95125 (US). GHOSAL, Sugata; J–230, 2nd floor, Saket, New Delhi 110017 (IN). OHTA, Kohji; 5–2–10, Todoroki, Setagaya–ku, Tokyo 158–0082 (JP). FUJIMORI, Yasuhiro; 11693 Westshore Court, Cupertino, CA 95014 (US). TAKAHASHI, Yasuaki; 205 Courthouse Sinkawasaki, 152–5 Shimohirama, Saiwai–ku, Kawasaki, Kanagawa 211–0953 (JP). NAKAYA, Hideo; 1215–12 Kozukue–chou, Kohoku–ku, Yokohamashi, Kanagawa–Prefecture 222–0036 (JP).

(74) Agents: SOBRINO, Maria, E. et al.; Blakely, Sokoloff, Taylor & Zafman, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025–1026 (US).

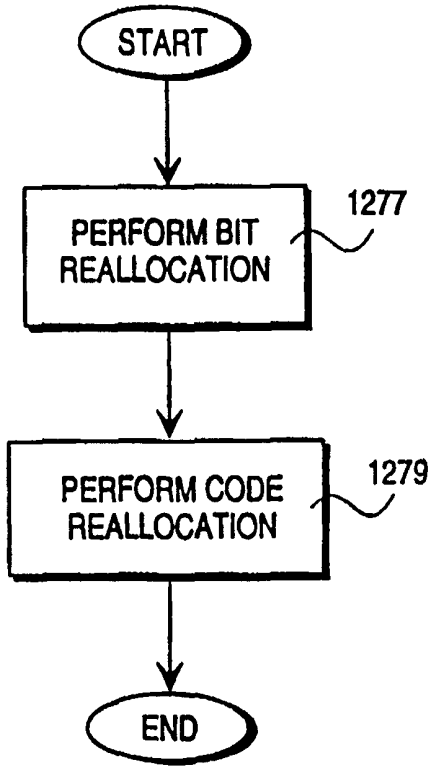
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SOURCE CODING TO PROVIDE FOR ROBUST ERROR RECOVERY

(57) Abstract

Data is encoded to maximize subsequent recovery of lost or damaged compression constants of encoded data. In one embodiment, a compression constant is used to define (1277) a randomization pattern and the data is randomized (1279) using the randomization pattern. In one embodiment, a bit reallocation process and code reallocation process are performed on the data to randomize the data.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SOURCE CODING TO PROVIDE FOR ROBUST ERROR RECOVERY

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to providing a robust error recovery due to data losses incurred during transmission of signals. More particularly, the present invention relates to a data shuffling method used in facilitating a robust error recovery.

2. Art Background

A number of techniques exist for reconstructing lost data due to random errors that occur during signal transmission. However, these techniques cannot handle the loss of consecutive packets of data. Consecutive loss of packets of data is described in the art as burst error. Burst errors result in a reconstructed signal with such a degraded quality that it is easily apparent to the end user. Additionally, compression methodologies used to facilitate high speed communications compound the signal degradation caused by burst errors, thus adding to the degradation of the reconstructed signal. An example of burst error loss affecting transmitted and/or stored signals is seen in high definition television ("HDTV") signals and mobile telecommunication applications wherein compression methodologies play an important role.

The advent of HDTV has led to television systems with a much higher resolution than the current standards proposed by the National Television Systems Committee ("NTSC"). Proposed HDTV signals are predominantly digital. Accordingly, when a color television signal is converted for digital use it is common that the luminance and chrominance signals are digitized using eight bits. Digital transmission of color television requires a nominal bit rate of two hundred and sixteen megabits per second. The transmission rate is greater for HDTV which would nominally require about 1200 megabits per second. Such high transmission rates are well beyond the bandwidths supported by current wireless standards. Accordingly, an efficient compression methodology is required.

Compression methodologies also play an important role in mobile telecommunication applications. Typically, packets of data are communicated between remote terminals in mobile telecommunication applications. The limited number of transmission channels in mobile communications requires an effective compression methodology prior to the transmission of packets. A number of compression techniques are available to facilitate high transmission rates.

Adaptive Dynamic Range Coding ("ADRC") and the discrete cosine transform ("DCT") coding provide image compression techniques known in the art. Both techniques take advantage of the local correlation within an image to achieve a high compression ratio. However, an efficient compression algorithm results in compounded error propagation because errors in an encoded signal are more prominent when subsequently decoded. This error multiplication results in a degraded video image that is readily apparent to the user.

SUMMARY OF THE INVENTION

Data is encoded to maximize subsequent recovery of lost or damaged compression constants of encoded data. In one embodiment, a compression constant is used to define a randomization pattern and the data is randomized using the randomization pattern. In one embodiment, a bit reallocation process and code reallocation process are performed on the data to randomize the data.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the present invention will be apparent to one skilled in the art in light of the following detailed description in which:

Figure 1 generally illustrates the processes of signal encoding, transmission, and decoding.

Figure 2 illustrates one embodiment of a packet structure.

Figure 3 is a flow diagram illustrating one embodiment of the encoding process in accordance with the teachings of the present invention.

Figure 4 is a flow diagram illustrating one embodiment of the decoding process in accordance with the teachings of the present invention.

Figure 5 illustrates one embodiment of image-to-block mapping in accordance with the teachings of the present invention.

Figure 5a illustrates one embodiment of a shuffling pattern used in image-to-block mapping.

Figure 6 is an illustration of exemplary complementary and interlocking block structures.

Figures 7a, 7b, 7c, 7d illustrate one embodiment of shuffling patterns for Y blocks within a frame set.

Figure 8 is an illustration of one embodiment of cumulative DR distribution for Buffer 0.

Figure 8a is an illustration of one embodiment of a partial buffering process in accordance with the teachings of the present invention.

Figure 9 illustrates one embodiment of the intra buffer YUV block shuffling process in accordance with the teachings of the present invention.

Figure 10 illustrates one embodiment of the intra group VL-data shuffling process in accordance with the teachings of the present invention.

Figure 11 illustrates one embodiment of Q code concatenation within a 3-block group in accordance with the teachings of the present invention.

Figure 11a illustrates one embodiment of Q code concatenation for frame pairs including motion blocks in accordance with the teachings of the present invention.

Figure 12 illustrates one embodiment of pixel data error caused by a 1/6 burst error loss.

Figure 12a illustrates one embodiment of shuffling Q codes and distributing Q code bits in accordance with the teachings of the present invention.

Figure 12b illustrates one embodiment of pixel data error caused by a 1/6 burst error loss of redistributed Q codes.

Figure 12c illustrates one embodiment of pixel data error caused by a 1/6 burst error loss of reassigned Q codes.

Figure 12d illustrates one embodiment of a randomization process.

Figures 12e, 12f, 12g and **12h** are examples of randomization processes.

Figure 13 illustrates one embodiment of MIN shuffling in accordance with the teachings of the present invention.

Figure 13a illustrates one embodiment of Motion Flag shuffling and of a fixed length data loss in one frame pair.

Figure 14 illustrates one embodiment of a modular shuffling.

Figure 14a illustrates one embodiment of a modular shuffling result and the fixed length data loss associated with the modular shuffling.

Figure 14b illustrates an alternate embodiment of a modular shuffling result and the fixed length data loss associated with the modular shuffling.

Figure 14c illustrates an alternate embodiment of a modular shuffling result and the fixed length data loss associated with the modular shuffling.

Figure 15 illustrates one embodiment of variable length data buffering in a frame set.

Figure 16 illustrates one embodiment of inter segment VL-data shuffling in accordance with the teachings of the present invention.

DETAILED DESCRIPTION

The present invention provides a method for coding and arranging a signal stream to provide for a robust error recovery. In the following description, for purposes of explanation, numerous details are set forth, in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well known electrical structures and circuits are shown in block diagram form in order not to obscure the present invention unnecessarily.

The signal processing methods and structures are described from the perspective of one embodiment in which the signals are video signals. However, it is contemplated that the methods and apparatus described herein are applicable to a variety of types of signals including audio signals or other digital bitstreams of data, wherein each signal is composed of multiple signal elements. Furthermore

the embodiment of the process described herein utilizes the Adaptive Dynamic Range Coding ("ADRC") process to compress data; however a variety of coding techniques and algorithms may be used. For a more detailed discussion on ADRC, see "Adaptive Dynamic Range Coding Scheme for Future HDTV Digital VTR", Kondo, Fujimori and Nakaya, Fourth International Workshop on HDTV and Beyond, September 4-6, 1991, Turin, Italy.

In the above paper, three different kinds of ADRC are explained. These are achieved according to the following equations:

Non-edge-matching ADRC:

$$DR = MAX - MIN + 1$$

$$q = \left\lfloor \frac{(x - MIN + 0.5) \cdot 2^Q}{DR} \right\rfloor$$

$$x' = \left\lfloor \frac{(q + 0.5) \cdot DR}{2^Q} + MIN \right\rfloor$$

Edge-matching ADRC:

$$DR = MAX - MIN$$

$$q = \left\lfloor \frac{(x - MIN) \cdot (2^Q - 1)}{DR} + 0.5 \right\rfloor$$

$$x' = \left\lfloor \frac{q \cdot DR}{2^Q - 1} + MIN + 0.5 \right\rfloor$$

Multi-stage ADRC:

$$DR = MAX - MIN + 1$$

$$q = \left\lfloor \frac{(x - MIN + 0.5) \cdot 2^Q}{DR} \right\rfloor$$

$$x' = \left\lfloor \frac{(q + 0.5) \cdot DR}{2^Q} + MIN \right\rfloor$$

Where MAX' is the averaged value of x' in the case of $q = 2^Q - 1$;

MIN' is the averaged value of x' in the case of $q = 0$; and

$DR' = MAX' - MIN'$

$$q = \left\lfloor \frac{(x - MIN') \cdot (2^Q - 1)}{DR'} + 0.5 \right\rfloor$$

$$x' = \left\lfloor \frac{q \cdot DR'}{(2^Q - 1)} + MIN' + 0.5 \right\rfloor$$

where MAX represents the maximum level of a block, MIN represents the minimum level of a block, x represents the signal level of each sample, Q represents the number of quantization bits, q represents the quantization code (encoded data), x' represents the decoded level of each sample, and the square brackets $\lfloor \cdot \rfloor$ represent a truncation operation performed on the value within the square brackets.

The signal encoding, transmission, and subsequent decoding processes are generally illustrated in **Figure 1**. Signal 100 is a data stream input to Encoder 110. Encoder 110 follows the Adaptive Dynamic Range Coding ("ADRC") compression algorithm and generates Packets 1, . . . N for transmission along Transmission Media 135. Decoder 120 receives Packets 1, . . . N from Transmission Media 135 and generates Signal 130. Signal 130 is a reconstruction of Signal 100.

Encoder 110 and Decoder 120 can be implemented a variety of ways to perform the functionality described herein. In one embodiment, Encoder 110 and/or Decoder 120 are embodied as software stored on media and executed by a general purpose or specifically configured computer system, typically including a central processing unit, memory and one or more input/output devices and co-processors. Alternately, the Encoder 110 and/or Decoder 120 may be implemented as logic to perform the functionality described herein. In addition, Encoder 110 and/or Decoder 120 can be implemented as a combination of hardware, software or firmware.

In the present embodiment Signal 100 is a color video image comprising a sequence of video frames, each frame including information representative of an image in an interlaced video system. Each frame is composed of two fields, wherein one field contains data of the even lines of the image and the other field containing the odd lines of the image. The data includes pixel values which

describe the color components of a corresponding location in the image. For example, in the present embodiment, the color components consist of the luminance signal Y, and color difference signals U, and V. It is readily apparent the process of the present invention can be applied to signals other than interlaced video signals. Furthermore, it is apparent that the present invention is not limited to implementations in the Y, U, V color space, but can be applied to images represented in other color spaces.

Referring back to **Figure 1**, Encoder 110 divides the Y, U, and V signals and processes each group of signals independently in accordance with the ADRC algorithm. The following description, for purposes of simplifying the discussion, describes the processing of the Y signal; however, the encoding steps are replicated for the U and V signals.

In the present embodiment, Encoder 110 groups Y signals across two subsequent frames, referred to herein as a frame pair, of Signal 100 into three dimensional blocks ("3D") blocks. For one embodiment, a 3D block is generated from grouping two 2D blocks from the same localized area across a given frame pair, wherein a two dimensional 2D block is created by grouping localized pixels within a frame or a field. It is contemplated that the process described herein can be applied to different block structures. The grouping of signals will be further described in the image-to-block mapping section below.

Continuing with the present embodiment, for a given 3D block, Encoder 110 calculates whether there is a change in pixel values between the 2D blocks forming the 3D block. A Motion Flag is set if there are substantial changes in values. As is known in the art, use of a Motion Flag allows Encoder 110 to reduce the number of quantization codes when there is localized image repetition within each frame pair. Encoder 110 also detects the maximum pixel intensity value ("MAX") and the minimum pixel intensity value ("MIN") within a 3D block. Using values MAX and MIN, Encoder 110 calculates the dynamic range ("DR") for a given 3D block of data. For one embodiment DR = MAX - MIN + 1 in the case of non-edge-matching ADRC. For edge-matching ADRC, DR = MAX - MIN. In an alternative

embodiment, Encoder 110 encodes signals on a frame by frame basis for a stream of frames representing a sequence of video frames. In another embodiment, Encoder 110 encodes signals on a field by field basis for a stream of fields representing a sequence of video fields. Accordingly, Motion Flags are not used and 2D blocks are used to calculate the MIN, MAX, and DR values.

In the present embodiment, Encoder 110 references the calculated DR against a threshold table (not shown) to determine the number of quantization bits ("Qbits") used to encode pixels within the block corresponding to the DR. Encoding of a pixel results in a quantization code ("Q code"). The Q codes are the relevant compressed image data used for storage or transmission purposes.

In one embodiment, the Qbit selection is derived from the DR of a 3D block. Accordingly, all pixels within a given 3D block are encoded using the same Qbit, resulting in a 3D encoded block. The collection of Q codes, MIN, Motion Flag, and DR for a 3D encoded block is referred to as a 3D ADRC block. Alternately, 2D blocks are encoded and the collection of Q codes, MIN, and DR for a given 2D block results in 2D ADRC blocks.

A number of threshold tables can be implemented. In one embodiment, the threshold table consists of a row of DR threshold values. A Qbit corresponds to the number of quantization bits used to encode a range of DR values between two adjacent DRs within a row of the threshold table. In an alternative embodiment, the threshold table includes multiple rows and selection of a row depends on the desired transmission rate. Each row in the threshold table is identified by a threshold index. A detailed description of one embodiment of threshold selection is described below in the discussion of partial buffering. A further description of ADRC encoding and buffering is disclosed in US Patent no. 4,722,003 entitled "High Efficiency Coding Apparatus" and US Patent no. 4,845,560 also entitled "High Efficiency Coding Apparatus", assigned to the assignee of the present invention.

Here forth the Q codes are referred to as variable length data ("VL-data"). In addition, the DR, MIN, and Motion Flag are referred to as block attributes. The block attributes, together with the threshold index, constitute the fixed length data

("FL-data"). Furthermore, in view of the above discussion, the term block attribute describes a parameter associated with a component of a signal element, wherein a signal element includes multiple components.

In an alternate embodiment, the FL-data includes a Qbit code. The advantage is that the Qbit information does not have to be derived from the DR during the decoding process. Thus, if the DR information is lost or damaged, the Qbit information can still be determined from the Qbit code. Furthermore, if the Qbit code is lost or damaged, the Qbit information can be derived from DR. Thus the requirement to recover the DR and Qbit is reduced.

The disadvantage to including the Qbit code is the additional bits to be transmitted for each ADRC block. However, in one embodiment, Qbit codes for groups of ADRC blocks are combined, for example, in accordance with a function such as addition or concatenation. For example, if ADRC blocks are grouped in threes and if the Qbit values for each ADRC block are respectively 3, 4 and 4, the summed value that is encoded into the FL-data is 11. Thus the number of bits required to represent the sum is less than the number of bits required to represent each individual value and undamaged Qbit values of the group can be used to determine the Qbit value without performing a Qbit recovery process such as the one described subsequently.

Other embodiments are also contemplated. For example, Motion Flag data may also be encoded. A tag with Qbit and Motion Flag data can be generated and used to reference a table of codes. The configuration and function of the coding can vary according to application.

Frames, block attributes, and VL-data describe a variety of components within a video signal. The boundaries, location, and quantity of these components are dependent on the transmission and compression properties of a video signal. In the present embodiment, these components are varied and shuffled within a bitstream of the video signal to ensure a robust error recovery during transmission losses.

For illustrative purposes, the following description provides for a 1/6 consecutive packet transmission loss tolerance, pursuant to an ADRC encoding and shuffling of a video signal. Accordingly, the following definition and division of components exist for one embodiment. Other embodiments also are contemplated. A data set includes a partition of data of a video or other type of data signal. Thus, in one embodiment, a frame set is a type of data set that includes one or more consecutive frames. A segment includes a memory with the capacity to store a one-sixth division of the Q codes and block attributes included in a frame set. Further, a buffer includes a memory with the capacity to store a one-sixtieth division of the Q codes and block attributes included in a frame set. The shuffling of data is performed by interchanging components within segments and/or buffers. Subsequently, the data stored in a segment is used to generate packets of data for transmission. Thus, in the following description if a segment is lost all the packets generated from the segment are lost during transmission. Similarly, if a fraction of a segment is lost then a corresponding number of packets generated from the segment are lost during transmission.

Although, the following description refers to a 1/6 consecutive packet loss for data encoded using ADRC encoding, it is contemplated that the methods and apparatus described herein are applicable to a design of a 1/n consecutive packets loss tolerance coupled to a variety of encoding/decoding schemes.

Figure 2 illustrates one embodiment of Packet Structure 200 used for the transmission of the data across point-to-point connections as well as networks. Packet Structure 200 is generated by Encoder 110 and is transmitted across Transmission Media 135. For one embodiment, Packet Structure 200 comprises five bytes of header information, eight DR bits, eight MIN bits, a Motion Flag bit, a five bit threshold index, and 354 bits of Q codes. The packet structure described herein is illustrative and may typically be implemented for transmission in an asynchronous transfer mode ("ATM") network. However, the present invention is not limited to the packet structure described and a variety of packet structures that are used in a variety of networks can be utilized.

As noted earlier, Transmission Media (e.g., media) 135 is not assumed to provide error-free transmission and therefore packets may be lost or damaged. As noted earlier, conventional methods exist for detecting such loss or damage, but substantial image degradation will generally occur. The system and methods of the present invention therefore teach source coding to provide robust recovery from such loss or damage. It is assumed throughout the following discussion that a burst loss, that is the loss of several consecutive packets, is the most probable form of error, but some random packet losses might also occur.

To ensure a robust recovery for the loss of one or more consecutive packets of data, the system and methods of the present invention provide multiple level shuffling. In particular, the FL-data and the VL-data included in a transmitted packet comprise data from spatially and temporally disjointed locations of an image. Shuffling data ensures that any burst error is scattered and facilitates error recovery. As will be described below, the shuffling allows recovery of block attributes and Qbit values.

Data Encoding/Decoding

Figure 3 is a flow diagram illustrating one embodiment of the encoding process performed by Encoder 110. **Figure 3** further describes an overview of the shuffling process used to ensure against image degradation and to facilitate a robust error recovery.

In step one of **Figure 3**, an input frame set, also referred to as a display component, is decimated to reduce the transmission requirements. The Y signal is decimated horizontally to three-quarters of its original width and the U and V signals are each decimated to one-half of their original height and one-half of their original width. This results in a 3:1:0 video format with 3960 Y blocks, 660 U blocks and 660 V blocks in each frame pair. As noted earlier, the discussion will describe the processing of Y signals; however, the process is applicable to the U and V signals. At step two, the two Y frame images are mapped to 3D blocks. At step three, 3D blocks are shuffled. At step four, ADRC buffering and encoding is used. At step five, encoded Y, U and V blocks are shuffled within a buffer.

At step six, the VL-data for a group of encoded 3D blocks and their corresponding block attributes are shuffled. At step seven, the FL-data is shuffled across different segments. At step eight, post-amble filling is performed in which variable space at the end of a buffer is filled with a predetermined bitstream. At step nine, the VL-data is shuffled across different segments.

For illustrative purposes the following shuffling description provides a method for manipulation of pixel data before and after encoding. For an alternative embodiment, independent data values are shuffled/deshuffled via hardware. In particular, the hardware maps the address of block values to different addresses to implement the shuffling/deshuffling process. However, address mapping is not possible for data dependent values because shuffling has to follow the processing of data. The intra group VL-data shuffling described below includes the data dependent values. Further, for illustrative purposes the following shuffling description occurs on discrete sets of data. However, for alternative embodiments a signal is defined based on multiple data levels ranging from bits, to pixels, and to frames. Shuffling is possible for each level defined in the signal and across different data levels of the signal.

