US00RE49727E

(19) **United States**
(12) **Reissued Patent**
Xu et al.

(10) **Patent Number: US RE49,727 E**
(45) **Date of Reissued Patent: Nov. 14, 2023**

(54) **SYSTEM AND METHOD FOR DECODING USING PARALLEL PROCESSING**

(71) Applicant: **GOOGLE LLC**, Mountain View, CA (US)

(72) Inventors: **Yaowu Xu**, Saratoga, CA (US); **Paul Wilkins**, Cambridge (GB); **James Bankoski**, Los Gatos, CA (US)

(21) Appl. No.: **17/200,761**

(22) Filed: **Mar. 12, 2021**

Related U.S. Patent Documents

Reissue of:
(64) Patent No.: **10,230,986**
Issued: **Mar. 12, 2019**
Appl. No.: **15/165,577**
Filed: **May 26, 2016**

U.S. Applications:
(60) Continuation of application No. 13/565,364, filed on Aug. 2, 2012, now Pat. No. 9,357,223, which is a
(Continued)

(51) **Int. Cl.**
H04N 19/61 (2014.01)
H04N 19/91 (2014.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04N 19/61*** (2014.11); ***H04N 19/174*** (2014.11); ***H04N 19/176*** (2014.11);
(Continued)

(58) **Field of Classification Search**
CPC H04N 19/436; H04N 19/176; H04N 19/82; H04N 19/174; H04N 19/91; H04N 19/44; H04N 19/593; H04N 19/51; H04N 19/61
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,825,832	A	7/1974	Frei et al.
4,719,642	A	1/1988	Lucas

(Continued)

FOREIGN PATENT DOCUMENTS

JP	3510433	B2	3/2004
JP	2007-166625	A	6/2007

(Continued)

OTHER PUBLICATIONS

Armando J. Pinho, "Encoding of Image Partitions Using a Standard Technique for Lossless Image Compression," Dep. Electrónica e Telecomunicacoes/ INESC Universidade de Aveiro, Portugal (IEEE 1999), 4 pp.

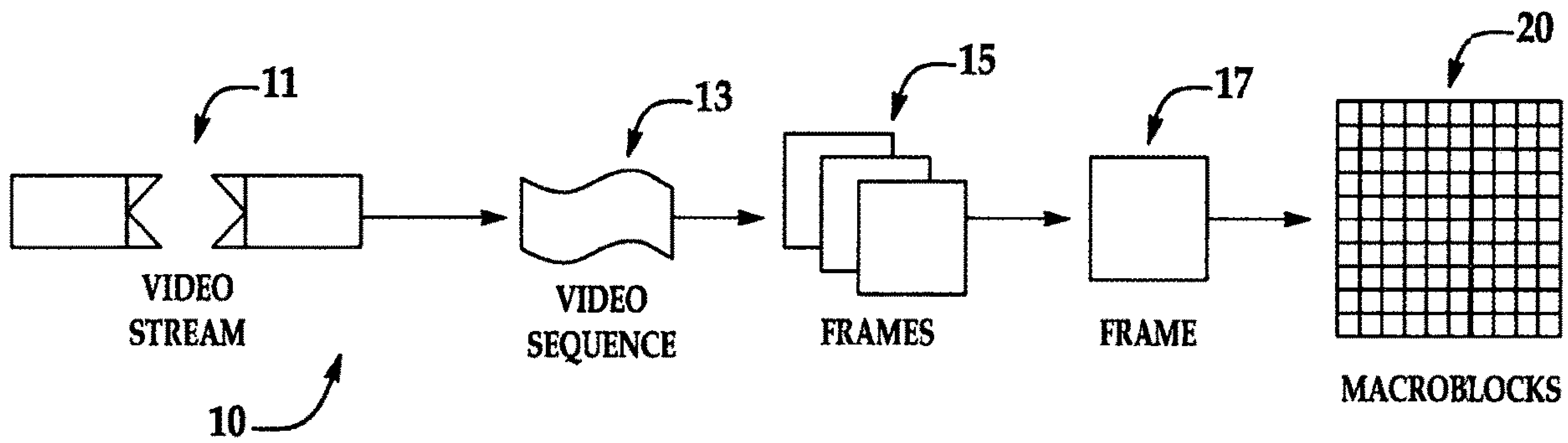
(Continued)

Primary Examiner — Ovidio Escalante
(74) *Attorney, Agent, or Firm* — Young Basile Hanlon & MacFarlane, P.C.

(57) **ABSTRACT**

An apparatus for decoding frames of a compressed video data stream having at least one frame divided into partitions, includes a memory and a processor configured to execute instructions stored in the memory to read partition data information indicative of a partition location for at least one of the partitions, decode a first partition of the partitions that includes a first sequence of blocks, decode a second partition of the partitions that includes a second sequence of blocks identified from the partition data information using decoded information of the first partition.

20 Claims, 5 Drawing Sheets



Related U.S. Application Data

division of application No. 12/329,248, filed on Dec. 5, 2008, now Pat. No. 8,311,111.

(60) Provisional application No. 61/096,223, filed on Sep. 11, 2008.

(51) **Int. Cl.**

H04N 19/82	(2014.01)
H04N 19/17	(2014.01)
H04N 19/593	(2014.01)
H04N 19/44	(2014.01)
H04N 19/174	(2014.01)
H04N 19/176	(2014.01)
H04N 19/436	(2014.01)
H04N 19/51	(2014.01)

