PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : **H04N 7/12**	**A1**	**(11) International Publication Number:** **WO 96/36178** **(43) International Publication Date:** 14 November 1996 (14.11.96)

(21) International Application Number: PCT/US96/06510

(22) International Filing Date: 8 May 1996 (08.05.96)

(30) Priority Data:

08/439,085	10 May 1995 (10.05.95)	US
08/440,464	10 May 1995 (10.05.95)	US

(71) Applicant: THE 3DO COMPANY [US/US]; 600 Galveston Drive, Redwood City, CA 94063 (US).

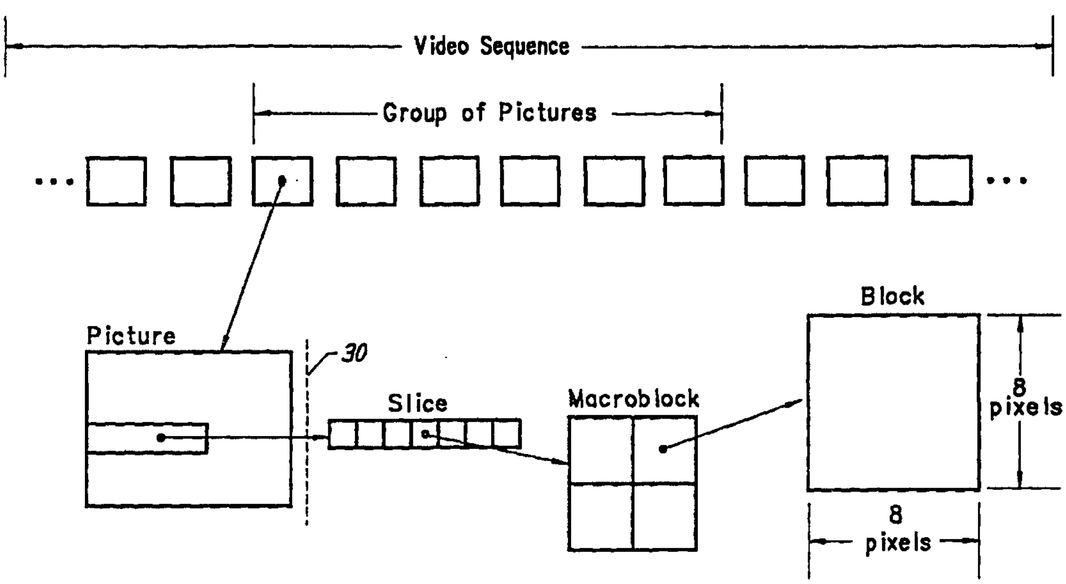
(72) Inventors: WASSERMAN, Steve, C.; 10455 North Blaney Avenue, Cupertino, CA 95014 (US). BALDWIN, James, Armand; 85 Paul Avenue, Mountain View, CA 94041 (US). MITSUOKA, George; 928 Wright Avenue #701, Mountain View, CA 94043 (US).

(74) Agents: WOLFELD, Warren, S. et al.; Fliesler, Dubb, Meyer and Lovejoy, Suite 400, Four Embacardero Center, San Francisco, CA 94111-4156 (US).

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

Published
With international search report.

(54) Title: MULTIPLE SEQUENCE MPEG DECODER AND PROCESS FOR CONTROLLING SAME



(57) Abstract

The process comprises the steps of: (a) extracting macroblock information from MPEG encoded image data (262); (b) extracting a series of parameters from the MPEG encoded image data (264); (c) determining quantization factors from the encoded image data (265); (d) configuring the configurable image decoding apparatus (266, 268, 270, 280), including (i) configuring a means for parsing the macroblock data into motion vectors and image data with the series of parameters with the parameters for decoding the encoded data; (ii) configuring a means for performing inverse quantization with the quantization co-efficients; (e) determining a decoding order of the extracted macroblock information to be decoded (270); (f) providing said extracted macroblock information to the parsing means in the decoding order (274); (g) combining decoded image data with motion vectors extracted by the parsing means (290); and (h) storing the combined data in the system memory (292).

FOR THE PURPOSES OF INFORMATION ONLY

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

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

MULTIPLE SEQUENCE MPEG DECODER AND PROCESS FOR CONTROLLING SAME

CROSS-REFERENCE TO RELATED APPLICATIONS

United States Patent Application Serial No. ________________, entitled CONFIGURABLE VIDEO DISPLAY SYSTEM HAVING LIST-BASED CONTROL MECHANISM FOR TIME-DEFERRED INSTRUCTING OF 3D RENDERING ENGINE THAT ALSO RESPONDS TO SUPERVISORY IMMEDIATE COMMANDS, inventors: Adrian Sfarti, Nicholas Baker, Robert Laker, and Adam Malamy, filed May 10, 1995.

United States Patent Application Serial No. ________________ entitled CONFIGURABLE VIDEO DISPLAY SYSTEM HAVING LIST-BASED CONTROL MECHANISM FOR BY-THE-LINE AND BY-THE-PIXEL MODIFICATION OF DISPLAYED FRAMES AND METHOD OF OPERATING SAME, inventors Richard W. Thaik, Robert Joseph Mical, Stephen Harland Landrum, and Steve C. Wasserman filed May 10, 1995.

PCT Patent Application Serial No. PCT/US92/09342, entitled RESOLUTION ENHANCEMENT FOR VIDEO DISPLAY USING MULTI-LINE INTERPOLATION, by inventors Mical et al., filed November 2, 1992, and also to U.S. Patent Application Serial No. 07/970,287, bearing the same title, same inventors and also filed November 2, 1992;

PCT Patent Application Serial No. PCT/US92/09349, entitled AUDIO/VIDEO COMPUTER ARCHITECTURE, by inventors Mical et al., filed November 2, 1992, and also to U.S. Patent Application Serial No. 07/970,308, bearing the same title, same inventors and also filed

November 2, 1992;

PCT Patent Application Serial No. PCT/US92/09350, entitled METHOD FOR CONTROLLING A SPRYTE RENDERING PROCESSOR, by inventors Mical et al., filed November 2, 1992, and also to U.S. Patent Application Serial No. 07/970,278, bearing the same title, same inventors and also filed November 2, 1992;

PCT Patent Application Serial No. PCT/US92/09462, entitled SPRYTE RENDERING SYSTEM WITH IMPROVED CORNER CALCULATING ENGINE AND IMPROVED POLYGON-PAINT ENGINE, by inventors Needle et al., filed November 2, 1992, and also to U.S. Patent Application Serial No. 07/970,289, bearing the same title, same inventors and also filed November 2, 1992;

PCT Patent Application Serial No. PCT/US92/09460, entitled METHOD AND APPARATUS FOR UPDATING A CLUT DURING HORIZONTAL BLANKING, by inventors Mical et al., filed November 2, 1992, and also to U.S. Patent Application Serial No. 07/969,994, bearing the same title, same inventors and also filed November 2, 1992;

PCT Patent Application Serial No. PCT/US92/09467, entitled IMPROVED METHOD AND APPARATUS FOR PROCESSING IMAGE DATA, by inventors Mical et al., filed November 2, 1992, and also to U.S. Patent Application Serial No. 07/970,083, bearing the same title, same inventors and also filed November 2, 1992;

PCT Patent Application Serial No. PCT/US94/12521, entitled DISPLAY LIST MANAGEMENT MECHANISM FOR REAL-TIME CONTROL OF BY-THE-LINE MODIFIABLE VIDEO DISPLAY SYSTEM, by inventors Robert Joseph Mical et al., filed November 1, 1994, and also to U.S. Patent Application Serial No. 08/146,505, bearing the same title, same inventors and filed November 1, 1993; and

U.S. Patent Application Serial No. 08/311,192 entitled REAL TIME DECOMPRESSION AND POST-DECOMPRESS

MANIPULATION OF COMPRESSED FULL MOTION VIDEO, by inventors Steve C. Wasserman et al., filed September 23, 1994.

The related patent applications are all commonly assigned with the present application and are all incorporated herein by reference in their entirety.

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to a system for decoding motion image data, and particularly to a process for controlling decoding data hardware.

Description of the Related Art

To address the growing need for a common format of representing compressed video on various digital storage media, the ISO/IEC standard 11172-2 has been adopted as one standard for compression of such image data. The standard is more commonly referred to as the Moving Picture Expert's Group (MPEG) standard or "MPEG-1". A second standard, ISO/IEC standard 13818, is a more robust version of video decoding and is more commonly known as MPEG-2. MPEG-1 is a subset of MPEG-2. Both standards have several basic compression algorithms in common, including motion compensation, application of the discrete cosine transform (DCT), quantization, variable length coding and run-length encoding.

In an MPEG-1 system, data is provided in a stream that is generally made up of two layers: a system layer contains timing and other information needed to multiplex audio and video and user data streams and to synchronize audio and video during playback; and a compression layer includes the user data, compressed audio and video streams. A system de-multiplexer extracts the timing information from the MPEG stream and sends it to other system components. The system de-multiplexer also de-multiplexes the video and audio streams and sends each to an appropriate decoder.

A video decoder in accordance with the MPEG standard will decompress the video stream. Each video stream is arranged in a data hierarchy with each lower level of the hierarchy comprising a component of a higher level of the hierarchy. The video stream data hierarchy comprises: the video sequence; the group of pictures; a picture; a slice; a macroblock; and a block. This hierarchy is represented graphically in Figure 1A. The video sequence is the highest level of the video bitstream. The video sequence always consists of a sequence sender, one or more groups of pictures, and an end of sequence code. The video sequence is another term for the video stream. The sequence may contain any number of instances of the "group of pictures" layer, as well as information such as picture size, aspect ratio, frame rate, bit rate, input buffer size, quantization tables, a "constrained parameters" flag, information about buffer sizes, and optional user data.

The group of pictures layer consists of one or more pictures intended to allow random access into a sequence. The group of pictures encompasses a series of pictures that are to be displayed contiguously. The group of pictures may possibly depend on reference frames from a previous group of pictures. A so-called "closed" group of pictures has no such pictures while an "open" group of pictures contains references to a previous group of pictures. A group of pictures will begin with a header that contains a time code and optional user data, followed by any number of pictures.

The picture is the primary coding unit of a video sequence. The picture generally consists of three rectangular matrices representing luminance (Y) and two chrominance (CbCr) values. The Y matrix has an even number of rows and columns. The Cb and Cr matrices are one-half the size of the Y matrix in each direction (horizontal and vertical). Thus, for every four Y samples, there is one Cr sample and one Cb sample. The most commonly used size for movie encoding are 352 x 240 pixels at 29.97 or 24 frames per second (NTSC) and 352 x 288 at 25 frames per second (PAL).

The picture contains decoded information for one frame of video. Each picture may be one of four possible types. An "intra" picture or "I-picture" is coded using only information present in the picture itself. "I" pictures provide random access points into the compressed video data. "I" pictures use only quantization, run length and VLC coding and therefore provide moderate compression. A predicted or "P-picture" is coded with respect to the previous I- or P-picture. This technique is called forward prediction. Predicted pictures provide more compression and serve as a reference for B-pictures (described below) and future P-pictures. (I-pictures may also serve as a reference for B-pictures.) P-pictures use motion compensation to provide more compression than is possible with I-pictures. "Bidirectional" or B-pictures are pictures that use both a past and future picture as a reference. Bidirectional pictures provide the most compression, and do not propagate errors because they are never used as a reference. The final type of picture is a "DC-coded" picture or "D-picture", which is coded using only information from itself and intended for use in fast-forward searching.

Below the picture layer of the video bitstream is the slice layer. The slice layer contains series of

16-pixel x 16 line sections of luminance (Y) components and the corresponding 8-pixel by 8 line sections of the chrominance (CrCb) components. A macroblock thus contains four Y-blocks, one Cb block and one Cr block, as noted above.

Each data block is an 8x8 set of values of a luminance or chrominance component. As discussed below, a data block may also be comprised of motion vectors and error terms.

In general, MPEG compression of image data involves a translation of pixel data from the red/green/blue (RGB) colorspace to the Y-CbCr color space, an application of the discrete cosine transform (DCT) to remove data redundancy, quantization of the DCT coefficients using weighting functions optimized for the human visual system, and encoding the quantized AC coefficient by first using zero run-length coding, followed by compression using entropy encoding, such as Huffman coding.

The combination of DCT and quantization results in many of the frequency coefficients being zero, especially the coefficients for high spatial frequencies. To take maximum advantage of this, the coefficients are organized in a zig-zag order to produce long runs of zeroes. This is represented in Figure 1B. The coefficients are then converted to a series of run amplitude pairs, each pair indicating a number of zero coefficients and the amplitude of a non-zero coefficient.

Some blocks of pixels need to be coded more accurately than others. For example, blocks with smooth intensity gradients need accurate coding to avoid visible block boundaries. The MPEG algorithm allows the amount of quantization to be modified for each 16x16 block of pixels, and this mechanism can also be used to provide smooth adaptation to a particular bit rate. The MPEG video bitstream includes the capacity for carrying quantization tables, to allow for modification of the degree of quantization.

In addition, motion compensation is a technique used for enhancing the compression of P- and B-pictures by eliminating temporal redundancy. Motion compensation typically improves compression by a factor of 2-5 compared to intra-picture coding. Motion compensation algorithms work at the macroblock level. When a macroblock is compressed by motion compensation, the compressed file contains: motion vectors -- the spatial difference between the reference picture(s) and the macroblock being coded; and error terms -- content differences between the reference and the macroblock being coded. When a macroblock in a P- or B-picture cannot be well predicted by motion compensation, it is coded in the same way a macroblock in an I-picture is coded, by using transform coding techniques. Macroblocks in a B-picture can be coded using either a previous or future reference picture as a reference so that four codings are possible.

A timing mechanism ensures synchronization between audio and video. In the MPEG-1 standard, a system clock reference and a presentation time stamp are utilized by the decoder. Additional standards are added by the MPEG-2 standard. System clock references and presentation time stamps in MPEG-1 are 33 bit values, which can represent any clock cycle in a 24-hour period.

A system clock reference (SCR) is a reflection of the encoder system clock. SCRs used by an audio and a video decoder must have approximately the same value. SCRs are inserted into the MPEG stream at least as often 0.7 seconds by the MPEG encoder, and are extracted by the system decoder and sent to the audio and video decoders, which update their internal clocks using the SCR value by the system decoder.

Presentation time stamps are samples of the encoder system clock that are associated with some video or audio presentation units. The presentation unit is a decoded video picture or a decoded audio time sequence. The encoder inserts presentation time stamps into the MPEG stream at least as often as every 0.7 seconds. The PTS represents the time at which the video picture is to be displayed or the starting playback time for the audio sequence.

Model MPEG decoders are set forth in the ISO/IEC 1172-2 standard. In appendix D thereof, the general decoder model includes an input buffer and a picture decoder. The input buffer stores data at a fixed rate and at regular intervals, set by the picture rate, the picture decoder instantaneously removes all the bits from the next picture from the input buffer.

In general, decoding a video sequence for forward playback involves first decoding the sequence header including the sequence parameters. These parameters will include the horizontal and vertical resolutions and aspect ratio, the bit rate, and the quantization tables or matrices. Next the decoder will decode the group of pictures' header, including the "closed GOP and broken LINK information," and take appropriate action. It will decode the first picture header in the group of pictures and read the VBV_delay_field. If playback begins from a random point in the bitstream, the decoder should discard all the bits until it finds a sequence start code, a group of pictures start code, or a picture start code which introduces an I-picture. The slices and macroblocks in the picture are decoded and written into a display buffer, and perhaps into another buffer. The decoded pictures may be post-processed and displayed in the order defined by the temporal reference at the picture rate defined in the sequence header.

The decoding sequence of pictures may not be the same as the display sequence. Thus, some mechanism of ordering the display sequence, and storing decoded image data, is required.

MPEG decoders can be implemented in a series of hardware and software configurations. For example, in an IBM PC-type computer, the system's CPU, internal data bus, and data storage unit can be programmed to perform all buffering and decoding functions. Software decoders capable of performing stream decoding include Xingit! from Xing Technology Corp., Arroyo Grande, California. Hardware processors such as the COM4100 family of multimedia processors available from C-Cube Microsystems provide hardware/software implemented processing of MPEG-encoded data. In addition, the C-Cube CL550 and CL560 JPEG (Joint Photographic Expert's Group) processors, which perform the JPEG baseline sequential process (a process which is essentially incorporated into the MPEG compression algorithm), include capabilities to allow for user-defined Huffman tables and quantization tables to be programmed into hardware component blocks which perform Huffman coding and decoding and quantization on 8x8 blocks of JPEG picture data.

In general, MPEG decoding streams consist of around 9,900 macroblocks per second (plus audio). In many multimedia applications, it would be beneficial to provide decoding potential in excess of the 9,900 macroblock per second rate to allow interactive applications, which will require different MPEG streams to be decoded simultaneously (or in a "multi-threaded" capacity), to be implemented. For example, in multimedia applications where different portions of the display screen will need to be reacting to actions of the user, and such applications are based on the video data which is stored in an MPEG format, multi-threaded decoding capability would be essential.

SUMMARY OF THE INVENTION

These and other objects of the invention are provided in a process for decoding MPEG encoded image data stored in a system memory utilizing a configurable image decoding apparatus. The process comprises the steps of: (a) extracting macroblock information from said MPEG encoded image data, the macroblocks containing image data and motion compensation data; (b) extracting a series of parameters from the MPEG encoded image data for decoding the MPEG encoded data; (c) determining quantization factors from the encoded image data; (d) configuring the configurable image decoding apparatus, including (i) configuring a means for parsing the macroblock data into motion vectors and image data with the series of parameters with the parameters for decoding the encoded data; (ii) configuring a means for performing inverse quantization with the quantization co-efficients; (e) determining a decoding order of the extracted macroblock information to be decoded; (f) providing said extracted macroblock information to the parsing means in the decoding order; (g) combining decoded image data with motion vectors extracted by the parsing means; and (h) storing the combined data in the system memory.

In a further aspect, the invention comprises an apparatus for processing encoded image data wherein image data is used to produce an image composed of a matrix of pixels, the apparatus being included in a host system, the host system including a system memory and a processor. The apparatus includes a first input port for receiving a first encoded image-defining signal, where said first encoded image defining signal is divisible into at least one pixel defining component, where each pixel defining component may comprise motion vector data or pixel value data. A first input/output port for receiving and outputting a handshaking signal is also included. A second input/output port is provided for outputting motion vector data and receiving reference data defining a reference frame relative to the motion vector data. An output port for outputting decoded image data is provided. The system further includes control instructions, operatively instructing the central processing unit to provide encoded image information into the first input port, operatively instructing decoded data from the output port to be written to system memory, instructing reference information to be input to the second input/output port and instructing decoded data and reference information to be directed to an video output formatter.

In yet another aspect, the invention comprises a process for decoding coded image data in a host computer, the host computer including a central processing unit (CPU) and system memory, the computer including a decoding processor, comprising the steps of:(a)directing the CPU to perform the steps of parsing the system memory into a series of buffers, including a display buffer, a reference buffer and a strip buffer; reading the coded image data and ascertaining context information regarding information in the data to be decoded; parsing the coded data into the slice level information and providing the information to the decoding processor; (b) directing the decoding processor to perform the steps of distributing coded motion vector information blocks and image data information blocks; decoding the image data blocks into quantized coefficient blocks; performing an inverse quantization on said quantized coefficient blocks to form pixel value blocks; converting the pixel value blocks to pixel coefficients; calculating the inverse discrete cosine transform of the pixel coefficients to produce pixel display values; decoding the motion vector blocks into pixel motion vectors; and adding the pixel motion vectors and pixel display values; and (c) directing the CPU to perform the steps of: retrieving decoded picture data from the decoding hardware; storing said decoded picture data in said system memory; directing the reference buffer data to the decoding hardware; and storing formatted decoded picture data in a display buffer in said system memory.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described with respect to the particular embodiments thereof. Other objects, features, and advantages of the invention will become apparent with reference to the specification and drawings in which:

Figure 1 is a block diagram of the MPEG coding structure and the breakdown of the distribution of functions in the system of the present invention.