Figure 4 is a flow diagram illustrating one embodiment of decoding process performed by Decoder 120. Preferably, the conversion and de-shuffling processes are the inverse of the processes represented in **Figure 3**.

Image-to-Block Mapping

In the present embodiment, a single frame typically comprises 5280 2D blocks wherein each 2D block comprises 64 pixels. Thus, a frame pair comprises 5280 3D blocks as a 2D block from a first frame and a 2D block from a subsequent frame are collected to form a 3D block.

Image-to-block mapping is performed for the purpose of dividing a frame or frame set of data into 2D blocks or 3D blocks respectively. Moreover, image-to-block mapping includes using a complementary and/or interlocking pattern to divide pixels in a frame to facilitate robust error recovery during transmission

losses. However, to improve the probability that a given DR value is not too large, each 2D block is constructed from pixels in a localized area.

Figure 5 illustrates one embodiment of an image-to-block mapping process for an exemplary 16 pixel section of an image. Image 500 comprises 16 pixels forming a localized area of a single frame. Each pixel in Image 500 is represented by an intensity value. For example, the pixel in the top left hand side of the image has an intensity value equal to 100 whereas the pixel in the bottom right hand side of the image has an intensity value of 10.

In one embodiment, pixels from different areas of Image 500 are used to create 2D Blocks 510, 520, 530, and 540. 2D Blocks 510, 520, 530, and 540 are encoded, shuffled (as illustrated below), and transmitted. Subsequent to transmission, 2D Blocks 510, 520, 530, and 540 are recombined and used to form Image 550. Image 550 is a reconstruction of Image 500.

To ensure accurate representation of Image 500 despite a possible transmission loss, **Figure 5** is an interlocking complementary block structure, one embodiment of which is illustrated in **Figure 5**, is used to reconstruct Image 550. In particular, the pixel selection used to create 2D Blocks 510, 520, 530, and 540 ensures that a complementary and/or interlocking pattern is used to recombine the blocks when Image 550 is reconstructed. Accordingly, when a particular 2D block's attribute is lost during transmission, contiguous sections of Image 550 are not distorted during reconstruction. For example, as illustrated in **Figure 5** the DR of 2D Block 540 is lost during data transmission. However, during reconstruction of Image 550, the decoder utilizes multiple neighboring pixels of neighboring blocks through which a DR can be recovered for the missing DR of 2D Block 540. In addition, as will be subsequently described, the combination of complementary patterns and shifting increases the number of neighboring pixels, preferably maximizing the number of neighboring pixels that originate from other blocks, significantly improving DR and MIN recovery.

Figure 5a illustrates one embodiment of a shuffling pattern used to form 2D blocks in one embodiment of the image-to-block mapping process. An image is

decomposed into two sub-images, Sub-Image 560 and Sub-Image 570, based on alternating pixels. Rectangular shapes are formed in Sub-Image 560 to delineate the 2D block boundaries. For purposes of discussion, the 2D blocks are numbered 0, 2, 4, 7, 9, 11, 12, 14, 16, 19, 21, and 23. Tile 565 illustrates the pixel distribution for a 2D block within Sub-Image 560.

In Sub-Image 570, the 2D block assignment is shifted by eight pixels horizontally and four pixels vertically. This results in a wrap around 2D block assignment and overlap when Sub-Images 560 and 570 are combined during reconstruction. The 2D blocks are numbered 1, 3, 5, 6, 8, 10, 13, 15, 17, 18, 20, and 22. Tile 575 illustrates the pixel distribution for a 2D block within Sub-Image 570. Tile 575 is the complementary structure of Tile 565. Accordingly, when a particular block's attribute is lost during transmission, neighboring pixels through which a block attribute can be recovered for the missing 2D block exists. Additionally, an overlapping 2D block of pixels with a similar set of block attributes exist. Therefore, during reconstruction of the image the decoder has multiple neighboring pixels from adjacent 2D blocks through which a lost block attribute can be recovered.

Figure 6 illustrates other complementary and interlocking 2D block structures. Other structures may also be utilized. Similar to **Figure 5**, these 2D block structures illustrated in **Figure 6**, ensure surrounding 2D blocks are present despite transmission losses for a given 2D block. However, Patterns 610a, 610b, and 610d use horizontal and/or vertical shifting during the mapping of pixels to subsequent 2D blocks. Horizontal shifting describes shifting the tile structure in the horizontal direction a predetermined number of pixels prior to beginning a new 2D block boundary. Vertical shifting describes shifting the tile structure in the vertical direction a predetermined number of pixels prior to beginning a new 2D block boundary. In application, horizontal shifting only may be applied, vertical shifting may only be applied, or a combination of horizontal and vertical shifting may be applied.

Pattern 610a illustrates a spiral pattern used for image-to-block mapping. The spiral pattern follows a horizontal shifting to create subsequent 2D blocks during the image-to-block mapping process. Patterns 610b and 610d illustrate complementary patterns wherein pixel selection is moved by a horizontal and vertical shifting to create subsequent 2D blocks during the image-to-block mapping process. Further, Patterns 610b and 610d illustrate alternating offsets on pixels selection between 2D blocks. Pattern 610c illustrates using an irregular sampling of pixels to create a 2D block for image-to-block mapping. Accordingly, the image-to-block mapping follows any mapping structure provided a pixel is mapped to a 2D block only once.

Figure 5, Figure 5a and **Figure 6** describe image-to-block mapping for 2D block generation. It is readily apparent that the processes are applicable to 3D blocks. As described above, 3D block generation follows the same boundary definition as a 2D block, however the boundary division extends across a subsequent frame resulting in a 3D block. In particular, a 3D block is created by collecting the pixels used to define a 2D block in a first frame together with pixels from a 2D block in a subsequent frame. In one embodiment, both pixels in the 2D block from the first frame and the 2D block from the subsequent frame are from the exact same location.

Intra Frame Set Block Shuffling

The pixels values for a given image are closely related for a localized area. However, in another area of the same images the pixel values may have significantly different values. Thus, subsequent to encoding the DR and MIN values for spatially close 2D or 3D blocks in a section of an image have similar values, whereas the DR and MIN values for blocks in another section of the image may be significantly different. Accordingly, when buffers are sequentially filled with encoded data from spatially close 2D or 3D blocks of an image, a disproportionate usage of buffer space occurs. Intra frame set block shuffling occurs prior to ADRC encoding and includes shuffling the 2D or 3D blocks

generated during the image-to-block mapping process. This shuffling process ensures an equalized buffer usage during a subsequent ADRC encoding.

Figures 7a - 7d illustrate one embodiment of shuffling 3D Y-blocks. The 3D Y-blocks in **Figures 7a-7d** are generated from applying the image-to-block mapping process described above to a frame pair containing only Y signals. The 3D Y-blocks are shuffled to ensure that the buffers used to store the encoded frame pair contain 3D Y-blocks from different parts of the frame pair. This leads to similar DR distribution during ADRC encoding. A similar DR distribution within each buffer leads to consistent buffer utilization.

Figure 7a -7d also illustrate 3D block shuffling using physically disjointed 3D blocks to ensure that transmission loss of consecutive packets results in damaged block attributes scattered across the image, as opposed to a localized area of the image.

The block shuffling is designed to widely distribute block attributes in the event of small, medium, or large, burst packet losses occur. In the present embodiment, a small burst loss is thought of as one where a few packets are lost; a medium loss is one in which the amount of data that can be held in one buffer is lost; and a large loss is one in which the amount of data that can be held in one segment is lost. During the 3D block shuffling each group of three adjacent blocks are selected from relatively remote parts of the image. Accordingly, during the subsequent intra group VL-data shuffling (to be detailed later), each group is formed from 3D blocks that have differing statistical characteristics. Distributed block attribute losses allow for a robust error recovery because a damaged 3D block is surrounded by undamaged 3D blocks and the undamaged 3D blocks can be used to recover lost data.

Figure 7a illustrates a frame pair containing 66 3D Y-blocks in the horizontal direction and 60 3D Y-blocks in the vertical direction. The 3D Y-blocks are allocated into Segments 0 - 5. As illustrated, the 3D Y-block assignment follows a two by three column section such that one 3D Y-block from each section is associated with a segment. Thus, if no further shuffling is performed and a burst loss of the first 880

packets occurs, all the block attributes associated with Segment 0 are lost. However, as later described, FL-data shuffling is performed to further disperse block attribute losses.

Figure 7b illustrates the scanning order of 3D Y-blocks numbered "0" used to enter into Segment 0. Each "0" 3D Y-block of **Figure 7a** is numbered 0, 1, 2, 3, ..., 659 to illustrate their location in the stream that is inputted into Segment 0. Using the block numbering to allocate segment assignment the remaining 3D Y-blocks are inputted into Segments 1 - 5, thus resulting in a frame pair shuffled across multiple segments.

Figure 7c illustrates the 660 3D Y-blocks comprising one segment. The 3D Y-blocks numbered 0 - 65 are inputted into Buffer 0. Similarly the 3D Y-blocks adjacent to the numbered 3D Y-blocks are inputted into Buffer 1. The process is repeated to fill Buffers 2 - 9. Accordingly, damage to a buffer during data transmission results in missing 3D Y-blocks from different parts of the image.

Figure 7d illustrates the final ordering of the "0" 3D Y-blocks across a buffer. 3D Y-blocks 0, 1, and 2 occupy the first three positions in the buffer. The process is repeated for the rest of the buffer. Accordingly, loss of three 3D Y-blocks during data transmission results in missing 3D Y-blocks from distant locations within the image.

Figures 7a-d illustrate one embodiment of 3D block distributions for 3D Y-blocks of a frame set. In alternative embodiments, however, 3D block distributions for 3D U-blocks and 3D V-blocks are available. The 3D U-blocks are generated from applying the image-to-block mapping process, described above, to a frame set containing only U signals. Similarly, 3D V-blocks are generated from applying the image-to-block mapping process to a frame set containing only V signals. Both the 3D U-block and the 3D V-block follow the 3D Y-block distribution described above. However, as previously described, the number of 3D U-blocks and 3D V-blocks each have a 1:6 proportion to 3D Y-blocks.

Figures 7a-d are used to illustrate one embodiment of intra frame set block shuffling for a Y signal such that burst error of up to 1/6 of the packets lost during

transmission is tolerated and further ensures equalized buffer use. It will be appreciated by one skilled in the art that segment, buffer, and ADRC block assignments can be varied to ensure against 1/n burst error loss or to modify buffer utilization.

Partial Buffering

As illustrated in **Figure 3**, the ADRC encoding and buffering processes occur in step four. Dependent on the encoding technique, 2D or 3D blocks generated during the image-to-block mapping process are encoded resulting in 2D or 3D ADRC blocks. A 3D ADRC block, contains Q codes, a MIN value, a Motion Flag, and a DR. Similarly, a 2D ADRC block contains Q codes, a MIN, and a DR. A 2D ADRC block, however, does not include a Motion Flag because the encoding is performed on a single frame or a single field.

A number of buffering techniques are found in the prior art (see for example, High Efficiency Coding Apparatus, U.S. Patent 4,845,560 of Kondo et. al. and High Efficiency Coding Apparatus, U.S. Patent 4,722,003 of Kondo). Both High Efficiency Coding Apparatus patents are hereby incorporated by reference.

The partial buffering process set forth below, describes an innovative method for determining the encoding bits used in ADRC encoding. In particular, partial buffering describes a method of selecting threshold values from a threshold table designed to provide a constant transmission rate between remote terminals while restricting error propagation. In an alternative embodiment, the threshold table is further designed to provide maximum buffer utilization. In one embodiment, a buffer is a memory that stores a one-sixtieth division of encoded data from a given frame set. The threshold values are used to determine the number of Qbits used to encode the pixels in 2D or 3D blocks generated from the image-to-block mapping process previously described.

The threshold table includes rows of threshold values, also referred to as a threshold set, and each row in the threshold table is indexed by a threshold index. In one embodiment, the threshold table is organized with threshold sets that generate a higher number of Q code bits located in the upper rows of the threshold

table. Accordingly, for a given buffer having a predetermined number of bits available, Encoder 110 moves down the threshold table until a threshold set that generates less than a predetermined number of bits is encountered. The appropriate threshold values are used to encode the pixel data in the buffer.

In one embodiment, a transmission rate of no more than 30 Mbps is desired. The desired transmission rate results in 31,152 bits available for VL-data storage in any given buffer. Accordingly, for each buffer a cumulative DR distribution is computed and a threshold set is selected from the threshold table to encode the pixels in 3D or 2D blocks into VL-data.

Figure 8 illustrates one embodiment of selected threshold values and the DR distribution for Buffer 0. The vertical axis of **Figure 8** includes the cumulative DR distribution. For example, the value "b" is equal to the number of 3D or 2D blocks whose DR is greater than or equal to L_3. The horizontal axis includes the possible DR values. In one embodiment, DR values range from 0 to 255. Threshold values L_4, L_3, L_2, and L_1 describe a threshold set used to determine the encoding of a buffer.

In one embodiment, all blocks stored in Buffer 0 are encoded using threshold values L_4, L_3, L_2, and L_1. Accordingly, blocks with DR values greater than L_4 have their pixel values encoded using four bits. Similarly, all pixels belonging to blocks with DR values between L_3 and L_4 are encoded using three bits. All pixels belonging to blocks with DR values between L_2 and L_3 are encoded using two bits. All pixels belonging to blocks with DR values between L_1 and L_2 are encoded using one bit. Finally, all pixels belonging to blocks with DR values smaller than L_1 are encoded using zero bits. L_4, L_3, L_2, and L_1 are selected such that the total number of bits used to encode all the blocks in Buffer 0 is as close as possible to a limit of 31,152 bits without exceeding the limit of 31,152.

Figure 8a illustrates the use of partial buffering in one embodiment. Frame 800 is encoded and stored in Buffers 0 - 59. Provided a transmission error inhibits data recovery, the decoding process is stalled for Frame 800 until error recovery is

performed on the lost data. However, partial buffering restricts the error propagation within a buffer, thus allowing decoding of the remaining buffers. In one embodiment, a transmission error inhibits the Qbit and Motion Flag recovery for Block 80 in Buffer 0. Partial buffering limits the error propagation to the remaining blocks within Buffer 0. Error propagation is limited to Buffer 0 because the end of Buffer 0 and the beginning of Buffer 1 are known due to the fixed buffer length. Accordingly, Decoder 120 can begin processing of blocks within Buffer 1 without delay. Additionally, the use of different threshold sets to encode different buffers allows Encoder 110 to maximize/control the number of Q codes bits included in a given buffer, thus allowing a higher compression ratio. Furthermore, the partial buffering process allows for a constant transmission rate because Buffers 0 - 59 consist of a fixed length.

In one embodiment, a buffer's variable space is not completely filled with Q code bits because a limited number of threshold sets exist. Accordingly, the remaining bits in the fixed length buffer are filled with a predetermined bitstream pattern referred to as a post-amble. As will be described subsequently, the post-amble enables bidirectional data recovery because the post-amble delineates the end of the VL-data prior to the end of the buffer.

Intra Buffer YUV Block Shuffling

Y, U, and V, signals each have unique statistical properties. To improve the Qbit and Motion Flag recovery process (described below) the Y, U, and V signals are multiplexed within a buffer. Accordingly, transmission loss does not have a substantial effect on a specific signal.

Figure 9 illustrates one embodiment of the intra buffer YUV block shuffling process in which YUV ADRC blocks are derived from the Y, U, and V signals respectively. Buffer 900 illustrates the ADRC block assignments after intra frame set block shuffling. Buffer 900 comprises 66 Y-ADRC blocks followed by 11 U-ADRC blocks which are in turn followed by 11 V-ADRC blocks. Buffer 910 shows the YUV ADRC block organization after intra buffer YUV block shuffling. As illustrated, three Y-ADRC blocks are followed by a U-ADRC block or three Y-

ADRC blocks are followed by a V-ADRC block. Intra buffer YUV block shuffling reduces similarity between adjacent block's bitstreams within the buffer. Alternative embodiments of intra buffer YUV block shuffling with a different signal, i.e., YUV ratios or other color spaces are possible dependent on the initial image format.

Intra Group VL-Data Shuffling

In one embodiment, Intra group VL-data shuffling comprises three processing steps. The three processing steps include Q code concatenation, Q code reassignment, and randomizing concatenated Q codes. **Figure 10** illustrates one embodiment of intra group VL-data shuffling wherein three processing steps are applied consecutively to Q codes stored in a buffer. In alternative embodiments, one or more processing steps discussed herein may be applied to perform intra group VL-data shuffling. Each processing step independently assists in the error recovery of data lost during transmission. Accordingly, each processing step is described independently.

1. Q code concatenation

Q code concatenation ensures that groups of ADRC blocks are decoded together. Group decoding facilitates error recovery because additional information is available from neighboring blocks during the data recovery process detailed below. For one embodiment, Q code concatenation is applied independently to each group of three ADRC blocks stored in a buffer. In an alternative embodiment, a group includes ADRC block(s) from different buffers. The concatenation of Q codes across three ADRC blocks is described as generating one concatenated ADRC tile. **Figure 11** and **Figure 11a** illustrate one embodiment of generating concatenated ADRC tiles.

Figure 11 illustrates one embodiment of generating a concatenated ADRC tile from 2D ADRC blocks. Specifically, the concatenation is performed for each Q code (q_0 - q_{63}) included in 2D ADRC Blocks 0, 1, and 2 resulting in the sixty four Q codes of Concatenated ADRC Tile A. For example, the first Q code $q_{0,0}$ (0th

quantized value) of 2D ADRC Block 0 is concatenated to the first Q code $q_{0,1}$ of 2D ADRC Block 1. The two concatenated Q codes are in turn concatenated to the first Q code $q_{0,2}$ of 2D ADRC Block 2, thus resulting in Q_0 of Concatenated ADRC Tile A. The processes is repeated until Q_{63} is generated. Alternatively, the generation of Q_i in Concatenated ADRC Tile A is described by the equation

$$Q_i = [\, q_{i,0} , q_{i,1} , q_{i,2} \,] \qquad i= 0, 1, 2, \ldots 63$$

Additionally, associated with each Q_i in Concatenated ADRC Tile A there is a corresponding number of N bits that represents the total number of bits concatenated to generate a single Q_i.

Figure 11a illustrates one embodiment of generating a concatenated ADRC tile from frame pairs including motion blocks. A motion block is a 3D ADRC block with a set Motion Flag. The Motion Flag is set when a predetermined number of pixels within two 2D blocks structure created by image-to-block mapping process described earlier, change in value between a first frame and a subsequent frame. In an alternative embodiment, the Motion Flag is set when the maximum value of each pixel change between the 2D block of a first frame and a subsequent frame exceeds a predetermined value. In contrast, non-motion (i.e., stationary) block includes a 3D ADRC block with a Motion Flag that is not set. The Motion Flag remains un-set when a predetermined number of pixels within the two 2D blocks of a first frame and a subsequent frame do not change in value. In an alternative embodiment, the Motion Flag remains un-set when the maximum value of each pixel change between a first frame and a subsequent frame does not exceed a predetermined value.

A motion block includes Q codes from an encoded 2D block in a first frame and an encoded 2D block in a subsequent frame. The collection of Q codes corresponding to a single encoded 2D block are referred to as an ADRC tile. Accordingly, a motion block generates two ADRC tiles. However, due to the lack of motion, a stationary block need only include one-half of the number of Q codes of a motion block, thus generating only one ADRC tile. In the present embodiment,

the Q codes of a stationary block are generated by averaging corresponding pixels values between a 2D block in a first frame and a corresponding 2D block in a subsequent frame. Each averaged pixel value is subsequently encoded resulting in the collection of Q codes forming a single ADRC tile. Accordingly, Motion Blocks 1110 and 1130 generate ADRC Tiles 0, 1, 3, and 4. Stationary Block 1120 generates ADRC Tile 2.

The concatenated ADRC tile generation of **Figure 11a** concatenates the Q codes for ADRC Tiles 0 - 4 into Concatenated ADRC Tile B. Specifically, the concatenation is performed for each Q code (q_0 - q_{63}) included in ADRC Tiles 0, 1, 2, 3 and 4 resulting in the sixty four Q codes of Concatenated ADRC Tile B. Alternatively, the generation of each Q code, Q_i, in Concatenated ADRC Tile B is described by the mathematical equation

$$Q_i = [\, q_{i,0} , q_{i,1} , q_{i,2} , q_{i,3} , q_{i,4} \,] \qquad i= 0, 1, 2, \ldots 63$$

2. Q code reassignment

Q code reassignment ensures that bit errors caused by transmission losses are localized within spatially disjointed pixels. In particular, during Q code reassignment, Q codes are redistributed and the bits of the redistributed Q codes are shuffled. Accordingly, Q code reassignment facilitates error recovery because undamaged pixels surround each damaged pixel. Furthermore, DR and MIN recovery is aided because pixel damage is distributed evenly throughout an ADRC block.