(52) **U.S. Cl.**

CPC ***H04N 19/436*** (2014.11); ***H04N 19/44*** (2014.11); ***H04N 19/51*** (2014.11); ***H04N 19/593*** (2014.11); ***H04N 19/82*** (2014.11); ***H04N 19/91*** (2014.11)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,729,127	A	3/1988	Chan et al.	
4,736,446	A	4/1988	Reynolds et al.	
4,797,729	A	1/1989	Tsai	
4,868,764	A	9/1989	Richards	
4,891,748	A	1/1990	Mann	
5,068,724	A	11/1991	Krause et al.	
5,083,214	A	1/1992	Knowles	
5,091,782	A	2/1992	Krause et al.	
5,136,371	A	8/1992	Savatier et al.	
5,136,376	A	8/1992	Yagasaki et al.	
5,164,819	A	11/1992	Music	
5,225,832	A	7/1993	Wang et al.	
5,270,812	A	12/1993	Richards	
5,274,442	A	12/1993	Murakami et al.	
5,313,306	A	5/1994	Kuban et al.	
5,341,440	A	8/1994	Earl et al.	
5,381,145	A	1/1995	Allen et al.	
5,432,870	A	7/1995	Schwartz	
5,452,006	A	9/1995	Auld	
5,561,477	A	10/1996	Polit	
5,576,765	A	11/1996	Cheney et al.	
5,576,767	A	11/1996	Lee et al.	
5,589,945	A	12/1996	Abecassis	
5,604,539	A	2/1997	Ogasawara et al.	
5,646,690	A	7/1997	Yoon	
5,659,539	A	8/1997	Porter et al.	
5,696,869	A	12/1997	Abecassis	
5,734,744	A	3/1998	Wittenstein et al.	
5,737,020	A	4/1998	Hall et al.	
5,748,247	A	5/1998	Hu	
5,774,593	A	6/1998	Zick et al.	
5,793,647	A	8/1998	Hageniers et al.	
5,794,179	A	8/1998	Yamabe	
5,818,530	A *	10/1998	Canfield	H04N 19/136 375/E7.161
5,818,969	A	10/1998	Astle	
5,828,370	A	10/1998	Moeller et al.	
5,835,144	A	11/1998	Matsumura et al.	
5,883,671	A	3/1999	Keng et al.	
5,903,264	A	5/1999	Moeller et al.	
5,929,940	A	7/1999	Jeannin	
5,930,493	A	7/1999	Ottesen et al.	
5,963,203	A	10/1999	Goldberg et al.	
5,999,641	A	12/1999	Miller et al.	
6,014,706	A	1/2000	Cannon et al.	
6,041,145	A	3/2000	Hayashi et al.	
6,061,397	A	5/2000	Ogura	
6,084,908	A	7/2000	Chiang et al.	
6,108,383	A	8/2000	Miller et al.	
6,112,234	A	8/2000	Leiper	
6,115,501	A	9/2000	Chun et al.	
6,119,154	A	9/2000	Weaver et al.	
6,141,381	A	10/2000	Sugiyama	
6,160,846	A	12/2000	Chiang et al.	
6,167,164	A	12/2000	Lee	
6,181,742	B1	1/2001	Rajagopalan et al.	
6,181,822	B1	1/2001	Miller et al.	
6,185,363	B1	2/2001	Dimitrova et al.	
6,188,799	B1	2/2001	Tan et al.	
6,240,135	B1	5/2001	Kim	
6,292,837	B1	9/2001	Miller et al.	
6,327,304	B1	12/2001	Miller et al.	
6,366,704	B1	4/2002	Ribas-Corbera et al.	
6,370,267	B1	4/2002	Miller et al.	
6,400,763	B1	6/2002	Wee	
6,496,537	B1 *	12/2002	Kranawetter	H04N 19/61 375/240.24
6,522,784	B1	2/2003	Zlotnick	
6,529,638	B1	3/2003	Westerman	
6,560,366	B1	5/2003	Wilkins	
6,594,315	B1	7/2003	Schultz et al.	
6,687,303	B1	2/2004	Ishihara	
6,697,061	B1	2/2004	Wee et al.	
6,707,952	B1	3/2004	Tan et al.	
6,765,964	B1	7/2004	Conklin	
6,876,703	B2	4/2005	Ismaeil et al.	
6,934,419	B2	8/2005	Zlotnick	
6,985,526	B2	1/2006	Bottreau et al.	
6,987,866	B2	1/2006	Hu	
7,003,035	B2	2/2006	Tourapis et al.	
7,023,916	B1	4/2006	Pandel et al.	
7,027,654	B1	4/2006	Ameres et al.	
7,170,937	B2	1/2007	Zhou	
7,227,589	B1	6/2007	Yeo et al.	
7,236,524	B2	6/2007	Sun et al.	
7,330,509	B2	2/2008	Lu et al.	
7,499,492	B1	3/2009	Ameres et al.	
7,606,310	B1	10/2009	Ameres et al.	
7,764,739	B2	7/2010	Yamada et al.	
7,813,570	B2	10/2010	Shen et al.	
8,175,161	B1	5/2012	Anisimov	
8,213,518	B1 *	7/2012	Wang	H04N 19/436 348/721
8,265,144	B2 *	9/2012	Christoffersen	H04N 19/91 375/240.1
8,401,084	B2	3/2013	MacInnis	
8,520,734	B1	8/2013	Xu	
8,743,979	B2	6/2014	Lee et al.	
8,767,817	B1	7/2014	Xu et al.	
8,948,267	B1 *	2/2015	Khan	H04N 19/44 375/240.24
9,100,509	B1	8/2015	Jia et al.	
9,100,657	B1	8/2015	Jia et al.	
2002/0012396	A1	1/2002	Pau et al.	
2002/0031184	A1	3/2002	Iwata	
2002/0039386	A1	4/2002	Han et al.	
2002/0168114	A1	11/2002	Valente	
2003/0023982	A1	1/2003	Lee et al.	
2003/0189982	A1 *	10/2003	MacInnis	H04N 19/176 375/240.24
2003/0215018	A1	11/2003	MacInnis et al.	
2003/0219072	A1 *	11/2003	MacInnis	H04N 19/17 375/E7.199
2004/0028142	A1	2/2004	Kim	
2004/0066852	A1	4/2004	MacInnis	
2004/0120400	A1	6/2004	Linzer	
2004/0228410	A1 *	11/2004	Ameres	H04N 19/129 375/E7.113
2004/0240556	A1	12/2004	Winger et al.	
2004/0258151	A1	12/2004	Spampinato	
2005/0050002	A1	3/2005	Slotznick	
2005/0053157	A1 *	3/2005	Lillevold	H04N 19/44 375/240.25
2005/0117655	A1	6/2005	Ju	
2005/0147165	A1	7/2005	Yoo et al.	
2005/0169374	A1	8/2005	Marpe et al.	

(56) **References Cited**

U.S. PATENT DOCUMENTS

2005/0210145 A1 9/2005 Kim et al.
2005/0259747 A1* 11/2005 Schumann H04N 19/42 375/E7.176
2005/0265447 A1 12/2005 Park
2005/0265461 A1 12/2005 Raveendran
2005/0276323 A1 12/2005 Martemyanov et al.
2006/0072674 A1 4/2006 Saha et al.
2006/0098737 A1 5/2006 Sethuraman et al.
2006/0109912 A1 5/2006 Winger et al.
2006/0114985 A1 6/2006 Linzer
2006/0126726 A1 6/2006 Lin et al.
2006/0126740 A1 6/2006 Lin et al.
2006/0150151 A1 7/2006 Dubinsky
2006/0215758 A1 9/2006 Kawashima
2006/0239345 A1 10/2006 Taubman et al.
2006/0256858 A1 11/2006 Chin
2006/0291567 A1 12/2006 Filippini et al.
2007/0025441 A1 2/2007 Ugur et al.
2007/0053443 A1 3/2007 Song
2007/0086528 A1 4/2007 Mauchly et al.
2007/0092006 A1 4/2007 Malayath
2007/0140342 A1 6/2007 Karczewicz et al.
2007/0229704 A1 10/2007 Mohapatra et al.
2007/0286288 A1 12/2007 Smith et al.
2008/0056348 A1 3/2008 Lyashevsky et al.
2008/0152014 A1 6/2008 Schreier et al.
2008/0159407 A1 7/2008 Yang et al.
2008/0198270 A1 8/2008 Hobbs et al.
2008/0198920 A1 8/2008 Yang et al.
2008/0212678 A1 9/2008 Booth et al.
2008/0215317 A1* 9/2008 Fejzo G10L 19/025 704/E19.044
2008/0240254 A1* 10/2008 Au G06T 1/20 375/240.24
2008/0267295 A1* 10/2008 Sung H04N 7/012 375/240.24
2008/0298469 A1* 12/2008 Liu H04N 19/186 375/E7.14
2008/0317364 A1* 12/2008 Gou H04N 19/176 382/233
2009/0002379 A1* 1/2009 Baeza H04N 19/51 345/522
2009/0003447 A1 1/2009 Christoffersen et al.
2009/0052529 A1* 2/2009 Kim H04N 13/15 375/E7.126
2009/0080534 A1 3/2009 Sekiguchi et al.
2009/0168893 A1* 7/2009 Schlanger H04N 19/44 375/E7.2
2009/0225845 A1 9/2009 Veremeev et al.
2009/0238277 A1 9/2009 Meehan
2009/0245349 A1* 10/2009 Zhao H04N 19/70 375/240.03
2009/0249178 A1 10/2009 Ambrosino et al.
2010/0061455 A1 3/2010 Xu et al.
2010/0158109 A1* 6/2010 Dahlby H04N 21/8543 375/240.03
2010/0177826 A1 7/2010 Bhaumik et al.
2010/0183076 A1 7/2010 Yoon
2010/0189179 A1 7/2010 Gu et al.
2010/0215263 A1 8/2010 Imanaka
2010/0239181 A1 9/2010 Lee et al.
2010/0246665 A1 9/2010 Brederson et al.
2010/0316132 A1* 12/2010 MacInnis H04N 19/00 375/240.24
2011/0261884 A1 10/2011 Rubinstein et al.
2012/0014451 A1 1/2012 Lee et al.
2012/0128069 A1 5/2012 Sato
2012/0147958 A1 6/2012 Ronca et al.
2012/0213448 A1 8/2012 Malmborg et al.
2012/0294376 A1 11/2012 Tanaka et al.
2012/0307892 A1 12/2012 Xu et al.
2013/0034150 A1 2/2013 Sadafale
2013/0083161 A1 4/2013 Hsu et al.
2013/0259137 A1 10/2013 Kuusela
2015/0043645 A1 2/2015 Ventela
2015/0326888 A1 11/2015 Jia et al.
2017/0188024 A9* 6/2017 Wang H04N 19/105