Figure 2 is a block overview diagram of the system hardware and MPEG decoding unit hardware in accordance with the present invention.

Figure 3 is a block diagram of the video bitstream DMA controller shown in Figure 2.

Figure 4 is a block diagram of the parsing unit shown in Figure 2.

Figure 5A is a block diagram of the

interconnections of the zig-zagging unit, inverse discrete cosine transform unit, and motion compensation units shown in Figure 2.

Figure 5B is a block diagram of the de-zig-zag unit shown in Figure 5A.

Figure 6A is a block diagram of the inverse discrete cosine transform (IDCT) unit.

Figure 6B is a flow diagram of the control logic process utilized in the IDCT unit shown in Figure 6A.

Figure 6C is a representation of the calculations performed by the IDCT circuit of Figure 6A.

Figure 7 is a logic diagram of the motion vector processor of the present system.

Figure 8 is a block diagram of the macroblock configuration utilized in accordance with the present invention.

Figure 9 is a table of the byte offsets for inserting the values from the macroblocks into the system memory.

Figure 10A is a block diagram of the data pipe for the motion compensation unit of the present system.

Figure 10B is an exemplary luminance and chrominance predictable macroblock.

Figure 11A is a block diagram of the video output display functions in accordance with the system of the present invention.

Figure 11B is a representation of the raster conversion of chroma data to YUV444 format.

Figure 11C is a flowchart of the colorspace conversion matrix utilized in the CSC/dither circuit.

Figure 12 is a process flow chart of a process for decoding a single MPEG data stream in accordance with the present invention.

Figure 13 is a block diagram of the data flow between a host system memory and the MPEG decoding hardware in accordance with the present invention.

Figure 14 is a flow chart indicating a multiple decode sequence for the method of decoding MPEG video data in accordance with the present invention.

Figure 15 is a table showing the inputs and outputs of each block of data during a typical video sequence.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The invention provides a flexible MPEG decoding system which is implemented in both hardware and software. A key aspect of the hardware and software system of the present invention is the division of labor between decoding functions performed by the software and decoding functions performed by the hardware. This allows the MPEG decoding system of the present invention to be highly flexible, and with the proper instructions, to decode multiple MPEG streams, in effect, simultaneously. Hence, multi-threaded moving video, still images, and varied image sizes can be decoded by the system of the present invention. The hardware architecture allows all these situations to coexist with the software controlling distribution of image data, and sequencing of data to the hardware decoding functions.

Figure 1 shows the breakdown of the division of labor between the hardware and software decoding functions of the system of the present invention. As shown in Figure 1, a typical video sequence is broken down into a group of pictures, comprised of an I, P, and B-type pictures, which is comprised of slices of macroblocks, each macroblock containing an image block of 8x8 pixels and, possibly, encoded motion vector data. Line 30 represents the division of labor between the software portion of the system and the hardware portion of the system. Thus, in decoding a video sequence, the software portion of the system will search the video sequence, determine the group of pictures ordering, and sequence the ordering of the pictures to be decoded to the hardware portion of the system. The hardware component of the system decodes image and motion vector information at the slice, macroblock, and block level in accordance with the MPEG-1 decoding standard and the following description.

SYSTEM OVERVIEW

Figure 2 shows a general overview of the hardware components of a decoding system in accordance with the present invention.

The hardware architecture of the present invention as shown in Figure 2 may reside in a host system, or be incorporated as part of an application specific integrated circuit (ASIC) 150 which is itself incorporated into a host system. For example, the host system will include a system memory 110, a central processing unit (CPU) 102, an address and data bus 104, and a system memory controller 106. MPEG unit hardware control registers 112, which are accessible to the CPU and decoding hardware, may be provided and include system status and configuration information. The control registers 112 are configurable by the CPU 102 for use by the decoding system of the present invention. Such control registers are defined herein in conjunction with their function relative to given components. System memory 110 generally comprises synchronous dynamic random access memory (SDRAM). As shown in Figure 2, MPEG decoding hardware 200 may be included on ASIC 150. The host system or ASIC 150 may include other hardware components for performing multimedia application specific processing such as, for example, digital signal processing, advanced video processing, and interfacing with other components of the host system. CPU 102 may comprise a PowerPC class microprocessor manufactured by IBM Microelectronics and Motorola.

System memory 110 will contain MPEG-encoded video data which must be decoded by the MPEG decoding system in a coded data buffer. System memory 110 is configured to include reference buffers, display (or "output") buffers, and a strip buffer which are accessible by decoding hardware 200 and the system CPU 102.

As shown in Figure 2, a memory controller interface and arbiter 160 handles all communication between system memory 110 and the MPEG decoding hardware 200. Memory controller interface 160 will handle requests from a video bitstream DMA controller 170 which issues requests to read bitstream data into a buffer contained in the DMA controller 170; requests from a motion compensation unit 175 to read data into the motion compensation unit 175; requests from a video output DMA controller to write to the video output DMA controller 180; and read and write requests from a video output formatter 185. Arbitration between all the MPEG requestors is handled by memory controller 160. Memory controller 160 can handle simultaneous, independent requests to several memory groups as discussed herein.

Video bitstream DMA controller 170 supplies coded data to the MPEG decoding unit 200. As explained in further detail below, a FIFO unit in DMA controller 170 contains data waiting to be transferred to a parsing unit 210, which is present in the MPEG decoding hardware 200. As space becomes available in the FIFO, video bitstream DMA controller 170 initiates memory requests to the memory arbiter 160 to refill the FIFO.

MPEG decompression hardware 200 performs the video decompression algorithm on the slice layer and below, including parsing of the video bitstream, entropy (Huffman or, more generally, variable length decoding (VLD)), inverse quantization, the inverse discrete cosine transform, and motion compensation. Three interfaces are provided to the MPEG decompression hardware 200: the coded data interface 202, the motion compensator interface 204, and the decoded data interface 206. Decoded data interface 202 includes a data provision interface 202a, and a communication protocol interface 202b. Communication protocol interface 202b utilizes a request/acknowledge protocol to communicate with the video bitstream DMA controller 170. When decompressing predicted macroblocks, MPEG core unit 200, and specifically motion vector processor 212, supplies the pixel location of the prediction data in advance of the time the data is actually needed on line 204. Motion compensation unit 175 may then fetch the appropriate data from system memory 110. Decoded data comes out of port 206 in a block order, but without the zig-zag configuration. Five logical blocks are shown as comprising the MPEG core decoding hardware 200: the parsing unit 210, a motion vector processor 212, an inverse quantization unit 214, a "de-zig-zag unit" 216 and an inverse discrete cosine transform unit 218.

Motion compensation unit 175 converts pixel addresses of reference macroblocks supplied by the MPEG core hardware 200 to physical memory addresses in system memory 110 and initiates memory transactions with system memory 110 to acquire necessary data for motion compensation via the memory controller 160. The motion compensation unit will perform half-pixel interpolation, if necessary, and store the prediction value in a local register until the corresponding pel is available at the output 206 of core hardware 200. At that time, the prediction data and the output of the core hardware 200 (specifically IDCT 218) are combined by the motion compensator unit 175. The combined data may be stored in a strip buffer by video output DMA controller 180. There is sufficient storage in the motion compensation unit 175 to ensure that no memory transaction has to be repeated during the duration of a macroblock.

Video output DMA controller 180 transfers decompressed data from the motion compensation unit 175 and the MPEG core hardware 200 to system memory 110. A buffer in the output DMA controller 180 temporarily stores decompressed pixels on their way to system memory 110. After the output DMA controller 180 accumulates enough data for a bus transaction, the output DMA controller calculates an address in system memory 110 where the data should be written and initiates the appropriate memory transaction via the memory controller interface 160. The DMA controller passes entire frames to the output formatter 185.

Video output formatter 185 converts images from the native MPEG format to one of several formats utilized by the host system. As discussed in further detail below, the output formatter contains a color space converter, dither circuit, and quantizer.

If the luminance/chrominance data is in a 4:4:4 format, it may also be directly passed to the output. The color space converter transforms the MPEG data to the RGB (red/green/blue) domain for use in three-dimensional rendering. The quantizer optionally converts 24 bit pixels to 16 bit pixels.

System Control Registers

As noted above, control registers 112 have a default configuration and may be configured by software instructions to CPU 102. Specific registers configured for functions of individual hardware elements are described in the following sections pertaining to such elements. Registers 112 are configured for system configurations and system interrupts as follows:

Table 1 -
MPEGUnit Configuration Register Bit Descriptions

Name	Bit(s)	Type	Description
(reserved)	0:18	x	reserved
vofRdEnable	19	RW	output formatter read enable
vofWrEnable	20	RW	output formatter write enable
vofReset_n	21	RW	output formatter reset
vodEnable	22	RW	Video Output DMA Enable
vodReset_n	23	RW	Video Output DMA Reset
motEnable	24	RW	Motion Estimator Enable
motReset_n	25	RW	Motion Estimator Reset
mvdReset_n	26	RW	Decompressor Reset
parserStep	27	RW	Parser Step Control
parserEnable	28	RW	Parser Enable
parserReset_n	29	RW	Parser Reset
vbdEnable	30	RW	Video Bitstream DMA Enable
vbdReset_n	31	RW	Video Bitstream DMAReset

Table 2 - Interrupt Enable

Name	Bit(s)	Type	Description
(reserved)	0:24	x	reserved
Strip Buffer Error	25	RW	error in output dma with strip buffer enabled
Everything Done	26	RW	output formatter, parser done
Output Formatter	27	RW	formatting complete
Output DMA	28	RW	DMA complete
Bitstream Error	29	RW	parser bitstream error
End Of Picture	30	RW	from parser
Video Bitstream DMA	31	RW	buffer exhausted

Table 3 - Interrupt Status

Name	Bit(s)	Type	Description
(reserved)	0:24	x	reserved
Strip Buffer Error	25	RW	error in output dma with strip buffer enabled
Everything Done	26	RW	output formatter, parser done
Output Formatter	27	R	formatting complete
Output DMA	28	R	DMA complete
Bitstream Error	29	R	parser bitstream error
End Of Picture	30	R	from parser
Video Bitstream DMA	31	R	buffer exhausted

<u>Video BitStream DMA Controller</u>

Figure 3 is a hardware block diagram of the video bitstream DMA controller block 170 shown in Figure 2. As shown in Figure 3, the bitstream DMA controller 170 includes a 16 x 32 RAM 220, a multiplexer 222, a FIFO controller 224, and an address generator 226.

Video bitstream DMA controller 170 reads coded data from system memory 110 and places it into FIFO register 220. Generally, the parser unit 210 takes the data from the FIFO at a highly variable rate depending on the characteristics of the coded video bitstream.

Coded data buffers (see Figure 13) in system memory 110 may begin on any byte boundary and may be any number of bytes long. DMA controller 170 has its own queue of two address and length registers that tell it where in system memory 110 the coded data resides. Each time video bitstream DMA controller 170 exhausts a coded data buffer in main memory 110, it returns an interrupt to the CPU and begins reading coded data from the next valid address in the DMA controller queue of addresses. The queue of two buffer addresses is provided in a Current Address Register (Table 4) and a Next Address Register (Table 6) in DMA controller 170 and reduces the urgency of the end of buffer interrupt of DMA controller 170. Each buffer address consists of a (byte-aligned) memory address (Tables 4, 6) and a length in bytes (Tables 5, 7). To place a buffer address in the queue, the CPU must first write a 23-bit physical memory address to the Next Address Register (Table 6) and then a 16-bit length to the Next Length Register (Table 7) in the DMA controller 170. When a data buffer is exhausted, the DMA controller 170 optionally generates an interrupt, and moves on to the next buffer specified in the Next Address Register. After an end-of-picture interrupt is generated by the parsing unit 210, registers in the DMA controller 170 may be examined to determine where the first start code following the end-of-picture occurred.

The hardware registers for implementing the aforementioned description are as follows:

Table 4 - Bitstream Unit DMA Current Address Register

Name	Bit(s)	Type	Description
(reserved)	0:6	x	reserved
Current Address	7:31	R	next read address

Table 5 - Video Bitstream DMA Current Length

Name	Bit(s)	Type	Description
(reserved)	0:14	x	reserved
Current Length	15:31	R	bytes remaining in current buffer

Table 6 - Video Bitstream DMA Next Address

Name	Bit(s)	Type	Description
(reserved)	0:6	x	reserved
Next Address	7:31	RW	next buffer address

Table 7 - Video Bitstream DMA Next Length

Name	Bit(s)	Type	Description
(reserved)	0:14	x	reserved
Next Length	15:31	RW	next buffer length

Table 8 - Video Bitstream DMA Config/Status

Name	Bit(s)	Type	Description
(reserved)	0:14	x	reserved
vbd snoop enable	15	RW	enable snooping on vbd reads
(reserved)	16:26	x	reserved
Buffer Byte Count	27:31	R	number of bytes buffered

FIFO controller 224 monitors the fullness of the 16x32 RAM 220 containing coded data on its way to parsing unit 220. Each time the data request from the parser unit 210 becomes valid, FIFO controller 224 moves on to the next 16 bits to be transferred. The memory address queue is provided in address generator 226 and is incremented every four bytes. When RAM 220 becomes half empty or more, FIFO controller 224 makes a request to the address generator 226. Address generator 226 initiates a memory transfer via memory controller 160. When the data becomes available, the address generator inserts a write signal to FIFO controller 224.

A soft reset and enable for bitstream DMA controller 170 are provided in the MPEG unit configuration register. A zero in the vbdReset bit location disables operation of the DMA controller 170; for normal operation, a "1" is written to this bit. If during normal operation, the bit transfers from a "1" to a zero, the DMA address queue is flushed and the remaining contents of the bitstream FIFO are immediately invalidated. Setting this bit to "0" is equivalent to a soft reset of the DMA controller 170. The vbdEnable bit is a bitstream enable bit, which, when disabled, pauses DMA controller 170.

The DMA controller next address queue includes a bitstream unit address queue control bit (Next Address) which, when written to, places a new value in the next location of the address queue. Note that the address does not become a valid entry in the queue until the corresponding write to the length register (Next Length) occurs. The address is 25 bits long and the 25 bits uniquely specify a byte location in system memory 110. Any byte alignment is allowed. Registers implementing the address queue may be individually read via a direct memory mapping for diagnostic purposes.

The bitstream unit current length queue (Video Bitstream DMA Current Length) corresponds to the address queue (Video Bitstream DMA Current Address). Each entry in the length specifies the number of bytes to be read from the segment of the bitstream beginning at the address contained in the corresponding entry of the address queue. Entries in the length queue are 16 bits long, allowing each buffer segment to be up to 64 Kbytes. Writing the length queue actually causes a new entry to be placed in the queue; a write to the address queue does not cause an update. Therefore, the address should be written before the length when adding a new segment to the length queue. If there are no valid addresses in the address queue, the address/length immediately becomes the current address for the DMA controller 170. If there is one valid address, the address length becomes the current value only after the buffer presently being read is exhausted.

The Bitstream Unit DMA current status register of the DMA controller allows the CPU to determine where in memory the DMA controller unit is currently reading from. This is particularly useful at the end of a picture in the case of a bitstream error.

MPEG CORE HARDWARE

The MPEG core hardware 200 is defined as the parsing unit 210, inverse quantization unit 214, motion vector processor 212, de-zig-zag unit 216, and inverse discrete cosine transform unit 218.

In general, parsing unit 210 turns the MPEG bitstream (slice layer and below) into a series of motion vectors and run/level pairs that are passed on to the motion vector processor 212 and inverse quantization unit, respectively. The inverse quantization unit decodes the run/level pairs (using Q-tables decoded from the bitstream by the system CPU 102), reconstructs the frequency domain discrete cosine transform samples as per the MPEG-1 specification, and passes them to the de-zig-zag unit 216. The de-zig-zag unit contains memory to "de-zig-zag" the data recovered from the MPEG stream. The inverse discrete cosine transform unit transforms the frequency domain data into the spatial domain. Motion vectors from the parser unit 210 are transferred to the motion vector processor 212. Motion compensation unit 175 combines the prediction generated by the motion vector processor with the output from the inverse discrete cosine transform unit 218 and passes the results on to the video output DMA controller 180.

Parsing Unit

Figure 4 shows a block diagram of the parsing unit 210 utilized in the MPEG core decompression hardware 200. Parser unit 210 includes a bit shifter 230, parser state machine 232 and registers 234. The parsing unit 210 must be programmed with the variables picture_Coding_Type, forward_R_Size and backward_R_Size as decoded from the bitstream by the CPU under the instructions provided in the system of the present invention. It should be recognized that these variables need not be present in a bitstream format, but can be decoded from coded data in a different data structure more suitably used for interactive formats. The following values reside in the parser configuration register set forth below:

Table 9 - Parser Configuration

Name	Bit(s)	Type	Description
(reserved)	0:2	x	reserved
full_pel_backward_ vector	3	RW	from picture header
(reserved)	4	x	reserved
backward_r_size	5:7	RW	from picture header
(reserved)	8:10	x	reserved
full_pel_forward_ vector	11	RW	from picture header
(reserved)	12	x	reserved
forward_r_size	13:15	RW	from picture header
(reserved)	16	x	reserved
priorityMode	17:19	RW	priority request control
(reserved)	20:28	x	reserved
picture_coding_type	29:31	RW	from picture header

The parser configuration register contains the reset and enable bits for the parser. The parser configuration register contains parameters that must be decoded from the picture layer of the bitstream. This register is only written while the parser is in reset mode.

The image size register, produced below, allows the parser to determine the relative addresses of the prediction of a predictive coded (p-picture) macroblock. It should only be modified while the parser is in reset. MPEG Specification 11172-2 specifies the proper decoding of the variables mp_height and mp_width.

Table 10 - Image Size

Name	Bit(s)	Type	Description
(reserved)	0:15	x	reserved
mb_height	16:23	RW	image width in macroblocks
mb_width	24:31	RW	image height in macroblocks

The parser status register contains information for the CPU from parser 210. It is utilized for debugging and retrieving details about bitstream errors by the CPU from parser 210.

Table 11 - Parser Status 0

Name	Bit(s)	Type	Description
(reserved)	0	x	reserved
mb_row	1:7	R	current macroblock row
(reserved)	8	x	reserved
bitstreamError	9	R	1 = bitstream error detected
error state	10:15	R	state where error occurred
eval bits	15:31	R	current bit shifter output

Table 12 - Parser Status 1

Name	Bit(s)	Type	Description
(reserved)	0:2	x	reserved
blockNumber	3:5	R	current block number
macroblock_type	6:10	R	as in MPEG spec
numBits Valid	11:15	R	# of valid eval bits, from left
lastStartCode	16:23	R	last start code parsed
(reserved)	24	x	reserved
mb_column	25:31	R	current macroblock column

Parser state machine 232 includes control logic 236 and a state register 238. Bit shifter 230 also includes a register 231. The bit shifter 230 shifts bitstream data up to 12 bit increments as defined by the control logic 236. As shown in Table 13, the control logic determines the amount of the shift necessary dependent upon the data being examined. The parser handles elements in the bitstream as data units, such as the quantizer information, macroblock stuffing, address increment, etc. Table 14 outlines the time necessary for each element to be handled by the parser. The amount of the bit shift allowed bit shifter 230 is directly dependent upon the data unit being handled. When information from the bitstream DMA unit is provided to the parser, the control logic will search the data for a start code in 12 bit increment shifts, 12 bits/clock cycle.

