Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2736888 A1 2010/03/18

(21) 2 736 888

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2009/09/10
(87) Date publication PCT/PCT Publication Date: 2010/03/18
(85) Entrée phase nationale/National Entry: 2011/03/10
(86) N° demande PCT/PCT Application No.: US 2009/056473
(87) N° publication PCT/PCT Publication No.: 2010/030752
(30) Priorités/Priorities: 2008/09/11 (US61/096,223);
2008/12/05 (US12/329,248)

(51) Cl.Int./Int.Cl. *H04N 7/24* (2011.01)

(71) Demandeur/Applicant:
GOOGLE INC., US

(72) Inventeurs/Inventors:
XU, YAOWU, US;
WILKINS, PAUL, US;
BANKOSKI, JAMES, US

(74) Agent: SIM & MCBURNEY

(54) Titre : SYSTEME ET PROCEDE DE DECODAGE AU MOYEN D'UN TRAITEMENT PARALLELE
(54) Title: SYSTEM AND METHOD FOR DECODING USING PARALLEL PROCESSING

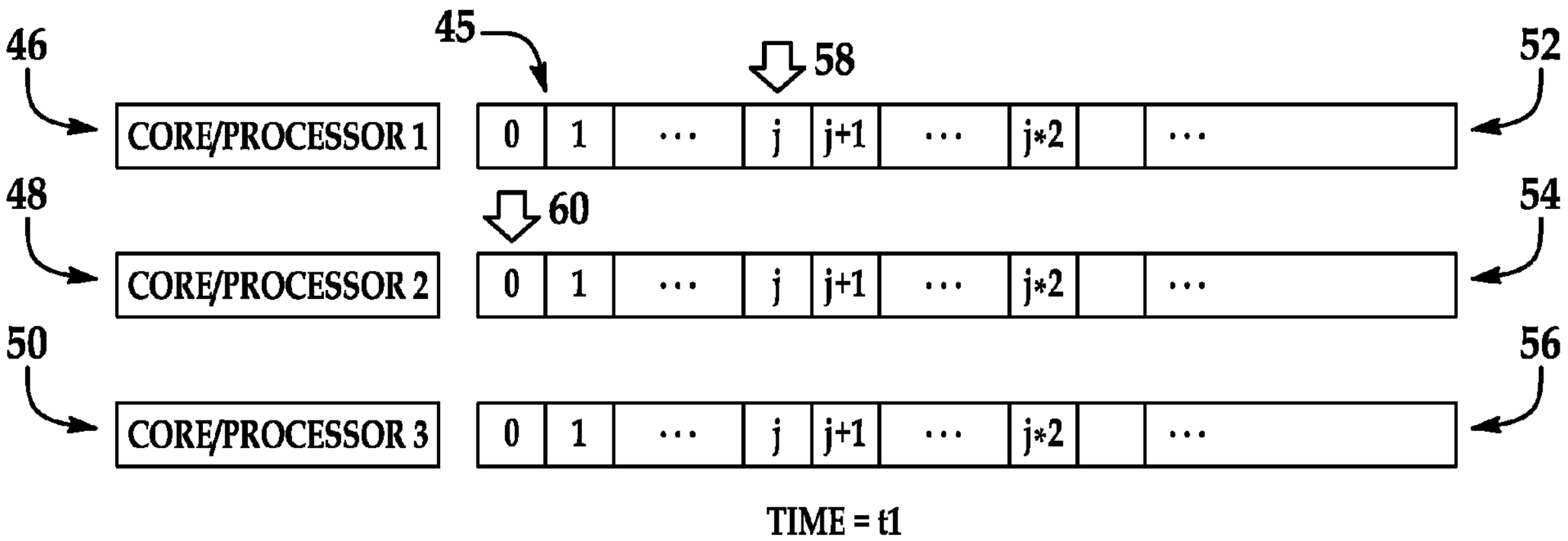

FIG. 6A

(57) Abrégé/Abstract:
A method for decoding a stream of encoded video data is disclosed. The video stream includes partitions that have been compressed using lossless encoding. Each partition includes rows that have also been encoded using intra-frame or inter-frame encoding, for example. During the decoding process, two or more of the partitions are entropy decoded on two or more processors in parallel, except that partitions containing adjacent rows in the frame are decoded with an offset so that at least a portion of the output of the entropy decoding of one partition can be used as input in the entropy and intra/inter-frame decoding of the other.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 March 2010 (18.03.2010)

PCT

(10) International Publication Number
WO 2010/030752 A3

(51) International Patent Classification:
H04N 7/24 (2006.01)

(21) International Application Number:
PCT/US2009/056473

(22) International Filing Date:
10 September 2009 (10.09.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/096,223 11 September 2008 (11.09.2008) US
12/329,248 5 December 2008 (05.12.2008) US

(71) Applicant *(for all designated States except US)*: **ON2 TECHNOLOGIES, INC.** [US/US]; 3 Corporate Drive, Suite 100, Clifton Park, New York 12065 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **XU, Yaowu** [CN/US]; c/o On2 Technologies, Inc., 3 Corporate Drive, Suite 100, Clifton Park, New York 12065 (US). **WILKINS, Paul** [GB/US]; c/o On2 Technologies, Inc., 3 Corporate Drive, Suite 100, Clifton Park, New York 12065 (US). **BANKOSKI, James** [US/US]; c/o On2 Technologies, Inc., 3 Corporate Drive, Suite 100, Clifton Park, New York 12065 (US).

(74) Agents: **BASILE, JR., Andrew, R.** et al.; Young Basile Hanlon & Macfarlane, PC, 3001 West Big Beaver Rd., Suite 624, Troy, MI 48084 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(88) Date of publication of the international search report:
14 May 2010

(54) Title: SYSTEM AND METHOD FOR DECODING USING PARALLEL PROCESSING

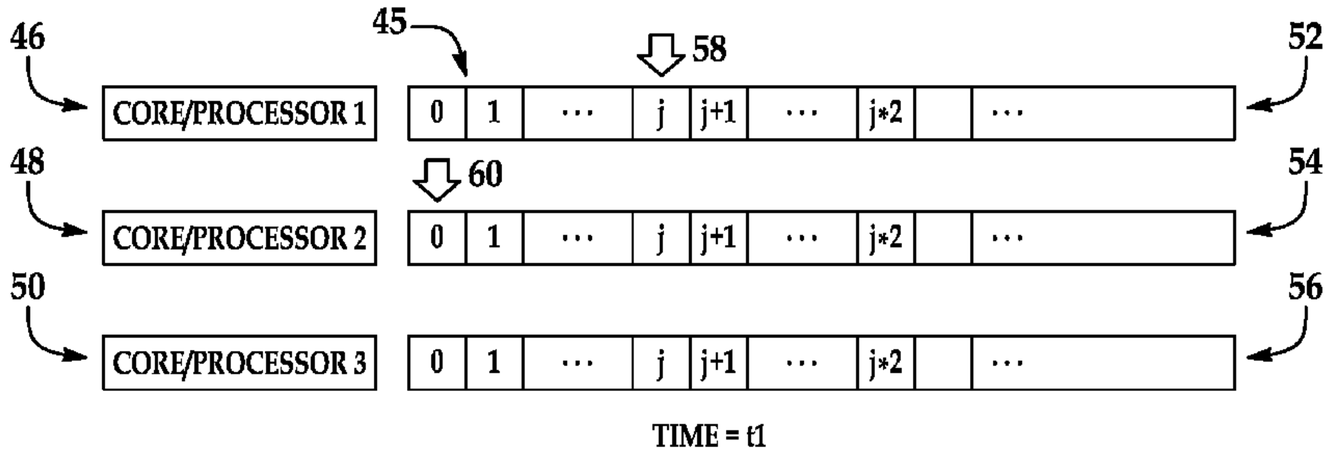

FIG. 6A

(57) Abstract: A method for decoding a stream of encoded video data is disclosed. The video stream includes partitions that have been compressed using lossless encoding. Each partition includes rows that have also been encoded using intra-frame or inter-frame encoding, for example. During the decoding process, two or more of the partitions are entropy decoded on two or more processors in parallel, except that partitions containing adjacent rows in the frame are decoded with an offset so that at least a portion of the output of the entropy decoding of one partition can be used as input in the entropy and intra/inter-frame decoding of the other.

WO 2010/030752 A3

SYSTEM AND METHOD FOR DECODING USING PARALLEL PROCESSING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. patent application no. 12/329,248, filed December 5, 2008, which claims priority to U.S. provisional patent application no. 61/096,223, filed September 11, 2008, both of which are hereby incorporated by reference in their entirety.