Figure 12 illustrates one embodiment of pixel corruption during the transmission loss of a 1/6 burst error loss. In particular, 2D ADRC Blocks 1210, 1220, and 1230 each include sixty four pixels encoded using three bits. Accordingly, each pixel, P_0 through P_{63}, of a 2D ADRC block is represented by three bits. 2D ADRC Block 1210 shows the bit loss pattern, indicated by a darkened square, of bits when the first bit of every six bits are lost. Similarly, the bit loss pattern when the second bit or fourth bit of every six bits are lost are shown in 2D ADRC Blocks 1220 and 1230, respectively. **Figure 12** illustrates that without Q code reassignment one-

half of all the pixels 2D ADRC Blocks 1210, 1220, and 1230 are corrupted for a 1/6 burst error loss.

For one embodiment, Q code reassignment is applied independently to each concatenated ADRC tile stored in a buffer, thus ensuring that bit errors are localized within spatially disjointed pixels upon deshuffling. In an alternative embodiment, Q code reassignment is applied to each ADRC block stored in a buffer.

Figure 12a illustrates one embodiment of Q code reassignment that generates a bitstream of shuffled Q code bits from a concatenated ADRC tile. Table 122 and Table 132 illustrate the Q code redistribution. Bitstreams 130 and 140 illustrate the shuffling of Q code bits.

Table 122 shows the concatenated Q codes for Concatenated ADRC Tile A. Q_0 is the first concatenated Q code and Q_{63} is the final concatenated Q code. Table 132 illustrates the redistribution of Q codes. For one embodiment Q_0, Q_6, Q_{12}, Q_{18}, Q_{24}, Q_{30}, Q_{36}, Q_{42}, Q_{48}, Q_{54}, and Q_{60} are included in a first set, partition 0. Following Table 132, the following eleven concatenated Q codes are included in partition 1. The steps are repeated for partitions 2 - 5. The boundary of a partition is delineated by a vertical line in Table 132. This disjointed spatial assignment of concatenated Q codes to six partitions ensures that a 1/6 burst error loss results in a bit loss pattern distributed across a group of consecutive pixels.

Figure 12b illustrates one embodiment of the bit pattern loss created by the 1/6 burst error loss of redistributed Q codes. In particular, 2D ADRC blocks 1215, 1225, and 1235 each include sixty four pixels encoded using three bits. Accordingly, each pixel P_0 through P_{63}, of each 2D ADRC block, is represented by three bits. In 2D ADRC Blocks 1215, 1225, and 1235 the bit loss pattern, indicated by a darkened square, is localized across a group of consecutive pixels. Accordingly, only eleven consecutive pixels within each 2D ADRC Block 1215, 1225, and 1235 are corrupted for a given segment loss. In an alternative embodiment, Q code assignment to partitions include Q codes from different motion blocks, thus providing both a

disjointed spatial and temporal assignment of Q codes to six segments. This results in additional undamaged spatial-temporal pixels during a 1/6 burst error loss and further facilitates a more robust error recovery.

Referring to **Figure 12a**, the bits of the redistributed Q codes in Table 132 are shuffled across a generated bitstream so that adjacent bits in the bitstream are from adjacent partitions. The Q code bits for all the partitions in Table 132 are concatenated into Bitstream 130. For a given partition adjacent bits in Bitstream 130 are scattered to every sixth bit location in the generated Bitstream 140. Accordingly, bits number zero through five, of Bitstream 140, include the first bit from the first Q code in each partition. Similarly, bits number six through eleven, of Bitstream 140, include the second bit from the first Q code in each partition. The process is repeated for all Q code bits. Accordingly, a 1/6 burst error loss will result in a spatially disjointed pixel loss.

Figure 12c illustrates one embodiment of the bit pattern loss created by the 1/6 burst error loss of reassigned (i.e. redistributed and shuffled) Q codes. In particular, 2D ADRC Blocks 1217, 1227, and 1237 each include sixty four pixels encoded using three bits. Accordingly, each pixel P_0 through P_{63}, of each 2D ADRC Block, is represented by three bits. In 2D ADRC Blocks 1217, 1227, and 1237, the bit loss pattern, indicated by a darkened square, is distributed across spatially disjointed pixels, thus facilitating pixel error recovery.

3. Randomization of Q codes bits

In one embodiment, Q code bit randomization process is composed of two steps as shown in **Figure 12d**; however, the process may be performed using only either one of the two steps.

Randomization may be applied to destroy the correlation of incorrect candidate decodings that may be generated during a subsequent data decoding process in order to estimate lost or damaged data. The randomization process does not change the properties of the correct candidate decoding, as it is restored to its original condition. In particular, by utilizing randomization across multiple blocks

of data, subsequent derandomized data will tend to result in candidate decodings that exhibit highly correlated properties indicative that the corresponding candidate decoding is not a good selection. The randomization process is chosen such that a correct derandomization results in candidate decoding exhibiting highly correlated properties and an incorrect derandomization results in a decoding exhibiting uncorrelated properties. Encoding parameters may be used to perform the randomization and derandomization processes. For example, a randomization pattern may be chosen based on the values for the compression constants.

One embodiment of a randomization process is illustrated in **Figure 12d**. At step 1277, a bit reallocation is performed. At step 1279 a code reallocation is performed. As noted above, steps 1277 and 1279 each may be performed independently and still realize some coding benefits. In addition, steps 1277 and 1279 may be executed in an order different than illustrated in **Figure 12d**.

In one embodiment, as discussed above, randomization is achieved using a code reallocation process. In one embodiment, reallocation is performed using a masking key. Thus, during the encoding process, a key, referred to herein as KEY, is used to mask a bitstream of Q codes. KEY may be used to mask a bitstream of Q codes corresponding to three blocks of data. Each key element (d_i) of the masking key is generated by the combination of certain compression constants used to encode a corresponding block of data. This process may enhance error localization.

For example, in one embodiment, the MF and Qbit values are used to define KEY. Alternately, the masking key is generated from DR and MIN values. More particularly, for 4 bit ADRC encoding which uses MR and Qbit values to generate KEY, the value of the key elements composing KEY are determined in accordance with the following equation:

$$d_i = 5 \bullet m_i + q_i \qquad \text{where } i = 0, 1, 2$$

and q_i represents the number of quantization bits; $q_i = 0, 1, 2, 3, 4$ and m_i represents the motion flag (MF) value, for example, 0 for a stationary block and 1 for a motion block.

Continuing with the present example, if KEY is generated using three blocks, KEY is formed according to the following:

$$KEY = d_0 + 10 \bullet d_1 + 100 \bullet d_2$$

If KEY is generated using N blocks, KEY is defined according to the following:

$$KEY = \sum_{i=0}^{N-1} 10^i \cdot d_i$$

The result of the process is a randomization of Q codes, for example, as shown in **Figure 12e**.

It therefore follows that during recovery of MF or Qbit data, a bit derandomization process is performed in which possible KEY values are regenerated depending upon the values used to create the masking keys. The regenerated KEY values are used to unmask the received bitstream of Q codes resulting in candidate encoded data. Thus, if the MF or Qbit value used to generate the mask is not correct, the corresponding Q codes will exhibit a low level of correlation, which will be typically readily detectable.

In another embodiment, a randomization process, referred to herein as bit reallocation, is applied to the data. In one embodiment, bit reallocation is achieved by simple bit weight inversion. The inversion pattern is determined according to the number of bits used for encoding (e.g., Qbit). This randomization process can improve recovery of MF and Qbit values. Examples are shown in **Figures 12e**, **12f**, **12g**, and **12h**. **Figure 12f** illustrates a bit reallocation process for 2 bit encoding, **Figure 12g** illustrates a bit reallocation for 3 bit encoding and **Figure 12h** illustrates a bit reallocation for 4 bit encoding.

Figures 10 - 12 illustrate intra group VL-data shuffling tolerated up to 1/6 packet data loss during transmission. It will be appreciated by one skilled in the art, that the number of total partitions and bit separation can be varied to ensure against 1/n burst error loss.

Inter Segment FL-Data Shuffling

Inter segment FL-data shuffling describes rearranging block attributes among different segments. Rearranging block attributes provides for a distributed

loss of data. In particular, when FL-data from a segment is lost during transmission the DR value, MIN value, and Motion Flag value lost do not belong to the same block. **Figures 13** and **14** illustrate one embodiment of inter segment FL-data shuffling.

Figure 13 illustrates the contents of Segments 0 to 5. For one embodiment, each segment comprises 880 DRs, 880 MINs, 880 Motion Flags, and VL-data corresponding to 660 Y-blocks, 110 U-blocks, and 110 V-blocks. As illustrated in graph MIN Shuffling 1300, the MIN values for Segment 0 are moved to Segment 2, the MIN values for Segment 2 are moved to Segment 4, and the MIN values for Segment 4 are moved to Segment 0. Additionally, the MIN values for Segment 1 are moved to Segment 3, the MIN values for Segment 3 are moved to Segment 5, and the Motion Flag values for Segment 5 are moved to Segment 1.

Figure 13a illustrates Motion Flag shuffling. As illustrated, in graph Motion Flag Shuffling 1305, the Motion Flag values for Segment 0 are moved to Segment 4, the Motion Flag values for Segment 2 are moved to Segment 0, and the Motion Flag values for Segment 4 are moved to Segment 2. Additionally, the Motion Flag values for Segment 1 are moved to Segment 5, the Motion Flag values for Segment 3 are moved to Segment 1, and the Motion Flag values for Segment 5 are moved to Segment 3. Loss pattern 1310 illustrates the FL-data loss after Segment 0 is lost during transmission.

For a specific block attribute, both **Figure 13** and **Figure 13a** illustrate shuffling all instances of the specific block attribute between segments. For example, in **Figure 13** the 880 MIN values from Segment 0 are collectively exchanged with the 880 MIN values in Segment 2. Similarly, in **Figure 13a** the 880 Motion Flags for Segment 0 are collectively exchanged with the 880 Motion Flags in Segment 4. During a transmission loss of consecutive packets, this collective shuffling of block attributes results in a disproportionate loss of a specific block attributes for a block group. In one embodiment, a block group includes three ADRC blocks.

Figure 14 illustrates one embodiment of a modular three shuffling process for DR, MIN, and Motion Flag values. A modular three shuffling describes a shuffling pattern shared across three blocks (i.e., a block group) in three different segments. The shuffling pattern is repeated for all block groups within the three different segments. However, a different shuffling pattern is used for different block attributes. Accordingly, the modular three shuffling process distributes block attributes over all three segments. In particular, for a given block group a modular three shuffling ensures that only one instance of a specific block attribute is lost during the transmission loss of a segment. Thus, during the data recovery process, described below, a reduced number of candidate decodings are generated to recover data loss within a block.

As illustrated in DR Modular Shuffle 1410, a segment stores 880 DR values. Accordingly, the DR values are numbered 0 - 879 dependent on the block from which a given DR value is derived. In a modular three shuffling the FL-data contents of three segments are shuffled. A count of 0 - 2 is used to identify each DR value in the three segments identified for a modular shuffling. Accordingly, DR's belonging to blocks numbered 0, 3, 6, 9 . . . belong to Count 0. Similarly, DR's belonging to blocks numbered 1, 4, 7, 10, . . . belong to Count 1 and DR's belonging to blocks numbered 2, 5, 8, 11 . . . belong to Count 2. Thus, for a given count the DR values associated with that count are shuffled across Segment 0, 2, and 4. Similarly, the DR values associated with the same count are shuffled across Segments 1, 3, and 5.

In DR Modular Shuffle 1410, the DR values belonging to Count 0 are left un-shuffled. The DR values belonging to Count 1 are shuffled. In particular, the Count 1 DR values in Segment A are moved to Segment B, the Count 1 DR values in Segment B are moved to Segment C, and the Count 1 DR values in Segment C are moved to Segment A.

The DR values belonging to Count 2 are also shuffled. In particular, the Count 2 DR values in Segment A are moved to Segment C, the Count 2 DR values

in Segment B are moved to Segment A, and the Count 2 DR values in Segment C are moved to Segment B.

MIN Modular Shuffle 1420 illustrates one embodiment of a modular three block attribute shuffling process for MIN values. A segment includes 880 MIN values. In MIN Modular Shuffle 1420, the shuffling pattern used for Count 1 and Count 2 in DR Modular Shuffle 1410 are shifted to Count 0 and Count 1. In particular, the shuffling pattern used for Count 1 in DR Modular Shuffle 1410 is applied to Count 0. The shuffling pattern used for Count 2 in DR Modular Shuffle 1410 is applied to Count 1 and the MIN values belonging to Count 2 are left un-shuffled.

Motion Flag Modular Shuffle 1430 illustrates one embodiment of a modular three block attribute shuffling process for Motion Flag values. A segment includes 880 Motion Flag values. In Motion Flag Modular Shuffle 1430, the shuffling pattern used for Count 1 and Count 2 in DR Modular Shuffle 1410 are shifted to Count 2 and Count 0 respectively. In particular, the shuffling pattern used for Count 2 in DR Modular Shuffle 1410 is applied to Count 0. The shuffling pattern used for Count 1 in DR Modular Shuffle 1410 is applied to Count 2 and the Motion Flag values belonging to Count 1 are left un-shuffled.

Figure 14a illustrates the modular shuffling result of Modular Shuffles 1410, 1420, and 1430. Modular Shuffle Result 1416 shows each attribute destination of blocks belonging to Segment 0. In this example, Segment 0 corresponds to Segment A of **Figure 14**. This destination is defined according to Modular Shuffles 1410, 1420, and 1430 of **Figure 14**. **Figure 14a** also illustrates the distribution loss of block attributes after Segment 0 is lost during transmission. In particular, Loss Pattern 1415 shows the DR, Motion Flag, and MIN values loss across six segments after a subsequent deshuffling is applied to the received data that was initially shuffled using Modular Shuffles 1410, 1420, and 1430. As illustrated in **Figure 14a**, the block attribute loss is distributed periodically across Segments 0, 2, and 4 while Segments 1, 3, and 5 have no block attribute loss. Additionally, Spatial Loss Pattern 1417 illustrates the deshuffled spatial distribution of damaged FL-data after Segment 0 is

lost during transmission. Spatial Loss Pattern 1417 shows the DR, Motion Flag, and MIN value loss after a subsequent deshuffling is applied to the received data. In Spatial Loss Pattern 1417, a damaged block is surrounded by undamaged blocks and damaged block attributes can be recovered with surrounding undamaged blocks.

Figure 14 and **Figure 14a** illustrate a modular three shuffling pattern and the distribution loss of block attributes after a segment is lost during transmission. In alternative embodiments, the count variables or the number of segments are varied to alternate the distribution of lost block attributes. **Figure 14b** illustrates Modular Shuffle Result 1421 and Loss Pattern 1420. Similarly, **Figure 14c** illustrates Modular Shuffle Result 1426 and Loss Pattern 1425. Both Loss Pattern 1420 and Loss Pattern 1425 illustrate the distribution loss of block attributes across six segments, as opposed to three segments as previously described.

It is contemplated that in alternate embodiments various combinations of block attributes will be distributed to perform the shuffling process.

Inter Segment VL-Data Shuffling

In the inter segment VL-data shuffling process, bits between a predetermined number of segments, for example, 6 segments, are arranged to ensure a spatially separated and periodic VL-data loss during an up to 1/6 packet transmission loss. **Figure 15** and **16** illustrate one embodiment of the inter segment VL-data shuffling process.

In the present embodiment, a transmission rate approaching 30 Mbps is desired. Accordingly, the desired transmission rate results in 31,152 bits available for the VL-data in each of the 60 buffers. The remaining space is used by FL-data for the eighty eight blocks included in a buffer. **Figure 15** includes the VL-data buffer organization within a frame set for a transmission rate approaching 30 Mbps. As previously described, partial buffering is used to maximize the usage of available VL-data space within each buffer, and the unused VL-data space is filled with a post-amble.

Figure 16 illustrates one embodiment of the shuffling process to ensure a spatially separated and periodic VL-data loss. The first row illustrates the VL-data from the 60 buffers in **Figure 15** rearranged into a concatenated stream of 1,869,120 bits. The second row illustrates the collection of every sixth bit into a new stream of bits. Thus, when the decoder subsequently reverses the process, a burst loss of up to 1/6 of the data transmitted is transformed into a periodic loss where at least 5 undamaged bits separate every set of two damaged bits.

The third row illustrates grouping every 10 bits of Stream 2 into a new stream of bits, Stream 3. The boundary of a grouping is also defined by the number of bits in a segment. Grouping of Stream 2 for every tenth bit ensures that a 1/60 data loss results in fifty-nine undamaged bits between every set of two damaged bits. This provides for a spatially separated and periodic VL-data loss in the event that 88 consecutive packets of data are lost.

The fourth row illustrates grouping every 11 bits of Stream 3 into Stream 4. The boundary of a grouping is also defined by the number of bits in a segment. Grouping of Stream 3 for every eleventh bit ensures that 1/660 data loss results in 659 undamaged bits between to damaged bits, resulting in a spatially separated and periodic VL-data loss during a transmission loss of 8 consecutive packets.

Each group of 31,152 bits within Stream 4 is consecutively re-stored in Buffers 0 - 59, with the first group of bits stored in Buffer 0 and the last group of bits stored in Buffer 59.

It will be appreciated by one skilled in the art that the grouping requirements of **Figure 16** are variable to ensure a spatially separated and periodic VL-data loss tolerance up to a 1/n transmission loss.

Transmission

The previously described shuffling process creates buffers with intermixed FL-data and VL-data. For one embodiment, packets are generated from each buffer, according to packet structure 200, and transmitted across Transmission media 135. The data received is subsequently decoded. Lost or damaged data may be recovered using data recovery processes.

The invention has been described in conjunction with the preferred embodiment. It is evident that numerous alternatives, modifications, variations and uses will be apparent to those skilled in the art in light of the foregoing description.

CLAIMS

What is claimed is:

1. A method for encoding data to maximize subsequent recovery of lost or damaged compression constants of encoded data, said method for encoding comprising the steps of:

using at least one compression constant to define (1277) a randomization pattern; and

randomizing (1279) the data using the randomization pattern.

2. The method as set forth in claim 1, wherein the step of randomizing comprises the step of performing a bit reallocation process.

3. The method as set forth in claim 1, wherein the step of randomizing comprises the step of performing a code reallocation process.

4. The method as set forth in claim 1, wherein the data comprises a plurality of blocks, said step of randomizing is applied to the plurality of blocks.

5. The method as set forth in claim 2, wherein the bit reallocation process is performed by bit weight inversion.

6. The method as set forth in claim 3, wherein code reallocation process is performed by masking the data using a masking key.

7. The method as set forth in claim 6, wherein compression constants are used to define the masking key.

8. The method as set forth in claim 6, wherein the masking key is determined according to the following:

$$KEY = 5 \cdot m + q$$

where q represents the number of quantization bits, and m represents a motion flag value indicative of motion of the data.

9. The method as set forth in claim 6, wherein the masking key for N blocks is determined according to the following:

$$KEY = \sum_{i=0}^{N-1} 10^i \cdot d_i$$

where $d_i = 5 \cdot m_i + q_i$, q_i represents the number of quantization bits of an i-th block, m_i represents the motion flag value of the i-th block, and d_i represents a key element of the i-th block.

10. A system for encoding data to maximize subsequent recovery of lost or damaged compression constants of encoded data, said system comprising:

at least one compression constant;

a randomization pattern defined by the at least one compression constant; and

a randomizer configured to receive the randomization pattern, said randomizer randomizing the data using the randomization pattern.

11. The system as set forth in claim 10, wherein the randomizer performs a bit reallocation process.

12. The system as set forth in claim 10, wherein randomizer performs a code reallocation process.

13. The system as set forth in claim 10, wherein the data comprises a plurality of blocks and the plurality of blocks are randomized.

14. The system as set forth in claim 11, wherein bit reallocation is performed by bit weight inversion.

15. The system as set forth in claim 12, wherein code reallocation is performed by masking the data using a masking key.

16. The system as set forth in claim 17, wherein compression constants are used to define the masking key.

17. The system as set forth in claim 15, wherein the masking key is determined according to the following:

$$KEY = 5 \cdot m + q$$

where q represents the number of quantization bits, and m represents a motion flag value indicative of motion of the data.