Control logic 236 determines the amount of the bit shift depending on the nature of the incoming data. The shift value is written to the bit shift register 231. For example, a start code in a video sequence comprises 23 consecutive zeros. The bit shifter will require 2 cycles, at 12 bits per cycle, to determine a start code. Table 13 outlines the number of cycles (and the MPEG 1 specification the size and type of data) which the parser requires to examine incoming data. The parser configuration registers 234 contain information derived from the stream header and allow the parser to determine the nature of the incoming data. Once the data type is determined, data can be shifted to the control logic which divides out the RLL and motion vector data to the IQ unit and the motion vector processor. The state register 238 in the parser state machine 232 can track the data type expected by the control logic by reading the bit shift register

231.

The following table details the number of cycles of the system timing clock expended by parser unit 210 in decoding various parts of the bitstream:

Table 13 - Parser Performance

Decoding Process	Performance
Look for slice header	12 bits/cycle
Quantizer Scale	1 tick
Slice Extra Information	1 cycle if none present, 1 cycle per code otherwise
Macroblock Stuffing	1 cycle if none present, 1 cycle per code otherwise
Macroblock Address Increment	1 cycle for codes less than 4 bits, 2 cycles for longer codes, plus 1 cycle for each escape
Macroblock Type	1 cycle
Motion Vectors	1 cycle for each vector not present; otherwise, 1 cycle for the motion code if less than 4 bits, 2 cycles otherwise; plus 1 cycle if R present -- times 2 to account for both H and V
Macroblock Pattern	1 cycle, whether present or not; 2 cycles for codes longer than 4 bits
Block	There is a 1 cycle overhead at the beginning of each block while the parser decides what to do next
DC Term in I-coded Macroblocks	3 cycles
Each R/L Code (including first in non-I-coded Macroblocks)	1 cycle if code less than 4 bits, 2 cycles otherwise
Each R/L Escape	2 cycles
End of Macroblock	1 cycle

Parser unit 212 directly detects certain error conditions. When the parser encounters an error, it generates an interrupt and freezes in its current state. The CPU can then examine the current state bits in the parser status register, and determine the type of error that caused the interrupt. The following table enumerates all of the detected error conditions and the state in which they are detected:

Table 14 - Parser State Table

Symbolic State Name	State Number (decimal)	Description of Error
HANDLE_START_CODE	9	Invalid slice start code (>mb_height)
QUANTIZER_SCALE	10	Quantizer_scale set to zero
MACROBLOCK_ADDRESS_ INCREMENT	13	Invalid VLC for macroblock address increment
MACROBLOCK_ADDRESS	14	Invalid macroblock_address_ increment after a slice start code (>mb_width) -- or -- decoded macroblock_address_increment causes decoding to go beyond the end of the picture (as defined by mb_height and mb_width)
MACROBLOCK_TYPE	15	Invalid macroblock type VLC
QUANTIZER_SCALE_MB	16	Quantizer_scale set to zero
MOTION_CODE	18	Invalid motion VLC
MACROBLOCK_PATTERN	20	Invalid coded_block_pattern VLC
DCT_DC_SIZE_ LUMINANCE	22	Invalid VLC for dct_dc_size_luminance
DCT_DC_SIZE_ CHROMINANCE	23	Invalid VLC for dct_dc_size_chrominance
DO_RUN	27	More than 64 samples decoded for one block
DECODE_RLP_STAGE1	32	Invalid run/level VLC -- or -- more than 64 samples decoded for one block

DECODE_RLP_STAGE2	33	Invalid run/level VLC -- or -- more than 64 samples decoded for one block
DECODE_RLP_ESCAPE_ LEVEL	35	More than 64 samples decoded for one block
DECODE_RLP_ESCAPE_ LONG	36	More than 64 samples decoded for one block
END_OF_SLICE	30	More than 12 consecutive zeros found that are not followed by a valid start code

In a worst-case macroblock decode, the total number of cycles required would be 790 cycles. A worst-case macroblock would consist of an address increment code with more than 4 bits, an M-quant, 2 motion vectors of the long variety, a long pattern code, an all-escape or long R/L pair codes. Macroblock stuffing and address escapes will add one cycle per instance to the worst case number. The inverse discrete transform unit 218 can transform an entire macroblock in 1056 cycles, giving the parser approximately a 50% higher performance than the inverse discrete cosine transform unit. If macroblock stuffing is present, the parser's performance degrades; however, more than 300 stuffing codes would have to be inserted to lower the parser's performance to the level of the inverse discrete cosine transform unit.

Inverse Quantization Unit

The inverse quantization unit 214 decodes the run/length pairs and performs an inverse quantization on the incoming image blocks in accordance with the process outlined in the MPEG-1 specification. The inverse quantization unit 214 contains a 128 bit long word space for reading and writing quantization tables in the IQ unit 214. As noted above, the quantization tables are decoded by the CPU 102 and provided to IQ unit 214. These tables should only be accessed while the IQ unit 214 is in re-set.

De-Zig-Zag Unit

Figure 5A shows the connections between the IDCT and the DZZ and motion compensation unit.

DZZ 216 includes a DZZ address generator, 64 x 12 RAM and flow control logic 256. Data from the IQ unit is written to RAM 254. Address generator 252 selects the data address for a data read so that data out of RAM 254 is in an inverse zig-zag format.

The IDCT/DZZ handshaking interface consists of the production of eight signals from the DZZ flow control 256 that indicate the availability of valid data (DZZ_Validlines). Each signal corresponds to one of eight vertical columns that comprise an 8x8 block of samples. After reading the data from a particular column of samples in RAM 254, IDCT 218 inserts the corresponding signal in the IDCT_invalidateDZZlines bus to inform DZZ 216 that the data has been read. DZZ 216 responds by lowering DZZ valid lines until the column contains new data.

The DZZ data interface provides a 6 bit read address from the IDCT 218 to the DZZ 216. The most significant 3 bits select the vertical column and the least significant bits select an individual sample within the column. The DZZ 216 latches the address from the IDCT 218 and provides the selected data before the end of the next clock cycle. IDCT 218 also provides an enable signal to allow power conservation of the random access memory within the DZZ.

Inverse Discrete Cosine Transform Unit

As noted above, inverse discrete cosine transform (IDCT) unit 218 transforms 8x8 blocks of frequency-domain input samples into 8x8 blocks of spatial domain pixels or differential pixels as specified in the MPEG-1 standard.

The IDCT 218 receives reconstructed frequency domain samples from DZZ 216, performs an inverse DCT to return the data to the spatial domain, and transfers the results to the motion compensator 175. Both interfaces to the IDCT 218 include handshaking. If data from DZZ 216 is unavailable, or the motion compensator 175 is not able to accept additional input, IDCT 218 will stall.

The IDCT and motion compensator handshaking interface includes a ready signal (MOT_spaceavailable) from the motion compensator 175 to the IDCT 218. Eight output values can be sent on the output data interface of IDCT 218. IDCT 218 responds to the request by the motion compensator 175 by asserting the IDCT_motAck to acknowledge that eight samples (comprising a horizontal row of pixels) will be available shortly. IDCT 218 asserts IDCT_dataOutValid when the samples actually appear at the output.

The IDCT data interface consists of a 9-bit, two's complement data bus (IDCT_dataOut) and a single bit data valid qualifier (IDCT_dataOUTVALID). The qualifier signal will be asserted for eight consecutive cycles following each assertion of IDCT_motAck. Each group consists of eight samples comprising a horizontal row of pixels (or differential pixel) outputs. The first group of eight corresponds to the uppermost row, and the outputs proceed downward to the bottom (8th) row of the macroblock. Within each row, the outputs occur in the following order, with zero as the leftmost output, 7 as the rightmost output: 0, 7, 3, 4, 1, 6, 2, 5. If mot_spaceAvailable remains asserted, and the IDCT 218 input data is not started, the IDCT would produce one group of eight results every ten cycles.

Figure 6A is a block diagram of the inverse discrete cosine transform unit. The inverse discrete cosine transform unit takes advantage of the separability of the discrete cosine transform unit by doing sixteen one-dimensional length eight inverse discrete transforms in order to calculate a single two-dimensional 8x8 inverse discrete cosine transform. Each one-dimensional inverse discrete cosine transform requires ten cycles to execute. A single two-dimensional inverse discrete cosine transform can be completed in 176 cycles per block or 1056 cycles per macroblock. The overall performance of the IDCT unit is thus 62,500 macroblocks per second at 66 Mhz. CCIR 601 video consists of 40,500 macroblocks per second, yielding more than 50% overhead above the CCIR 601 video rate. This allows for multiple threads of compressed data to essentially be decoded simultaneously.

As shown in Figure 6A, the IDCT comprises control logic 300, a 64x18 CRAM 302, multiplexers 306-320, registers 322-334, multipliers 336-348, result registers 350-364, sine magnitude to two's-complement converters 365-372, adders 375-382, partial sum registers 384, 386, adder/subtracter 388, final result register 389, two's complement to sine-magnitude converter 390, rounding logic 392, rounded result register 394, clipper logic 396, sine-magnitude to two's complement converter 398, and IDCT_dataout register 399.

The DZZ_DataOut is provided to multiplexer 304 and is distributed to seven multipliers 336-348 and multiplexers 308-320. Only seven multipliers are required as one multiplier MO is used twice (since its the result of which is equivalent to the result of M4). Figure 6C shows the multiplication ordering performed by the control logic for each of the eight iterations (0 - 7). Thus, the result of multiplier 336 is used in register 350 and register 352.

The separability of the DCT transform allows the performance of 16 one dimensional length IDCTs in order to obtain the single, two-dimensional 8x8 IDCT. The IDCT may not be halted in the middle of the computation of any one dimensional transform. It only stops after the transform has been completed, and either the motion compensation unit 175 is unable to accept new data or the DZZ cannot provide new input data. The IDCT will produce eight results each time it goes through the main sequence of states. These results will either be placed into CRAM 302 for vertical iterations, or loaded into the motion compensator 175 for horizontal iterations.

The IDCT control logic 300 loads the proper inputs from the DZZ or CRAM 302 and operates multiplexers 308-320 to control signals in the data path to produce the desired result. Normally, the IDCT control logic 300 cycles through a sequence of 11 states, however, three conditions cause the sequence to vary: assertion of reset, lack of valid data in the DZZ, and lack of space available in the motion compensation unit.

Figure 6B shows the possible states and transitions of the IDCT control logic state machine. The usual sequence of 11 states is shown in the center of the diagram, the reset/no data in the DZZ condition on the left and the motion compensator full state on the right.

The control logic performs eight horizontal and eight vertical iterations per two-dimensional IDCT. The iteration number is maintained in a separate register. The MSB of the iteration register determines whether a horizontal or vertical iteration is taking place. This is, in turn, used to create the read enable for the DZZ, write enable for the CRAM, and to make decisions the next state transition as outlined in Figure 6B.

The reset state of the control logic sets-up the first four multiplications necessary for calculation of the IDCT (LOAD_F0, LOAD_F4, LOAD_F2 and LOAD_F6). The normal 11 stages for the IDCT are LOAD_F1, LOAD_F3, LOAD_F5 AND LOAD_F7 to set up the multiplexers to calculate the multiplication ordering shown in Figure 6C, then computation and storing stages COMPUTE_R0_R7, RESULT_R0_R7, COMPUTE_R3_R4, RESULT_R3_R4, COMPUTE_R1_R6, RESULT_R1_R6, and COMPUTE_R2_R5. At this stage, depending on whether a horizontal or vertical iteration is being performed and whether space is available in the motion compensation unit, the control logic will either loop to the LOAD_F1 stage or store the COMPUTE_R2_R5 result in RESULT_R2_R5. If a vertical iteration is being performed and no preloading (DZZ_validLines-valid) is occurring, or a horizontal iteration is occurring and space is available in the motion compensation unit (mot_spaceAvailable), the LOAD_F1 sequence will be executed. If a vertical iteration is being performed and a preload is occurring, or if a horizontal iteration is being performed and no space is available in the motion compensation unit, the result of the COMPUTE_R2_R5 will be stored and the logic will wait at the WAIT_READY step until DZZ_validLines is valid during a vertical iteration, where the LOAD_F0 step will be executed, or the motion compensation unit has space available during a horizontal iteration.

The IDCT produces results during eight consecutive cycles out of 11 during normal operation. These eight cycles are qualified by the signal IDCT_dataOutValid.

Motion Vector Processor

Figure 7 is a block logic diagram of motion vector processor. The motion vector processor described with respect to Figure 7 implements the decoding of forward and backward motion vectors in accordance with the MPEG-1 specification.

Motion vector processor reconstructs the value of the motion vectors in p-type and b-type macroblock pictures. The macroblocks motion vectors are decoded in accordance with the standards set forth in the MPEG 1 standard. In p-type macroblocks, first the value of the forward motion vector for the macroblock is reconstructed and a prediction macroblock is formed. Then, the DCT coefficient information, stored for some or all of the blocks is decoded, dequantized, inverse DCT transformed and added, in motion compensation unit 180, to the prediction macroblock.

In B-type macroblocks, according to the invention, first, the value of the forward motion vector for the macroblock is reconstructed from the retrieved forward motion vector information, and the backward motion vector for the macroblock is reconstructed from the retrieved backward motion vector information. The forward prediction and the backward prediction are then computed. Finally, the computed prediction is added to the differential pixels from the IDCT.

In motion vector processor 212, horizontal and vertical motion vector data is input from the parser to a bit shifter 402. Shifter 402 is coupled to a forward/backward_r_size register (the forward/backward_r_size values being computed from the picture header information in accordance with the MPEG-1 standard) and the shift of bit register 402 is determined based on the input data. The data is then shifted to an intermediate result holding register 404.

An overflow mask 410 is also generated and comprises the r_size shifted a quantity FFF (hex) to allow for checking of overflows in the picture boundary and allow the reference to "wrap" around picture boundaries. In accordance with the MPEG-1 defined process for reconstructing motion vectors, the reconstruction method implemented by motion vector processor begins by generating a complement to the horizontal or vertical, forward or backward r_values. A sign change control input is provided to an exclusive-OR gate which has, as its other input, the data from register 404. The sign change is implemented dependent upon the values of the forward/backward_r_size, again in accordance with the MPEG-1 specification. The output of XOR gate 406 is provided to an adder 408, which sums the output of XOR gate 406 with the previously retrieved values for the motion_horizontal_forward_r, motion_vertical_forward_r, motion_horizontal_backward_r, and motion_vertical_backward_r stored in registers 412-418 depending upon whether a horizontal or vertical motion vector is being processed.

The output of adder 408 is provided to OR gates 420,422 and AND gates 424, 426 along with mask 410 and the output of registers 412-418, and the output of a selector 428, which adds four to the value of the output from adder 408. The gate array performs computation of the current motion vector being decoded based on the values in registers 412- 418 and the input data. A multiplexer determines the proper result of the gate array output being decoded, i.e., the positive values for the reconstructed horizontal or vertical motion vectors (recon_right, and recon_down, respectively).

The output of MUX 430 is provided to a bit shifter 432 and second MUX 434. The final portion of the motion vector reconstruction involves computing the whole and half-pel unit values from the reconstructed values. The reconstructed values the half-pel values are selected by MUX 434 and stored in register 440. Adder 444 sums the reconstructed value with a horizontal overhead selection value.

Motion Compensation Unit

Each macroblock is stored in memory as 384 continuous bytes. Organization within each macroblock is shown in Figure 8. Each luminance block is divided into two halves, T for top and B for bottom. The chrominance blocks are divided into quarters numbered 0-3 from top to bottom. The offset of the first byte of each of these elements in the macroblock is given by the table in Figure 9. The sort address for any macroblock is given by base + $[(H)(16)H_{size}+(V\%16)]$ x 384. This allows for easy calculation in hardware (since 384 is 3x128).

The motion unit soft reset and enable bits (motReset and motEnable) are present in the MPEG unit configuration register. The address for the reference buffers (Reference 0 and Reference 1) (shown in Figure 10) in system memory 110 must begin on a 4 KB boundary, giving 13 bits address for each buffer. The prediction address should be set to zero if the buffer is not present.

Figure 10A shows the data pipe for the motion compensation unit utilized in the system of the present invention. The pipe consists of a series of registers MUXs and adders which accept 32 byte data segments of prediction data in the order defined in an exemplary macroblock shown in Figure 10B. The addresses of the prediction buffers are held in registers as shown in the following tables:

Table 15 - Forward Prediction Buffer Address

Name	Bit(s)	Type	Description
(reserved)	0:6	x	reserved
forward prediction buffer address	7:19	RW	4K aligned address
(reserved)	20:31	x	reserved

Table 16 - Reverse Prediction Buffer Address

Name	Bit(s)	Type	Description
(reserved)	0:6	x	reserved
reverse prediction buffer address	7:19	RW	4K aligned address
(reserved)	20:31	x	reserved

Each scan is performed first for the forward prediction data, then for the backward prediction data. The pipe first performs horizontal compensation in stages 500(0) through 500(8), then vertical compensation in steps 500(8a) through 500(9) as described further below. In Figure 10B, four blocks comprising a single macroblock are shown for the luminance values. The following description will be limited to the luminance values, though it should be readily understood that the pipeline processing is similar for the chrominance blocks also shown in Figure

10B.

In Figure 10B, a worst case block prediction is shown at 460. In this instance, the block is not horizontally or vertically aligned with any block segment, and thus, to retrieve the eight 32 byte segments making up a single block, fifteen 32 byte segments (numbered 1-15) must be read, since data will be needed from each of these segments to interpolate the values for the selected block 460. Each segment contains four rows of eight pixels each. These rows are inserted into the pipeline as pixel words in the following order:

<u>column</u>	<u>0</u>	<u>1</u>
row 0	0	1
row 1	2	3
row 2	4	5
row 3	6	7

With reference to Figure 10A, data enters the pipe in a series 32 bit registers 500(0-7) and is advanced register to register each clock tick. In a simple case, data is transferred sequentially through the registers to adders 502-508 which perform interpolation (if necessary) by averaging pixel data in adjacent 8-bit segments in register 500(7). However, as will be noted, adder 508 is coupled to multiplexer 510 which has inputs from registers 500(0) and 500(6). For proper interpolation, the "right-most" byte of the even numbered words gets averaged with the "left-most" byte of the odd numbered words. The right-most of the odd words must get averaged with the left-most of the corresponding even word in the adjacent chunk. For two adjacent words:

A0	A1	B0	B1
A2	A3	B2	B3
A4	A5	B4	B5
A6	A7	B6	B7

which enter the pipe in the order: A0, A1, A2, A3, A4, A5, A6, A7, B0, B1, B2, B3, B4, B5, B6, B7, the even numbered words will find their right neighbor seven positions behind them in the pipe, while odd numbered words find their right neighbor 1 position behind. Thus, registers 500(0) and 500(6) provide selectable outputs to MUX 510 which allow the control logic for the motion compensator to average byte neighbors within a word, and the right-most byte of each word with the left-most byte of either the 1-tick or 7-tick delayed word. MUXs 511-514 allow for interpolation adders 502-508 to be bypassed when interpolation is not required (i.e., when the target block 460 is horizontally aligned within the luminance macroblock).

A 36 bit wide register 500(8) stores the interpolated (or non-interpolated) horizontal data in four 9-bit banks. Truncation is performed on the horizontally interpolated data during vertical interpolation and the end pixels are eventually thrown out.

Vertical interpolation is performed in a similar manner using adders 522-528 and multiplexers 525-528.

A 16 x 36 RAM 515 is provided to store the bottom row of each 32 byte segment and return it to the pipe at the proper instance. In vertical interpolation, each pel's neighbor directly above it can be found two clock ticks behind it in the data pipe. Thus registers 500(8a) and 500(8b) are provided to delay the data by two clock ticks before vertical averaging. In a luminance block, this means writing segments 6 and 7 into RAM 515, and reading them back into the pipeline via a MUX 517 into register 500(8a) before the top row of the next data segment reaches the vertical interpolation step.