TECHNICAL FIELD

[0002] The present invention relates in general to video decoding using multiple processors.

BACKGROUND

[0003] An increasing number of applications today make use of digital video for various purposes including, for example, remote business meetings via video conferencing, high definition video entertainment, video advertisements, and sharing of user-generated videos. As technology is evolving, people have higher expectations for video quality and expect high resolution video with smooth playback at a high frame rate.

[0004] There can be many factors to consider when selecting a video coder for encoding, storing and transmitting digital video. Some applications may require excellent video quality where others may need to comply with various constraints including, for example, bandwidth or storage requirements. To permit higher quality transmission of video while limiting bandwidth consumption, a number of video compression schemes are noted including proprietary formats such as VPx (promulgated by On2 Technologies, Inc. of Clifton Park, New York), H.264 standard promulgated by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), including present and future versions thereof. H.264 is also known as MPEG-4 Part 10 or MPEG-4 AVC (formally, ISO/IEC 14496-10).

[0005] There are many types of video encoding schemes that allow video data to be compressed and recovered. The H.264 standard, for example, offers more efficient methods of video coding by incorporating entropy coding methods such as Context-based Adaptive Variable Length Coding (CAVLC) and Context-based Adaptive Binary Arithmetic Coding (CABAC). For video data that is encoded using CAVLC, some modern decompression

systems have adopted the use of a multi-core processor or multiprocessors to increase overall video decoding speed.

SUMMARY

[0006] An embodiment of the invention is disclosed as a method for decoding a stream of encoded video data including a plurality of partitions that have been compressed using at least a first encoding scheme. The method includes selecting at least a first one of the partitions that includes at least one row of blocks that has been encoded using at least a second encoding scheme. A second partition is selected that includes at least one row of blocks encoded using the second encoding scheme. The first partition is decoded by a first processor, and the second partition is decoded by a second processor. The decoding of the second partition is offset by a specified number of blocks so that at least a portion of the output from the decoding of the first partition is used as input in decoding the second partition. Further, the decoding of the first partition is offset by a specified number of blocks so that at least a portion of the output from the decoding of the second partition is used as input in decoding the first partition.

[0007] Another embodiment of the invention is disclosed as a method for decoding a stream of video data representing at least one frame image composed of a plurality of rows of blocks. The data is encoded using a first encoding scheme and a second encoding scheme. The method includes providing a plurality of processors having shared primary memory space and reading a record in the video data that indicates a partition count and a partition location offset. The video data is divided into a plurality of partitions based on the partition count and partition location offset. A first one and a second one of the plurality of partitions are identified. The first one of the plurality of partitions includes encoded context information that can be used to decode the second one of the plurality of partitions in accordance with the first encoding scheme. The first one of the plurality of partitions is decoded in accordance with the second encoding scheme using a first one of the plurality of processors. The second one of the plurality of partitions is decoded in accordance with the first decoding scheme using a second one of the plurality of processors. The decoding of the second partition is offset by a specified number of blocks where the specified number is at least as large as the number of blocks in the encoded context information and at least a

portion of the output from the decoding of the first partition can be used as input in decoding the second partition.

[0008] Another embodiment of the invention is disclosed as a method for encoding video data including at least one frame having a plurality of rows of blocks. The method includes encoding the rows using a second encoding scheme. The input to encoding each row includes information contained in an adjacent row. The plurality of rows is divided into a plurality of partitions where at least two adjacent rows are placed into separate partitions. Each of the plurality of the partitions is further encoded using a first encoding scheme. Values are recorded indicative of the number or partitions into which the plurality of rows have been divided and the location of the partitions within the encoded video data.

[0009] These and other embodiments of the invention are described in additional detail hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

[00010] The description herein makes reference to the accompanying drawings wherein like reference numerals refer to like parts throughout the several views, and wherein:

[00011] FIG. 1 is a diagram of the hierarchy of layers in a compressed video bitstream in accordance with one embodiment of the present invention.

[00012] FIG. 2 is a block diagram of a video compression system in accordance with one embodiment of the present invention.

[00013] FIG. 3 is a block diagram of a video decompression system in accordance with one embodiment of the present invention.

[00014] FIG. 4 is a schematic diagram of a frame and its corresponding partitions outputted from the video compression system of FIG. 2.

[00015] FIG. 5 is a schematic diagram of an encoded video frame in a bitstream outputted from the video compression system of FIG. 2 and sent to the video decompression system of FIG. 3.

[00016] FIGS. 6A-6B are timing diagrams illustrating the staging and synchronization of cores on a multi-core processor used in the video decompression system of FIG. 3.

[00017] FIG. 7A is a schematic diagram showing data-dependent macroblocks and an offset calculation based used in the video compression and decompression systems of FIGS. 2 and 3.

[00018] FIG. 7B is a schematic diagram showing data-dependent macroblocks and an alternative offset calculation used in the video compression and decompression systems of FIGS. 2 and 3.

DETAILED DESCRIPTION

[00019] Referring to FIG. 1, video coding standards, such as H.264, provide a defined hierarchy of layers 10 for a video stream 11. The highest level in the layer can be a video sequence 13. At the next level, video sequence 13 consists of a number of adjacent frames 15. Number of adjacent frames 15 can be further subdivided into a single frame 17. At the next level, frame 17 can be composed of a series of fix-sized macroblocks 20, which contain compressed data corresponding to, for example, a 16x16 block of displayed pixels in frame 17. Each macroblock contains luminance and chrominance data for the corresponding pixels. Macroblocks 20 can also be of any other suitable size such as 16x8 pixel groups or 8x16 pixel groups. Macroblocks 20 are further subdivided into blocks. A block, for example, can be a 4x4 pixel group that can further describe the luminance and chrominance data for the corresponding pixels. Blocks can also be of any other suitable size such as 16×16, 16×8, 8×16, 8×8, 8×4, 4×8 and 4×4 pixels groups.

[00020] Although the description of embodiments are described in the context of the VP8 video coding format, alternative embodiments of the present invention can be implemented in the context of other video coding formats. Further, the embodiments are not limited to any specific video coding standard or format.

[00021] Referring to FIG. 2, in accordance with one embodiment, to encode an input video stream 16, an encoder 14 performs the following functions in a forward path (shown by the solid connection lines) to produce an encoded bitstream 26: intra/inter prediction 18, transform 19, quantization 22 and entropy encoding 24. Encoder 14 also includes a reconstruction path (shown by the dotted connection lines) to reconstruct a frame for encoding of further macroblocks. Encoder 14 performs the following functions in the reconstruction path: dequantization 28, inverse transformation 30, reconstruction 32 and loop filtering 34. Other structural variations of encoder 14 can be used to encode bitstream 26.

[00022] When input video stream 16 is presented for encoding, each frame 17 within input video stream 16 can be processed in units of macroblocks. At intra/inter prediction stage 18, each macroblock can be encoded using either intra prediction or inter prediction

mode. In the case of intra-prediction, a prediction macroblock can be formed from samples in the current frame that have been previously encoded and reconstructed. In the case of inter-prediction, a prediction macroblock can be formed from one or more reference frames that have already been encoded and reconstructed.

[0023] Next, still referring to FIG. 2, the prediction macroblock can be subtracted from the current macroblock to produce a residual macroblock (residual). Transform stage 19 transform codes the residual signal to coefficients and quantization stage 22 quantizes the coefficients to provide a set of quantized transformed coefficients. The quantized transformed coefficients are then entropy coded by entropy encoding stage 24. The entropy-coded coefficients, together with the information required to decode the macroblock, such as the type of prediction mode used, motion vectors and quantizer value, are output to compressed bitstream 26.

[0024] The reconstruction path in FIG. 2, can be present to permit that both the encoder and the decoder use the same reference frames required to decode the macroblocks. The reconstruction path, similar to functions that take place during the decoding process, which are discussed in more detail below, includes dequantizing the transformed coefficients by dequantization stage 28 and inverse transforming the coefficients by inverse transform stage 30 to produce a derivative residual macroblock (derivative residual). At the reconstruction stage 32, the prediction macroblock can be added to the derivative residual to create a reconstructed macroblock. A loop filter 34 can be applied to the reconstructed macroblock to reduce distortion.

[0025] Referring to FIG. 3, in accordance with one embodiment, to decode compressed bitstream 26, a decoder 21, similar to the reconstruction path of encoder 14 discussed previously, performs the following functions to produce an output video stream 35: entropy decoding 25, dequantization 27, inverse transformation 29, intra/inter prediction 23, reconstruction 31, loop filter 34 and deblocking filtering 33. Other structural variations of decoder 21 can be used to decode compressed bitstream 26.