18. The system as set forth in claim 15, wherein the masking key for N blocks is determined according to the following:

$$KEY = \sum_{i=0}^{N-1} 10^i \cdot d_i$$

where $d_i = 5 \cdot m_i + q_i$, q_i represents the number of quantization bits of an i-th block, m_i represents the motion flag value of the i-th block, and d_i represents a key element of the i-th block.

19. A computer readable medium containing instructions which, when executed by a processing system perform a process for encoding data to maximize subsequent recovery of lost or damaged compression constants of encoded data comprising:

using at least one compression constant to define (1277) a randomization pattern; and

randomizing (1279) the data using the randomization pattern.

20. The computer readable medium as set forth in claim 19, wherein the randomizing comprises instructions, which when executed, perform a bit reallocation process.

21. The computer readable medium as set forth in claim 19, wherein randomizing comprises instructions, which when executed, perform a code reallocation process.

22. The computer readable medium as set forth in claim 19, wherein the data comprises a plurality of blocks, and the randomizing is applied to the plurality of blocks.

23. The computer readable medium as set forth in claim 20, wherein bit reallocation is performed by bit weight inversion.

24. The computer readable medium as set forth in claim 21, wherein code reallocation is performed by masking the data using a masking key.

25. The computer readable medium as set forth in claim 24, wherein compression constants are used to define the masking key.

26. The computer readable medium as set forth in claim 24, wherein the masking key is determined according to the following:

$$KEY = 5 \cdot m + q$$

where q represents the number of quantization bits, and m represents a motion flag value indicative of motion of the data.

27. The computer readable medium as set forth in claim 24, wherein the masking key for N blocks is determined according to the following:

$$KEY = \sum_{i=0}^{N-1} 10^i \cdot d_i$$

where $d_i = 5 \cdot m_i + q_i$, q_i represents the number of quantization bits of an i-th block, m_i represents the motion flag value of the i-th block, and d_i represents a key element of the i-th block.

28. An apparatus for encoding data to maximize subsequent recovery of lost or damaged compression constants of encoded data comprising:

means for using at least one compression constant to define (1277) a randomization pattern; and

means for randomizing (1279) the data using the randomization pattern.

29. The apparatus as set forth in claim 28, wherein the means for randomizing performs a bit reallocation process.

30. The apparatus as set forth in claim 28, wherein the means for randomizing performs a code reallocation process.


SIGNAL 130
DECODER 120
TRANSMISSION MEDIA 135
PACKETS 1 • • • N
ENCODER 110
SIGNAL 100

FIG. 1

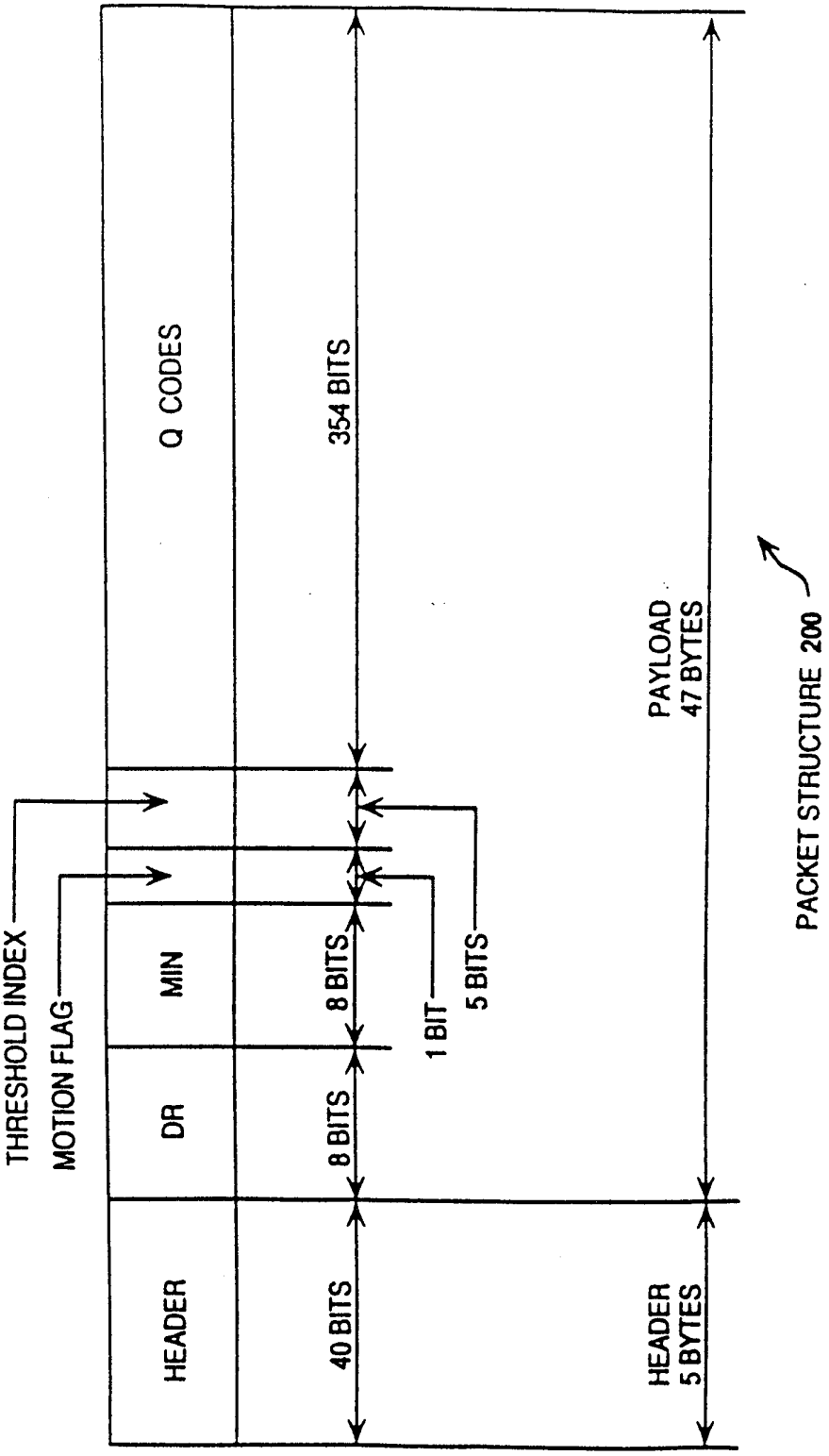

HEADER
DR
MIN
Q CODES
40 BITS
8 BITS
8 BITS
1 BIT
5 BITS
354 BITS
HEADER
5 BYTES
PAYLOAD
47 BYTES
PACKET STRUCTURE 200
THRESHOLD INDEX
MOTION FLAG

FIG. 2

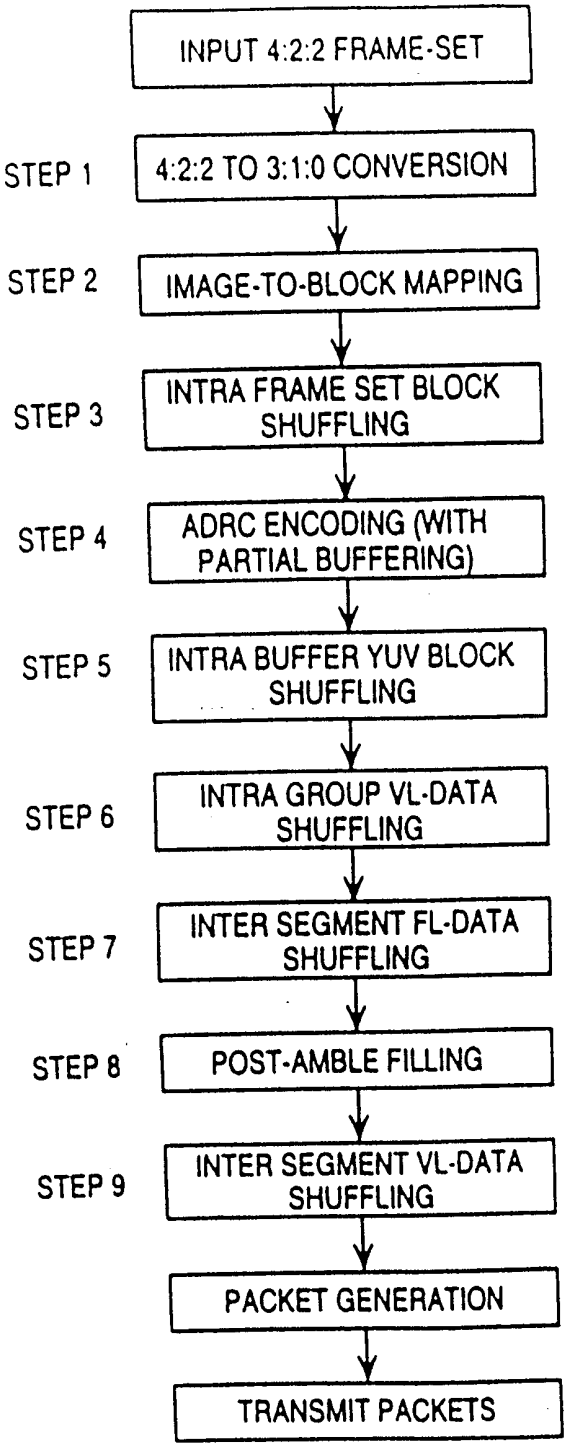

INPUT 4:2:2 FRAME-SET
STEP 1
4:2:2 TO 3:1:0 CONVERSION
STEP 2
IMAGE-TO-BLOCK MAPPING
STEP 3
INTRA FRAME SET BLOCK SHUFFLING
STEP 4
ADRC ENCODING (WITH PARTIAL BUFFERING)
STEP 5
INTRA BUFFER YUV BLOCK SHUFFLING
STEP 6
INTRA GROUP VL-DATA SHUFFLING
STEP 7
INTER SEGMENT FL-DATA SHUFFLING
STEP 8
POST-AMBLE FILLING
STEP 9
INTER SEGMENT VL-DATA SHUFFLING
PACKET GENERATION
TRANSMIT PACKETS
SIGNAL FLOWGRAPH IN THE ENCODER

FIG. 3

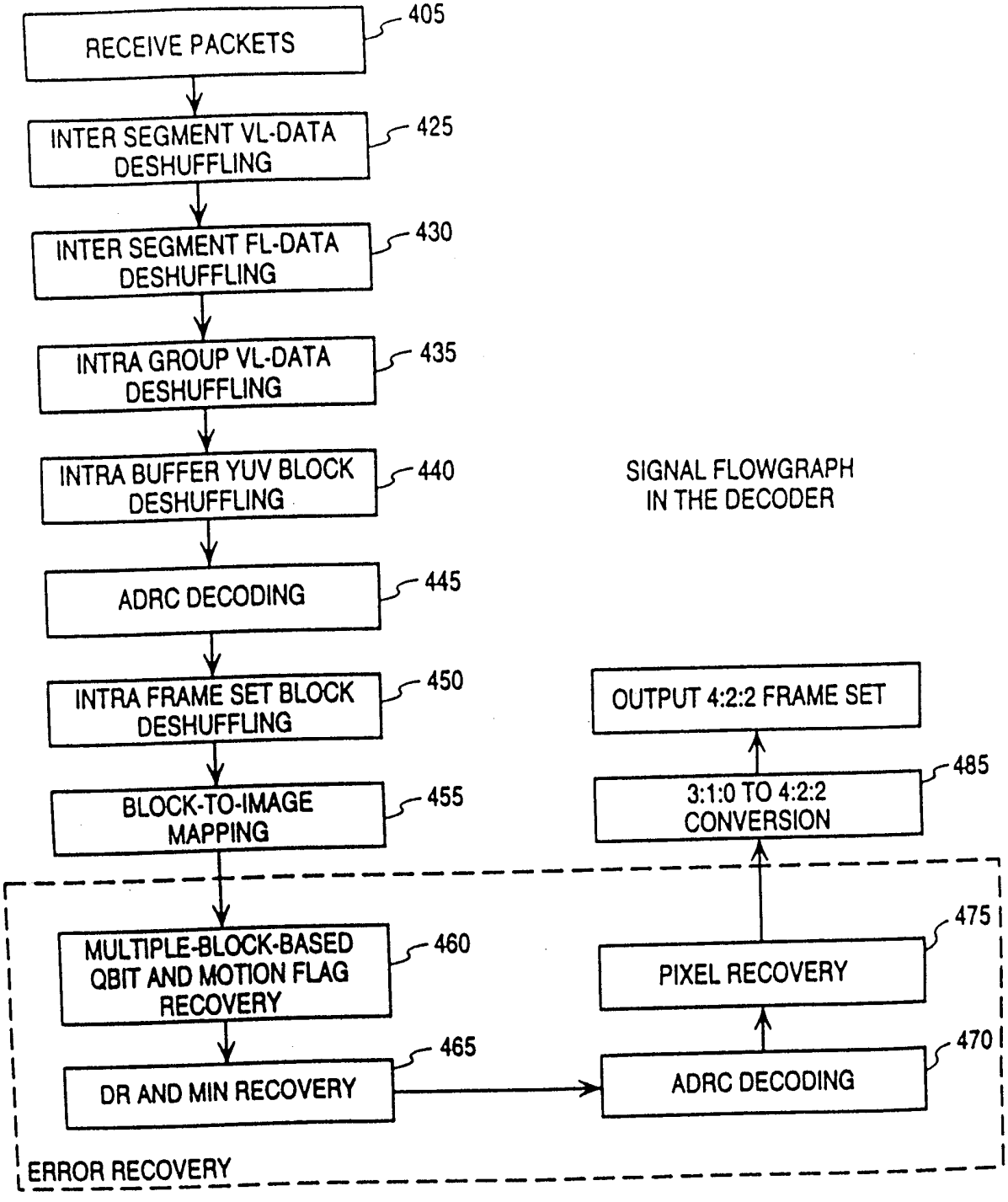

RECEIVE PACKETS
405
INTER SEGMENT VL-DATA DESHUFFLING
425
INTER SEGMENT FL-DATA DESHUFFLING
430
INTRA GROUP VL-DATA DESHUFFLING
435
INTRA BUFFER YUV BLOCK DESHUFFLING
440
ADRC DECODING
445
INTRA FRAME SET BLOCK DESHUFFLING
450
BLOCK-TO-IMAGE MAPPING
455
MULTIPLE-BLOCK-BASED QBIT AND MOTION FLAG RECOVERY
460
DR AND MIN RECOVERY
465
ERROR RECOVERY
ADRC DECODING
470
PIXEL RECOVERY
475
3:1:0 TO 4:2:2 CONVERSION
485
OUTPUT 4:2:2 FRAME SET
SIGNAL FLOWGRAPH IN THE DECODER

FIG. 4

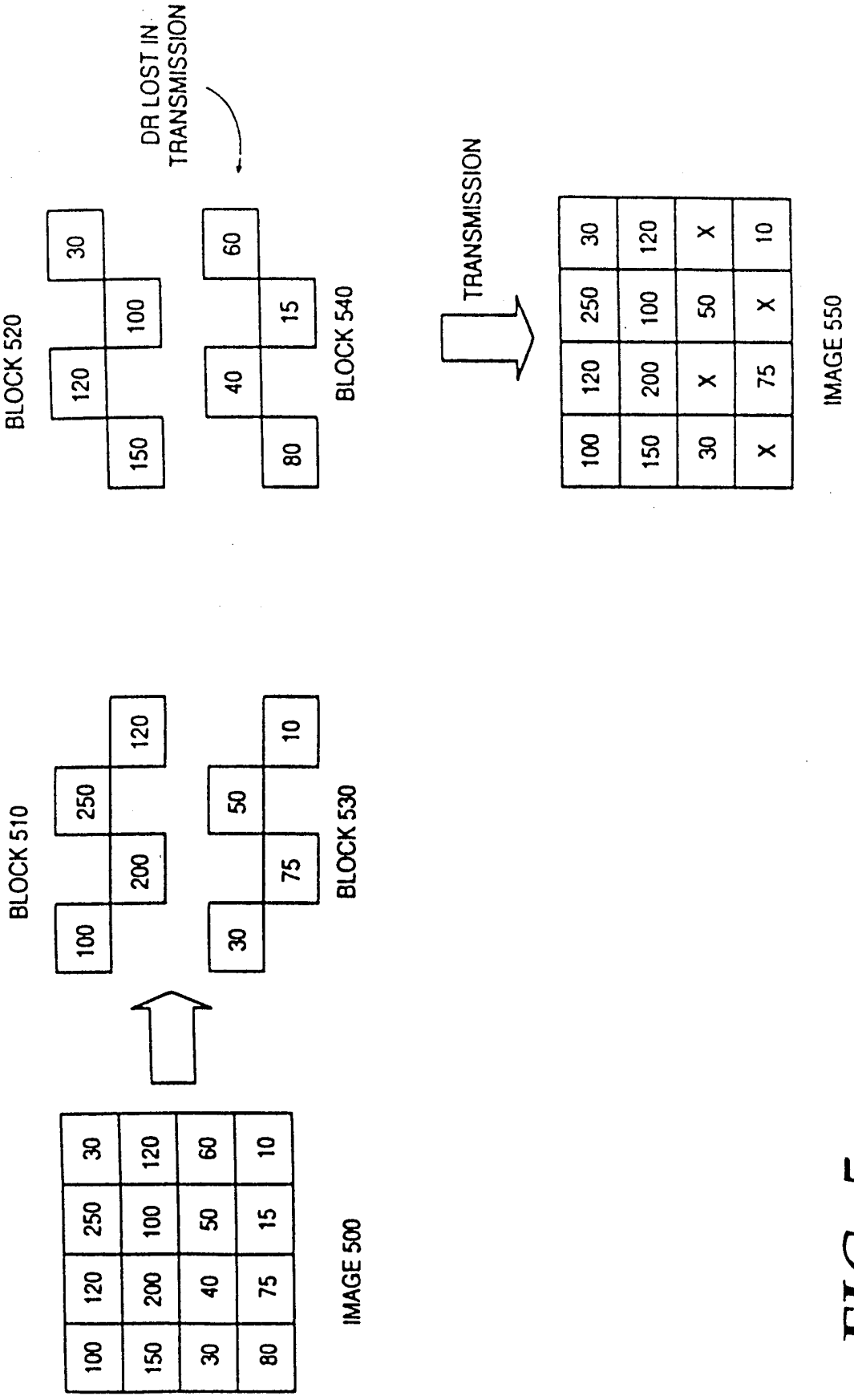
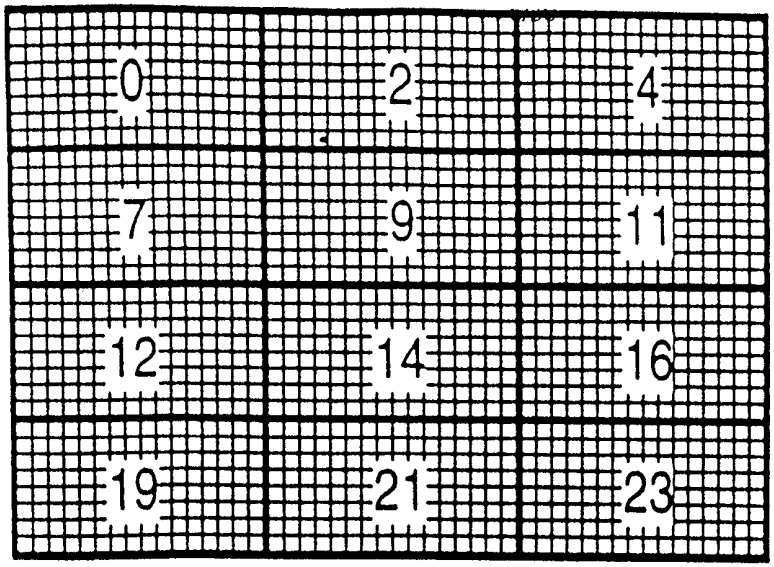
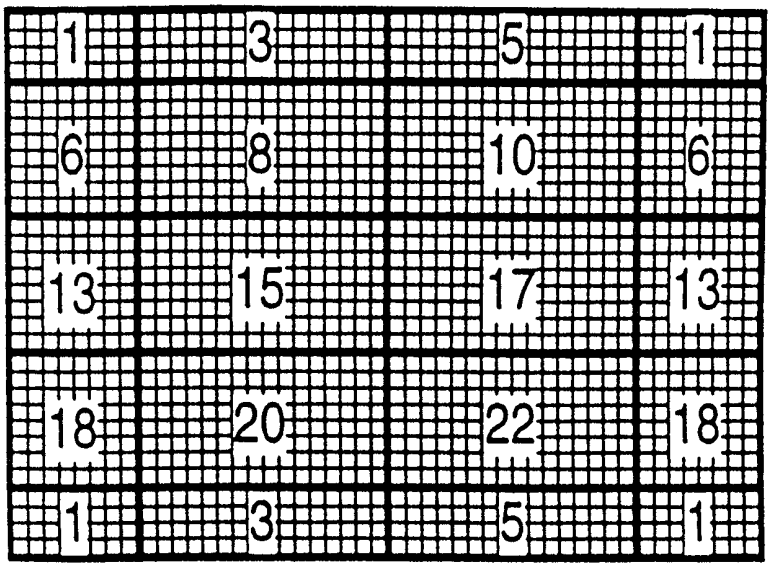