FOREIGN PATENT DOCUMENTS

WO 2008/0204 70 A1 2/2008
WO 2008/036237 A2 3/2008
WO 2010/063184 A1 6/2010

OTHER PUBLICATIONS

B. Vasudev & N. Merhav, "DCT Mode Conversions for Field/Frame Coded MPEG Video", IEEE 2d Workshop on Multimedia Signal Processing 605-610 (Dec. 1998).
Bankoski et al. "Technical Overview of VP8, an Open Source Video Codec for the Web". Dated Jul. 11, 2011.
Bankoski et al., "VP8 Data Format and Decoding Guide", Independent Submission RFC 6389, Nov. 2011, 305 pp.
Bankoski et al., "VP8 Data Format and Decoding Guide draft-bankoski-vp8-bitstream-02", Network Working Group, Internet-Draft, May 18, 2011, 288 pp.
Fore June "An Introduction to Digital Video Data Compression in Java", Chapter 12: DPCM video Codec, CreateSpace, Jan. 22, 2011.
International Search Report and Written Opinion Issued in co-pending PCT International Application No. PCT/US2013/034581 dated Jun. 11, 2013.
Li E Q et al., "Implementation of H.264 encoder on general-purpose processors with hyper-threading technology", Proceedings of SPIE, pp. 384-395, vol. 5308, No. 1, Jan. 20, 2004.
"Introduction to Video Coding Part 1: Transform Coding", Mozilla, Mar. 2012, 171 pp.
On2 Technologies Inc., White Paper TrueMotion VP7 Video Codec, Jan. 10, 2005, 13 pages, Document Version: 1.0, Clifton Park, New York.
On2 Technologies, Inc., White Paper On2's TrueMotion VP7 Video Codec, Jul. 11, 2008, pp. 7 pages, Document Version:1.0, Clifton Park, New York.
"Overview VP7 Data Format and Decoder", Version 1.5, On2 Technologies, Inc., Mar. 28, 2005, 65 pp.
Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services—Coding of moving video, Amendment 2: New profiles for professional applications, International Telecommunication Union, Apr. 2007, 75 pp.
Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services—Coding of moving video, Advanced video coding for generic audiovisual services, Amendment 1: Support of additional colour spaces and removal of the High 4:4:4 Profile, International Telecommunication Union, Jun. 2006, 16 pp.
Series H: Audiovisual and Multimedia Systems, Infrastructure of audiovisual services—Coding of moving video, Advanced video coding for generic audiovisual services, Version 3, International Telecommunication Union, Mar. 2005, 343 pp.
Sharp, "Entropy slices for parallel entropy decoding", ITU-T SG16 Meeting, Apr. 22, 2008-Feb. 5, 2008, Geneva.
Sze, "Massively Parallel CABAC", VCEG meeting, Jan. 7, 2009, London and MPEG meeting, Aug. 7, 2009, Geneva.
Chen, T, Y.H. Ng; Lossless Color Image Compression Technique for Multimedia Applications; IBM Technical Disclosure Bulletin; vol. 37 No. 10, Oct. 1994.
Tasdizen, et al.; "A High Performance Reconfigurable Motion Estimation Hardware Architecture", Design, Automation & Test in Europe Conference & Exhibition, Apr. 20, 2009, IEEE, Piscataway, NJ, US pp. 882-885.
Vos, Luc De and Stegherr, Michael; "Parameterizable VLSI Architectures for the Full-Search Block-Matching Algorithm", IEEE Transactions on Circuits and Systems, vol. 36, No. 10, Oct. 1989 NY US pp. 1309-1316.
VP6 Bitstream and Decoder Specification, Version 1.03, (On2 Technologies, Inc.), Dated Oct. 29, 2007.

(56) **References Cited**

OTHER PUBLICATIONS

Wiegand et al, "Overview of the H 264/AVC Video Coding Standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, No. 7, pp. 568, 569, Jul. 1, 2003.
Yao Wang, "Motion Estimation for Video coding", EE4414: Motion Estimation basics, 2005.
Youn et al., "Motion Vector Refinement for high-performance Transcoding" IEEE Transactions on Multimedia, vol. 1, No. 1, Mar. 1999.