[0026] When compressed bitstream 26 is presented for decoding, the data elements can be decoded by entropy decoding stage 25 to produce a set of quantized coefficients. Dequantization stage 27 dequantizes and inverse transform stage 29 inverse transforms the coefficients to produce a derivative residual that is identical to that created by the reconstruction stage in encoder 14. Using the type of prediction mode and/or motion vector

information decoded from the compressed bitstream 26, at intra/inter prediction stage 23, decoder 21 creates the same prediction macroblock as was created in encoder 14. At the reconstruction stage 33, the prediction macroblock can be added to the derivative residual to create a reconstructed macroblock. The loop filter 34 can be applied to the reconstructed macroblock to reduce blocking artifacts. A deblocking filter 33 can be applied to video image frames to further reduce blocking distortion and the result can be outputted to output video stream 35.

[00027] Current context-based entropy coding methods, such as Context-based Adaptive Arithmetic Coding (CABAC), are limited by dependencies that exploit spatial locality by requiring macroblocks to reference neighboring macroblocks and that exploit temporal localities by requiring macroblocks to reference macroblocks from another frame. Because of these dependencies and the adaptivity, encoder 14 codes the bitstream in a sequential order using context data from neighboring macroblocks. Such sequential dependency created by encoder 14 causes the compressed bitstream 26 to be decoded in a sequential fashion by decoder 21. Such sequential decoding can be adequate when decoding using a single-core processor. On the other hand, if a multi-core processor or a multi-processor system is used during decoding, the computing power of the multi-core processor or the multi-processor system would not be effectively utilized.

[00028] Although the disclosure has and will continue to describe embodiments of the present invention with reference to a multi-core processor and the creation of threads on the multi-core processor, embodiments of the present invention can also be implemented with other suitable computer systems, such as a device containing multiple processors.

[00029] According to one embodiment, encoder 14 divides the compressed bitstream into partitions 36 rather than a single stream of serialized data. With reference to FIG. 4 and by way of example only, the compressed bitstream can be divided into four partitions, which are designated as Data Partitions 1-4. Other numbers of partitions are also suitable. Since each partition can be the subject of a separate decoding process when they are decoded by decoder 21, the serialized dependency can be broken up in the compressed data without losing coding efficiency.

[00030] Referring to FIG. 4, frame 17 is shown with divided macroblock rows 38. Macroblock rows 38 consist of individual macroblocks 20. Continuing with the example, every Nth macroblock row 38 can be grouped into one of partitions 36 (where N is the total

number of partitions). In this example, there are four partitions and macroblock rows 0, 4, 8, 12, etc. are grouped into partition 1. Macroblock rows 1, 5, 9 and 13, etc. are grouped into partition 2. Macroblock rows 2, 6, 10, 14, etc. are grouped into partition 3. Macroblock rows 3, 7, 11, 15, etc. are grouped into partition 4. As a result, each partition 36 includes contiguous macroblocks, but in this instance, each partition 36 does not contain contiguous macroblock rows 38. In other words, macroblock rows of blocks in the first partition and macroblock rows in the second partition can be derived from two adjacent macroblock rows in a frame. Other grouping mechanisms are also available and are not limited to separating regions by macroblock row or grouping every Nth macroblock row into a partition. Depending on the grouping mechanism, in another example, macroblock rows that are contiguous may also be grouped into the same partition 36.

[00031] An alternative grouping mechanism may include, for example, grouping a row of blocks from a first frame and a corresponding row of blocks in a second frame. The row of blocks from the first frame can be packed in the first partition and the corresponding row of blocks in the second frame can be packed in the second partition. A first processor can decode the row of blocks from the first frame and a second processor can decode the row of blocks from the second frame. In this manner, the decoder can decode at least one block in the second partition using information from a block that is already decoded by the first processor.

[00032] Each of the partitions 36 can be compressed using two separate encoding schemes. The first encoding scheme can be lossless encoding using, for example, context-based arithmetic coding like CABAC. Other lossless encoding techniques may also be used. Referring back to FIG. 1, the first encoding scheme may be realized by, for example, entropy encoding stage 24.

[00033] Still referring to FIG. 1, the second encoding scheme, which can take place before the first encoding scheme, may be realized by at least one of intra/inter prediction stage 18, transform stage 19, and quantization 22. The second encoding scheme can encode blocks in each of the partitions 36 by using information contained in other partitions. For example, if a frame is divided into two partitions, the second encoding scheme can encode the second partition using information contained in the macroblock rows of the first partition.

[00034] Referring to FIG. 5, an encoded video frame 39 from compressed bitstream 26 is shown. For simplicity, only parts of the bitstream that are pertinent to embodiments of the

invention are shown. Encoded video frame 39 contains a video frame header 44 which contains bits for a number of partitions 40 and bits for offsets of each partition 42. Encoded video frame 39 also includes the encoded data from data partitions 36 illustrated as P_1-P_N where, as discussed previously, N is the total number of partitions in video frame 17.

[00035] Once encoder 14 has divided frame 17 into partitions 36, encoder 14 writes data into video frame header 44 to indicate number of partitions 40 and offsets of each partition 42. Number of partitions 40 and offsets of each partition 42 can be represented in frame 17 by a bit, a byte or any other record that can relay the specific information to decoder 21. Decoder 21 reads the number of data partitions 40 from video frame header 44 in order to decode the compressed data. In one example, two bits may be used to represent the number of partitions. One or more bits can be used to indicate the number of data partitions (or partition count). Other coding schemes can also be used to code the number of partitions into the bitstream. The following list indicates how two bits can represent the number of partitions:

	BIT 1	BIT 2	NUMBER OF PARTITIONS
[00037]	0	0	One partition
[00038]	0	1	Two partitions
[00039]	1	0	Four partitions
[00040]	1	1	Eight partitions

[00041] If the number of data partitions is greater than one, decoder 21 also needs information about the positions of the data partitions 36 within the compressed bitstream 26. The offsets of each partition 42 (also referred to as partition location offsets) enable direct access to each partition during decoding.

[00042] In one example, offset of each partition 42 can be relative to the beginning of the bitstream and can be encoded and written into the bitstream 26. In another example, the offset for each data partition can be encoded and written into the bitstream except for the first partition since the first partition implicitly begins in the bitstream 26 after the offsets of each partition 42. The foregoing is merely exemplary. Other suitable data structures, flags or records such words and bytes, can be used to transmit partition count and partition location offset information.

[00043] Although the number of data partitions can be the same for each frame 17 throughout the input video sequence 16, the number of data partitions may also differ from frame to frame. Accordingly, each frame 17 would have a different number of partitions 40.

The number of bits that are used to represent the number of partitions may also differ from frame to frame. Accordingly, each frame 17 could be divided into varying numbers of partitions.

[00044] Once the data has been compressed into bitstream 26 with the proper partition data information (i.e. number of partitions 40 and offsets of partitions 42), decoder 21 can decode the data partitions 36 on a multi-core processor in parallel. In this manner, each processor core may be responsible for decoding one of the data partitions 36. Since multi-core processors typically have more than one processing core and shared memory space, the workload can be allocated between each core as evenly as possible. Each core can use the shared memory space as an efficient way of sharing data between each core decoding each data partition 36.

[00045] For example, if there are two processors decoding two partitions, respectively, the first processor will begin decoding the first partition. The second processor can then decode macroblocks of the second partition and can use information received from the first processor, which has begun decoding macroblocks of the first partition. Concurrently with the second processor, the first processor can continue decoding macroblocks of the first partition and can use information received from the second processor. Accordingly, both the first and second processors can have the information necessary to properly decode macroblocks in their respective partitions.

[00046] Furthermore, as discussed in more detail below, when decoding a macroblock row of the second partition that is dependent on the first partition, a macroblock that is currently being processed in the second partition is offset by a specified number of macroblocks. In this manner, at least a portion of the output of the decoding of the first partition can be used as input in the decoding of the macroblock that is currently being processed in the second partition. Likewise, when decoding a macroblock row of the first partition that is dependent on the second partition, a macroblock that is currently being processed in the first partition is offset by a specified number of macroblocks so that at least a portion of the output of the decoding of the second partition can be used as input in the decoding of the macroblock that is currently being processed in the first partition.

[00047] When decoding the compressed bitstream, decoder 21 determines the number of threads needed to decode the data, which can be based on the number of partitions 40 in each encoded frame 39. For example, if number of partitions 40 indicates that there are four

partitions in encoded frame 39, decoder 21 creates four threads with each thread decoding one of the data partitions. Referring to FIG. 4, as an example, decoder 21 can determine that four data partitions have been created. Hence, if decoder 21 is using a multi-core processor, it can create four separate threads to decode the data from that specific frame.