IMAGE 500
100 120 250 30
150 200 100 120
30 40 50 60
80 75 15 10
BLOCK 510
100 250 120
150 200
BLOCK 530
30 50 10
75
BLOCK 520
150 120 100 30
BLOCK 540
80 40 15 60
DR LOST IN TRANSMISSION
TRANSMISSION
100 120 250 30
150 200 100 120
30 X 50 X
X 75 X 10
IMAGE 550

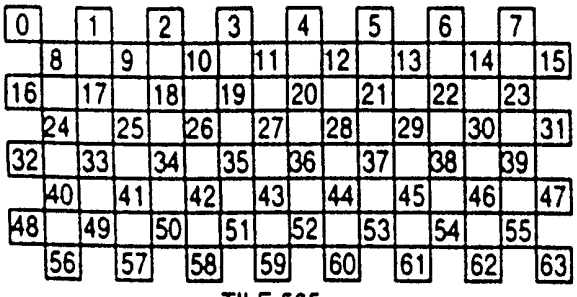
FIG. 5



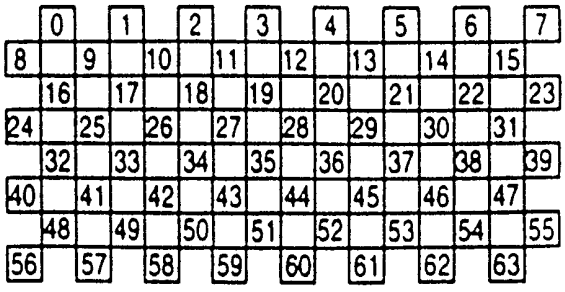
SUB-IMAGE 560



SUB-IMAGE 570

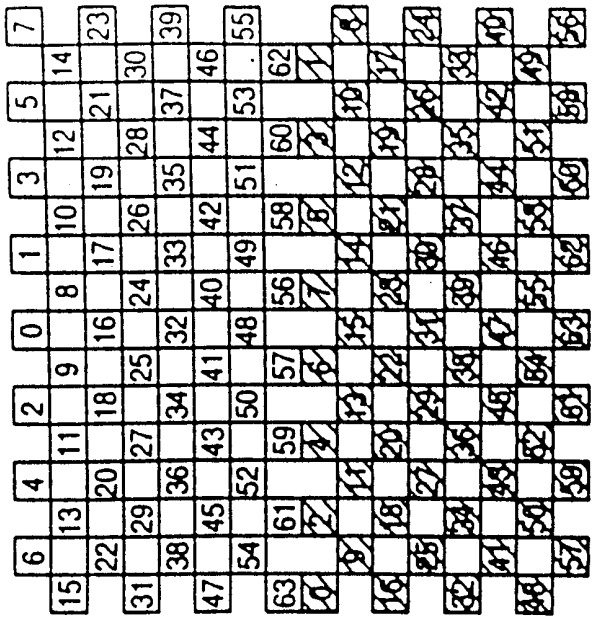


TILE 565

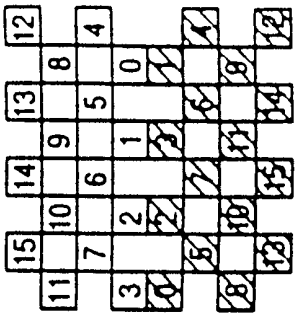


TILE 575

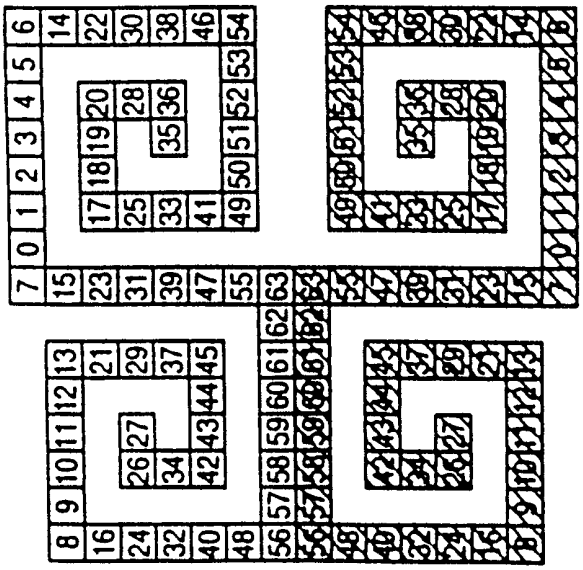
FIG. 5a



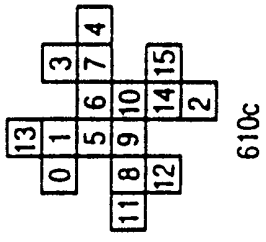
610b



610d



610a



610c

FIG. 6

INTRA FRAME SET BLOCK SHUFFLING
SEGMENT DEFINITION: Y BLOCKS

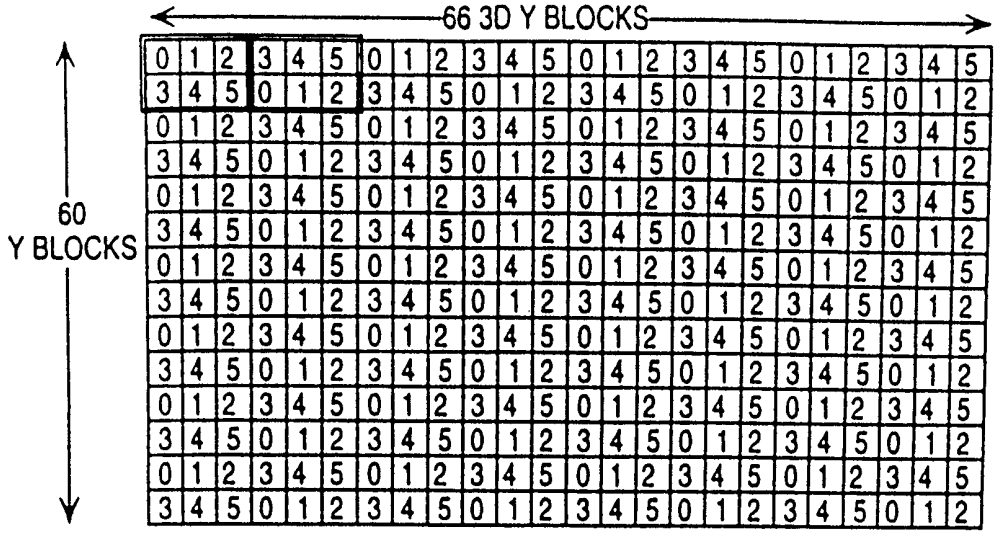


FIG. 7a

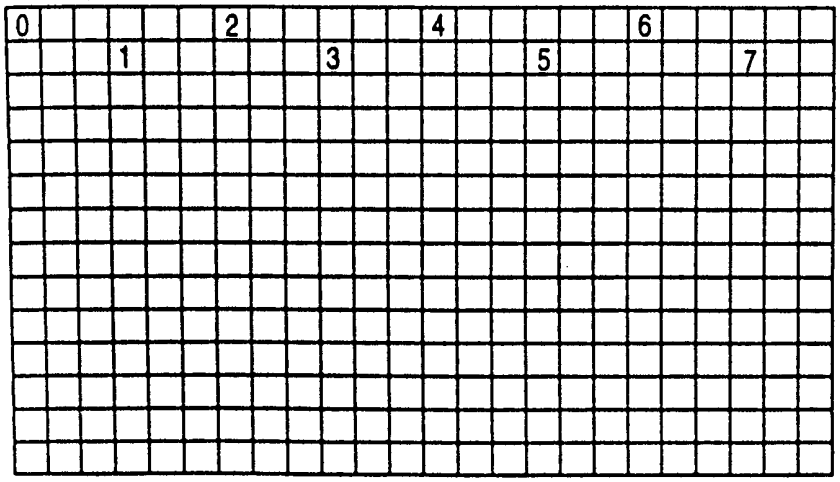


FIG. 7b

INTRA FRAME SET BLOCK SHUFFLING

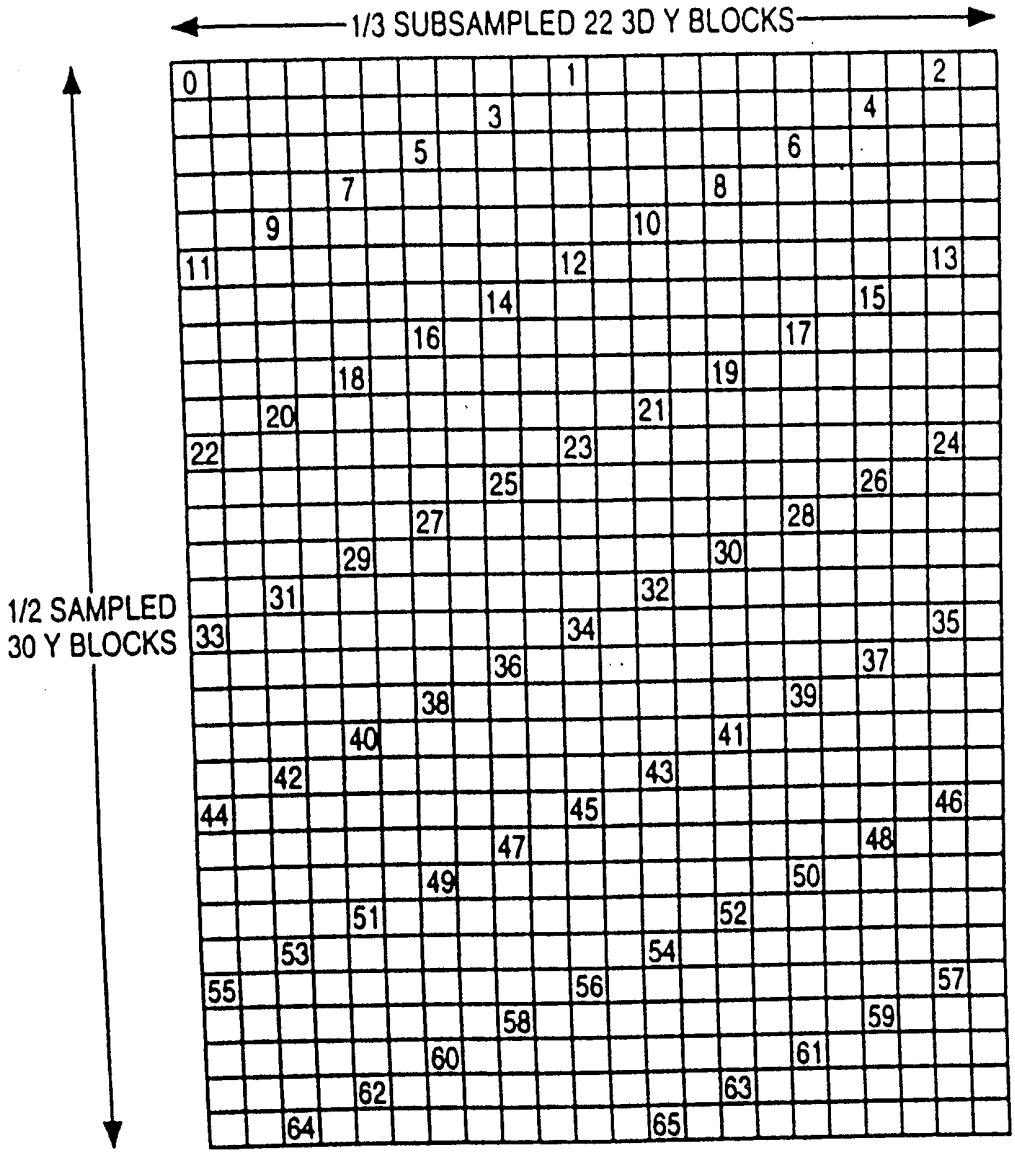


FIG. 7c

INTRA FRAME SET BLOCK SHUFFLING

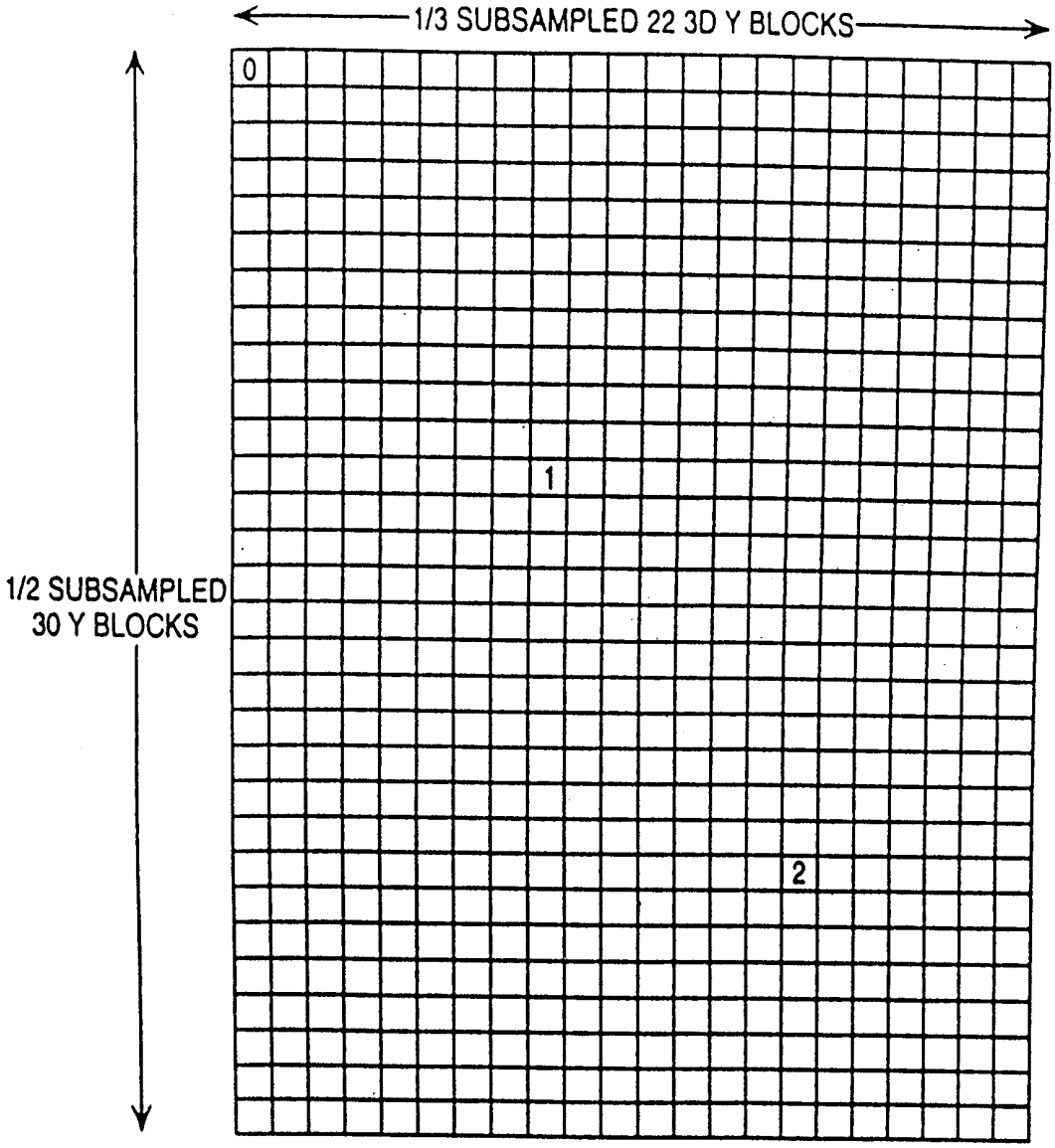


FIG. 7d

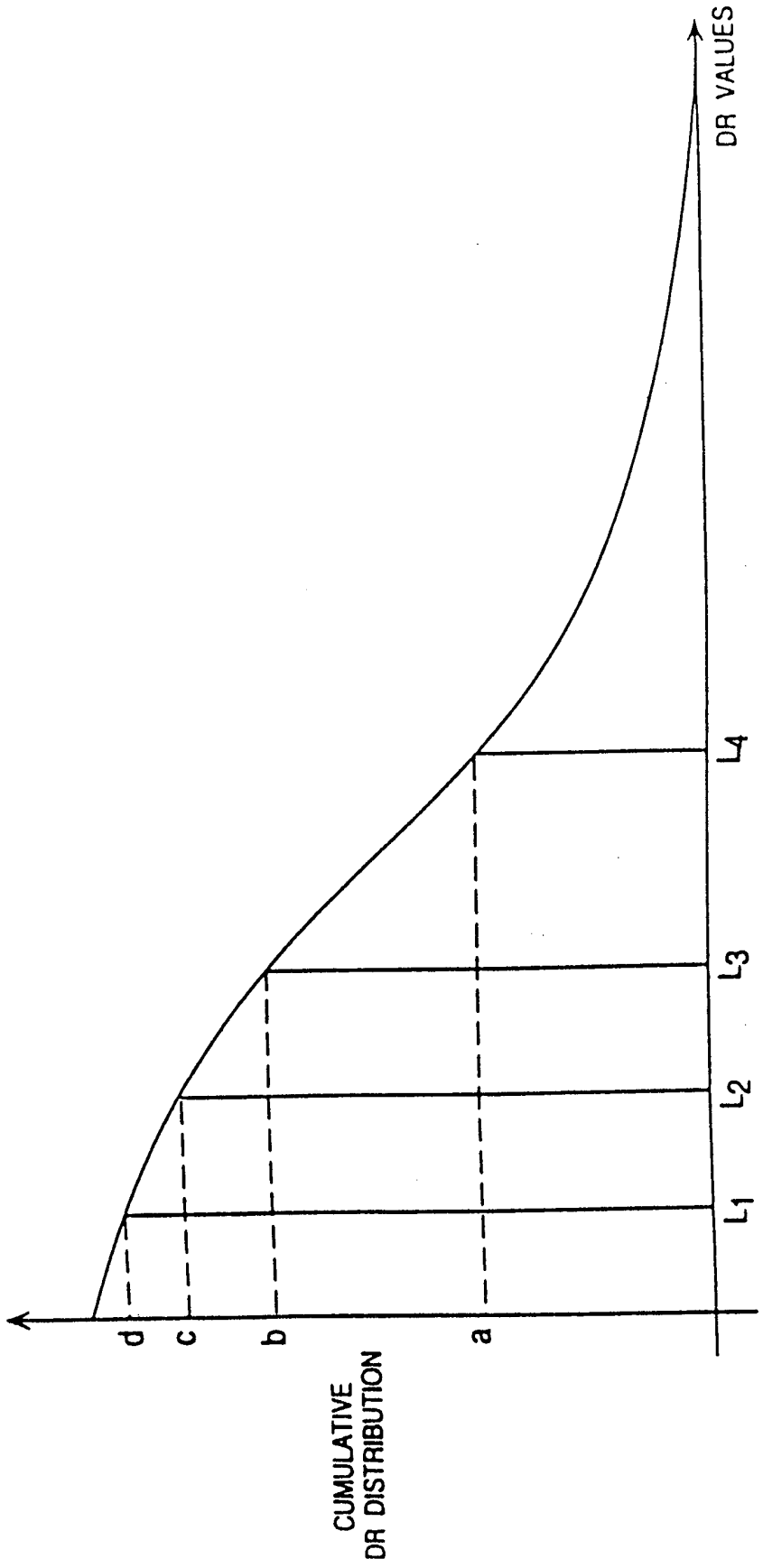


FIG. 8

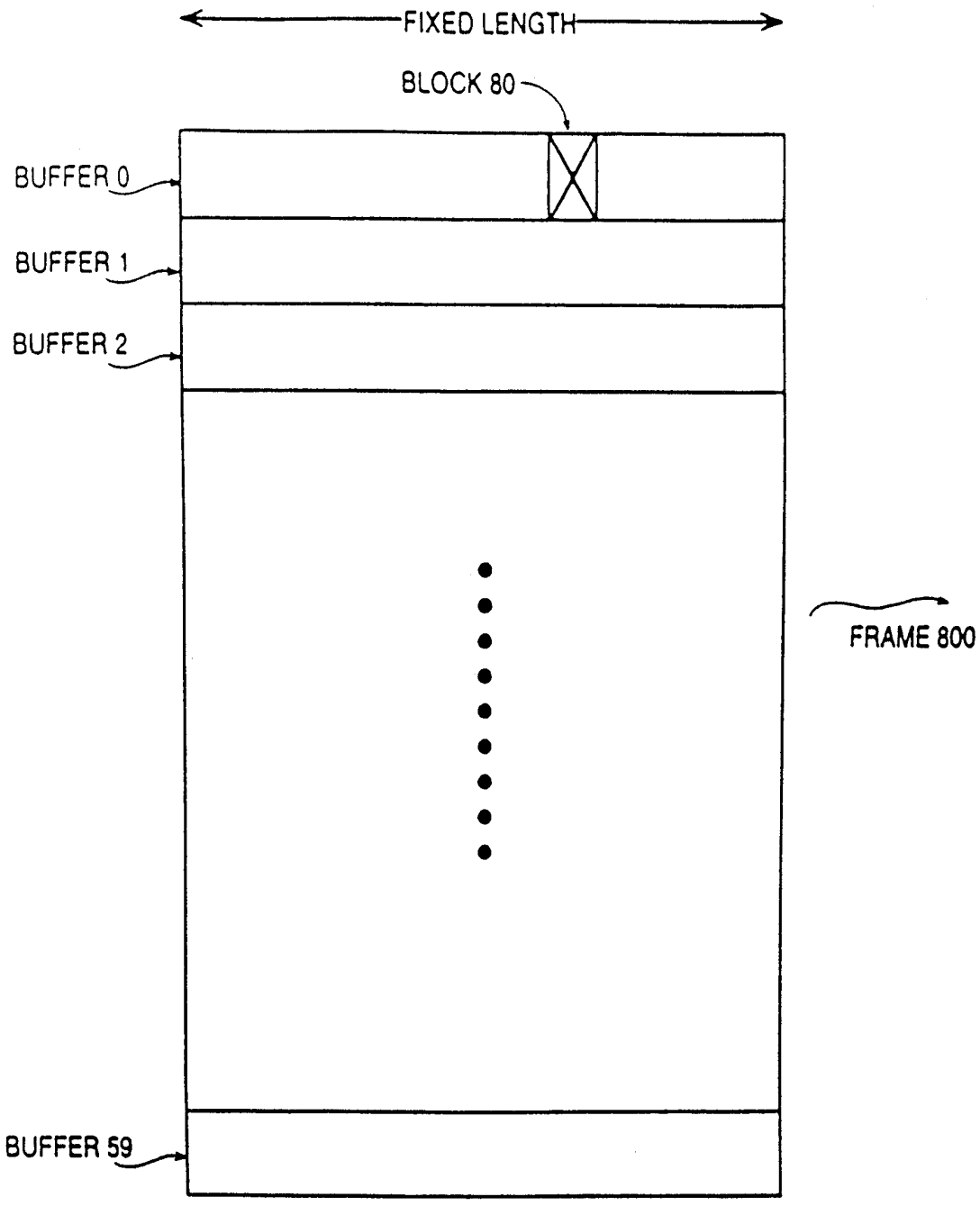

FIXED LENGTH