Register 500(9) stores the interpolation result in a 32 bit register. The data is still aligned in the same format it had in system memory, although interpolated. A MUX 450 utilizes the lower few bits of the prediction address to move bytes horizontally and drop words vertically to shave off all but the prediction data. The horizontal swizzling requires that up to three bytes per row of each chunk be saved and joined with data from the next data segment. Thus a 24 bit wide 4byte x 3 byte array of flip-flops 552 stores this information for rejoinder by MUXs 554-556.

The pipe outputs accurate predictions for either forward, reverse, or both motion vectors. As noted above, the forward and reverse data alternates with each row of data segments (4 pel rows) that come from memory, At the input to the pipe, control instructions ensure that data is provided from the motion vector processor in the right order such that the if both the forward and reverse motion vectors are being predicted, the forward data never gets more than three pixel rows ahead of reverse and vice-versa. Tracking is performed to follow which prediction is ahead, and if data is received for the prediction that is ahead, it is stored in a second 16 x 32 RAM 560. If data is received for the trailing prediction, it can be interpolated with the data stored previously by adders 560-568 and MUXs 571-574.

After both forward and reverse interpolation, a fully reconstructed prediction is ready for reconstruction.

Register 500(B) holds the forward or reverse interpolated data.

Chrominance data is placed into the pipe in a manner similar to luminance, except that the pipe must account for the interleaved structure. Horizontal half-pel interpolation is the same, except the vertical interpolation requires saving and restoring the last row of a block twice as often. The realignment requires setting the chroma prediction as only 8x8 (x2) and forward/reverse interpolation treats the component type as an additional row bit.

Output DMA Unit

Addresses for the video output DMA unit 180 are the same as those in the prediction base address register (Tables 15 and 16). The output DMA unit has two modes: a reference frame format and a strip buffer format. In reference frame format, all the output is written contiguously into reference frame format. A strip buffer (Figure 13) is used in system memory when passing data to save memory when passing non-reference frames to the output formatter. Data is written in 16KB programmable buffer in system memory 110 aligned on a 16K boundary. The following table lists the output unit control registers:

Table 17 - Output Control Register

Name	Bit(s)	Description
Output Address	19:31	Physical base address of reference frame being written
RFU	16:18	

Output Mode	15	OXX -- Reference frame format; XOX -- Reference frame format with handshaking 100 -- 16KB Strip; 101 -- 32 KB Strip; 110 -- 64KB Strip; 111 -- 128 KB Strip
RFU	0:13	

If any of the strip buffer output modes are enabled, the allocated buffer must be large enough to hold at least two rows of macroblocks. This number must be rounded to the next highest power of two (32KB for 352 pel wide video). A reference frame format with handshaking allows writing to a full reference from format in memory while performing output formatting at the same time.

<u>Video Output Formatter</u>

Figure 11 shows a block diagram of the video output formatter utilized in the system of the present invention. The video output formatter is operationally independent from the MPEG core hardware. This is another feature which allows multi-threaded decoding, since the core hardware may decode one stream while the formatter processes another.

The output formatter includes an input DMA interpolation raster 242, color space converter and dither filter 244, and a format conversion filter 246. The control registers set by CPU 102 in output formatter 185 are set forth as follows:

Table 18 - Output Formatter Configuration

Name	Bit(s)	Type	Description
(reserved)	0:2	x	reserved
vof snoop enable	3	RW	enable snooping on vof output

(reserved)	4:6	x	reserved
(reserved)	4:9	x	reserved
format	11	RW	0 = 32 bit 1 = 16 bit
(reserved)	16:18	x	reserved
enable CSC	19	RW	YCbCr->RGB conversion on
(reserved)	20:22	x	reserved
rowChunks	23:31	RW	number of 32B chunks per line

Table 19 - Output Formatter Cropping Control

Name	Bit(s)	Type	Description
(reserved)	0	x	reserved
hStart	1:7	RW	starting horizontal MB offset
(reserved)	8	x	reserved
vStart	9:15	RW	starting vertical MB offset
(reserved)	16	x	reserved
hStop	17:23	RW	ending horizontal MB offset
(reserved)	24	x	reserved
vStop	25:31	RW	ending vertical MB offset

Table 20 - Output Formatter Input Buffer Address

Name	Bit(s)	Type	Description
(reserved)	0:6	x	reserved
unformatted display buffer address	7:19	RW	4K aligned address
(reserved)	20:31	x	reserved

Table 21 - Output Formatter Output Buffer Address

Name	Bit(s)	Type	Description
(reserved)	0:6	x	reserved
formatted display buffer address	7:26	RW	32B aligned address
(reserved)	27:31	x	reserved

Table 22 - Dither Matrix, Upper Half

Name	Bit(s)	Type	Description
dither matrix (0,0)	0:3	RW	signed 4-bit error value
dither matrix (0,1)	4:7	RW	(set to 0 for no dithering)
dither matrix (0,2)	8:11	RW	...
dither matrix (0,3)	12:15	RW	...
dither matrix (1,0)	16:19	RW	...
dither matrix (1,1)	20:23	RW	...
dither matrix (1,2)	24:27	RW	...
dither matrix (1,3) through (1,3)	28:31	RW	...

Table 23 - Dither Matrix, Lower Half

Name	Bit(s)	Type	Description
dither matrix (2,0)	0:3	RW	signed 4-bit error value
dither matrix (2,1)	4:7	RW	(set to 0 for no dithering)
dither matrix (2,2)	8:11	RW	...
dither matrix (2,3)	12:15	RW	...
dither matrix (3,0)	16:19	RW	...
dither matrix (3,1)	20:23	RW	...
dither matrix (3,2)	24:27	RW	...
dither matrix (3,3) through (3,3)	28:31	RW	...

Table 24 - Output Formatter Alpha Fill Value

Name	Bit(s)	Type	Description
(reserved)	0:23	x	reserved
DSB	24	RW	control bit
alpha fill value	25:31	RW	7-bit alpha channel fill value

Table 25 - Output Formatter Image Size

Name	Bit(s)	Type	Description
(reserved)	0:15	x	reserved
mb_height	16:23	RW	image width in macroblocks
mb_width	24:31	RW	image height in macroblocks

As shown in Figure 11A, data from the output DMA controller is first converted to YUV444 format. Figure 11B graphically represents the interpolation of the 4:2:2 data to 4:4:4 format by interpolation of the chroma pixels.

The colorspace conversion and dither block includes adapted conversion coefficients to both convert and amplify the output values of the data. The color value conversion flow is represented in Figure 11C. The YCbCr components are first normalized, and the RGB components then computed at step 482. The dithering matrix from Tables 22 and 23 may then be applied. The format stage 246 is controlled relative to the type of format the data is to be written to.

Control bits for enabling the color space converter, a video output format bit for controlling which output mode of two defined modes will be used (32 bit, 16 bit), and a dithering circuit enable are notably provided. In addition, a separate image size register (Table 25) allows the video output formatter to operate independently of the MPEG core hardware so decoding and formatting can occur simultaneously.

Specific information on how the data formatted by the video output formatter is displayed can be found in copending application Serial Nos. ________ and ________ entitled CONFIGURABLE VIDEO DISPLAY SYSTEM HAVING LIST-BASED CONTROL MECHANISM FOR TIME-DEFERRED INSTRUCTING OF 3D RENDERING ENGINE THAT ALSO RESPONDS TO SUPERVISORY IMMEDIATE COMMANDS, filed May 10, 1995, cited above, and CONFIGURABLE VIDEO DISPLAY SYSTEM HAVING LIST-BASED CONTROL MECHANISM FOR BY-THE-LINE AND BY-THE-PIXEL MODIFICATION OF DISPLAYED FRAMES AND METHOD OF OPERATING SAME filed May 10, 1995, cited above.

Decoding a Single MPEG Stream

Figures 12 and 13 disclose the method of decoding an MPEG encoded video stream in accordance with the invention, and the data flow of video stream decompression in decompressing a single MPEG stream.

As shown in Figure 10, initially, the system instructions program system memory buffers and the configuration register information is set at default. This includes the configuration register bit descriptions to allow the decoder system of the present invention to operate.

At step 262, a read of the compressed video stream from system memory 110 (or another suitable data structure source) occurs to determine, at step 264, context information from the video stream, including image size variables, and information for the parser unit 212 (including picture coding type, the forward R and backward R coding and size, the full pel backward vector, the macroblock width and the macroblock height). At step 266, the context information is programmed into the configuration registers of parser unit 220. At step 268, the Q table values are determined and programmed in the registers of inverse quantization unit 222. At step 270, the decoding of the pictures is determined and the slice information provided to coded data address locations in system memory 110 which are accessible by video bitstream DMA controller 170. The bitstream read addresses are written to the bitstream read address status registers.

Steps 260 through 270 complete the software instruction operations of the system of the present invention with respect to decoding the video bitstream.

The system hardware then completes the video decoding process. At step 272, video bitstream DMA unit 170 controls reads the encoded, macroblock-level data into the FIFO of the video bitstream DMA controller 170 in accordance with the description set forth above. At step 274, parser unit 170 parses the macroblock data into run level pairs for inverse quantization unit 214 and motion vector data for motion vector processor 212. Motion vectors are sent to the motion vector processor 212 at step 276 in Figure 10. At step 280, the inverse quantizer unit 214 performs run level and Huffman decoding using the quantization tables provided at step 268. At step 282, 8x8 DCT coefficient blocks provided from inverse quantization unit 214 are provided to de-zig-zag unit 216 and DCT coefficients data are provided to IDCT unit 218. At step 284, inverse discrete cosine transform unit 218 performs an inverse discrete cosine transform on the decoded data. At step 290, motion compensation unit sums the motion prediction values and the IDCT decoded picture data by querying, when necessary, prediction reference data resident in the system memory, as will be explained with reference to Figure 13. The decoded data is provided to the video output DMA control at step 292 and the video output formatter at step 294.

Figure 13 represents a flow diagram of the data flow from specific locations in system memory 110 during video stream decompression. As should be generally understood by Figure 13, the system memory is divided into five buffers: a coded data buffer, a strip buffer, two reference buffers (reference 0, reference 1), and two output buffers (output 0, output 1).

Image data flow, represented by arrow 600, comprises encoded data at the bitstream slice level, parsed in accordance with steps 260-270 of Fig. 10, provided to MPEG core unit 200 from system memory 110. Decoded data is returned to system memory 110, and specifically to a strip buffer utilized to hold the information prior to display. Decoded prediction data from motion vector processor (step 275) is also written to reference buffers 0 and 1, as represented along line 604, for use by motion compensation unit 179 relative to decoding P-picture and B-picture macroblocks. The decoded prediction data from reference buffers 0 and 1 will, if necessary, be provided to motion compensation unit 175 as represented by line 606. As shown at line 610, reference buffer data may also be used by the output formatter. The output of the video output formatter is provided to output buffers 1 and 2 as represented by line 612.

System memory 110 may include a series of output buffers, and a series of reference buffers, all which may be utilized in accordance with a one-to-one mapping of streams to reference buffer sets when the decoding hardware is decoding multiple streams of data.

However, a unique feature of the system of the present invention is the use of a single set of strip and reference buffers. The reference buffers may be implemented as a cache buffer system where the newest P- or B-picture reference information from several sequences is written into the section of the code (Reference 0 or 1) containing the oldest previously written P- or B- data. This reduces the system memory bandwidth required to implement the system of the present invention.

A software sequence of a multiple threaded decoding algorithm is shown in Figure 14. In interactive bitstreams, the sequence layer, group of pictures layer, and picture layer may be absent. Because multiple sequence headers, group of pictures headers, and picture reference information is included in the stream, random access into the video sequence is possible and indeed contemplated by the MPEG-1 standard. However, to achieve such random access, the MPEG-1 standard relies on repetition of the sequence header. As shown in Fig. 14, each decoding sequence, at the video stream, group of pictures, picture or slice level, will require execution of steps 260-264. Thus, steps 260-264, 260n-264n, and 260-264n+1 are shown for 3 streams. A decision at step 265 is made by the control software dependent upon the nature of the display information being decoded. For example, if the information to be decoded is multiple small pixel array moving representations of baseball players on a field, decision step 265 would determine the ordering of decoding based upon the actions required of the players during the display. Thus, the specific criteria upon which ordering of streams occurs will be dependent upon the nature of the application being decoded, the information being displayed, the output format, and any number of other factors. Each stream from steps 266-264, 260n-264n, etc. may be selectively fed to the hardware processing steps 277-294. Because of the speed of the decoding hardware 200, an effective multiple-thread decode of image data is possible. In other words, multiple streams of data to be decoded could be provided to the decoding hardware for processing and, due to the speed of the hardware, each stream will be decoded and sent to system memory.

Figure 15 shows the inputs and outputs of each block of data and the direction of each block of data during a typical video sequence. The diagram assumes the common IBBPBBPBBI type frame ordering. The input frames are shown in coded as opposed to temporal ordering. The rows detail the input and output of each of the hardware blocks as well as the contents of each buffer over time.

The many features and advantages of the present invention will be readily apparent to one of average skill in the art. In accordance with the objectives of the invention, an efficient, configurable, low-cost MPEG decoding system is provided. The decoder utilizes a unique combination of hardware and software functions to decode an MPEG video stream. The system allows decoding of multiple streams of MPEG video data.

CLAIMS

What is claimed is:

1. A process for decoding MPEG encoded image data stored in a system memory utilizing a configurable image decoding apparatus, said process comprising the steps of:

(a) extracting macroblock information from said MPEG encoded image data, the macroblocks containing image data and motion compensation data;

(b) extracting a series of parameters from the MPEG encoded image data for decoding the MPEG encoded data;

(c) determining quantization factors from the encoded image data;

(d) configuring the configurable image decoding apparatus, including

(i) configuring a means for parsing the macroblock data into motion vectors and image data with the series of parameters with the parameters for decoding the encoded data;

(ii) configuring a means for performing inverse quantization with the quantization co-efficients;

(e) determining a decoding order of the extracted macroblock information to be decoded;

(f) providing said extracted macroblock information to the parsing means in the decoding order;

(g) combining decoded image data with motion vectors extracted by the parsing means; and

(h) storing the combined data in the system memory.

2. The process for decoding according to claim 1 wherein said step (a) comprises extracting a video sequence and parsing the video sequence to obtain the

macroblocks.

3. The process for decoding according to claim 2 wherein said step (a) includes extracting a series of parameters from the video sequence.

4. The process for decoding according to claim 1 wherein said step (a) comprises searching a data structure for said macroblocks.

5. The process for decoding according to claim 1 wherein said step (b) comprises searching a data structure for said context information.

6. The process for decoding according to claim 1 wherein the process includes, prior to step a, the step of

establishing in the system memory a series of buffers, including a display buffer, a reference buffer and a strip buffer.

7. The process for decoding according to claim 6 wherein said step (h) comprises

storing decoded image data in the strip buffer and the reference buffer.

8. The process for decoding according to claim 6 wherein

said step (d)(i) further comprises obtaining, from the series of parameters, image size data, forward and backward r values, and forward and backward prediction values, and writing said values to a configuration register in the means for parsing.

9. The process for decoding according to claim

1 further including the step of

(i) directing the data to an output formatter.

10. The process for decoding according to claim 9 further including the step of

(i) storing data from the output formatter in a display register in system memory.

11. A configurable decoding system in a host system, the host system including host system memory, a host system memory controller and a central processing unit, the system memory storing MPEG encoded video data including a video sequence comprising one or more groups of pictures, each picture comprised of a plurality of slices of macroblocks, each macroblock comprising at least four blocks, said blocks comprising coded picture data and coded motion compensation data, the system comprising:

instruction means for configuring the system memory to include a reference buffer, a display buffer, and a strip buffer;

instruction means for extracting, from said MPEG encoded data, the video sequence and for extracting context information from the video sequence, the context information for decoding the video sequence comprising header information, picture type, frame size, image size and quantization tables, and for extracting said slices of macroblocks from a picture in each group of pictures;

a configurable MPEG decoder, the MPEG decoder including configurable parsing means for extracting picture and motion vector data, means for performing entropy decoding on the picture data, programmable means for performing inverse quantization on the decoded picture data, means for performing inverse zig-zagging, and means for taking the inverse discrete cosine transform of the picture data co-efficients;

configuration control means, operatively coupled to the means for extracting and the configurable MPEG decoder, for configuring the MPEG decoder by programming the parsing means with said picture type, frame size and image size, for configuring the means for performing inverse quantization with said quantization tables;

configurable motion compensation means, coupled to the configurable MPEG decoding unit and the system memory;

configurable video output DMA controlling means, coupled to the motion compensation means and the system memory; and

configurable video output formatting means.

12. The decoding system of claim 11 wherein the instruction means for configuring further includes means for configuring the system memory to include a data buffer, wherein data to be decoded is provided in said data buffer and identified by a plurality of addresses.

13. The decoding system of claim 12 wherein the instruction means for configuring further includes means for configuring the configurable MPEG decoder with said context information.

14. The decoding system of claim 13 wherein said configurable parsing means includes a configuration register, said register being configured to contain said context information, the context information including image size data on the material being

decoded.

15. The decoding system of claim 13 wherein said context information includes at least the picture coding type, the forward r size, and the backward r size of the data to be decoded.

16. The decoding system of claim 15 wherein said programmable means for performing inverse quantization includes a quantization table register.

17. The decoding system of claim 12 wherein the reference buffers are operatively coupled to the motion compensation unit.

18. The decoding system of claim 12 wherein the buffers include a strip buffer, operatively coupled to the motion compensation unit and the video output DMA controller, storing decoded image data.

19. A process for decoding encoded video images in a host system, the host system including a system memory and a central processing unit, the system memory containing image data to be decoded, comprising:

providing a configurable parsing means, an configurable inverse quantization means, an inverse zig-zag unit, and an inverse discrete cosine transform unit;

defining, in said system memory, a first and second display buffers, a strip buffer, a first and second reference buffers, and a bitstream buffer;

extracting from the image data, a video sequence and a series of sequence parameters contained in the video sequence, said sequence parameters including information for decoding at least one picture in the stream;

outputting the sequence parameters to the configurable parsing means;

outputting the image data to the configurable parsing means;

writing decoded data to the strip buffer and to a video output formatter and to the first and second reference buffers; and

outputting from the display means to the first and second reference buffers.

20. An apparatus for processing encoded image data wherein image data is used to produce an image composed of a matrix of pixels, the apparatus being included in a host system, the host system including a system memory and a processor, the apparatus comprising:

a first input port for receiving a first encoded image-defining signal, where said first encoded image defining signal is divisible into at least one pixel defining component, where each pixel defining component may comprise motion vector data or pixel value data;

a first input/output port for receiving and outputting a handshaking signal;

a second input/output port for outputting motion vector data and receiving reference data defining a reference frame relative to the motion vector data;

an output port for outputting decoded image data;

control means, operatively instructing the central processing unit to provide encoded image information into the first input port, operatively instructing decoded data from the output port to be written to system memory, instructing reference information to be input to the second input/output port and instructing decoded data and reference information to be directed to an video output formatter.

21. The apparatus of claim 20 wherein the encoded image data is written from a coded data buffer in the system memory to the first input port.

22. The apparatus of claim 20 wherein the apparatus further includes an encoded data DMA controller, coupled to the first input port and the system memory, controlling writing of the encoded image information to the first input port.

23. The apparatus of claim 20 wherein the decoded data is written to a strip buffer and a reference buffer in the system memory from the output port.

24. The apparatus of claim 20 wherein the apparatus further includes an output DMA controller, coupled to the output port and the system memory, and the output DMA controller controls writing of the decoded image information to the system memory.

25. The apparatus of claim 20 wherein the reference information comprises decoded data from the reference buffer in system memory.

26. The apparatus of claim 20 wherein said apparatus includes a video output display system and reference data is written to the video output display system.