[00048] As discussed previously, macroblocks 20 within each frame use context data from neighboring macroblocks when being encoded. When decoding macroblocks 20, the decoder will need the same context data in order to decode the macroblocks properly. On the decoder side, the context data can be available only after the neighboring macroblocks have already been decoded by the current thread or other threads. In order to decode properly, the decoder includes a staging and synchronization mechanism for managing the decoding of the multiple threads.

[00049] With reference to FIGS. 6A and 6B, a time diagram shows the staging and synchronization mechanism to decode partitions 36 on threads of a multi-core processor in accordance with an embodiment of the present invention. FIGS. 6A and 6B illustrate an exemplary partial image frame 45 at various stages of the decoding process. The example is simplified for purposes of this disclosure and the number of partitions 36 is limited to three. Each partition 36 can be assigned to one of the three threads 46, 48 and 50. As discussed previously, each partition 36 includes contiguous macroblocks.

[00050] As depicted in FIGS. 6A and 6B, as an example, three threads 46, 48 and 50 are shown, and each of threads 46, 48 and 50 are capable of performing decoding in parallel with each other. Each of the three threads 46, 48 and 50 processes one partition in a serial manner while all three partitions 40 are processed in parallel with each other.

[00051] Each of FIGS. 6A and 6B contain an arrow that illustrates which macroblock is currently being decoded in each macroblock row, which macroblocks have been decoded in each macroblock row, and which macroblocks have yet to be decoded in each macroblock row. If the arrow is pointing to a specific macroblock, that macroblock is currently being decoded. Any macroblock to the left of the arrow (if any) has already been decoded in that row. Any macroblock to the right of the arrow has yet to be decoded. Although the macroblocks illustrated in FIGS. 6A and 6B all have similar sizes, the techniques of this disclosure are not limited in this respect. Other block sizes, as discussed previously, can also be used with embodiments of the present invention.

[00052] Referring to FIG. 6A, at time t1, thread 46 has initiated decoding of a first macroblock row 52. Thread 46 is currently processing macroblock j in first macroblock row 52 as shown by arrow 58. Macroblocks 0 to j-1 have already been decoded in first macroblock row 52. Macroblocks j+1 to the end of first macroblock row 52 have yet to be decoded in first macroblock row 52. Thread 48 has also initiated decoding of a second macroblock row 54. Thread 48 is currently processing macroblock 0 in second macroblock row 54 as shown by arrow 60. Macroblocks 1 to the end of second macroblock row 54 have been decoded in second macroblock row 54. Thread 50 has not begun decoding of a third macroblock row 56. No macroblocks have been decoded or are currently being decoded in third macroblock row 56.

[00053] Referring to FIG. 6B, at time t2, thread 46 has continued decoding of first macroblock row 52. Thread 46 is currently processing macroblock j*2 in first macroblock row 52 as shown by arrow 62. Macroblocks 0 to j*2-1 have already been decoded in first macroblock row 52. Macroblocks j*2+1 to the end of first macroblock row 52 have yet to be decoded in first macroblock row 52. Thread 48 has also continued decoding of second macroblock row 54. Thread 48 is currently processing macroblock j in second macroblock row 54 as shown by arrow 64. Macroblocks 0 to j-1 have already been decoded in second macroblock row 54. Macroblocks j+1 to the end of second macroblock row 54 have yet to be decoded in second macroblock row 54. Thread 50 has also initiated decoding of a third macroblock row 56. Thread 50 is currently processing macroblock 0 in third macroblock row 56 as shown by arrow 66. Macroblocks 1 to the end of third macroblock row 56 have yet to be decoded in third macroblock row 56.

[00054] Previous decoding mechanisms were unable to efficiently use a multi-core processor to decode a compressed bitstream because processing of a macroblock row could not be initiated until the upper adjacent macroblock row had been completely decoded. The difficulty of previous decoding mechanisms stems from the encoding phase. When data is encoded using traditional encoding techniques, spatial dependencies within macroblocks imply a specific order of processing of the macroblocks. Furthermore, once the frame has been encoded, a specific macroblock row cannot be discerned until the row has been completely decoded. Accordingly, video coding methods incorporating entropy coding methods such as CABAC created serialized dependencies which were passed to the decoder. As a result of these serialized dependencies, decoding schemes had limited efficiency

because information for each computer processing system (e.g. threads 46, 48 and 50) was not available until the decoding process has been completed on that macroblock row.

[00055] Utilizing the parallel processing staging and synchronization mechanism illustrated in FIGS. 6A and 6B allows decoder 21 to efficiently accelerate the decoding process of image frames. Because each partition 36 can be subject to a separate decoding process, interdependencies between partitions can be managed by embodiments of the staging and synchronization scheme discussed previously in connection with FIGS. 6A and 6B. Using this staging and synchronization decoding scheme, each thread 46, 48 and 50 that decodes an assigned partition can exploit context data from neighboring macroblocks. Thus, decoder 21 can decode macroblocks that contain context data necessary to decode a current macroblock before the preceding macroblock row has been completely decoded.

[00056] Referring again to FIGS. 6A and 6B, offset j can be determined by examining the size of the context data used in the preceding macroblock row (e.g. measured in a number of macroblocks) during the encoding process. Offset j can be represented in frame 17 by a bit, a byte or any other record that can relay the size of the context data to decoder 21. FIGS. 7A and 7B illustrate two alternatives for the size of offset j.

[00057] Referring to FIG. 7A, in one embodiment, current macroblock 68 is currently being processed. Current macroblock 68 uses context data from the left, top-left, top and top-right macroblocks during encoding. In other words, current macroblock 68 uses information from macroblocks: (r+1, c-1), (r, c-1), (r, c) and (r, c+1). In order to properly decode current macroblock 68, macroblocks (r+1, c-1), (r, c-1), (r, c) and (r, c+1) should be decoded before current macroblock 68. Since, as discussed previously, decoding of macroblocks can be performed in a serial fashion, macroblock (r+1, c-1) can be decoded before current macroblock 68. Further, in the preceding macroblock row (i.e. macroblock row r), since the encoding process uses (r, c+1) as the rightmost macroblock, the decoder can use (r, c+1) as the rightmost macroblock during decoding as well. Thus, offset j can be determined by subtracting the column row position of rightmost macroblock of the preceding row used during encoding of the current macroblock from the column row position of the current macroblock being processed. In FIG. 7A, offset j would be determined by subtracting the column row position of macroblock (r, c+1) from the column position of current macroblock 68 (i.e. (r+1, c)), or c+1-c, giving rise to an offset of 1.

[00058] Referring to FIG. 7B, in one embodiment, current macroblock 68' is currently being processed. Current macroblock 68' uses information from macroblocks: (r+1, c-1), (r, c-1), (r, c), (r, c+1), (r, c+2) and (r, c+3). In order to properly decode current macroblock 68', macroblocks (r+1, c-1), (r, c-1), (r, c) (r, c+1), (r, c+2) and (r, c+3) should be decoded before current macroblock 68'. Since, as discussed previously, decoding of macroblocks can be performed in a serial fashion, macroblock (r+1, c-1) can be decoded before current macroblock 68'. Further, in the preceding macroblock row (i.e. macroblock row r), since the encoding process uses (r, c+3) as the rightmost macroblock, the decoder can use (r, c+3) as the rightmost macroblock during decoding as well. As discussed previously, offset j can be determined by subtracting the column row position of rightmost macroblock of the preceding row used during encoding of the current macroblock from the column row position of the current macroblock being processed. In FIG. 7A, offset j would be calculated by subtracting the column row position of macroblock (r, c+3) from the column position of current macroblock 68' (i.e. (r+1, c)), or c+3-c, giving rise to an offset of 3.

[00059] In the preferred embodiment, the offset can be determined by the specific requirements of the codec. In alternative embodiments, the offset can be specified in the bitstream.

[00060] While the invention has been described in connection with certain embodiments, it is to be understood that the invention is not to be limited to the disclosed embodiments but, on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims, which scope is to be accorded the broadest interpretation so as to encompass all such modifications and equivalent structures as is permitted under the law.

CLAIMS

What is claimed is:

1. A method for decoding a stream of video data representing at least one frame image composed of a plurality of rows of blocks, wherein the data is encoded using a first encoding scheme and a second encoding scheme, comprising:

providing a plurality of processors having shared primary memory spaces;

reading a record in the video data that indicates a partition count and a partition location offset;

dividing the video data into a plurality of partitions based on the partition count and partition location offset;

identifying a first one and a second one of the plurality of partitions, wherein the first one of the plurality of partitions includes encoded context information that can be used to decode the second one of the plurality of partitions in accordance with the first encoding scheme;

decoding the first one of the plurality of partitions in accordance with the second encoding scheme, using a first one of the plurality of processors; and

decoding the second one of the plurality of partitions in accordance with the first decoding scheme, using a second one of the plurality of processors, wherein the decoding of the second partition is offset by a specified number of blocks, the specified number at least as large as the number of blocks in the encoded context information; wherein at least a portion of the output from the decoding of the first partition can be used as input in decoding the second partition.