BLOCK 80
BUFFER 0
BUFFER 1
BUFFER 2
BUFFER 59
FRAME 800

FIG. 8a

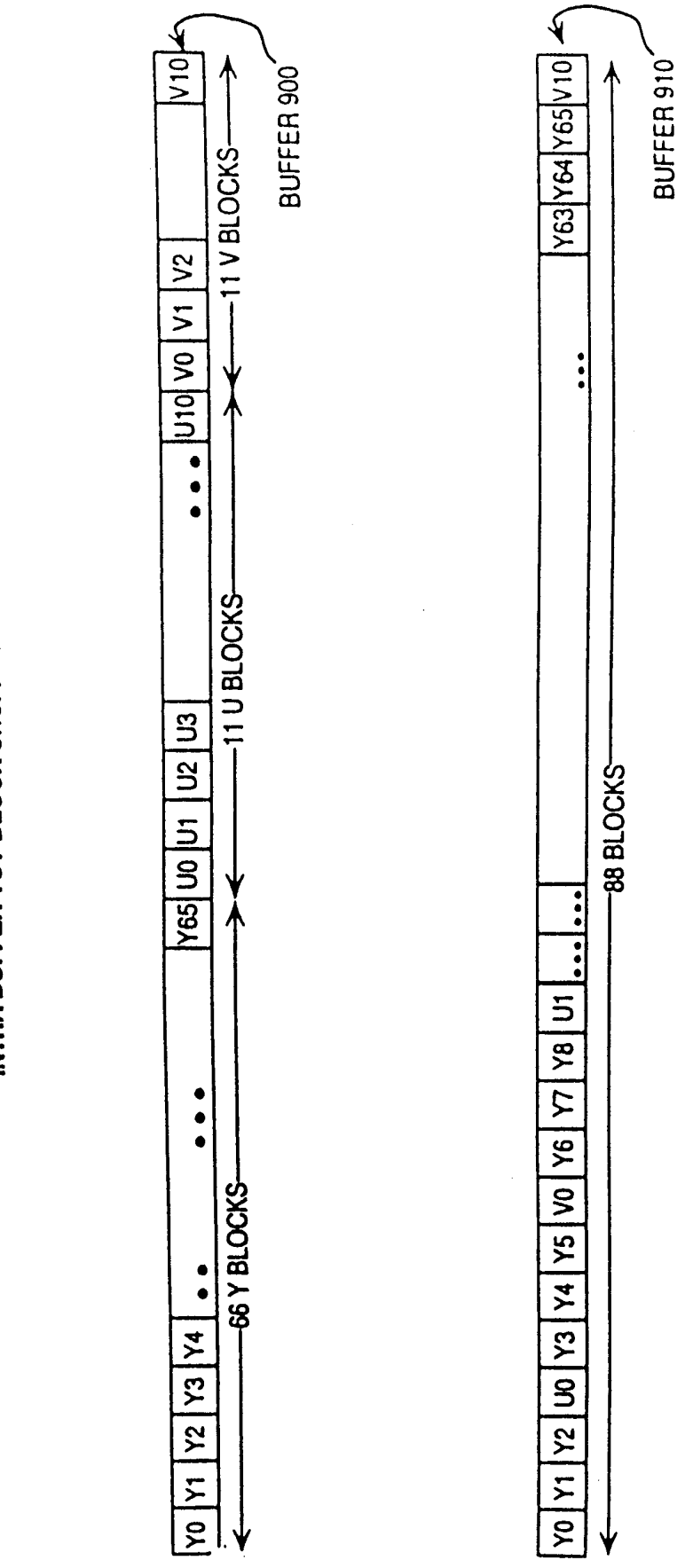

INTRA BUFFER YUV BLOCK SHUFFLING
Y0
Y1
Y2
Y3
Y4
Y65
U0
U1
U2
U3
U10
V0
V1
V2
V10
66 Y BLOCKS
11 U BLOCKS
11 V BLOCKS
BUFFER 900
Y0
Y1
Y2
U0
Y3
Y4
Y5
V0
Y6
Y7
Y8
U1
Y63
Y64
Y65
V10
88 BLOCKS
BUFFER 910

FIG. 9

INTRA GROUP VL-DATA SHUFFLING

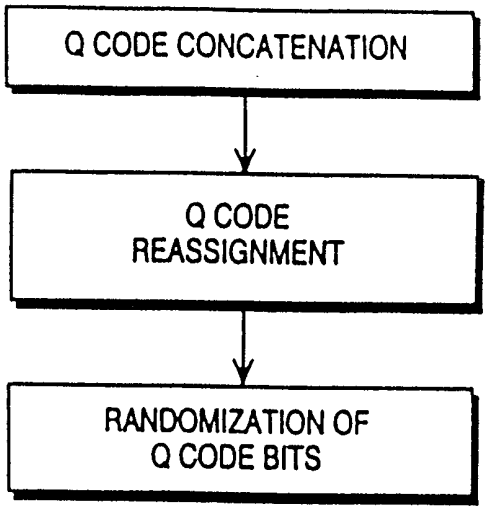


FIG. 10

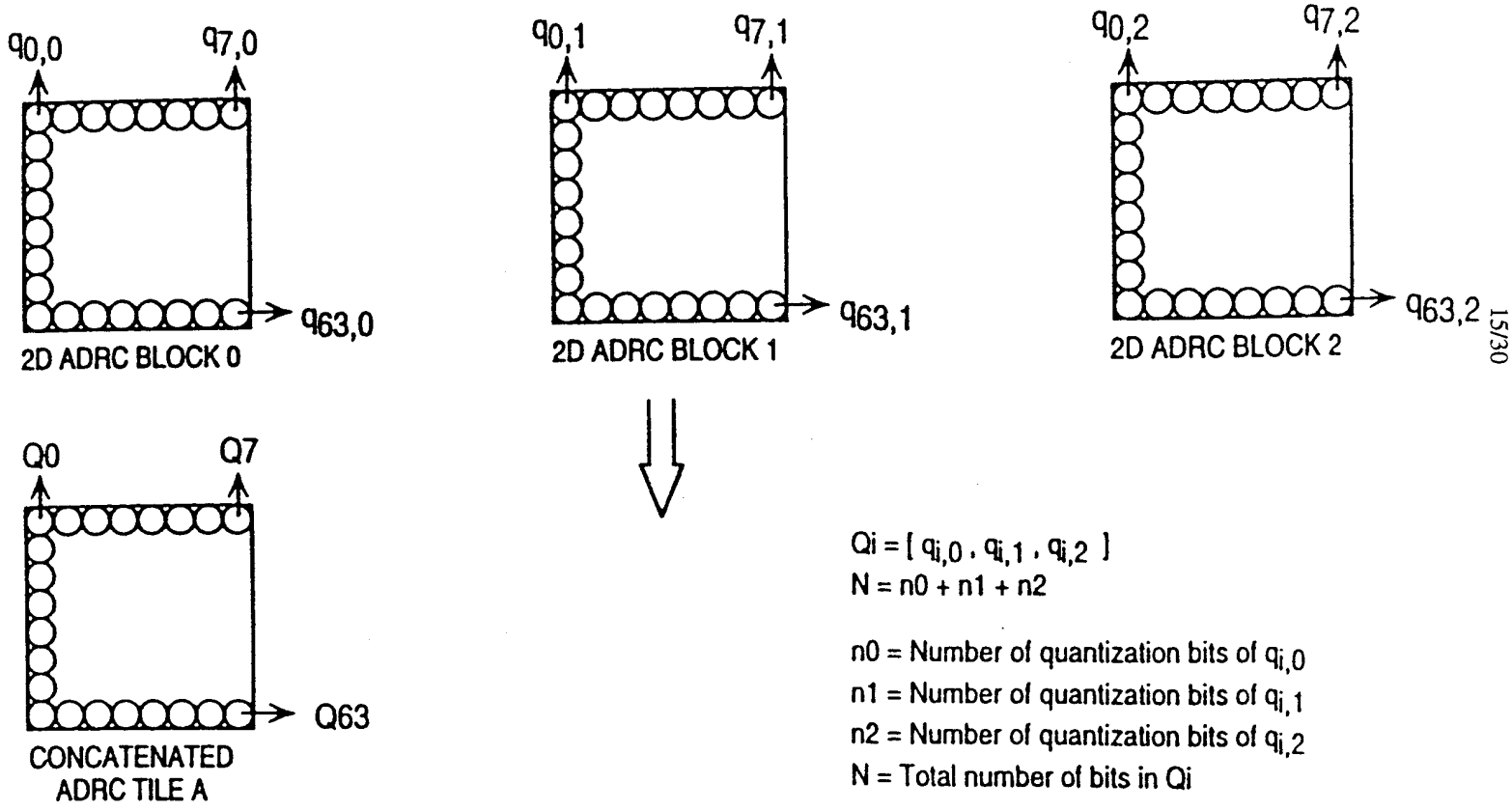

INTRA GROUP VL-DATA SHUFFLING
Q CODE CONCATENATION
q0,0
q7,0
q63,0
2D ADRC BLOCK 0
q0,1
q7,1
q63,1
2D ADRC BLOCK 1
q0,2
q7,2
q63,2
2D ADRC BLOCK 2
Q0
Q7
Q63
CONCATENATED
ADRC TILE A
Qi = [qi,0 , qi,1 , qi,2]
N = n0 + n1 + n2
n0 = Number of quantization bits of qi,0
n1 = Number of quantization bits of qi,1
n2 = Number of quantization bits of qi,2
N = Total number of bits in Qi

FIG. 11

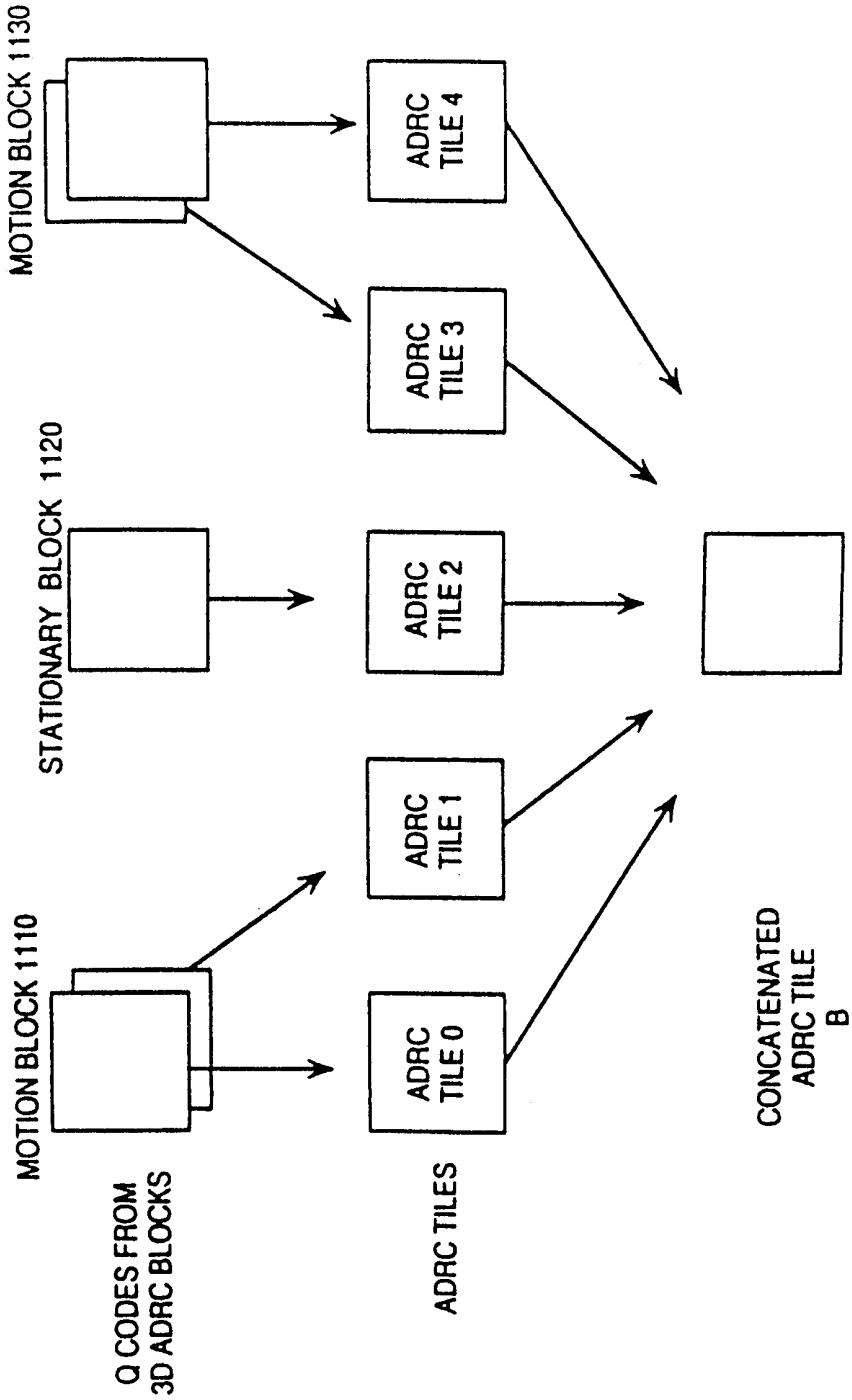

MOTION BLOCK 1130
STATIONARY BLOCK 1120
MOTION BLOCK 1110
Q CODES FROM 3D ADRC BLOCKS
ADRC TILE 0
ADRC TILE 1
ADRC TILE 2
ADRC TILE 3
ADRC TILE 4
ADRC TILES
CONCATENATED ADRC TILE B