27. A process for decoding coded image data in a host computer, the host computer including a host system memory, a central processing unit, decoding hardware, and video formatting hardware, the process

including:

directing the CPU to parse the system memory into a series of buffers, including a display buffer, a reference buffer and a strip buffer, the instruction means;

reading the coded image data and ascertaining context information regarding information in the data to be decoded;

parsing the coded data into the slice level information and providing the information to the decoding hardware;

retrieving decoded picture data from the decoding hardware;

storing said decoded picture data in said reference buffers and in said strip buffer;

directing the reference buffer data to the decoding hardware;

outputting reference buffer information and decoded picture data to the video formatting hardware;

storing formatted decoded picture data in a display buffer in said system memory.

28. A process for decoding coded image data in a host computer, the host computer including a central processing unit (CPU) and system memory, the computer including a decoding processor, comprising the steps of:

(a) directing the CPU to perform the steps of

parsing the system memory into a series of buffers, including a display buffer, a reference buffer and a strip buffer;

reading the coded image data and ascertaining context information regarding information in the data to be decoded;

parsing the coded data into the slice level information and providing the information to the decoding processor;

(b) directing the decoding processor to perform the steps of

distributing coded motion vector information blocks and image data information blocks;

decoding the image data blocks into quantized coefficient blocks;

performing an inverse quantization on said quantized coefficient blocks to form pixel value blocks;

converting the pixel value blocks to pixel coefficients;

calculating the inverse discrete cosine transform of the pixel coefficients to produce pixel display values;

decoding the motion vector blocks into pixel motion vectors; and

adding the pixel motion vectors and pixel display values; and

(c) directing the CPU to perform the steps of:

retrieving decoded picture data from the decoding hardware;

storing said decoded picture data in said system memory;

directing the reference buffer data to the decoding hardware; and

storing formatted decoded picture data in a display buffer in said system memory.

29. A process for decoding coded image data in a host computer, the host computer including a host system memory, a central processing unit, decoding hardware, and video formatting hardware, the coded data

including an nth stream of video data, an n + 1 stream of video data and an n + m stream of video data, where n and m are integers, the process including:

directing the CPU to parse the system memory into a series of buffers, including a display buffer, a reference buffer and a strip buffer, the instruction means;

reading the coded image data and, for each said stream, ascertaining context information regarding the coded image data to be decoded;

parsing, for each stream, the coded data into the slice level information;

ordering the coded data and the context information into a stream decoding order;

providing the coded data and context information to the decoding hardware;

retrieving decoded picture data from the decoding hardware;

storing said decoded picture data in said reference buffers and in said strip buffer;

directing the reference buffer data to the decoding hardware;

outputting reference buffer information and decoded picture data to the video formatting hardware;

storing formatted decoded picture data in a display buffer in said system memory.

30. A process for decoding coded image data in a host computer, the host computer including a central processing unit (CPU) and system memory, the coded data including an nth stream of video data, an n + 1 stream of video data and an n + m stream of video data, where n and m are integers the computer including a decoding processor, comprising the steps of:

(a) directing the CPU to perform the steps of parsing the system memory into a series of buffers, including a display buffer, a reference buffer and a strip buffer;

reading the coded image data;

determining, for each said stream, context information regarding information in the data to be decoded;

parsing, for each stream, the coded data into the slice level information and providing the information to the decoding processor;

(b) directing the decoding processor to perform the steps of

distributing coded motion vector information blocks and image data information blocks;

decoding the image data blocks into quantized coefficient blocks;

performing an inverse quantization on said quantized coefficient blocks to form pixel value blocks;

converting the pixel value blocks to pixel coefficients;

calculating the inverse discrete cosine transform of the pixel coefficients to produce pixel display values;

decoding the motion vector blocks into pixel motion vectors; and

adding the pixel motion vectors and pixel display values; and

(c) directing the CPU to perform the steps of:

retrieving decoded picture data from the decoding hardware;

storing said decoded picture data in said system memory;

directing the reference buffer data to the decoding hardware; and

storing formatted decoded picture data in a display buffer in said system memory.

31. An MPEG decoder in a host computer system, the host computer system including a host processor, a system memory, a system bus, and a system memory controller, comprising:

a memory controller interface coupled to the system memory controller;

a video stream DMA controller, coupled to the memory controller interface;

a parsing means for distributing coded motion vector information blocks and image data information blocks;

an entropy decoding means, coupled to the parsing means, receiving distributed image data blocks and decoding the image data blocks into quantized coefficient blocks;

an inverse quantization means for receiving the quantized coefficient blocks and performing an inverse quantization on said quantized coefficient blocks to form pixel value blocks;

an inverse zig-zag means for converting the pixel value blocks to pixel coefficients;

an inverse discrete cosine transform means for calculating the inverse discrete cosine transform of the pixel coefficients to produce pixel display values;

a motion vector processor means, coupled to the parsing means and receiving the distributed motion vector blocks, for decoding the motion vector blocks into pixel motion vectors;

a motion compensation unit, coupled to the motion vector analyzer and the inverse discrete cosine transform means, for adding the pixel motion vectors and pixel display values;

a video output DMA controller, coupled to the motion compensation unit and the memory controller interface, for ordering the pictures in an output order; and

a video output formatter, coupled to the video output DMA controller and the memory controller interface.

32. The decoder of claim 31 wherein the system memory includes data buffers, wherein data to be decoded is provided in said buffers identified by a plurality of addresses, and wherein said DMA controller is operatively coupled to said buffers.

33. The decoder of claim 32 wherein the bitstream DMA controller includes a FIFO RAM, a FIFO RAM controller, an end of picture detector, and an address generator for generating said addresses.

34. The decoder of claim 33 wherein the bitstream DMA controller includes an address register queue, said register queue containing system memory addresses for the coded data.

35. The decoder of claim 34 wherein the address register queue includes a current address register, a next address register, a current length register and a next length register.

36. The decoder of claim 31 wherein the parsing means comprises a bit shifter, a state machine and a FIFO RAM.

37. The decoder of claim 36 wherein the parsing

means contains programmable registers for receiving context information including image size data and picture coding information.

38. The decoder of claim 37 wherein the image size data includes an macroblock width and macroblock height.

39. The decoder of claim 38 wherein the picture coding information includes the picture coding type, the forward r size, backward r size, the forward pel vector and backward pel vector.

40. The decoder of claim 36 wherein the state machine is coupled to the motion vector processor means, and said FIFO.

41. The decoder of claim 40 wherein an output of said state machine comprises motion vector data provided to the motion vector processing means and another output of said state machine comprises picture data provided to said FIFO.

42. The decoder of claim 40 wherein said system memory includes a configuration register containing configuration information on said parsing means, entropy decoding means, motion compensation unit, video output DMA controller and video output formatter.

43. The decoder of claim 40 wherein said inverse quantization means includes a quantization table register.

44. An MPEG decoder in a host computer system, the host computer system including a host processor, a

system memory, a system bus, and a system memory controller, the system memory containing data to be decoded, comprising:

control means for instructing the host processor to deconstruct the encoded image data to extract macroblock level data comprising encoded picture data blocks and motion vector blocks, for instructing the host processor to determine the decoding order of the macroblock data, and for extracting picture data and quantization table data from the encoded image data;

a system memory controller interface coupled to the system memory controller via the system bus;

a video image data DMA controller, coupled to the system memory controller interface, the DMA controller including a video stream buffer receiving picture data from the system memory under direction of the control means;

a motion compensation unit, coupled to the system memory controller interface;

a slice and macroblock decompression unit, coupled to the video stream buffer and the motion compensation unit, the decompression unit comprising

a configurable parser, coupled to the video stream buffer for directing pixel data blocks and motion vector blocks;

a configurable decoding unit receiving pixel data blocks and performing entropy decoding and inverse quantization on said pixel data blocks;

a pixel data block inverse zig-zag scan unit, receiving pixel data blocks from the configurable decoding unit;

an inverse discrete cosine transform unit receiving pixel data blocks from the inverse zig-zag scan unit and performing and outputting pixel data blocks having decoded pixel value data; and a motion vector processor, coupled to the parser, receiving the motion vector blocks;

a motion compensation unit, coupled to the inverse discrete cosine transform unit and the motion vector processor;

a video output DMA controller, coupled to the system memory interface controller and the motion compensation unit; and

a video output formatter, coupled to the system memory interface controller and the motion video output DMA controller.

45. The decoder of claim 44 wherein said control means includes means for defining, in said system memory, a plurality of buffers including at least configuration buffers, data buffers, and display buffers.

46. The decoder of claim 45 wherein said data buffers include said encoded data blocks.

47. The decoder of claim 44 further including hardware configuration registers, said configuration registers including system configuration information for said configurable parser, said configurable decoding unit, said video output DMA controller, and said video output formatter.

48. The decoder of claim 47 wherein said parser includes a parser configuration register for storing context information.

49. The decoder of claim 47 wherein said parser configuration information includes at least the picture coding type, the forward r size, and the backward r

size of the data to be decoded.

50. The decoder of claim 45 wherein the buffers include a reference buffer, operatively coupled to the motion compensation unit, storing reference image data.

51. The decoder of claim 50 wherein two reference buffers are provided, and reference picture information is alternatively written to each said buffer.

52. The decoder of claim 45 wherein the registers include a strip buffer register, operatively coupled to the motion compensation unit and the video output DMA controller, storing decoded image data.

53. The decoder of claim 45 wherein the display buffers include display data output from the video output formatter.

54. The decoder of claim 45 wherein the entropy decoding unit includes a register to store quantization table data.

55. The decoder of claim 45 wherein the video image data DMA controller includes an address queue and a length queue, the address and length queues including current and future system memory addresses where coded data is located in system memory.

56. An integrated circuit for decoding coded data in a host system, the host system including a host system memory, a host system processor, a host system memory controller, and a host system bus, the integrated circuit comprising:

a memory controller interface, coupled to the host system memory controller by the host system bus;

a input/output bus, operatively coupled to the host system memory controller and the memory controller interface;

an encoded data DMA controller, coupled to the i/o bus;

a motion compensation unit, coupled to the i/o bus;

a output data DMA controller, coupled to the i/o bus;

an output formatter, coupled to the i/o bus; and

data decompression hardware, having a first i/o port operatively coupled to the encoded data DMA controller and a second i/o port coupled to the motion compensation unit, said hardware receiving encoded MPEG video macroblock information at the first I/O port and outputting decoded MPEG video data at the second I/O port.

57. The circuit of claim 56 wherein the system memory includes data buffers, wherein data to be decoded is provided in said buffers identified by a plurality of addresses, and wherein said encoded data DMA controller is operatively coupled to said buffers via the memory controller interface.

58. The circuit of claim 56 wherein the encoded data DMA controller includes a FIFO RAM, a FIFO RAM controller, an end of picture detector, and an address generator for generating said addresses.

59. The circuit of claim 58 wherein the address generator includes registers containing configuration information regarding the encoded data location in system memory.

60. The circuit of claim 59 wherein the registers include a current address register, a next address register, a current length register and a next length register.

61. The circuit of claim 56 wherein data decompression hardware includes

a configurable parser, coupled to the system memory, for directing encoded data in pixel data blocks and motion vector blocks;

a configurable decoding unit receiving pixel data blocks and performing entropy decoding and inverse quantization on said pixel data blocks;

a pixel data block inverse zig-zag scan unit, receiving pixel data blocks from the configurable decoding unit;

an inverse discrete cosine transform unit receiving pixel data blocks from the inverse zig-zag scan unit and performing and outputting pixel data blocks having decoded pixel value data; and

a motion vector processor, coupled to the parser, receiving the motion vector blocks and generating motion pixel motion data.

62. The circuit of claim 61 wherein the configurable parser includes configuration registers for storing context information on said encoded video data.

63. The circuit of claim 61 wherein the configurable decoding unit includes a register for storing quantization tables.

64. The circuit of claim 56 wherein the data decompression hardware further includes

a motion compensation unit, coupled to the inverse discrete cosine transform unit and the motion vector processor;

a video output DMA controller, coupled to the system memory interface controller and the motion compensation unit; and

a video output formatter, coupled to the system memory interface controller and the motion video output DMA controller.

65. An MPEG decoding system, comprising:

a host system including a host system memory, a host system memory controller, a host system processor, and a host system bus, the host system memory being divided into at least a storage area buffer, a first and a second display buffer buffers, a coded data buffer, and a first and second reference buffers;

MPEG video data decoding hardware including:

means for parsing image data blocks and motion vector blocks from macroblock data;

means for constructing motion vector data from coded motion vector blocks;

means for performing entropy decoding on coded image data blocks;

means for performing inverse quantization of the coded image data blocks ;

means for taking the inverse discrete cosine transform of the coded image data;

a motion compensation means, coupled to the means for taking the inverse discrete cosine transform and the motion vector processor, and operatively coupled to the system memory, for constructing picture data from the image data and motion vector blocks;

a video output DMA controller, operatively coupled to the system memory controller and the motion compensation means;

a video output formatter, coupled to the system memory and the video output DMA controller; and

instruction means, provided in the storage area and executable by the host system processor, for directing encoded image data to the parsing means in a decoding order, for configuring the means for parsing image data blocks, and interacting with the host system memory to store decoded image data, display image data, and configuration data for the decoding hardware.

66. The decoder of claim 65 wherein said instruction means includes means for defining, in said system memory, a plurality of buffers including at least data buffers, reference buffers, and display buffers.

67. The decoder of claim 66 wherein said data buffers include said encoded data blocks.

68. The decoder of claim 65 further including system configuration registers wherein said configuration registers include configuration information for said configurable parser, said configurable decoding unit, said video output DMA controller and Video Output Formatter.

69. The decoder of claim 68 wherein said parser includes a configuration data register including at least the picture coding type, the forward r size, and the backward r size of the data to be decoded.

70. The decoder of claim 66 wherein the plurality

of buffers further includes a strip buffer register, operatively coupled to the motion compensation unit and the video output DMA controller, storing decoded image data.

71. An MPEG decoding system, comprising:

software means for decoding a first portion of MPEG encoded data, including means for extracting macroblock data from said MPEG encoded data and for establishing a decoding order for said macroblock data; and

hardware means for decoding the macroblock data, including

means for extracting motion vector data and display data from said macroblocks,

means for decoding encoded AC coefficients and DC coefficients in said display data,

means for inversely quantizing said coefficients into a resulting array of decoded AC coefficient,

means for inversely scanning said array of decoded AC coefficients and DC coefficients in a zig-zag pattern to provide a block of discrete cosine transformed coefficients,

means for taking the inverse discrete cosine transform of the block of discrete cosine transformed coefficients to provide a first set of pel data;

means for decoding the motion vector data to provide a second set of pel data; and

means for adding the first and second sets of pel data.

72. An MPEG decoding system, comprising:

software means for decoding a nth stream of MPEG encoded data, an nth + 1 stream of MPEG encoded data, and an nth + m stream of MPEG encoded data, where m is an integer, including means for extracting macroblock data from each said MPEG encoded stream data and for establishing a decoding order for said macroblock data, and means for ordering the macroblock data from each said stream for decoding; and

hardware means for decoding the macroblock data, including

means for extracting motion vector data and display data from said macroblocks,

means for decoding encoded AC coefficients and DC coefficients in said display data,

means for inversely quantizing said coefficients into a resulting array of decoded AC coefficient,

means for inversely scanning said array of decoded AC coefficients and DC coefficients in a zig-zag pattern to provide a block of discrete cosine transformed coefficients,

means for taking the inverse discrete cosine transform of the block of discrete cosine transformed coefficients to provide a first set of pel data;

means for decoding the motion vector data to provide a second set of pel data; and

means for adding the first and second sets of pel data.

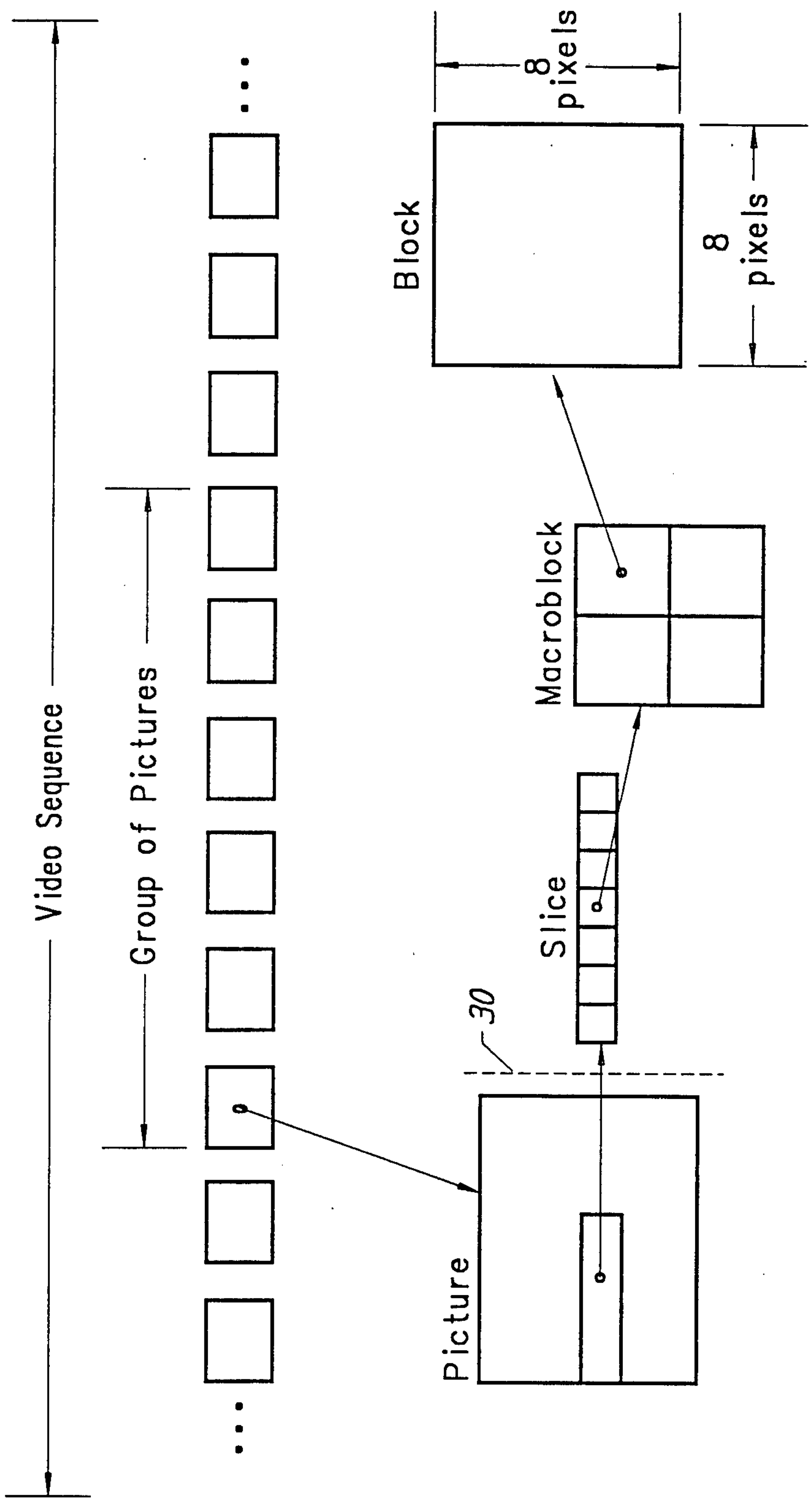

Video Sequence
Group of Pictures
Picture
30
Slice
Macroblock
Block
8
pixels
8
pixels