2. The method of claim 1, wherein the first encoding scheme includes lossless encoding and the second encoding scheme includes at least one of intra-frame prediction and inter-frame prediction.

3. The method of claims 2, wherein the lossless encoding is an entropy encoding scheme.

4. The method of any of claims 1 to 3, wherein at least one row of blocks in the first partition and at least one row of blocks in the second partition are derived from two adjacent rows in an image frame represented by the video data.

5. The method of any of claims 1 to 3, wherein at least a portion of a row of blocks of the output from the decoding of the second partition is used as input in decoding at least a portion of another row of blocks in the first partition in accordance with at least one of the first encoding scheme and the second encoding scheme.

6. The method of claim 5, wherein the row in the second partition is adjacent in a frame of the video information to the row in the first partition.

7. The method of any of claims 1 to 3, wherein the second encoding scheme is configured to encode at least one row of blocks in the first partition using context information contained in at least one row of blocks in the second partition.

8. The method of any of claims 1 to 3, wherein the second encoding scheme is configured to encode at least one row of blocks in the second partition using the context information contained in at least one row of blocks in the first partition.

9. The method of claim 8, wherein decoding the second partition further comprises:

using at least a portion of the information of the blocks most recently decoded by the first processor as the context information for decoding at least one block in the second partition.

10. The method of claim 8, wherein the specified number of blocks is determined based upon the size of the context infonmation used by the second encoding scheme.

11. The method of claim 8, wherein the one row of blocks in the first partition and the one row of blocks in the second partition are derived from corresponding rows in two successive frames of the video data, and wherein decoding the second partition further comprises:

decoding at least one block in the second partition using information from a block that is already decoded by the first one of the plurality of processors.

12. The method of any of claims 1 to 3, further comprising:

reading in the video data a record that indicates the number of partitions.

13. The method of any of claims 1 to 3, further comprising:

reading in the video data a record that indicates the size of the specified macroblock offset.

14. The method of claims 1 or 2, further comprising:

reading in the video data a record that indicates the location of the partitions within the encoded video data.

15. The method of claim 1, wherein the first encoding scheme is context-based arithmetic coding.

16. A method for encoding video data including at least one frame having a plurality of rows of blocks, comprising:

encoding the rows using a second encoding scheme, wherein the input to encoding each row includes information contained in an adjacent row;

dividing the plurality of rows into a plurality of partitions, wherein at least two adjacent rows are placed into separate partitions;

further encoding each of the plurality of the partitions using a first encoding scheme;

recording a value indicative of the number or partitions into which the plurality of rows have been divided; and

recording a value indicative of the location of the partitions within the encoded video data.

17. The method of claim 16, wherein the second encoding scheme includes at least one of intra-frame prediction and inter-frame prediction.

18. The method of claims 16, wherein the first encoding scheme is an entropy encoding scheme.

19. The method of claim 16, wherein the first encoding scheme is context-based arithmetic coding.

20. The method of any of claims 16-19, wherein the number of partitions N is a number greater than one.

21. The method of any of claims 16-19, wherein dividing the rows into a plurality of partitions further comprises:

grouping every Nth row of the frame into a different one of the plurality of partitions, so that each adjacent row in the frame is placed into a separate partition.

22. A method for decoding a stream of encoded video data including a plurality of partitions that have been compressed using at least a first encoding scheme, comprising:

selecting at least a first one of the partitions that includes at least one row of blocks that has also been encoded using at least a second encoding scheme;

selecting at least a second one of the partitions that includes at least one row of blocks that has also been encoded using the second encoding scheme;

decoding the first partition using a first processor; and

decoding the second partition using a second processor, wherein the decoding of the second partition is offset by a specified number of blocks so that at least a portion of the output from the decoding of the first partition is used as input in decoding the second partition in accordance with the first encoding scheme.

23. The method of claim 22, wherein the first encoding scheme includes lossless encoding and the second encoding scheme includes at least one of intra-frame prediction and inter-frame prediction.

24. The method of claims 22 or 23, wherein the one row of blocks in the first partition and the one row of blocks in the second partition are derived from two adjacent rows in an image frame represented by the video data.

25. The method of claims 22 or 23, wherein the lossless encoding is an entropy encoding scheme.

26. The method of claims 22 or 23, wherein at least a portion of a row of blocks of the output from the decoding of the second partition is used as input in decoding at least a portion of another row of blocks in the first partition in accordance with at least one of the first encoding scheme and the second encoding scheme.

27. The method of claim 26, wherein the row in the second partition is adjacent in a frame of the video information to the row in the first partition.

28. The method of claims 22 or 23, wherein the second encoding scheme is configured to encode the one row of blocks in the first partition using context information contained in the one row of blocks in the second partition.

29. The method of claims 22 or 23, wherein the second encoding scheme is configured to encode the one row of blocks in the second partition using context information contained in the one row of blocks in the first partition.

30. The method of claim 29, wherein the second encoding scheme includes at least one of intra-frame prediction and inter-frame prediction.

31. The method of claim 29, wherein the one row of blocks in the first partition and the one row of blocks in the second partition are derived from two adjacent rows in an image frame represented by the video data, and wherein decoding the second partition further comprises:

decoding at least one block in the second partition using information from a block that is already decoded by the first processor, wherein the at least one block in the second partition and the already decoded block in first partition are adjacent to each other the image frame.

32. The method of claim 29, wherein decoding the second partition further comprises:

using at least a portion of the information of the blocks most recently decoded by the first processor as context data for decoding at least one block in the second partition.

33. The method of claim 29, wherein the specified offset is determined based upon the size of the context used by the second encoding scheme.

34. The method of claim 29, wherein the one row of blocks in the first partition and the one row of blocks in the second partition are derived from corresponding rows in two successive frames of the video data, and wherein decoding the second partition further comprises:

decoding at least one block in the second partition using information from a block that is already decoded by the first processor.

35. The method of claims 22 or 23, further comprising:

reading in the video data a record that indicates the number of partitions.

36. The method of claims 22 or 23, further comprising:

reading in the video data a record that indicates the size of the specified macroblock offset.

37. The method of claims 22 or 23, further comprising:

reading in the video data a record that indicates the location of the partitions within the encoded video data.