* cited by examiner

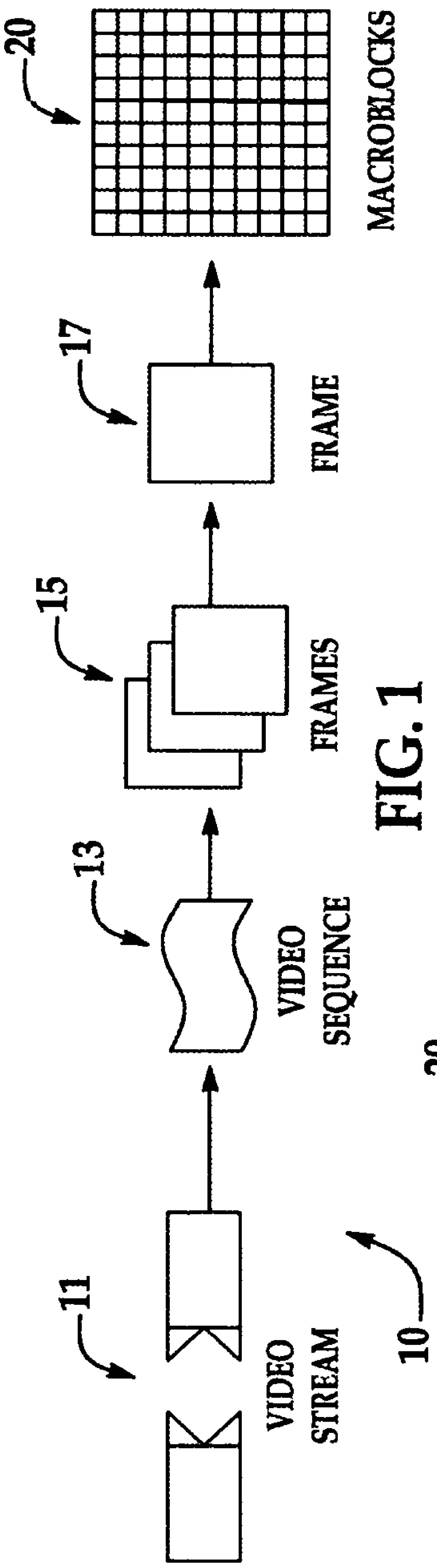

11
VIDEO STREAM
13
VIDEO SEQUENCE
15
FRAMES
17
FRAME
20
MACROBLOCKS
10

FIG. 1

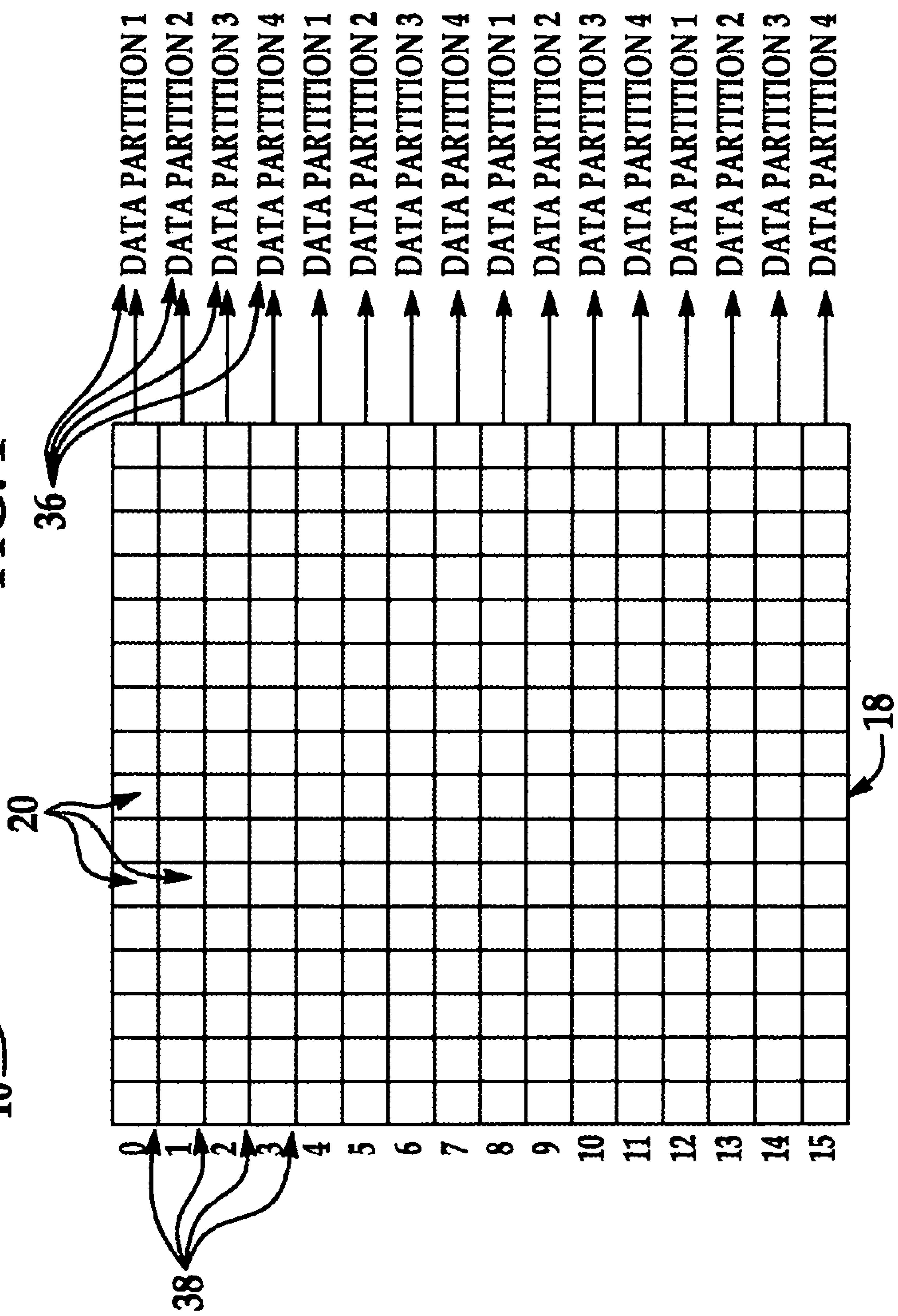

36
DATA PARTITION 1
DATA PARTITION 2
DATA PARTITION 3
DATA PARTITION 4
DATA PARTITION 1
DATA PARTITION 2
DATA PARTITION 3
DATA PARTITION 4
DATA PARTITION 1
DATA PARTITION 2
DATA PARTITION 3
DATA PARTITION 4
DATA PARTITION 1
DATA PARTITION 2
DATA PARTITION 3
DATA PARTITION 4
20
18
0
1
2
3
4
5
6
7
8
9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