FIG. 1

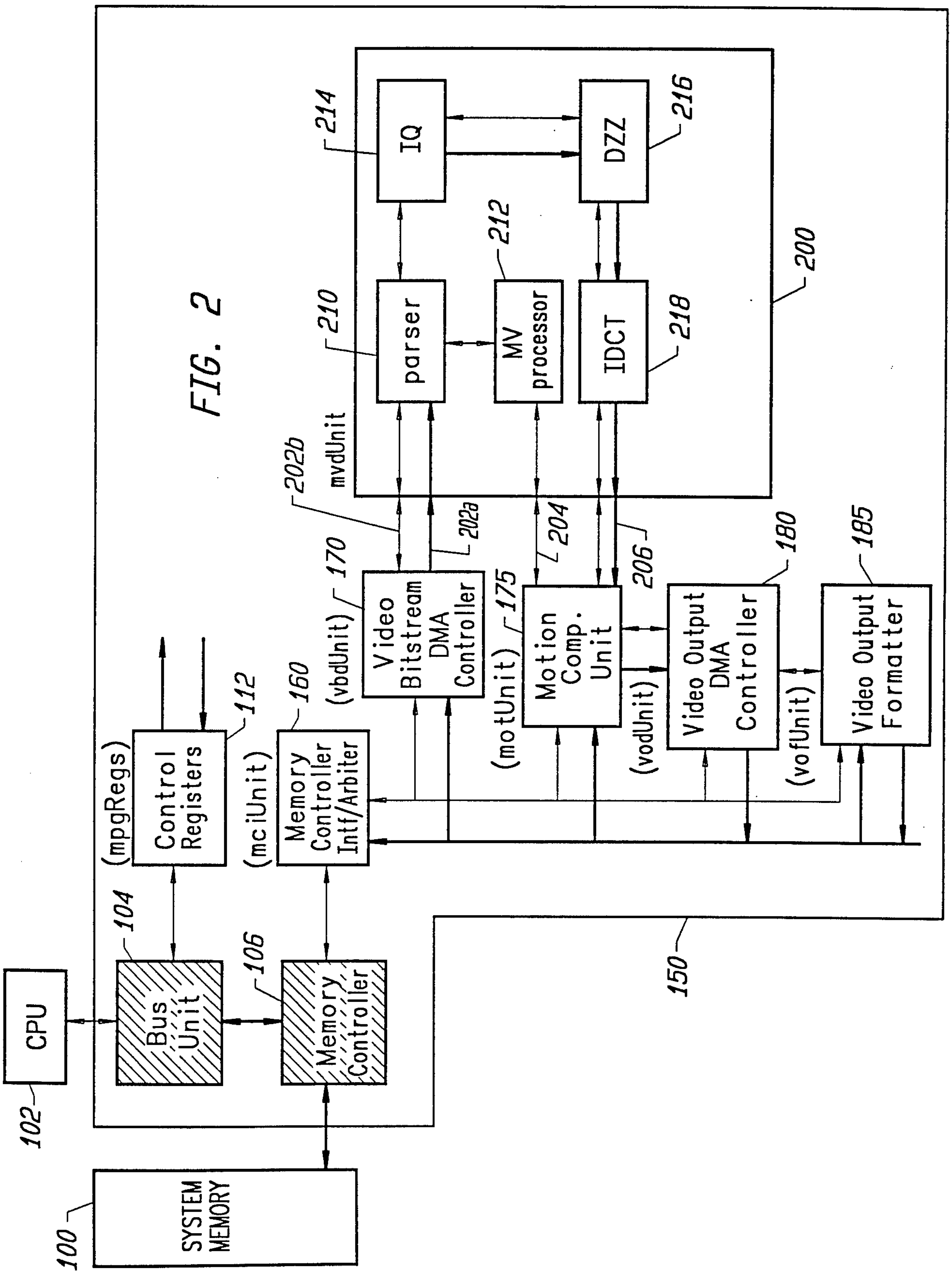

IQ
214
DZZ
216
212
200
parser
210
MV processor
IDCT
218
mvdUnit
202b
202a
204
206
170
Video Bitstream DMA Controller
(vbdUnit)
175
Motion Comp. Unit
(motUnit)
180
Video Output DMA Controller
(vodUnit)
185
Video Output Formatter
(vofUnit)
112
Control Registers
(mpgRegs)
160
Memory Controller Intf/Arbiter
(mciUnit)
104
Bus Unit
106
Memory Controller
CPU
102
150
SYSTEM MEMORY
100

FIG. 2

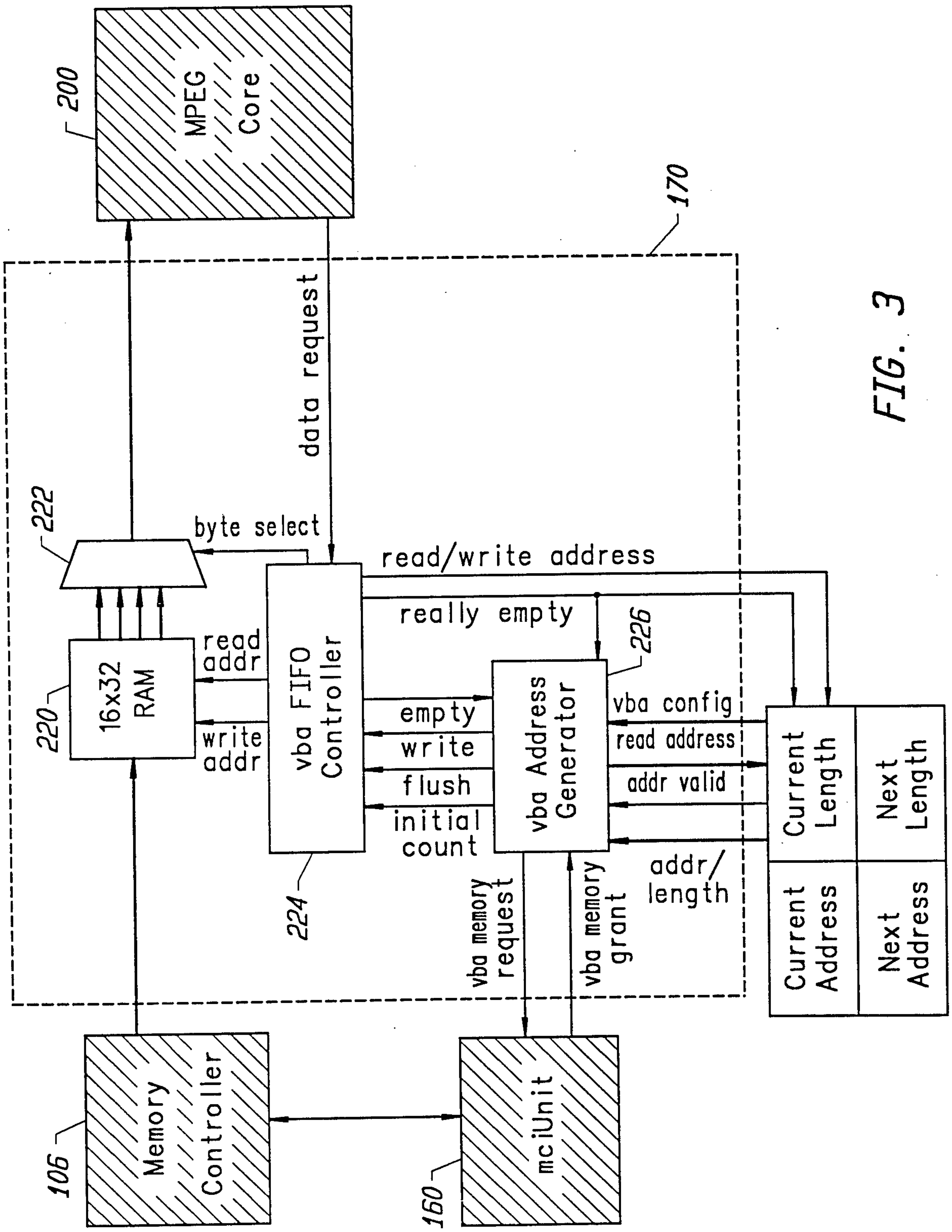
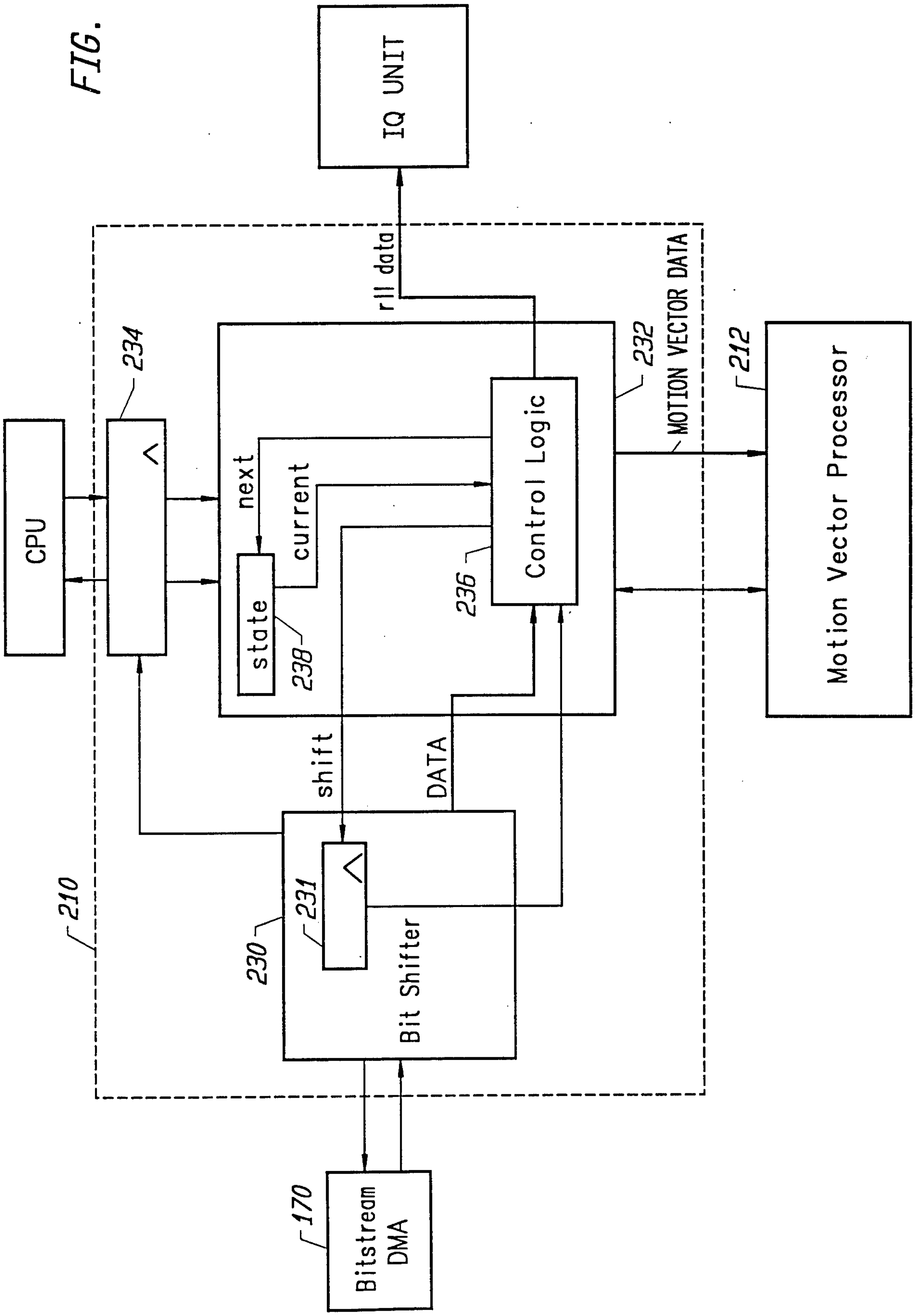

MPEG Core
200
170
data request
byte select
222
read/write address
really empty
226
read addr
16x32 RAM
220
write addr
vba FIFO Controller
224
empty
write
flush
initial count
vba Address Generator
vba config
read address
addr valid
addr/ length
Current Length
Next Length
Current Address
Next Address
vba memory request
vba memory grant
Memory Controller
106
mciUnit
160

FIG. 3

FIG. 4


IQ UNIT
rll data
MOTION VECTOR DATA
CPU
234
232
212
Control Logic
next
current
state
238
236
Motion Vector Processor
shift
DATA
210
230
231
Bit Shifter
170
Bitstream DMA

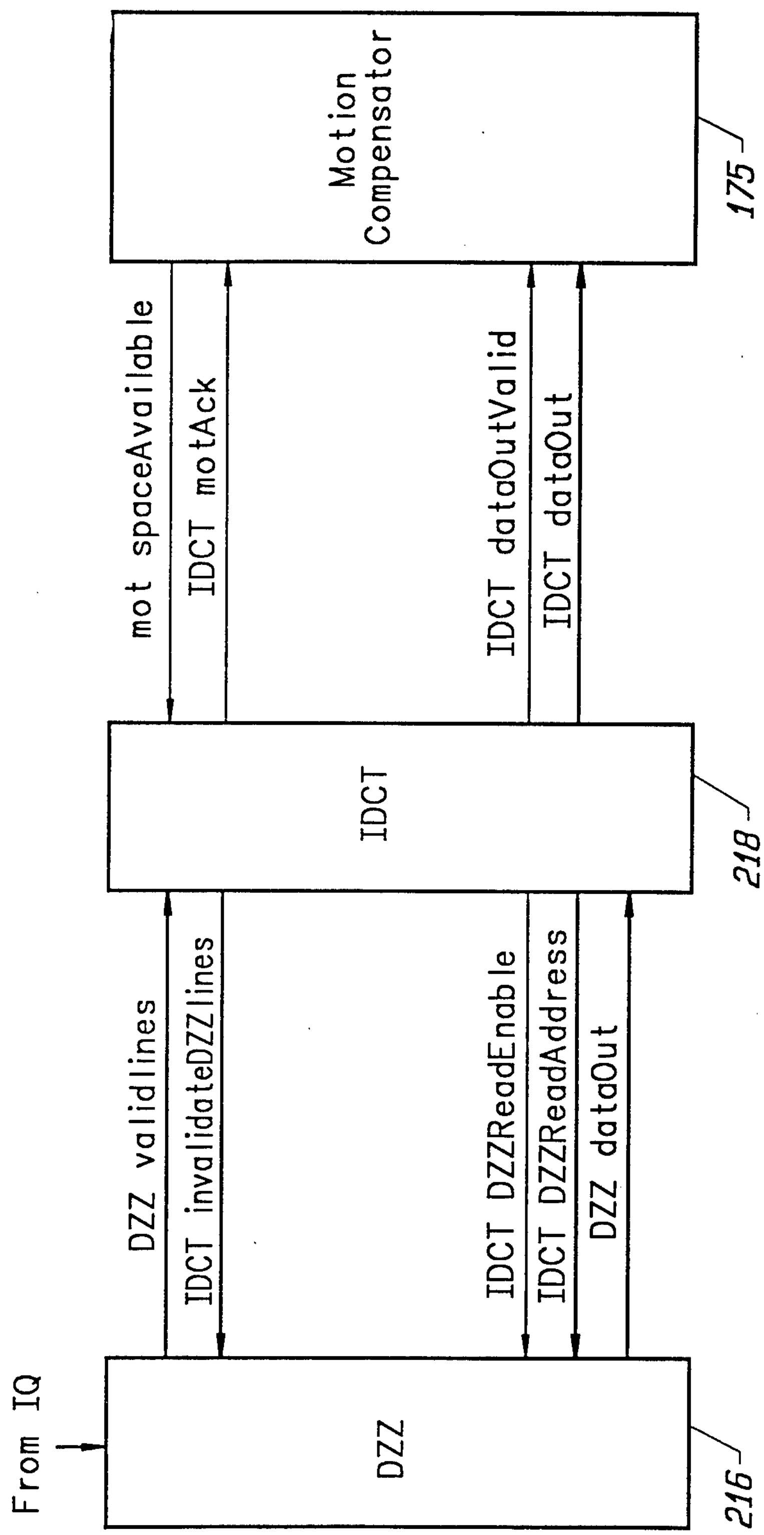
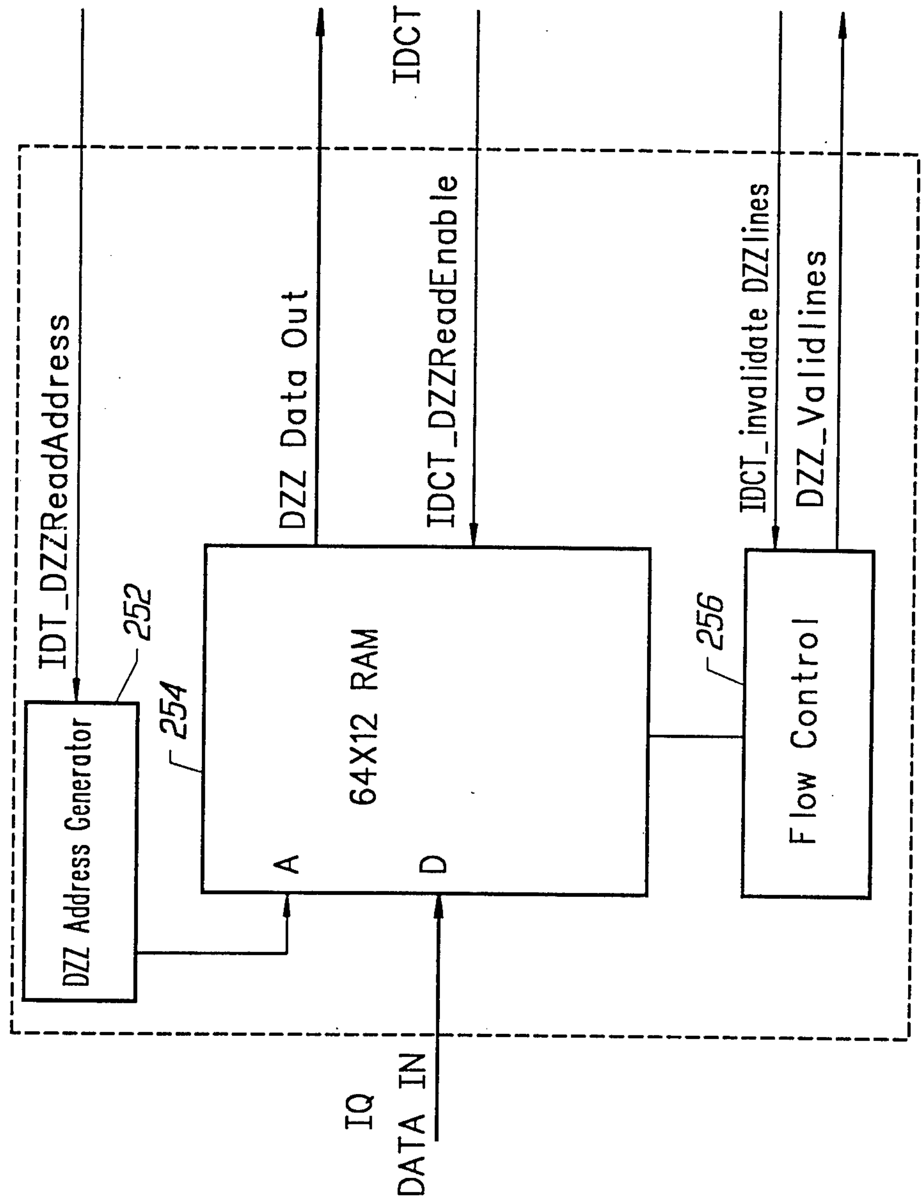

Motion Compensator
175
mot spaceAvailable
IDCT motAck
IDCT dataOutValid
IDCT dataOut
IDCT
218
DZZ validlines
IDCT invalidateDZZlines
IDCT DZZReadEnable
IDCT DZZReadAddress
DZZ dataOut
From IQ
DZZ
216

FIG. 5A

FIG. 5B


IDT_DZZReadAddress
DZZ Data Out
IDCT
IDCT_DZZReadEnable
IDCT_invalidate DZZlines
DZZ_Validlines
252
254
256
DZZ Address Generator
64X12 RAM
A
D
Flow Control
IQ
DATA IN