38. The method of claims 22 or 23, wherein the first encoding scheme is context-based arithmetic coding.

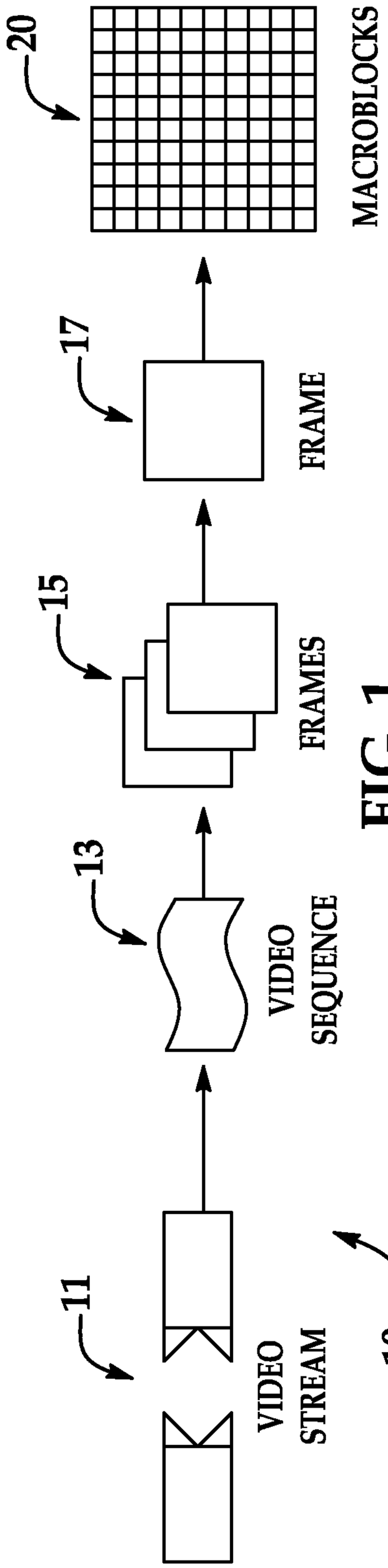
11
VIDEO STREAM
13
VIDEO SEQUENCE
15
FRAMES
17
FRAME
20
MACROBLOCKS

FIG. 1

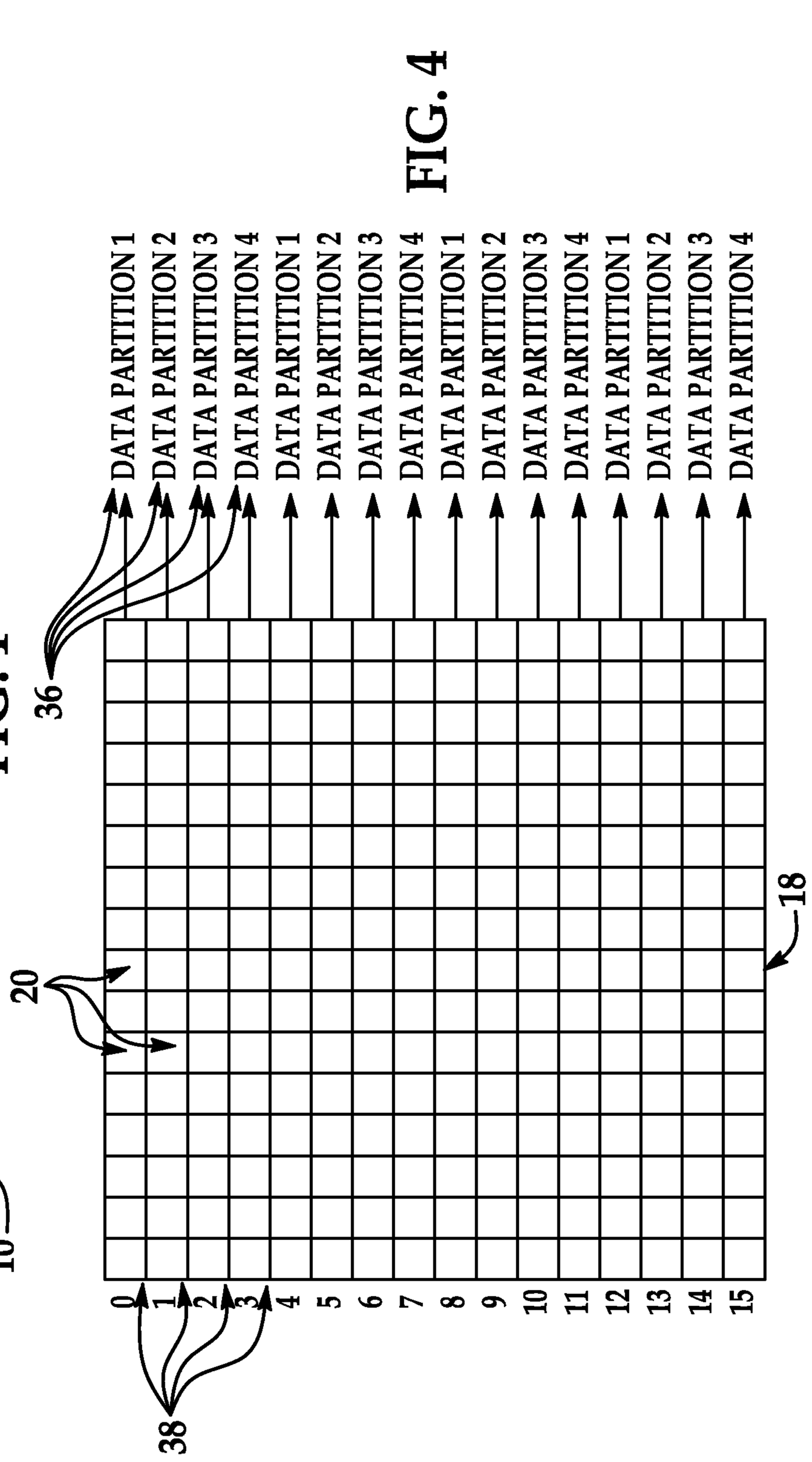
10
18
20
36
38
DATA PARTITION 1
DATA PARTITION 2
DATA PARTITION 3
DATA PARTITION 4
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

FIG. 4

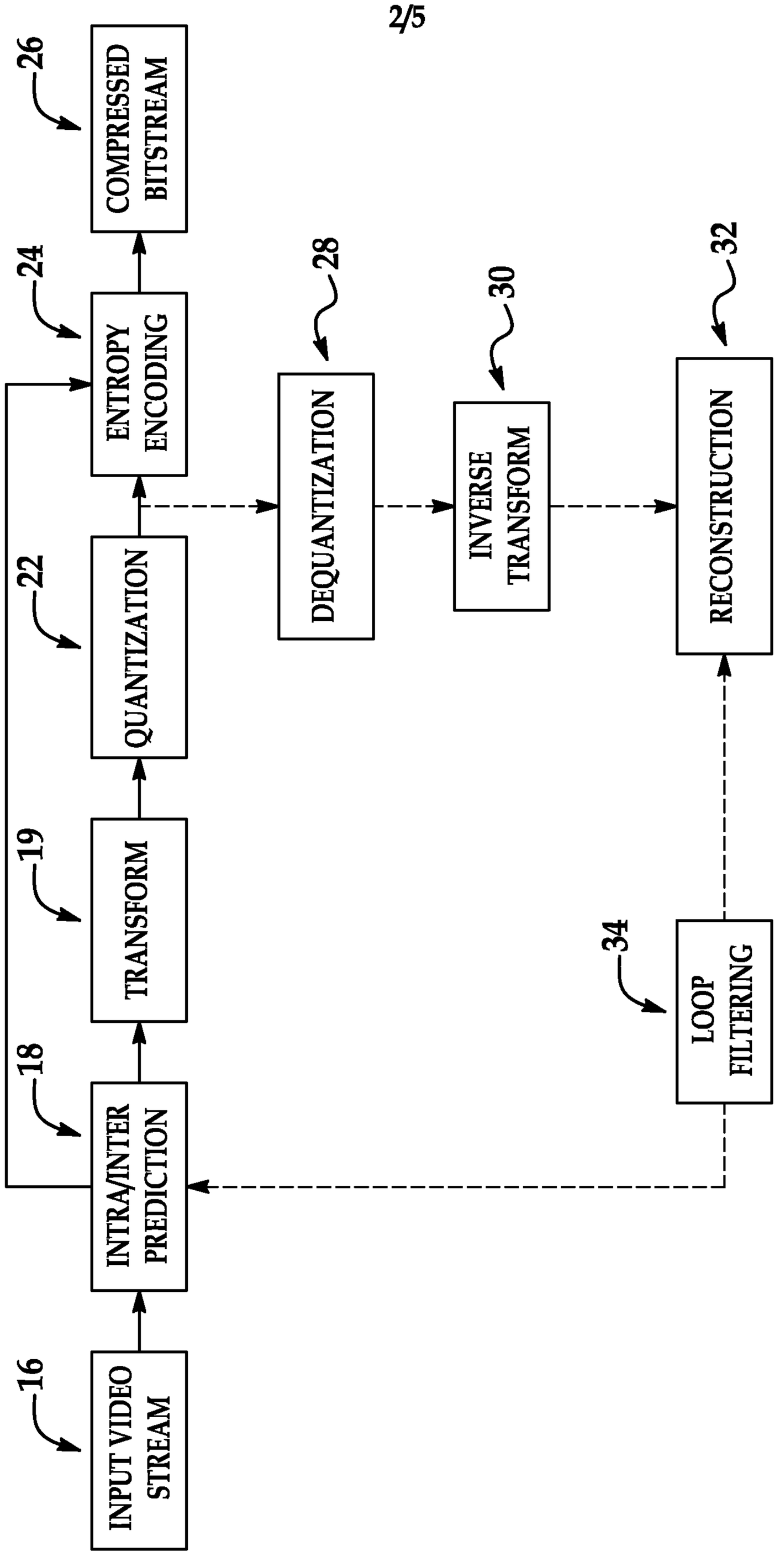
16
INPUT VIDEO STREAM
18
INTRA/INTER PREDICTION
19
TRANSFORM
22
QUANTIZATION
24
ENTROPY ENCODING
26
COMPRESSED BITSTREAM
28
DEQUANTIZATION
30
INVERSE TRANSFORM
32
RECONSTRUCTION
34
LOOP FILTERING
14