FIG. 11a

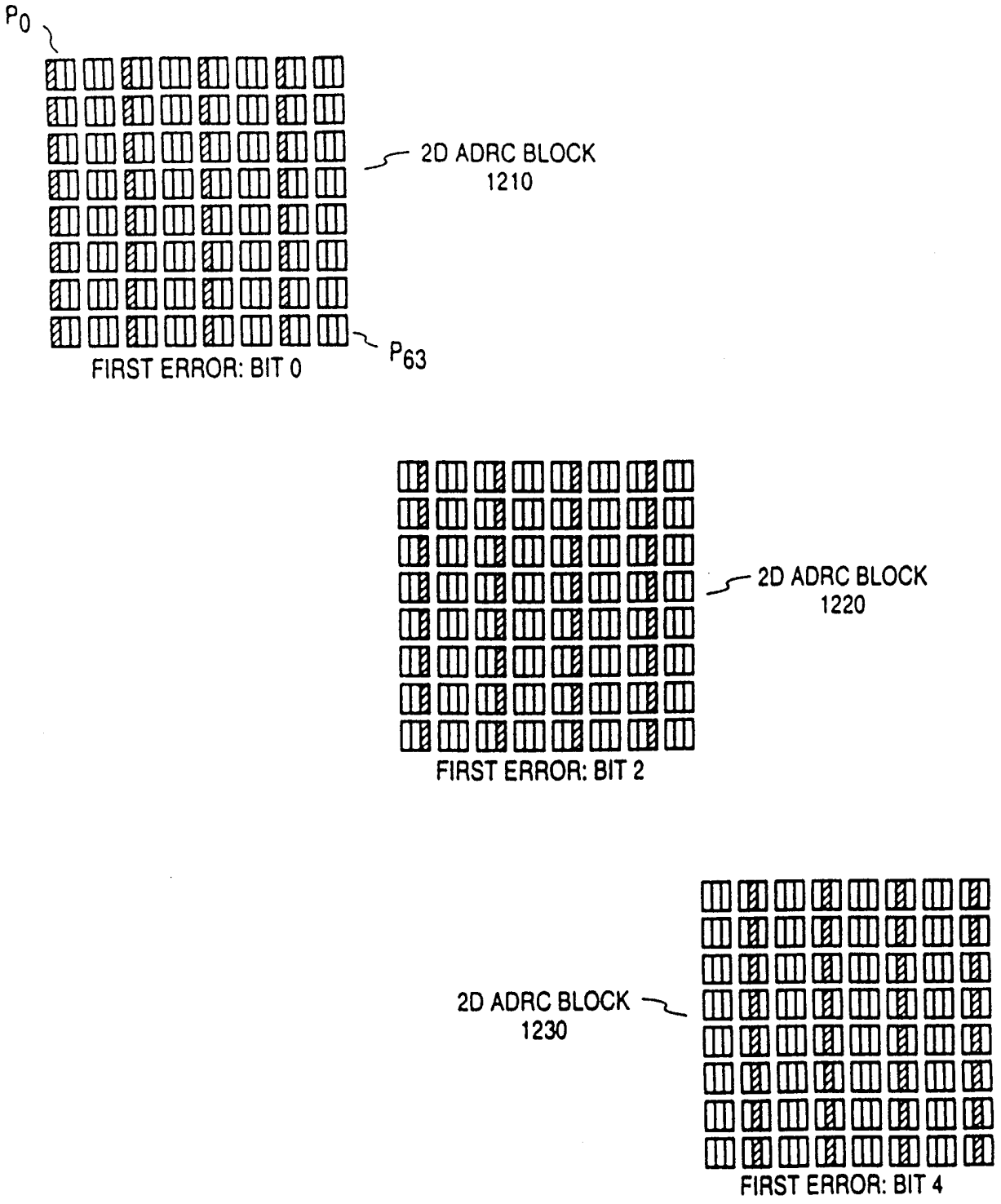

P0
2D ADRC BLOCK
1210
P63
FIRST ERROR: BIT 0
2D ADRC BLOCK
1220
FIRST ERROR: BIT 2
2D ADRC BLOCK
1230
FIRST ERROR: BIT 4

FIG. 12

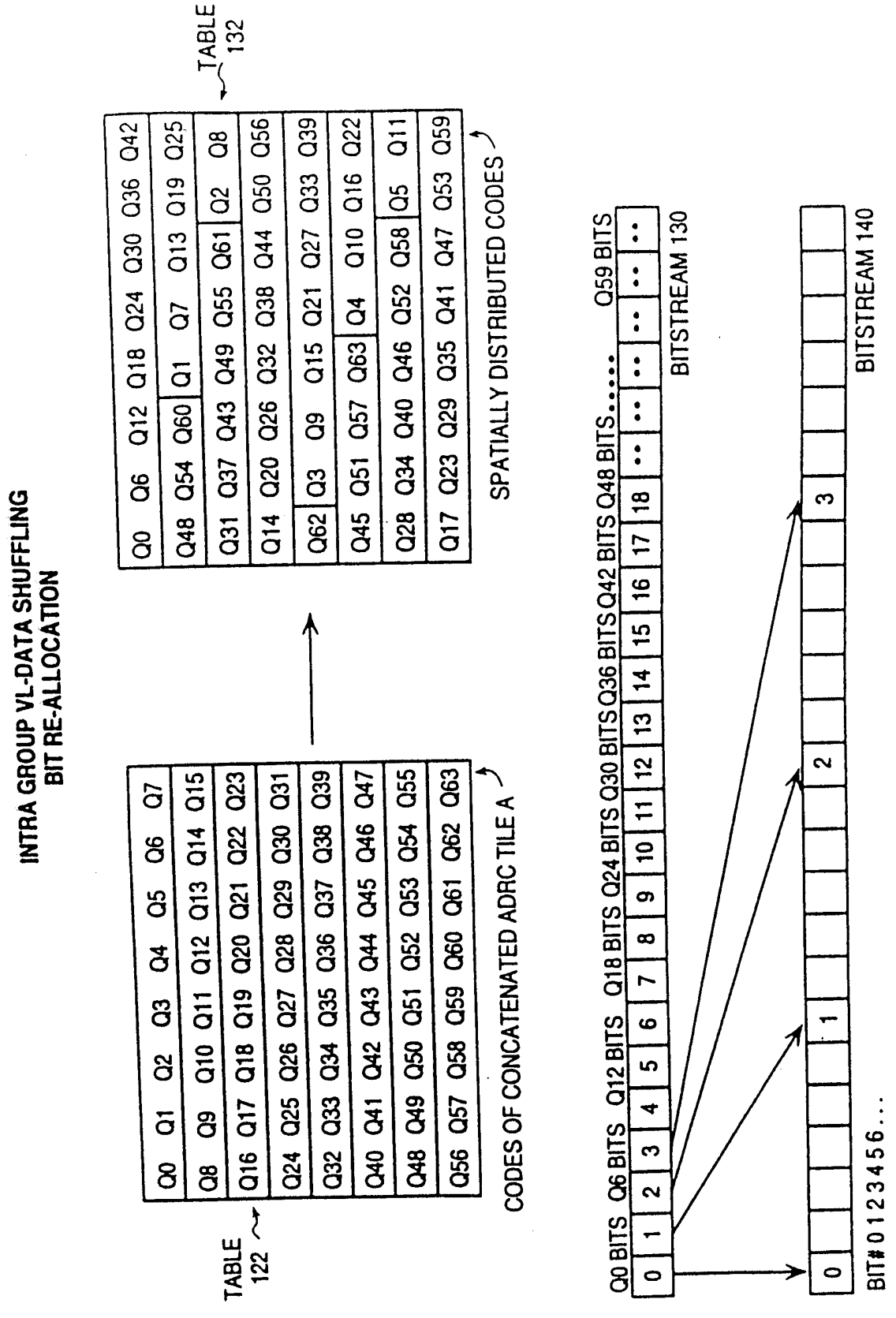

INTRA GROUP VL-DATA SHUFFLING
BIT RE-ALLOCATION
TABLE 122
CODES OF CONCATENATED ADRC TILE A
TABLE 132
SPATIALLY DISTRIBUTED CODES
Q0 BITS Q6 BITS Q12 BITS Q18 BITS Q24 BITS Q30 BITS Q36 BITS Q42 BITS Q48 BITS..... Q59 BITS
BITSTREAM 130
BIT# 0 1 2 3 4 5 6....
BITSTREAM 140

FIG. 12a

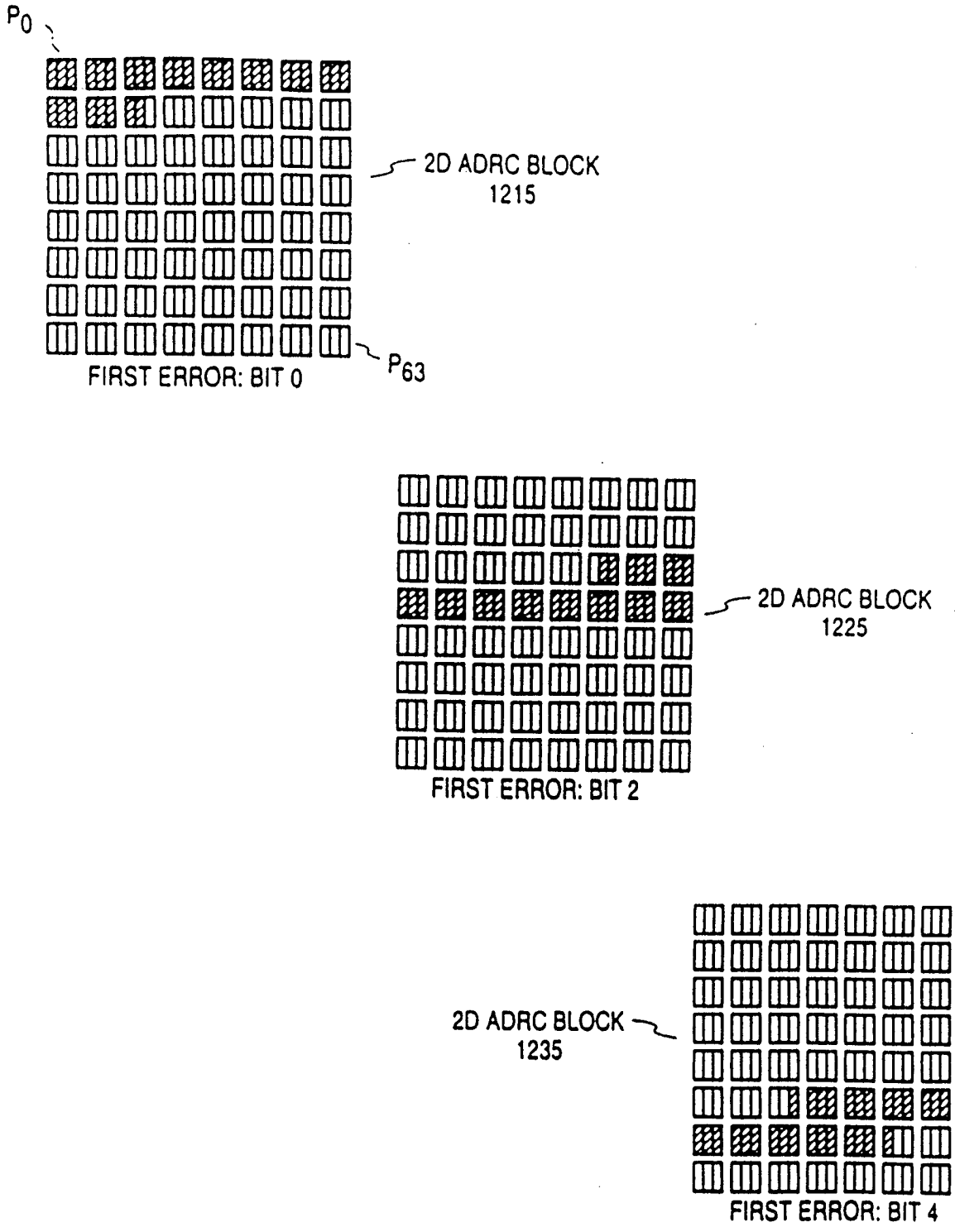

P0
2D ADRC BLOCK
1215
P63
FIRST ERROR: BIT 0
2D ADRC BLOCK
1225
FIRST ERROR: BIT 2
2D ADRC BLOCK
1235
FIRST ERROR: BIT 4

FIG. 12b

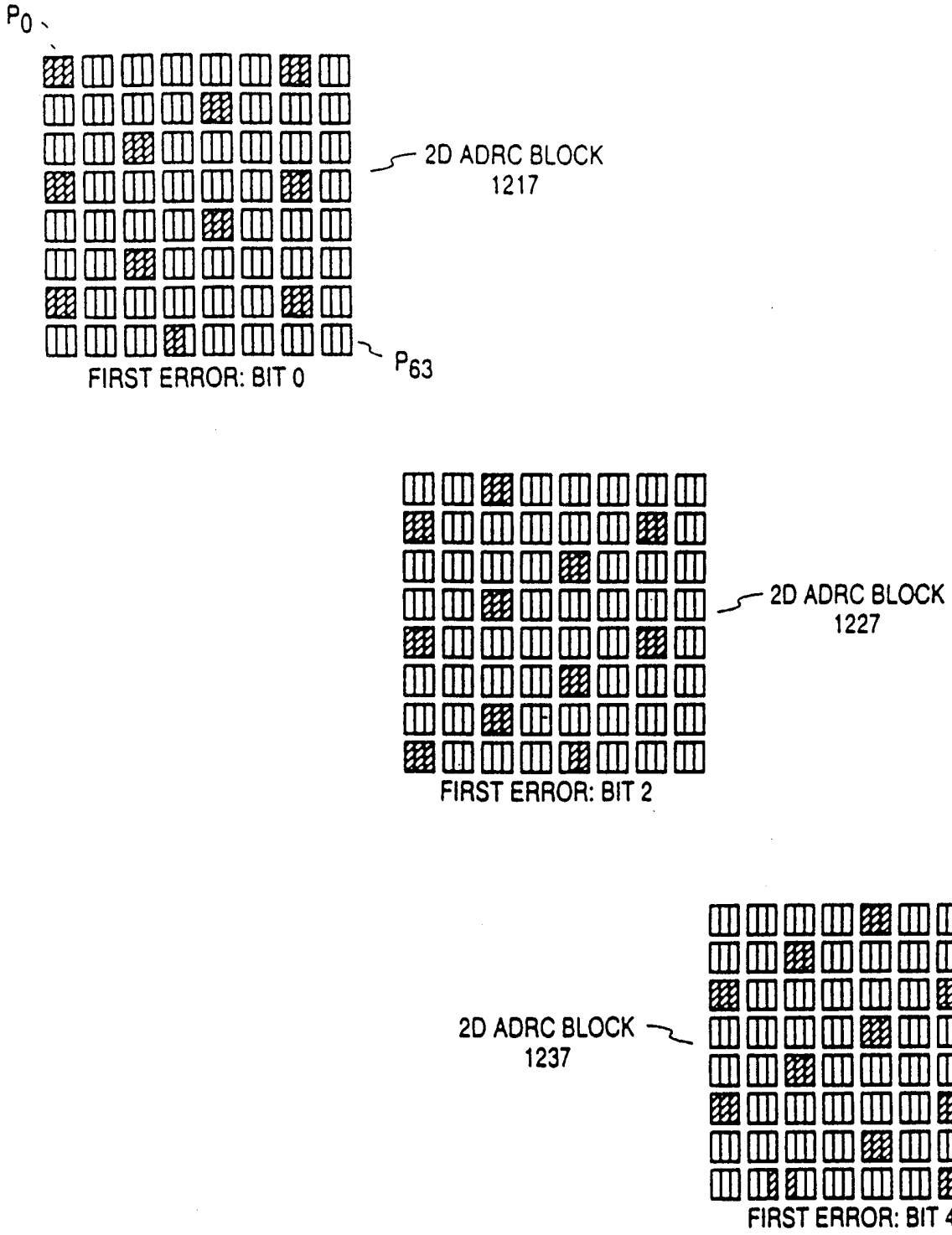

P0
2D ADRC BLOCK
1217
P63
FIRST ERROR: BIT 0
2D ADRC BLOCK
1227
FIRST ERROR: BIT 2
2D ADRC BLOCK
1237
FIRST ERROR: BIT 4

FIG. 12c

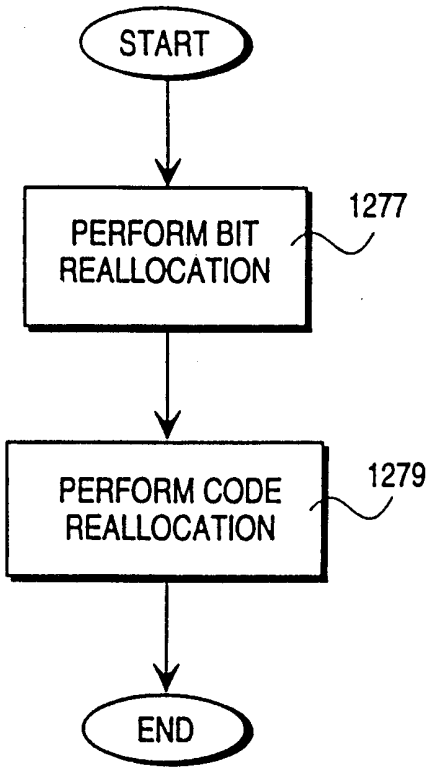

START
PERFORM BIT REALLOCATION
1277
PERFORM CODE REALLOCATION
1279
END

FIG. 12d

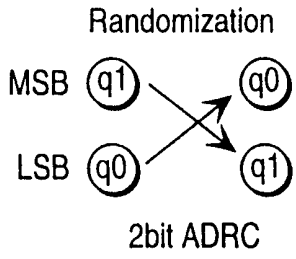


FIG. 12f

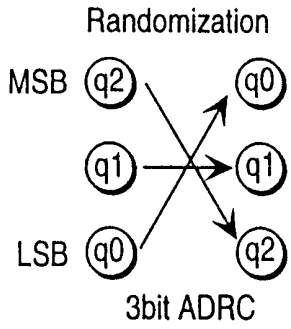


FIG. 12g

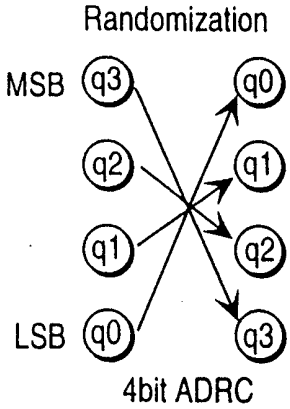


FIG. 12h

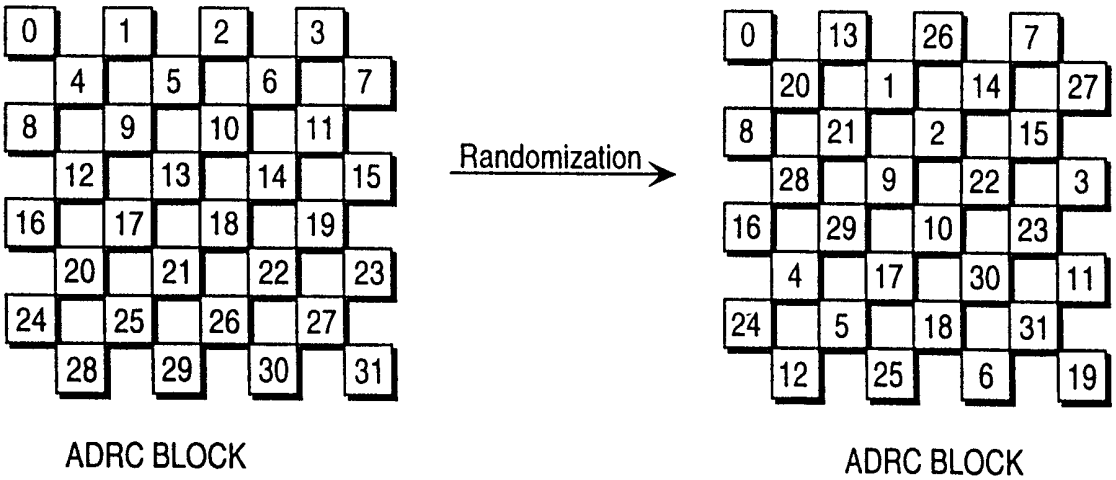
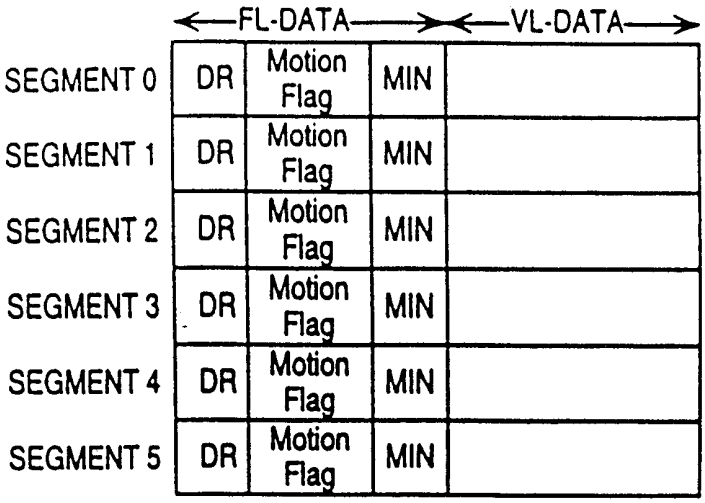


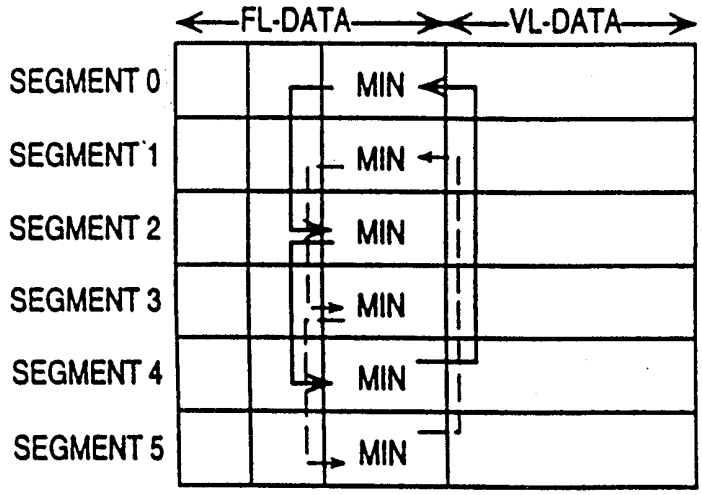
FIG. 12e

INTER SEGMENT FL-DATA SHUFFLING

ORIGNAL



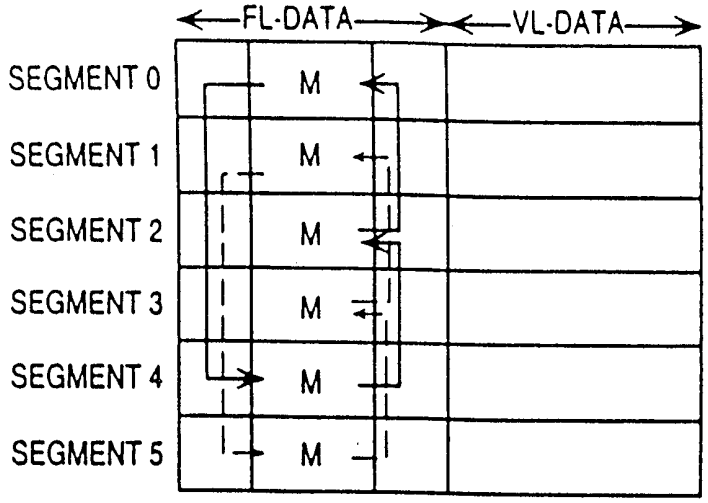
MIN Shuffling 1300



Original	Shuffled
Segment 0	--> Segment 2
Segment 2	--> Segment 4
Segment 4	--> Segment 0
Segment 1	--> Segment 3
Segment 3	--> Segment 5
Segment 5	--> Segment 1

FIG. 13

Motion Flag Shuffling



Original	Shuffled
Segment 0	--> Segment 4
Segment 2	--> Segment 0
Segment 4	--> Segment 2
Segment 1	--> Segment 5
Segment 3	--> Segment 1
Segment 5	--> Segment 3

After FL-Data Shuffling
FL-Data Loss Pattern for Segment 0

SEGMENT \ BLOCK#	0	1	2	3	4	5	• • • 879
0	DR	DR	DR	DR	DR	DR	• • •
1							• • •
2	M	M	M	M	M	M	• • •
3							• • •
4	MIN	MIN	MIN	MIN	MIN	MIN	• • •
5							

LOSS PATTERN 1310

M:MOTION FLAG

FIG. 13a

BLOCK #	0	1	2	3	4	5	6	7	8	...	879
COUNT	0	1	2	0	1	2	0	1	2	...	
SEGMENT A	◇	◇	◇	◇	◇	◇	◇	◇	◇	...	
SEGMENT B	○	○	○	○	○	○	○	○	○	...	
SEGMENT C	□	□	□	□	□	□	□	□	□	...	