FIG. 4

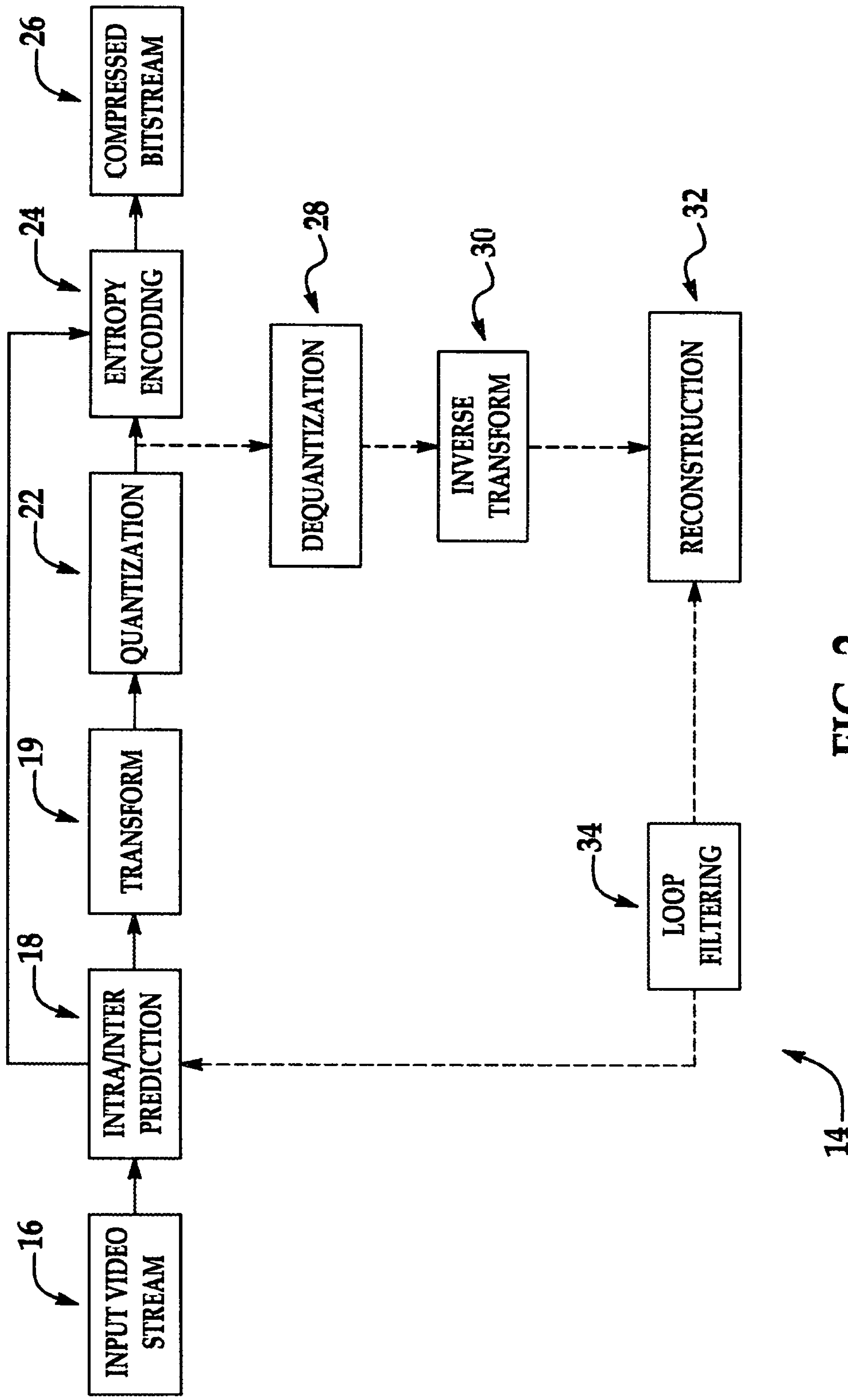

INPUT VIDEO STREAM
16
INTRA/INTER PREDICTION
18
TRANSFORM
19
QUANTIZATION
22
ENTROPY ENCODING
24
COMPRESSED BITSTREAM
26
DEQUANTIZATION
28
INVERSE TRANSFORM
30
RECONSTRUCTION
32
LOOP FILTERING
34
14

FIG. 2

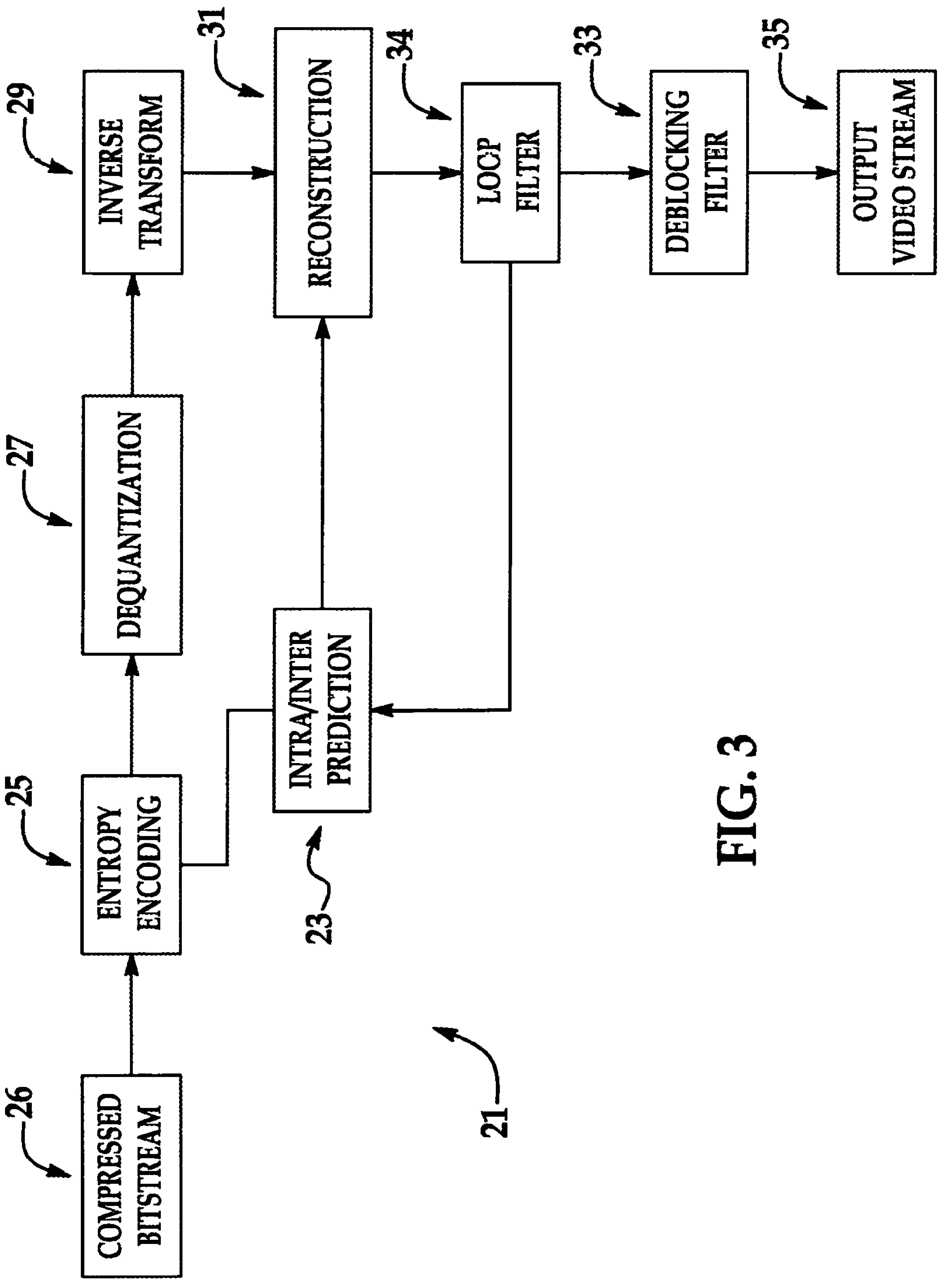

COMPRESSED BITSTREAM
26
ENTROPY ENCODING
25
DEQUANTIZATION
27
INVERSE TRANSFORM
29
RECONSTRUCTION
31
INTRA/INTER PREDICTION
23
LOCP FILTER
34
DEBLOCKING FILTER
33
OUTPUT VIDEO STREAM
35
21