FIG. 2

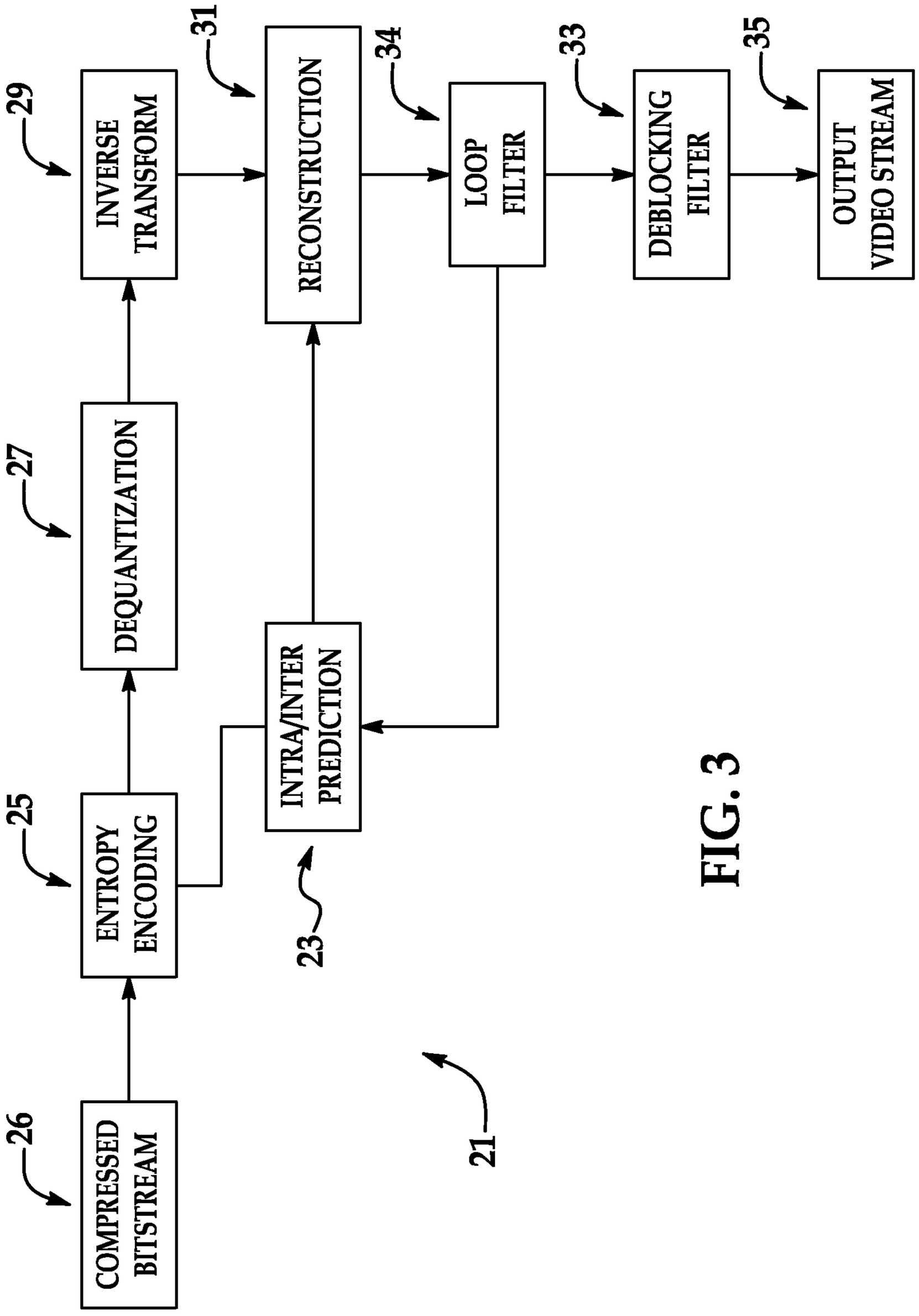
COMPRESSED BITSTREAM
26
ENTROPY ENCODING
25
DEQUANTIZATION
27
INVERSE TRANSFORM
29
INTRA/INTER PREDICTION
23
RECONSTRUCTION
31
LOOP FILTER
34
DEBLOCKING FILTER
33
OUTPUT VIDEO STREAM
35
21

FIG. 3

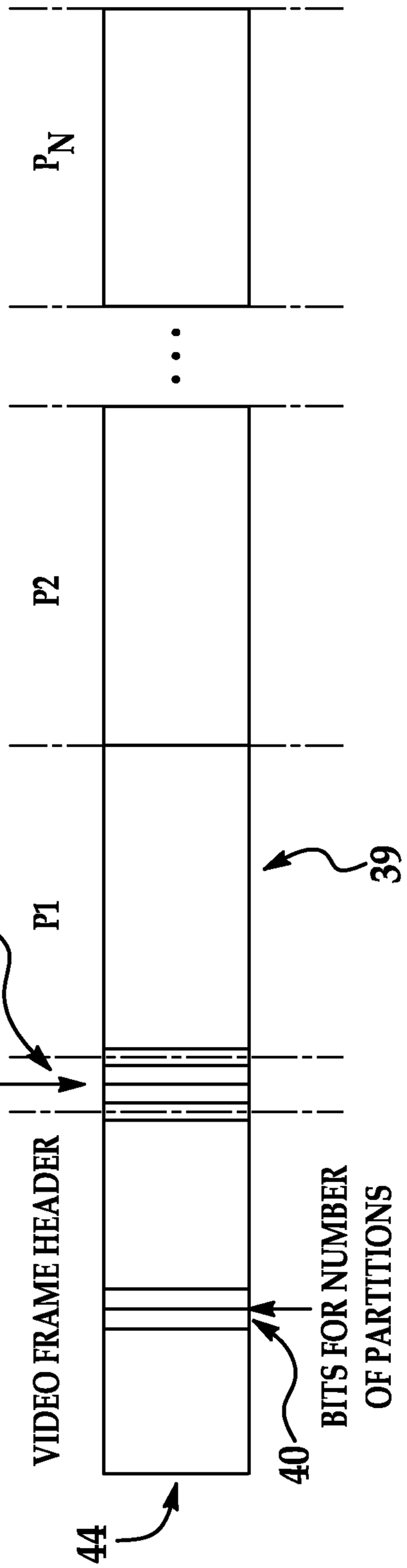
VIDEO FRAME HEADER
OFFSETS OF DATA PARTITION
42
P1
P2
P_N
39
BITS FOR NUMBER OF PARTITIONS
40
44

FIG. 5

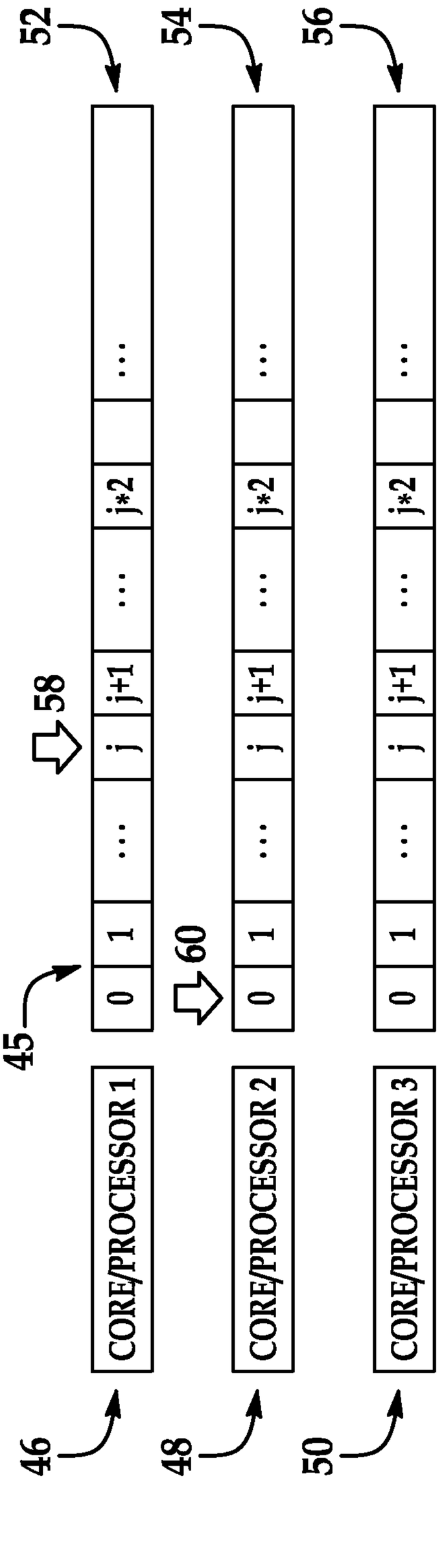
52
54
56
58
60
45
j*2
j+1
j
0
1
CORE/PROCESSOR 1
CORE/PROCESSOR 2
CORE/PROCESSOR 3
46
48
50
TIME = t1