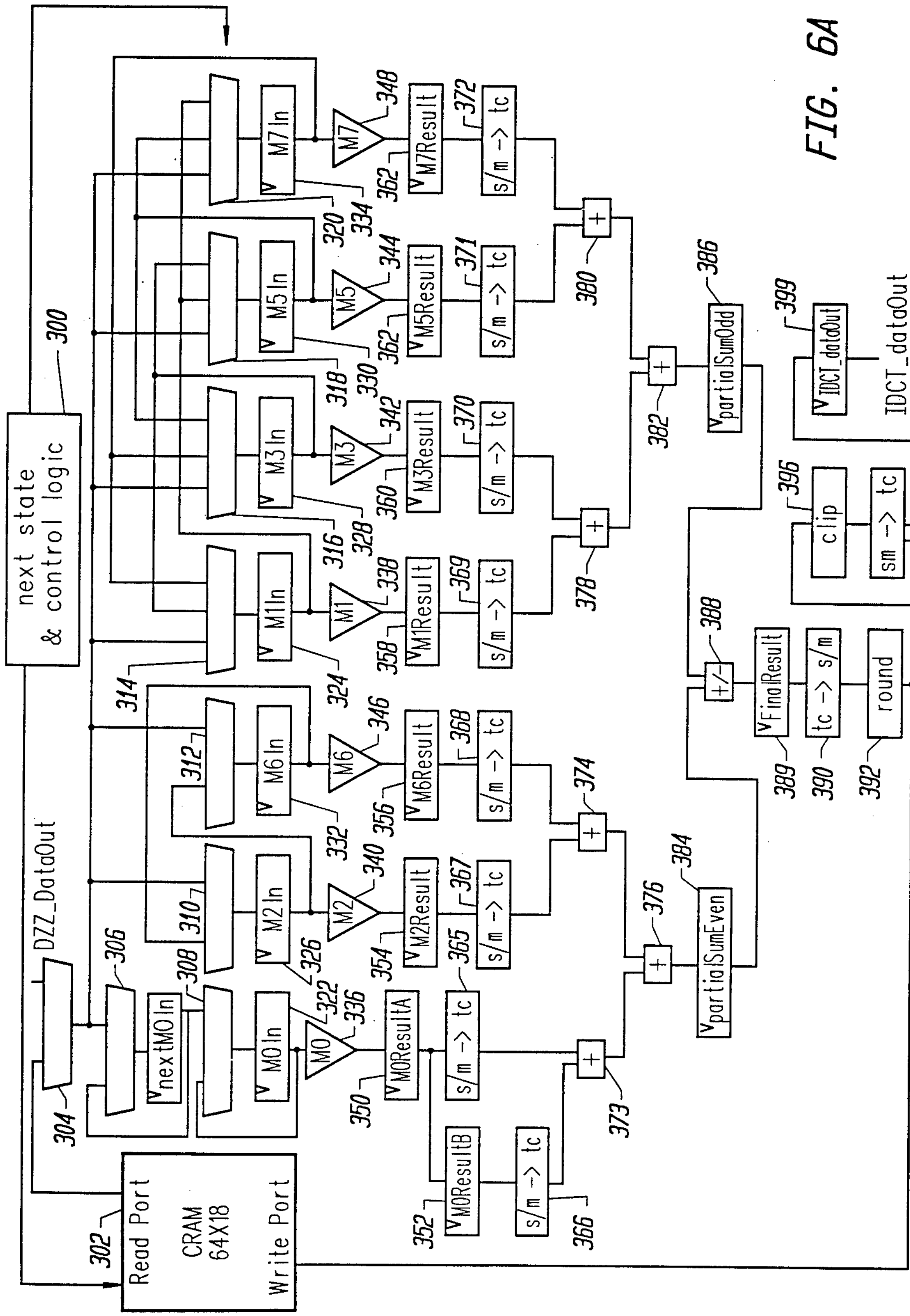

next state & control logic
300
CRAM 64X18
Read Port
Write Port
302
DZZ_DataOut
304
306
308
310
312
314
316
318
320
V nextM0In
V M0In
V M1In
V M2In
V M3In
V M5In
V M6In
V M7In
M0
M1
M2
M3
M5
M6
M7
V M0ResultA
V M0ResultB
V M1Result
V M2Result
V M3Result
V M5Result
V M6Result
V M7Result
s/m -> tc
V partialSumEven
V partialSumOdd
+/-
V FinalResult
tc -> s/m
round
V roundedResult
clip
sm -> tc
V IDCT_dataOut
IDCT_dataOut
322
324
326
328
330
332
334
336
338
340
342
344
346
348
350
352
354
356
358
360
362
365
366
367
368
369
370
371
372
373
374
376
378
380
382
384
386
388
389
390
392
394
396
398
399

FIG. 6A

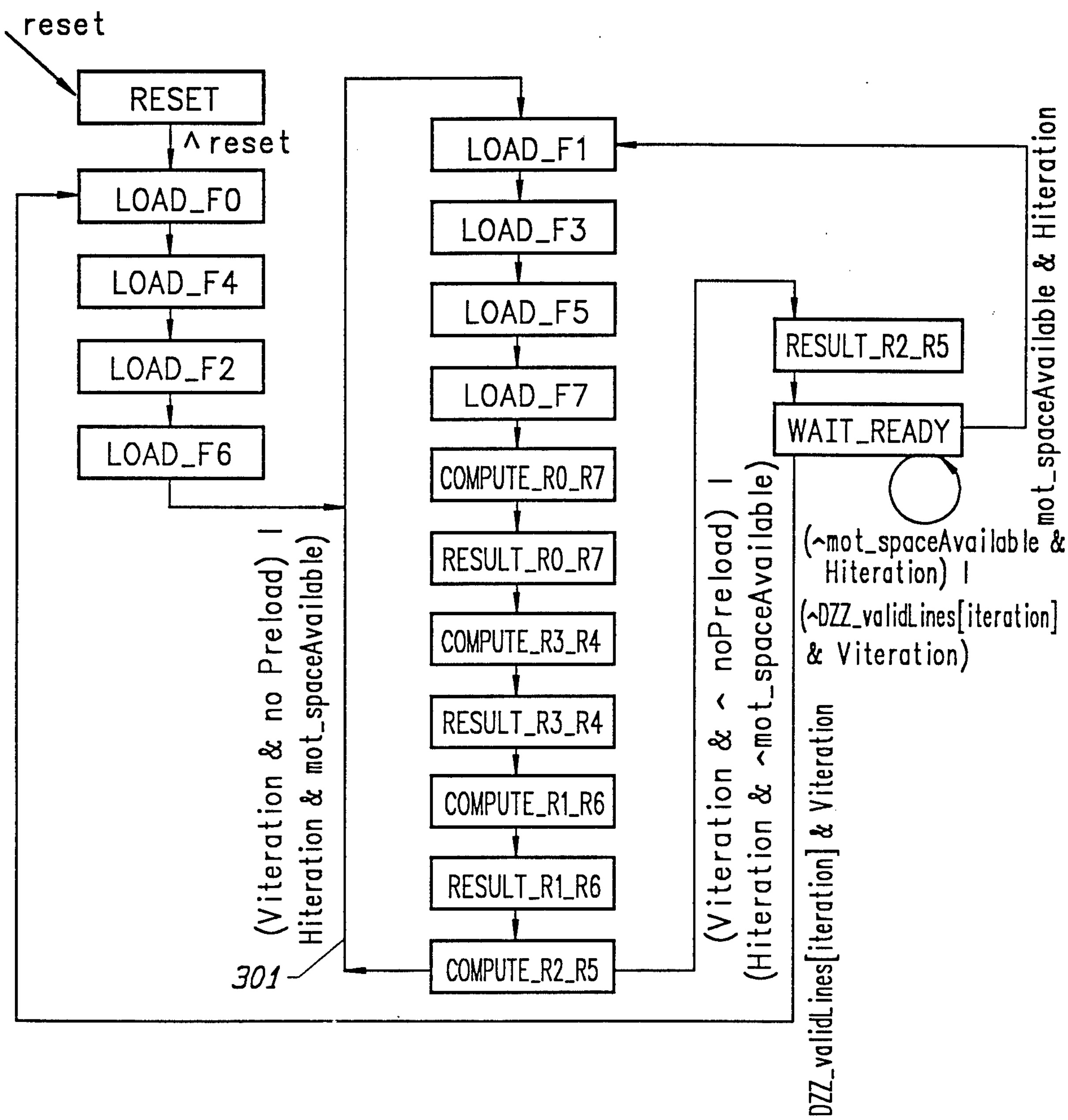

reset
RESET
^reset
LOAD_F0
LOAD_F4
LOAD_F2
LOAD_F6
LOAD_F1
LOAD_F3
LOAD_F5
LOAD_F7
COMPUTE_R0_R7
RESULT_R0_R7
COMPUTE_R3_R4
RESULT_R3_R4
COMPUTE_R1_R6
RESULT_R1_R6
COMPUTE_R2_R5
RESULT_R2_R5
WAIT_READY
(Viteration & no Preload) |
Hiteration & mot_spaceAvailable)
301
(Viteration & ^ noPreload) |
(Hiteration & ~mot_spaceAvailable)
DZZ_validLines[iteration] & Viteration
mot_spaceAvailable & Hiteration
(~mot_spaceAvailable & Hiteration) |
(~DZZ_validLines[iteration] & Viteration)

FIG. 6B

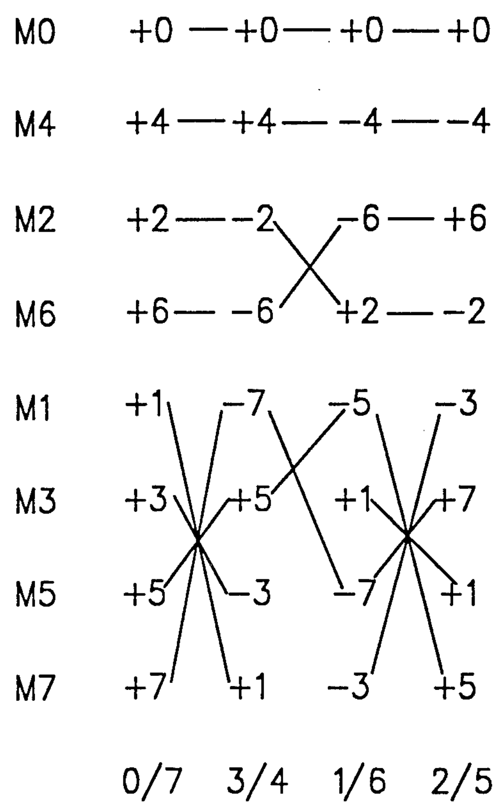

M0 +0 +0 +0 +0
M4 +4 +4 -4 -4
M2 +2 -2 -6 +6
M6 +6 -6 +2 -2
M1 +1 -7 -5 -3
M3 +3 +5 +1 +7
M5 +5 -3 -7 +1
M7 +7 +1 -3 +5
0/7 3/4 1/6 2/5

FIG. 6C

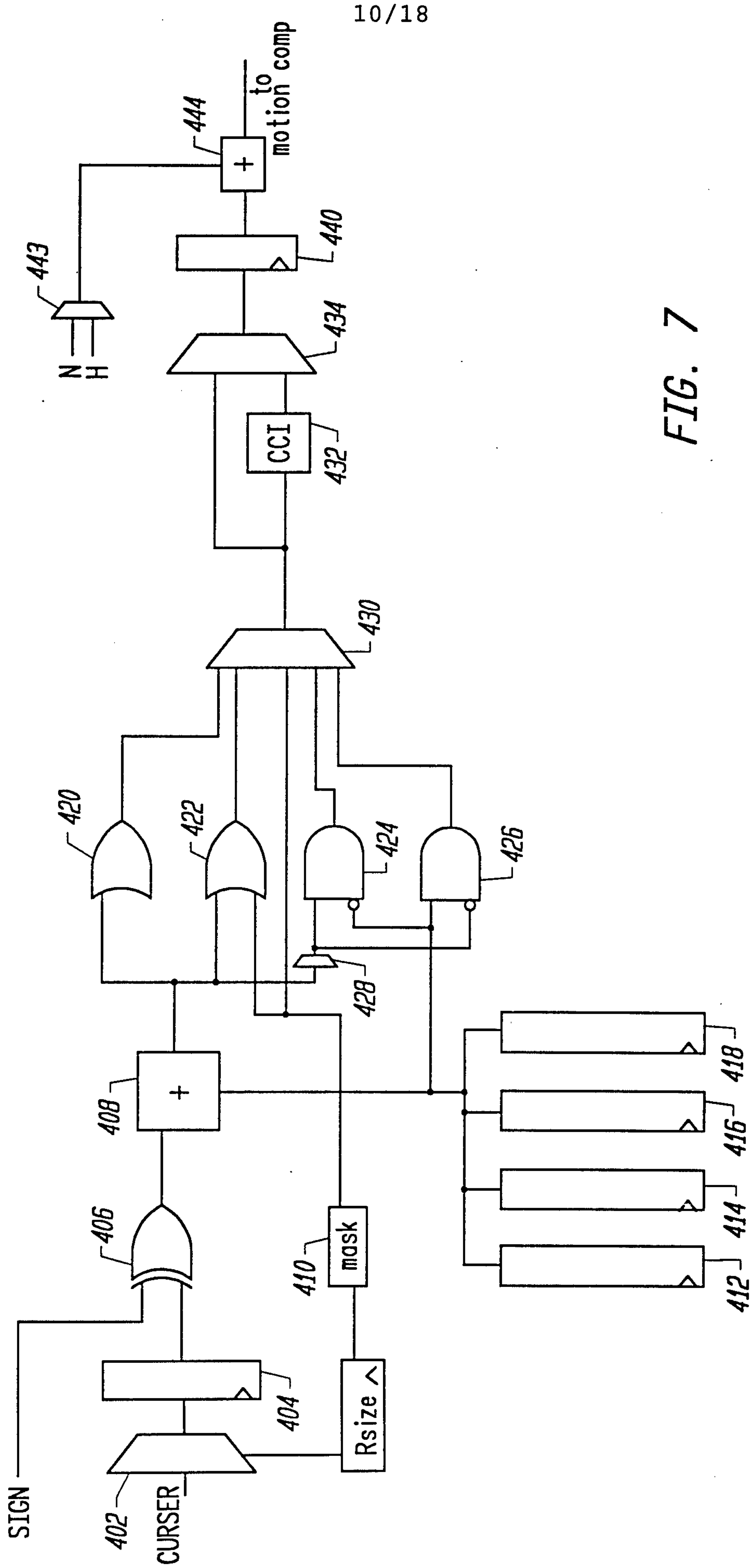

to motion comp
444
+
440
443
N
H
434
CCI
432
430
420
422
424
426
428
408
+
418
416
414
412
406
410
mask
404
Rsize
402
SIGN
CURSER

FIG. 7

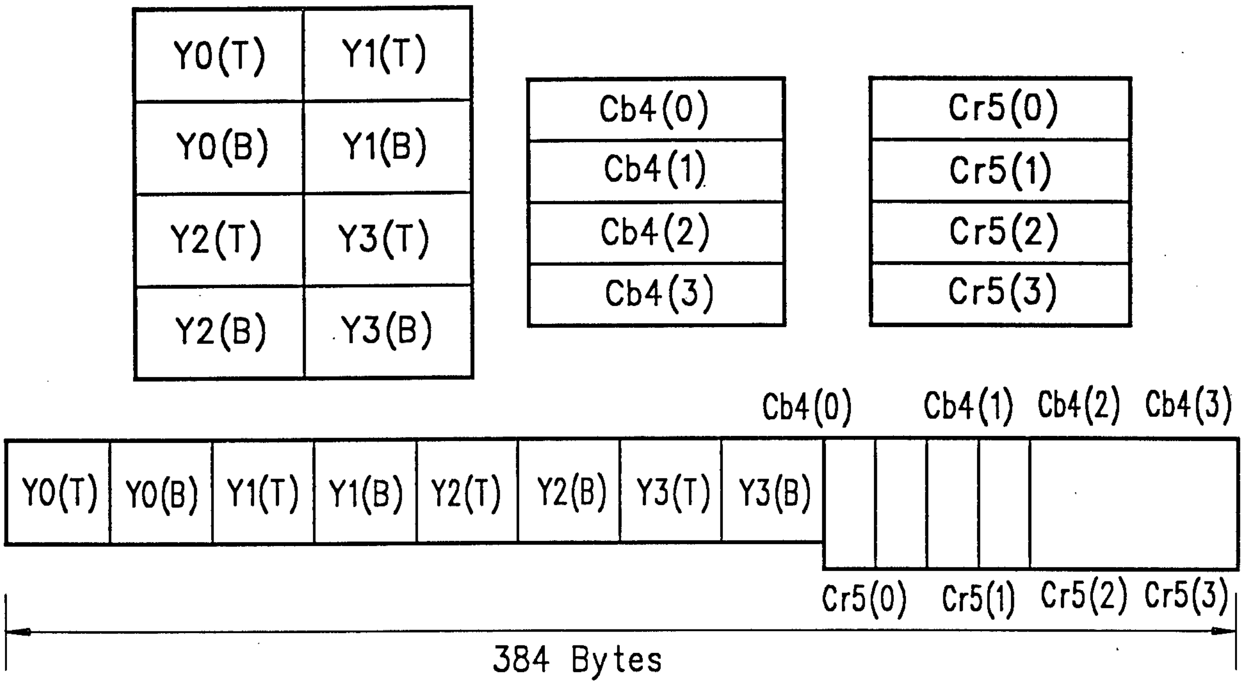


FIG. 8

Byte Offsets

Block	Offset
Y0(T)	0
Y0(B)	32
Y1(T)	64
Y1(B)	96
Y2(T)	128
Y2(B)	160
Y3(T)	192
Y3(B)	224
Cb4(0)	256
Cb4(1)	272
Cb4(2)	288
Cb4(3)	304
Cr5(0)	320
Cr5(1)	336
Cr5(2)	352
Cr5(3)	368