FIG. 3

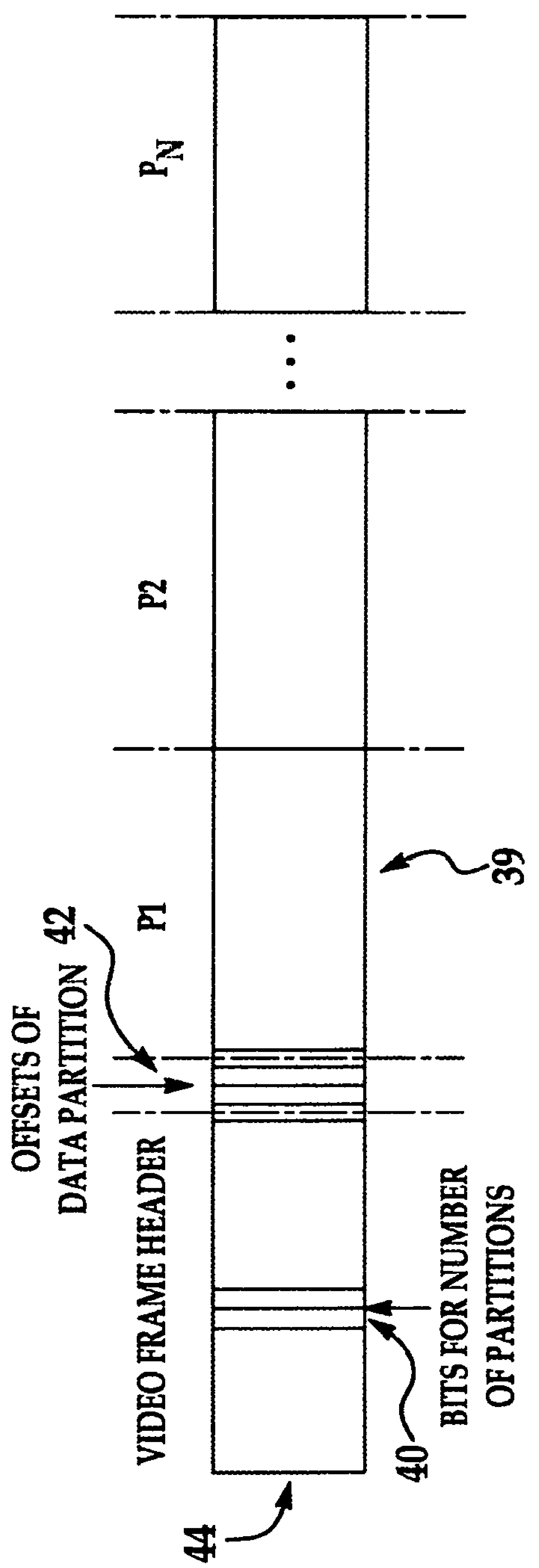

OFFSETS OF DATA PARTITION
42
VIDEO FRAME HEADER
BITS FOR NUMBER OF PARTITIONS
40
44
P1
P2
P_N
39

FIG. 5

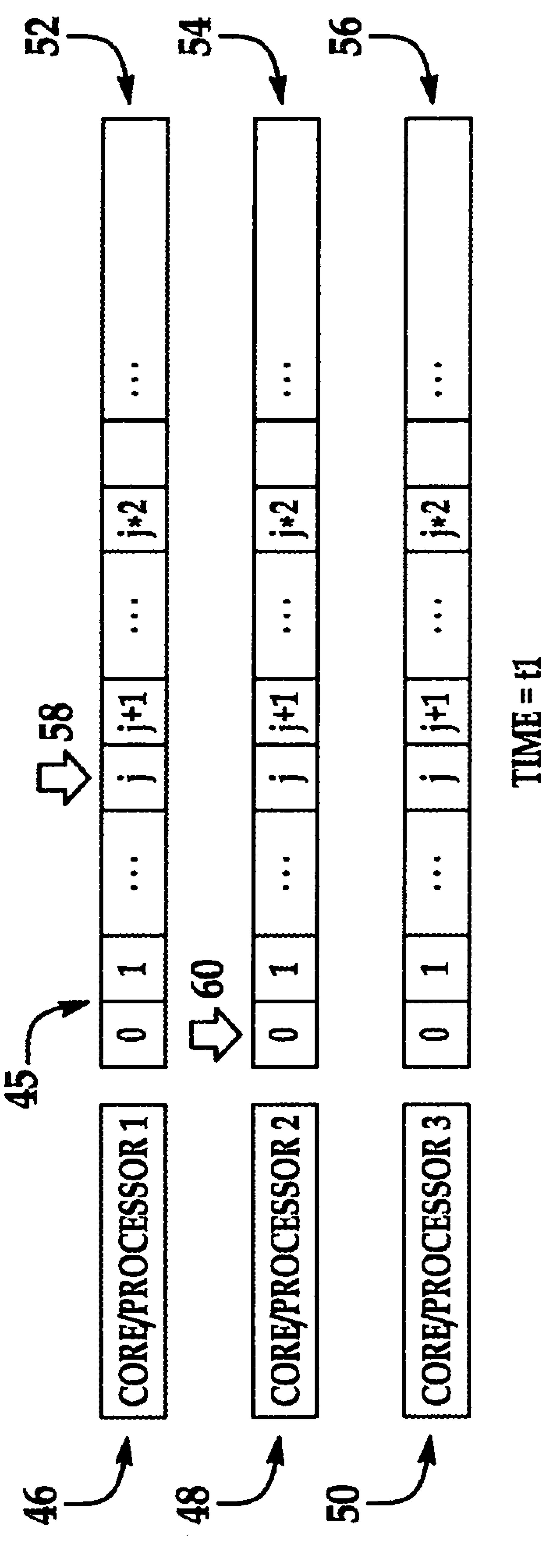

CORE/PROCESSOR 1
CORE/PROCESSOR 2
CORE/PROCESSOR 3
46
48
50
45
52
54
56
58
60
0
1
...
j
j+1
j+2
TIME = t1

FIG. 6A

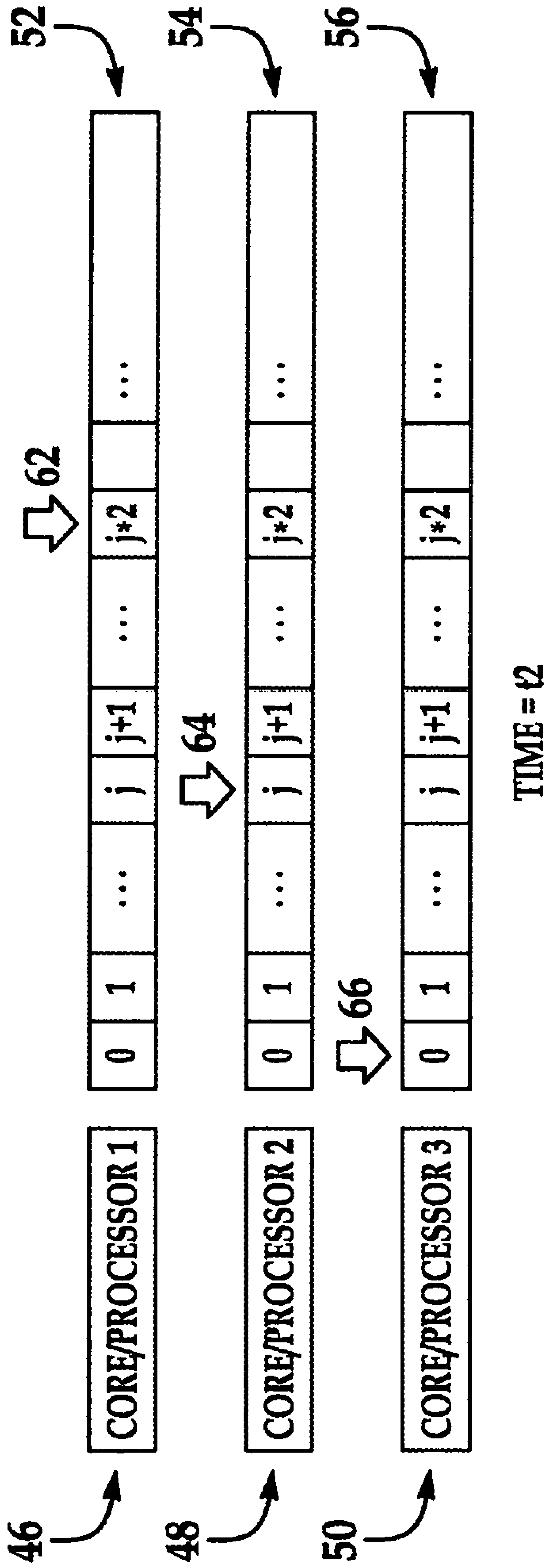


FIG. 6B

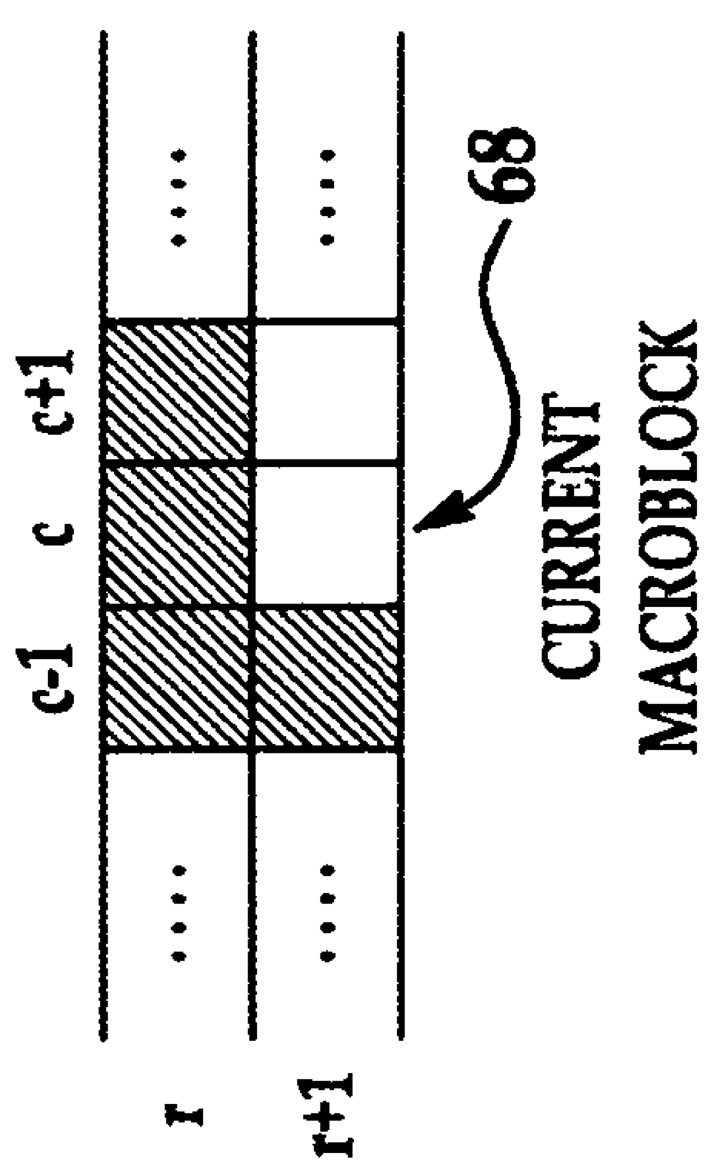


FIG. 7A

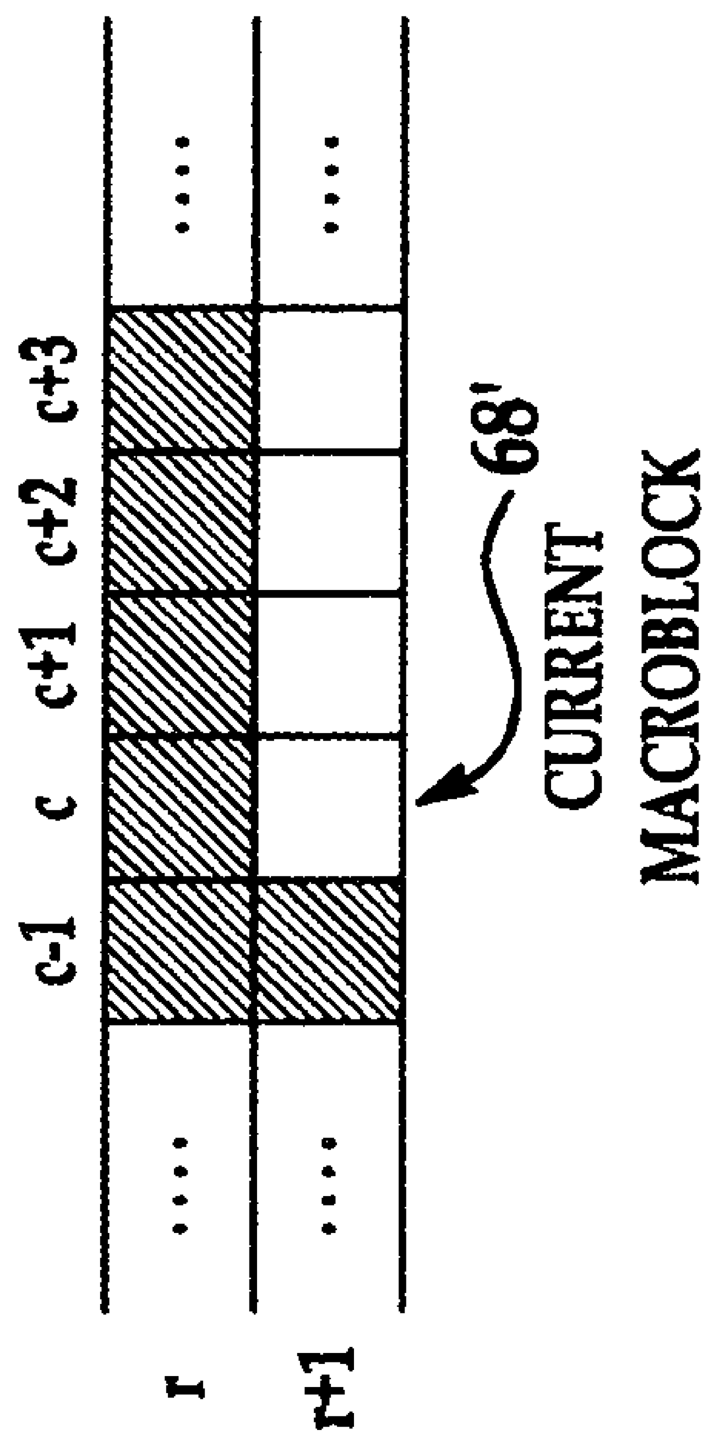


FIG. 7B

SYSTEM AND METHOD FOR DECODING USING PARALLEL PROCESSING

Matter enclosed in heavy brackets [ ] appears in the original patent but forms no part of this reissue specification; matter printed in italics indicates the additions made by reissue; a claim printed with strikethrough indicates that the claim was canceled, disclaimed, or held invalid by a prior post-patent action or proceeding.

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. nonprovisional patent application Ser. No. 13/565,364, filed Aug. 2, 2012, now U.S. Pat. No. 9,357,223, which is a divisional of U.S. nonprovisional patent application Ser. No. 12/329,248, filed Dec. 5, 2008, which claims priority to U.S. provisional patent application No. 61/096,223, filed Sep. 11, 2008, which are incorporated herein in entirety by reference.