FIG. 6A

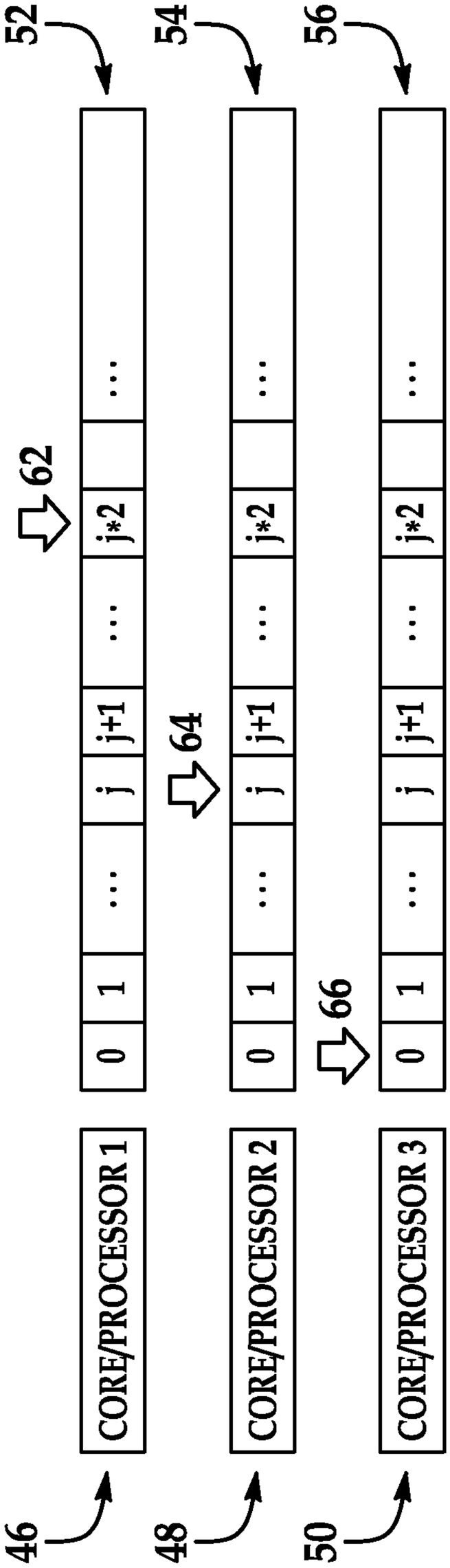

FIG. 6B

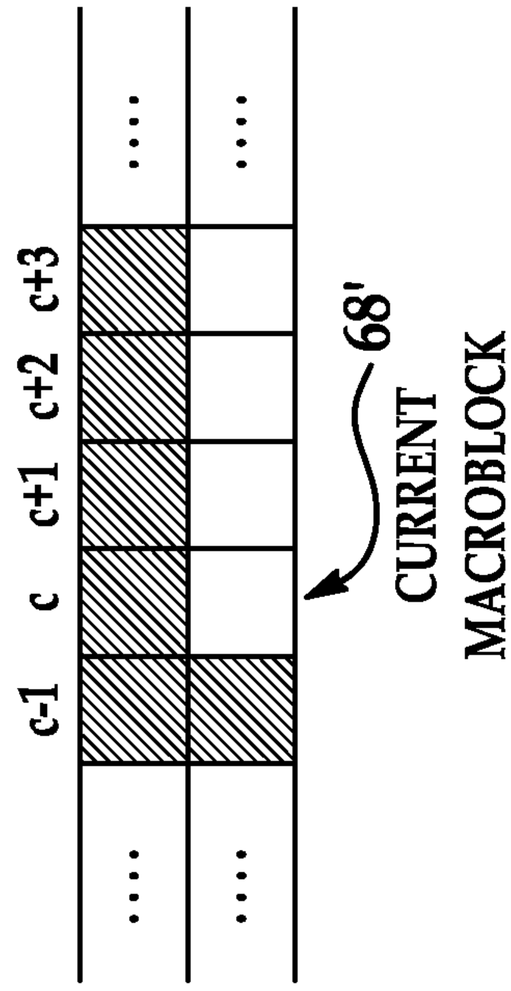

FIG. 7B

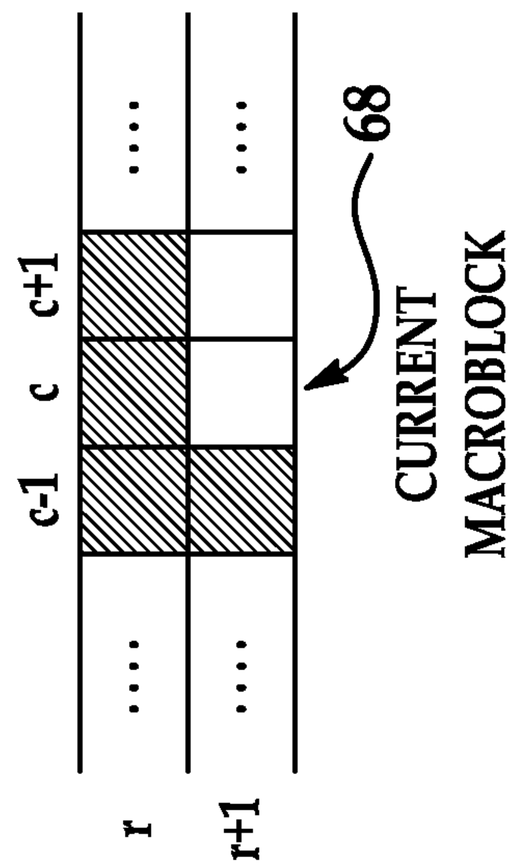

FIG. 7A

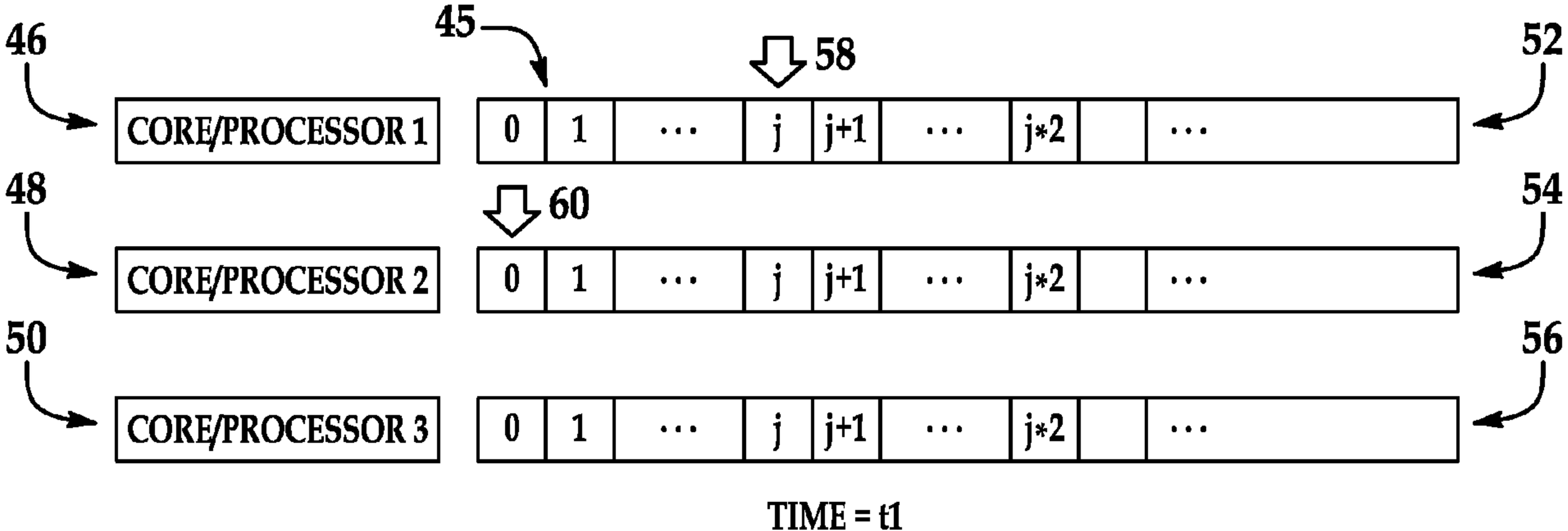
45
46
52
58
CORE/PROCESSOR 1
0
1
...
j
j+1
...
j*2
...
48
54
60
CORE/PROCESSOR 2
0
1
...
j
j+1
...
j*2
...
50
56
CORE/PROCESSOR 3
0
1
...
j
j+1
...
j*2
...
TIME = t1

FIG. 6A