DR MODULAR SHUFFLE 1410

BLOCK #	0	1	2	3	4	5	6	7	8	...	879
COUNT	0	1	2	0	1	2	0	1	2	...	
SEGMENT A	◇	◇	◇	◇	◇	◇	◇	◇	◇	...	
SEGMENT B	○	○	○	○	○	○	○	○	○	...	
SEGMENT C	□	□	□	□	□	□	□	□	□	...	

MIN MODULAR SHUFFLE 1420

BLOCK #	0	1	2	3	4	5	6	7	8	...	879
COUNT	0	1	2	0	1	2	0	1	2	...	
SEGMENT A	◇	◇	◇	◇	◇	◇	◇	◇	◇	...	
SEGMENT B	○	○	○	○	○	○	○	○	○	...	
SEGMENT C	□	□	□	□	□	□	□	□	□	...	

MOTION FLAG MODULAR SHUFFLE 1430

FIG. 14

BLOCK#	0	1	2	3	4	5	• • • 879
COUNT	0	1	2	0	1	2	• • •
DATA DR	0	2	4	0	2	4	• • •
MIN	2	4	0	2	4	0	• • •
M	4	0	2	4	0	2	• • •

MODULAR SHUFFLE RESULT 1416

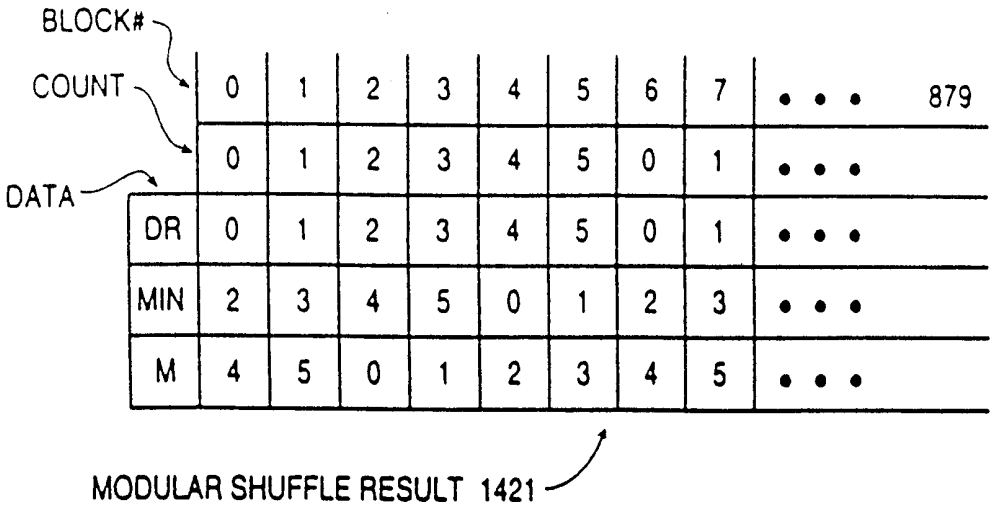
BLOCK#	0	1	2	3	4	5	• • • 879
COUNT	0	1	2	0	1	2	• • •
SEGMENT# 0	DR	M	MIN	DR	M	MIN	• • •
1							• • •
2	MIN	DR	M	MIN	DR	M	• • •
3							• • •
4	M	MIN	DR	M	MIN	DR	• • •
5							

LOSS PATTERN 1415

DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN

SPATIAL LOSS PATTERN 1417

FIG. 14a


BLOCK#
COUNT
DATA
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
DR 0 1 2 3 4 5 0 1 • • •
MIN 2 3 4 5 0 1 2 3 • • •
M 4 5 0 1 2 3 4 5 • • •
MODULAR SHUFFLE RESULT 1421

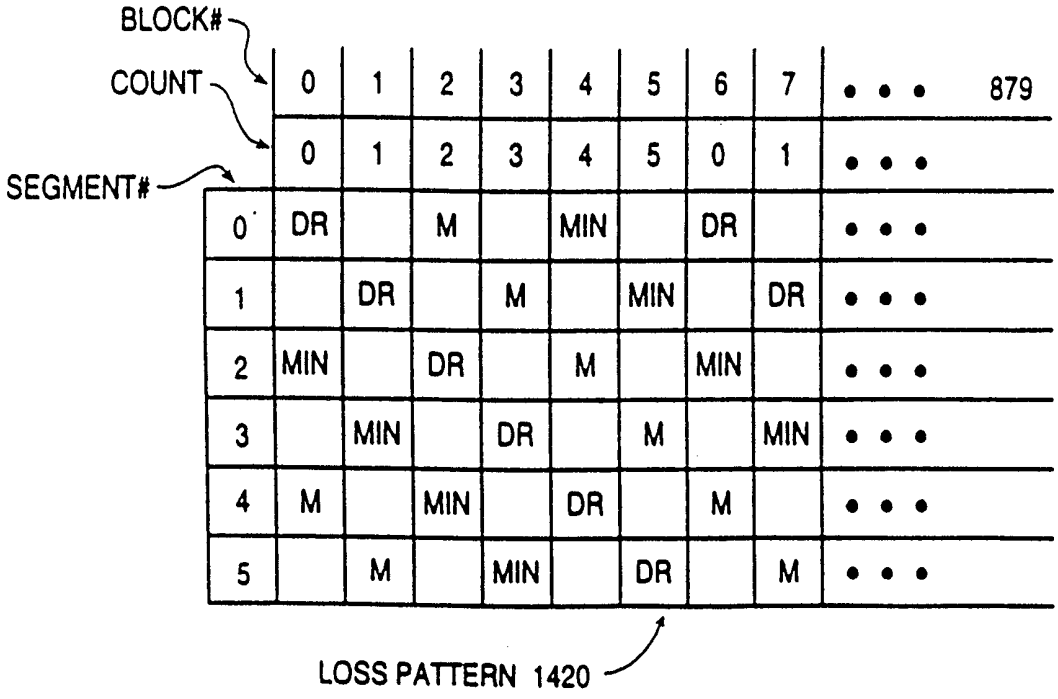
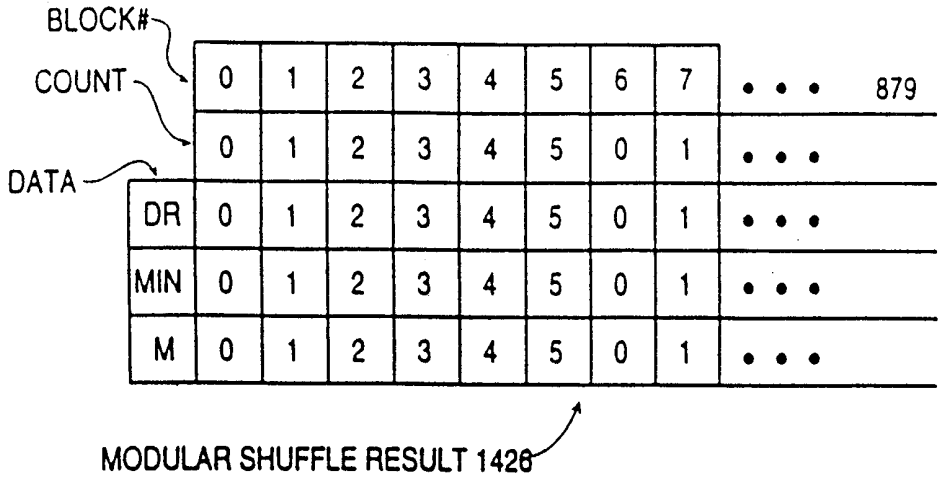

BLOCK#
COUNT
SEGMENT#
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
0 DR M MIN DR • • •
1 DR M MIN DR • • •
2 MIN DR M MIN • • •
3 MIN DR M MIN • • •
4 M MIN DR M • • •
5 M MIN DR M • • •
LOSS PATTERN 1420

FIG. 14b


BLOCK#
COUNT
DATA
DR
MIN
M
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
0 1 2 3 4 5 0 1 • • •
0 1 2 3 4 5 0 1 • • •
0 1 2 3 4 5 0 1 • • •
MODULAR SHUFFLE RESULT 1426

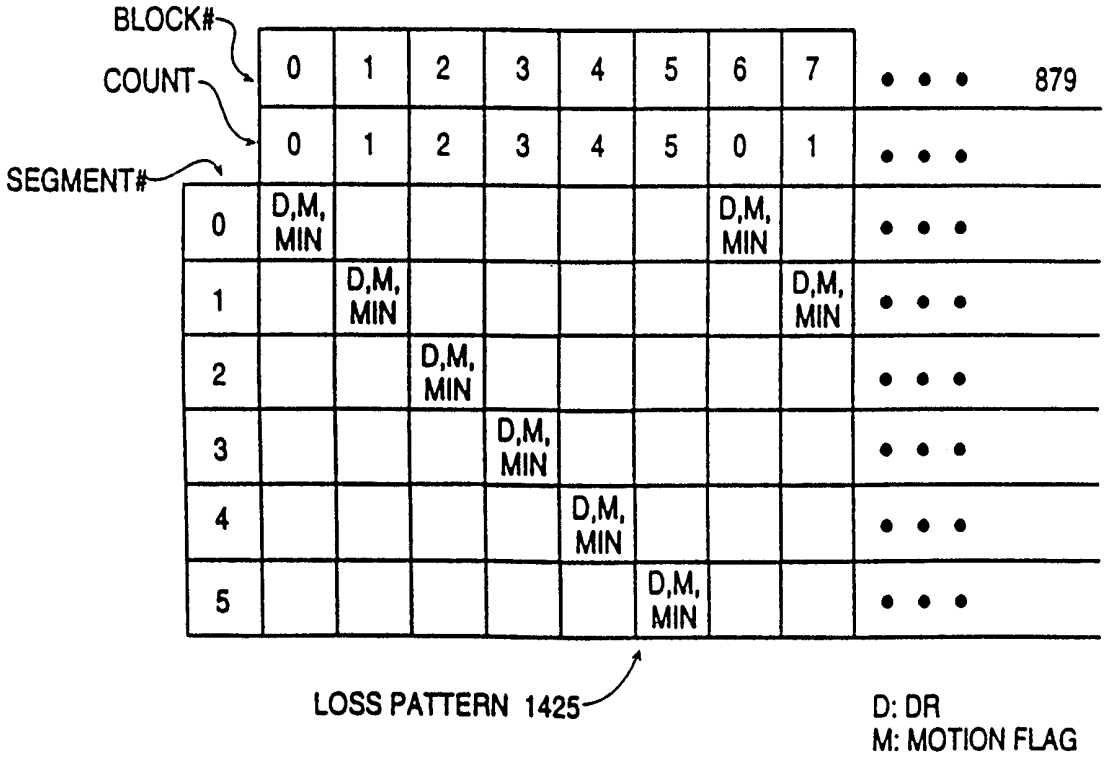

BLOCK#
COUNT
SEGMENT#
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
0 D,M, MIN D,M, MIN • • •
1 D,M, MIN D,M, MIN • • •
2 D,M, MIN • • •
3 D,M, MIN • • •
4 D,M, MIN • • •
5 D,M, MIN • • •
LOSS PATTERN 1425
D: DR
M: MOTION FLAG

FIG. 14c

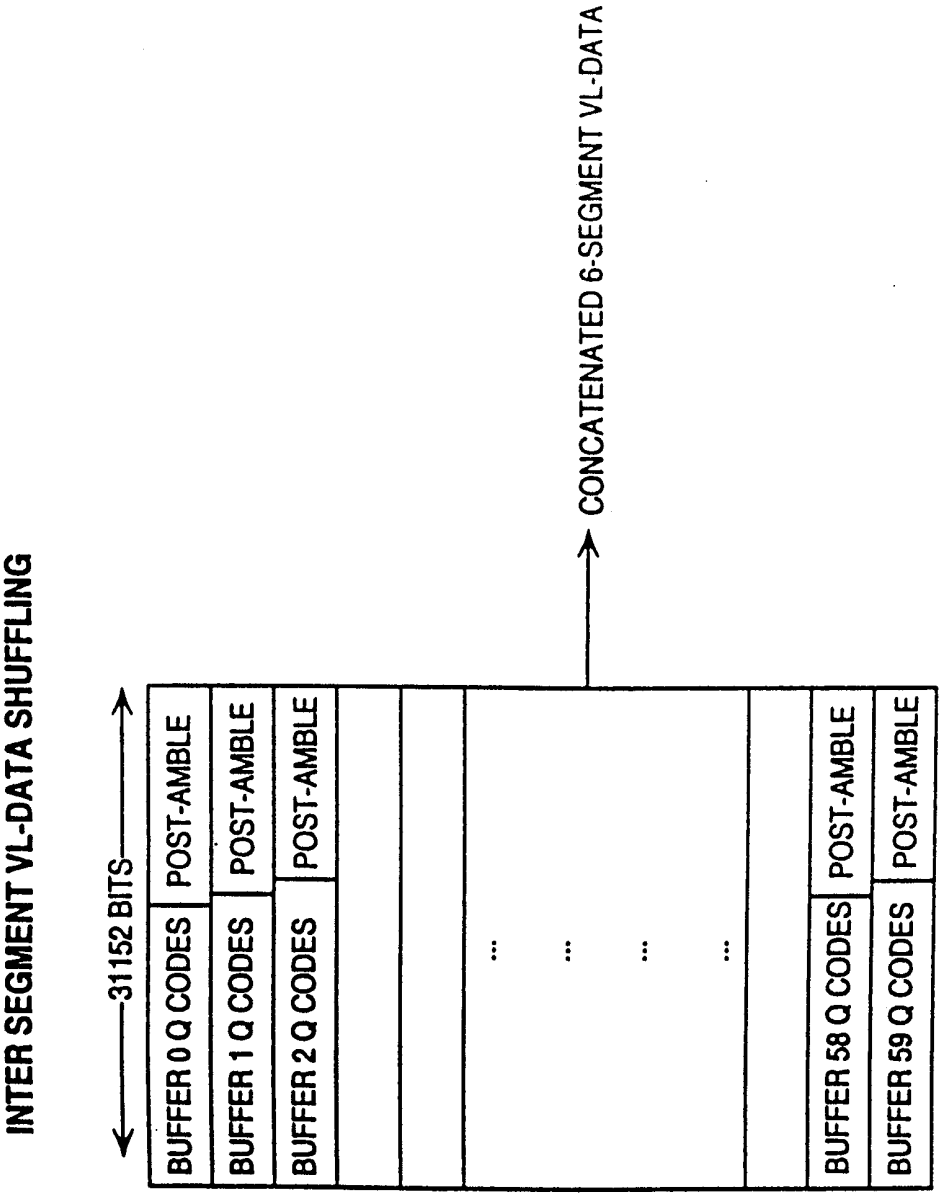

INTER SEGMENT VL-DATA SHUFFLING
31152 BITS
BUFFER 0 Q CODES
POST-AMBLE
BUFFER 1 Q CODES
POST-AMBLE
BUFFER 2 Q CODES
POST-AMBLE
BUFFER 58 Q CODES
POST-AMBLE
BUFFER 59 Q CODES
POST-AMBLE
CONCATENATED 6-SEGMENT VL-DATA

FIG. 15

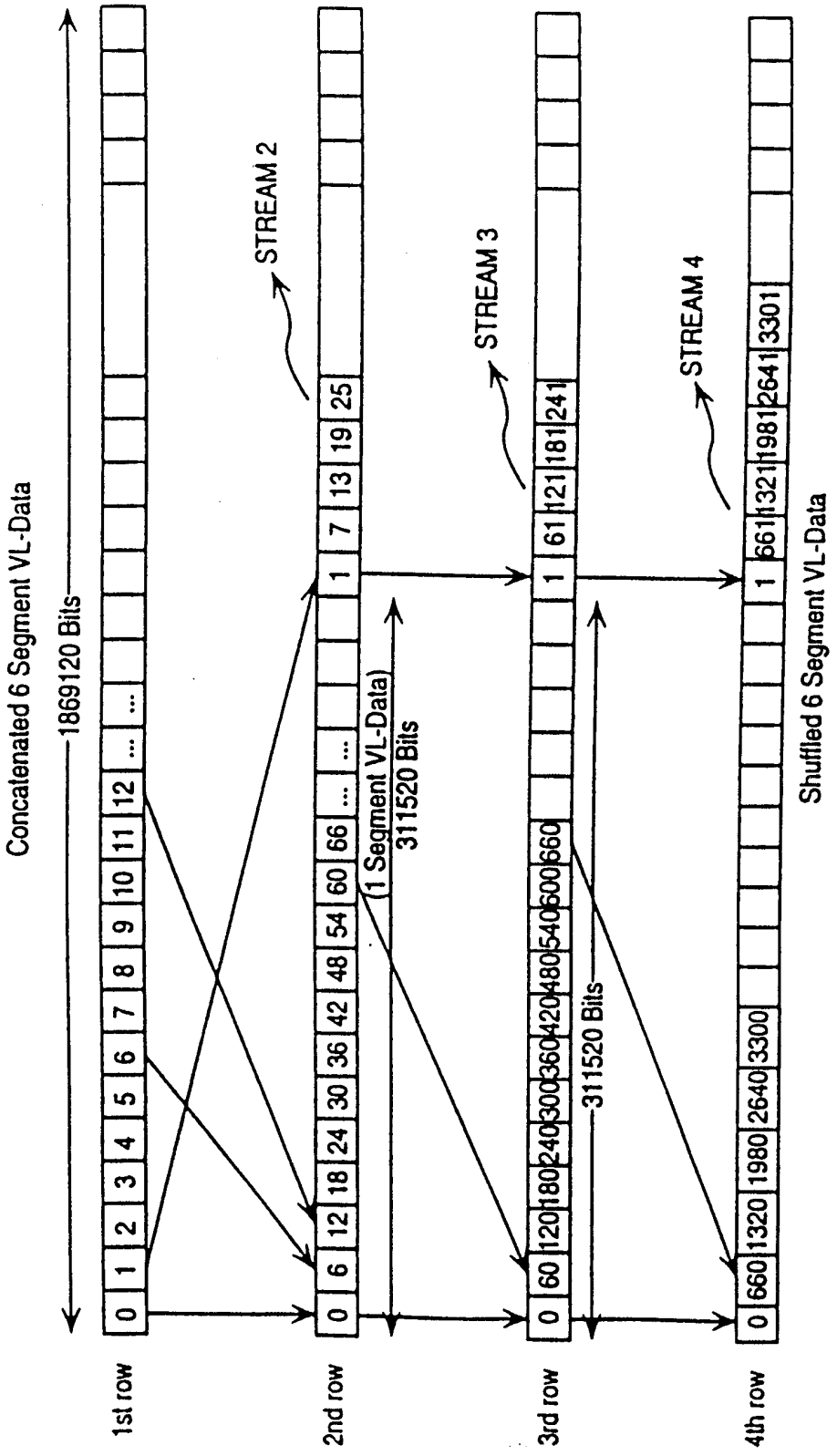

INTER SEGMENT VL-DATA SHUFFLING
Concatenated 6 Segment VL-Data
1869120 Bits
1st row
2nd row
3rd row
4th row
STREAM 2
STREAM 3
STREAM 4
(1 Segment VL-Data)
311520 Bits
311520 Bits
Shuffled 6 Segment VL-Data

FIG. 16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/03599

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :HO3M 13/03
US CL : 714/761, 787

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 714/761, 787, 788, 786, 807, 738, 739, 52; 341/143

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,159,452 A (KINOSHITA et al) 27 October 1992, col. 3 line 15-column 4 line 39	1-30
X,P	US 5,878,183 A (SUGIYANA et al) 02 March 1999, col. 5 line 41-col. 6 line 63.	1-30

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search

16 MAY 2000

Date of mailing of the international search report

13 JUN 2000

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

For CHRISTINE T. L. TU

Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998) ★