TECHNICAL FIELD

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

BACKGROUND

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.

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, N.Y., 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).

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

SUMMARY

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.

BRIEF DESCRIPTION OF THE DRAWINGS

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

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

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

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

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

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**.

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

FIG. **7**A 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**.

FIG. **7**B 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

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 16×16 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 16×8 pixel groups or 8×16 pixel groups. Macroblocks **20** are further subdivided into blocks. A block, for example, can be a 4×4 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.

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.

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**.

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.

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**.

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 arc 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 he applied to the reconstructed macroblock to reduce distortion.

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**.

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 he outputted to output video stream **35**.

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.

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.

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.

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 arc also available and arc 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**.

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.

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**.

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.

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**.

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
0	0	One partition
0	1	Two partitions
1	0	Four partitions
1	1	Eight partitions

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.

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.

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.

Once the data has been compressed into bit stream **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**.

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.

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.

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.

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.

With reference to FIGS. **6**A and **6**B, 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. **6**A and **6**B 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.

As depicted in FIGS. **6**A and **6**B, as an example, three threads **46**, **48** and **50** arc 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.

Each of FIGS. **6**A and **6**B 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. **6**A and **6**B 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.

Referring to FIG. **6**A, at time t**1**, 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**.

Referring to FIG. **6**B, at time t**2**, 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**.

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.

Utilizing the parallel processing staging and synchronization mechanism illustrated in FIGS. **6**A and **6**B 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. **6**A and **6**B. 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.

Referring again to FIGS. **6**A and **6**B, 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. **7**A and **7**B illustrate two alternatives for the size of offset j.

Referring to FIG. **7**A, 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. **7**A, 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.

Referring to FIG. **7**B, 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. **7**A, 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.

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.

While the invention has been described in connection with certain embodiments, it is to be understood that the invention is not to he 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.

What is claimed is:

1. An apparatus for decoding frames of a compressed video data stream, including at least one frame divided into partitions, the apparatus comprising:
- a memory; and
- a processor configured to execute instructions stored in the memory to:
  - read, from the compressed video data stream, partition data information indicative of a partition location with respect to the compressed video data stream for at least one of the partitions;
  - decode, from the compressed video data stream, a first partition of the partitions that includes a first sequence of blocks; and
  - decode, from the compressed video data stream, a second partition of the partitions that includes a second sequence of blocks identified based on the partition location indicated by the partition data information, using decoded information of the first partition;
  - wherein the first partition and the second partition have each been individually compressed; and
  - output or store a decoded frame including the first sequence of blocks and the second sequence of blocks.

2. The apparatus of claim **1**, wherein the decoding includes decoding lossless coded information.

[**3**. The apparatus of claim **1**, wherein the processor is further configured to identify one row of blocks in the frame having a boundary between the first partition and the second partition.]

[**4**. The apparatus of claim **1**, wherein the first partition includes contiguous blocks in at least a portion of a first row and at least a portion of a first subsequent row.]

[**5**. The apparatus of claim **4**, wherein the second partition includes contiguous blocks in at least a portion of a second row and at least a portion of a second subsequent row.]

[**6**. The apparatus of claim **1**, wherein the first partition comprises blocks of two or more contiguous rows of blocks.]

7. The apparatus of claim **1**, wherein the processor is further configured to decode the second sequence of blocks using context information contained in the first sequence of blocks.

[**8**. The apparatus of claim **7**, wherein the processor is further configured to perform at least one of intra-frame prediction or inter-frame prediction on the second partition.]

9. The apparatus of claim **7**, wherein the processor decodes the second partition with an offset of a specified number of blocks such that at least a portion of decoded context information of the first sequence of blocks is available as context data for decoding at least one block in the second partition.

10. The apparatus of claim **9**, wherein the offset is determined based upon size of the context.

11. The apparatus of claim **7**, wherein the first sequence of blocks in the first partition and the second sequence of blocks in the second partition are derived from corresponding rows of blocks in two successive frames of the video data, and wherein the processor is further configured to:
- decode at least one block in the second partition using information from a block previously decoded.

12. The apparatus of claim **1**, wherein the decoding includes context-based arithmetic coding.

13. A non-transitory computer-readable storage medium having stored thereon an encoded bitstream, including at least one frame divided into individually compressed partitions, wherein the encoded bitstream is configured for decoding by operations comprising:
- *reading, from the encoded bitstream, partition data information indicative of a partition location with respect to the encoded bitstream for at least one of the partitions;*
- *decoding, from the encoded bitstream, a first partition of the partitions that includes a first sequence of blocks;*
- *decoding, from the encoded bitstream, a second partition of the partitions that includes a second sequence of blocks identified based on the partition location indicated by the partition data information, using decoded information of the first partition; and*
- *outputting or storing a decoded frame including the first sequence of blocks and the second sequence of blocks.*

14. The non-transitory computer-readable storage medium of claim 13, wherein the decoding includes decoding losslessly coded information.

15. The non-transitory computer-readable storage medium of claim 13, wherein the decoding includes decoding the second sequence of blocks using context information contained in the first sequence of blocks.

16. The non-transitory computer-readable storage medium of claim 15, wherein the decoding includes decoding the second partition with an offset of a specified number of blocks such that at least a portion of decoded context *information of the first sequence of blocks is available as context data for decoding at least one block in the second partition.*

17. The non-transitory computer-readable storage medium of claim 16, wherein the offset is determined based upon size of the context.

18. The non-transitory computer-readable storage medium of claim 15, wherein the first sequence of blocks in the first partition and the second sequence of blocks in the second partition are derived from corresponding rows of blocks in two successive frames of the video data, and wherein the decoding includes:

decoding at least one block in the second partition using information from a block previously decoded.

19. The non-transitory computer-readable storage medium of claim 13, wherein the decoding includes context-based arithmetic coding.

20. An apparatus for encoding frames of a video, including at least one frame divided into partitions, the apparatus comprising:

a memory; and

a processor configured to execute instructions stored in the memory to:

encode, into a compressed video data stream, a first partition of the partitions that includes a first sequence of blocks;

encode, into the compressed video data stream, at a partition location with respect to the compressed video data stream, a second partition of the partitions that includes a second sequence of blocks encoded using information of the first partition;

include, in the compressed video data stream, partition data information indicative of the partition location with respect to the compressed video data stream for at least one of the partitions; and

output or store the compressed video data stream.

21. The apparatus of claim 20, wherein the processor is configured to execute the instructions to use context information from the first sequence of blocks to encode the second sequence of blocks.

22. The apparatus of claim 21, wherein the processor is configured to execute the instructions to encode the second partition with an offset of a specified number of blocks such that at least a portion of context information from the first sequence of blocks is available as context data for encoding at least one block in the second partition.

23. The apparatus of claim 22, wherein the offset is determined based upon size of the context.

24. The apparatus of claim 22, wherein the first sequence of blocks in the first partition and the second sequence of blocks in the second partition are derived from corresponding rows of blocks in two successive frames of the video data, and the processor is configured to execute the instructions to:

use information from a block previously decoded to encode at least one block in the second partition.

25. The apparatus of claim 24, wherein to encode the processor is configured to execute the instructions to use context-based arithmetic coding.

* * * * *