FIG. 9

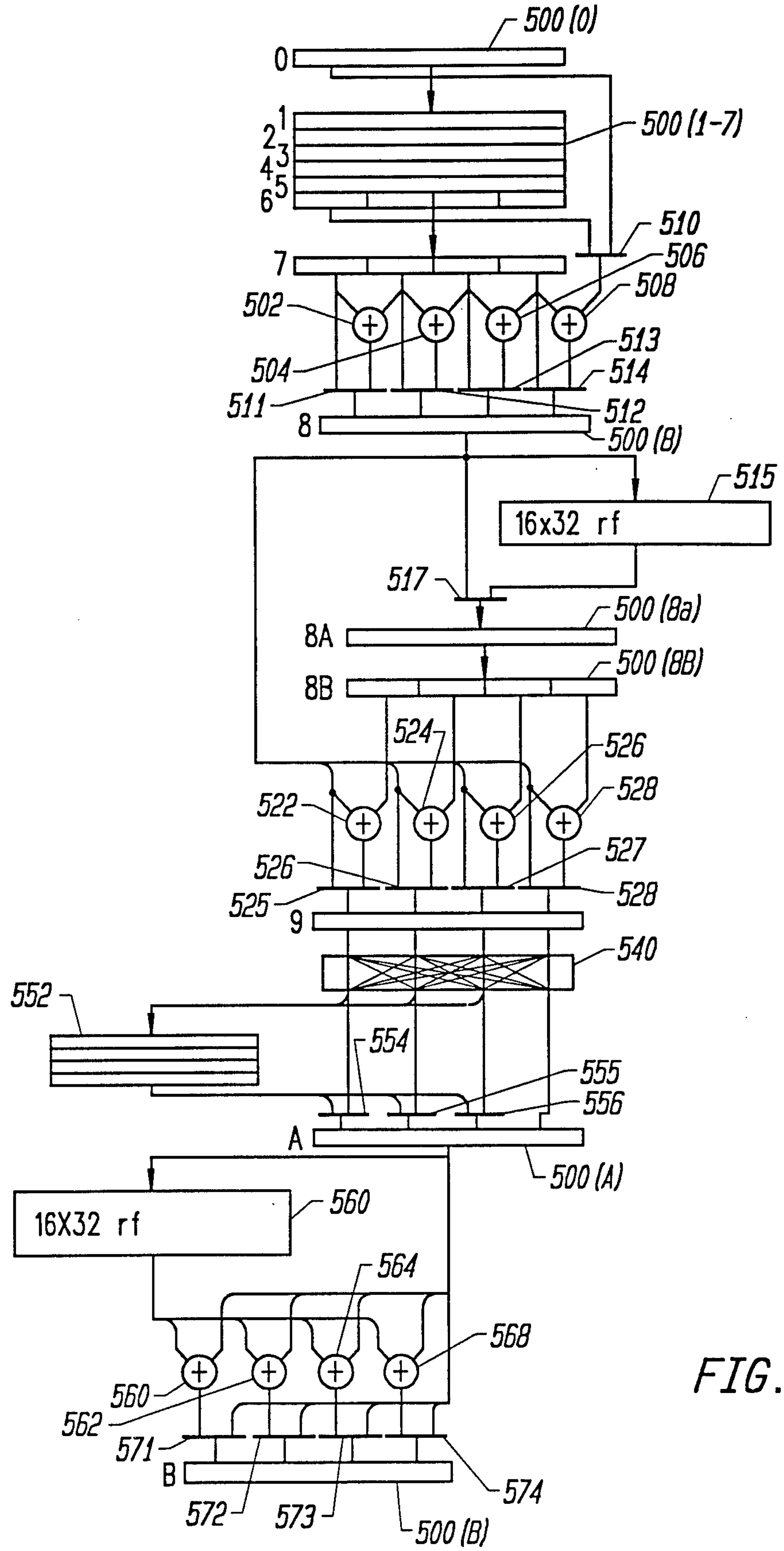

500 (0)
0
1
2
3
4
5
6
500 (1-7)
510
506
7
508
502
513
504
514
511
512
8
500 (8)
515
16x32 rf
517
500 (8a)
8A
500 (8B)
8B
524
526
522
528
526
527
525
528
9
540
552
554
555
556
A
500 (A)
16X32 rf
560
564
568
560
562
571
B
574
572
573
500 (B)

FIG. 10A

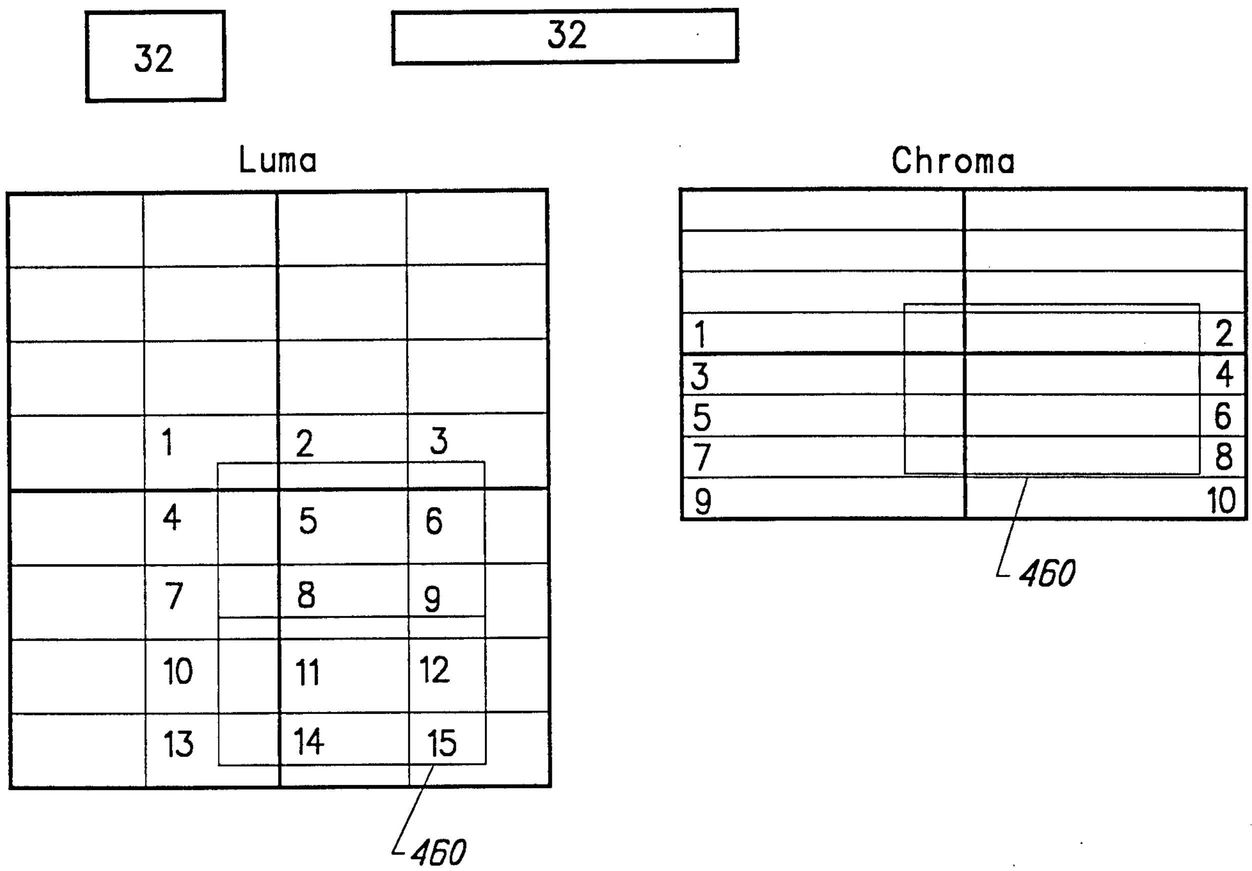

32
32
Luma
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
460
Chroma
1
2
3
4
5
6
7
8
9
10
460

FIG. 10B

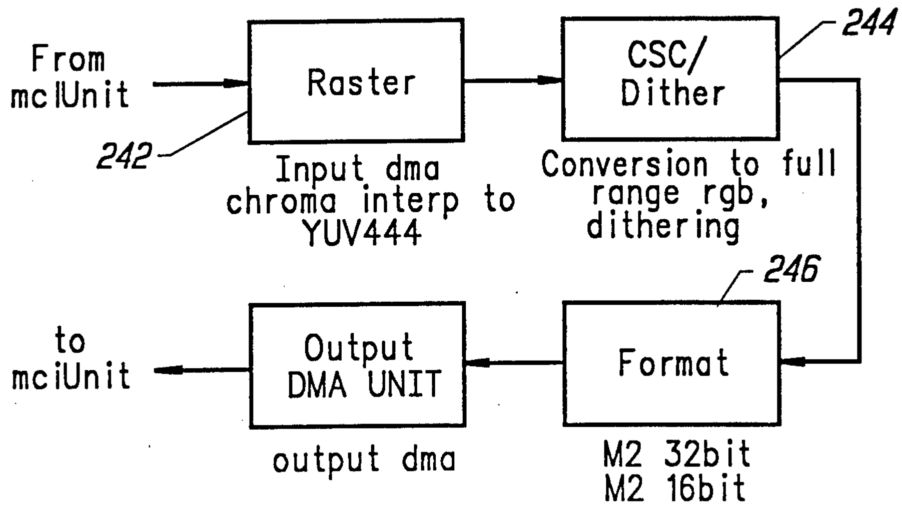


FIG. 11A

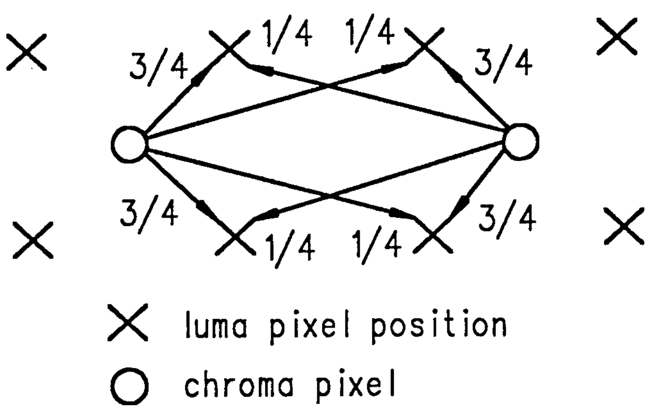


FIG. 11B

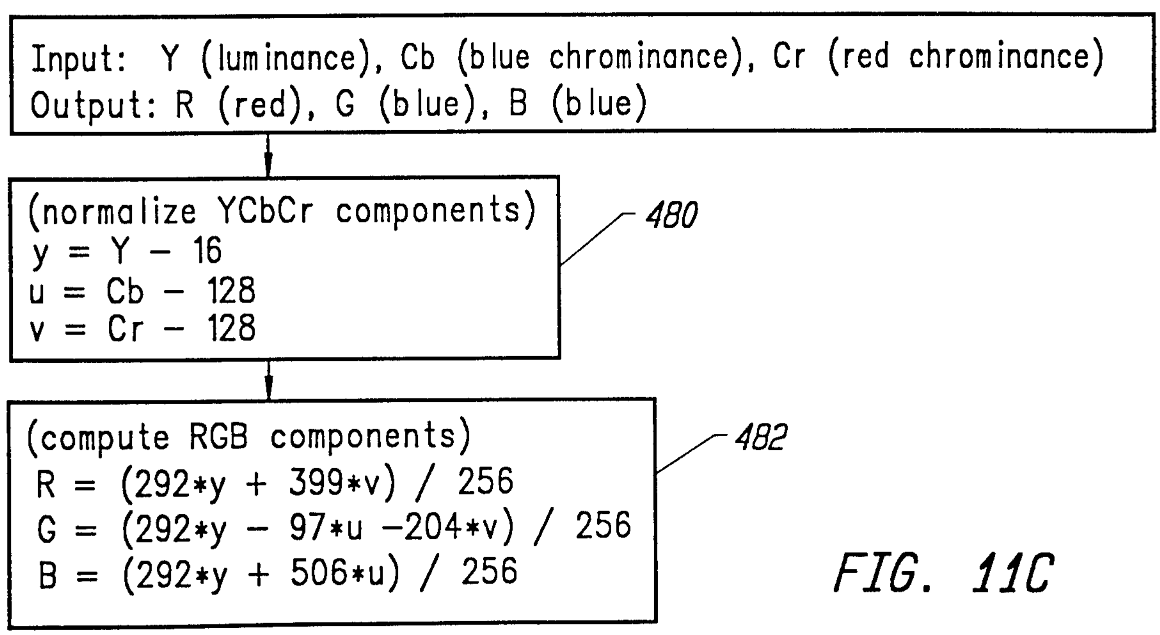


FIG. 11C

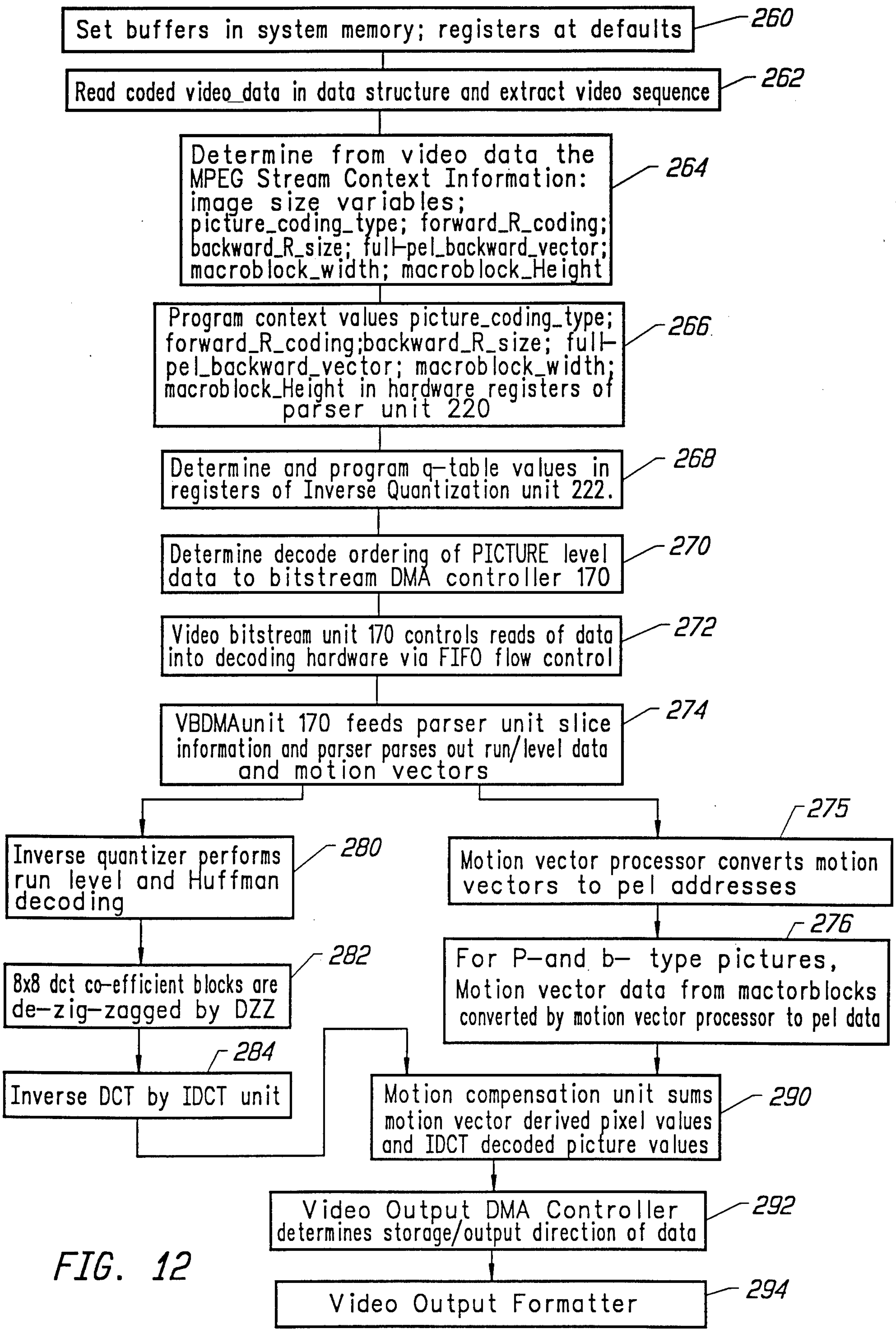

Set buffers in system memory; registers at defaults
260
Read coded video_data in data structure and extract video sequence
262
Determine from video data the MPEG Stream Context Information: image size variables; picture_coding_type; forward_R_coding; backward_R_size; full-pel_backward_vector; macroblock_width; macroblock_Height
264
Program context values picture_coding_type; forward_R_coding;backward_R_size; full-pel_backward_vector; macroblock_width; macroblock_Height in hardware registers of parser unit 220
266
Determine and program q-table values in registers of Inverse Quantization unit 222.
268
Determine decode ordering of PICTURE level data to bitstream DMA controller 170
270
Video bitstream unit 170 controls reads of data into decoding hardware via FIFO flow control
272
VBDMAunit 170 feeds parser unit slice information and parser parses out run/level data and motion vectors
274
Inverse quantizer performs run level and Huffman decoding
280
Motion vector processor converts motion vectors to pel addresses
275
8x8 dct co-efficient blocks are de-zig-zagged by DZZ
282
For P-and b- type pictures, Motion vector data from mactorblocks converted by motion vector processor to pel data
276
Inverse DCT by IDCT unit
284
Motion compensation unit sums motion vector derived pixel values and IDCT decoded picture values
290
Video Output DMA Controller determines storage/output direction of data
292
Video Output Formatter
294

FIG. 12

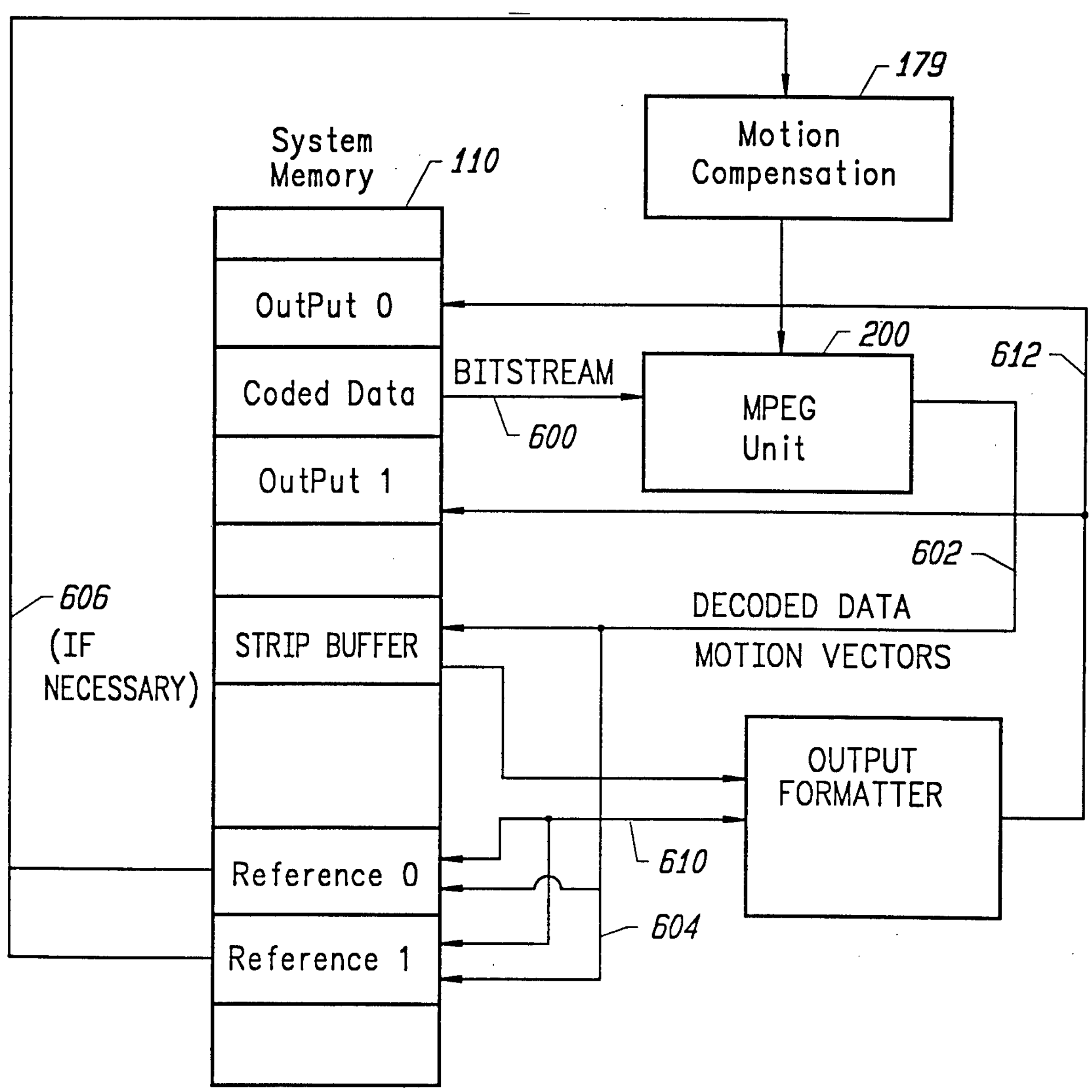

179
Motion Compensation
System Memory
110
OutPut 0
Coded Data
OutPut 1
STRIP BUFFER
Reference 0
Reference 1
BITSTREAM
600
200
MPEG Unit
612
602
DECODED DATA
MOTION VECTORS
606
(IF NECESSARY)
OUTPUT FORMATTER
610
604

FIG. 13

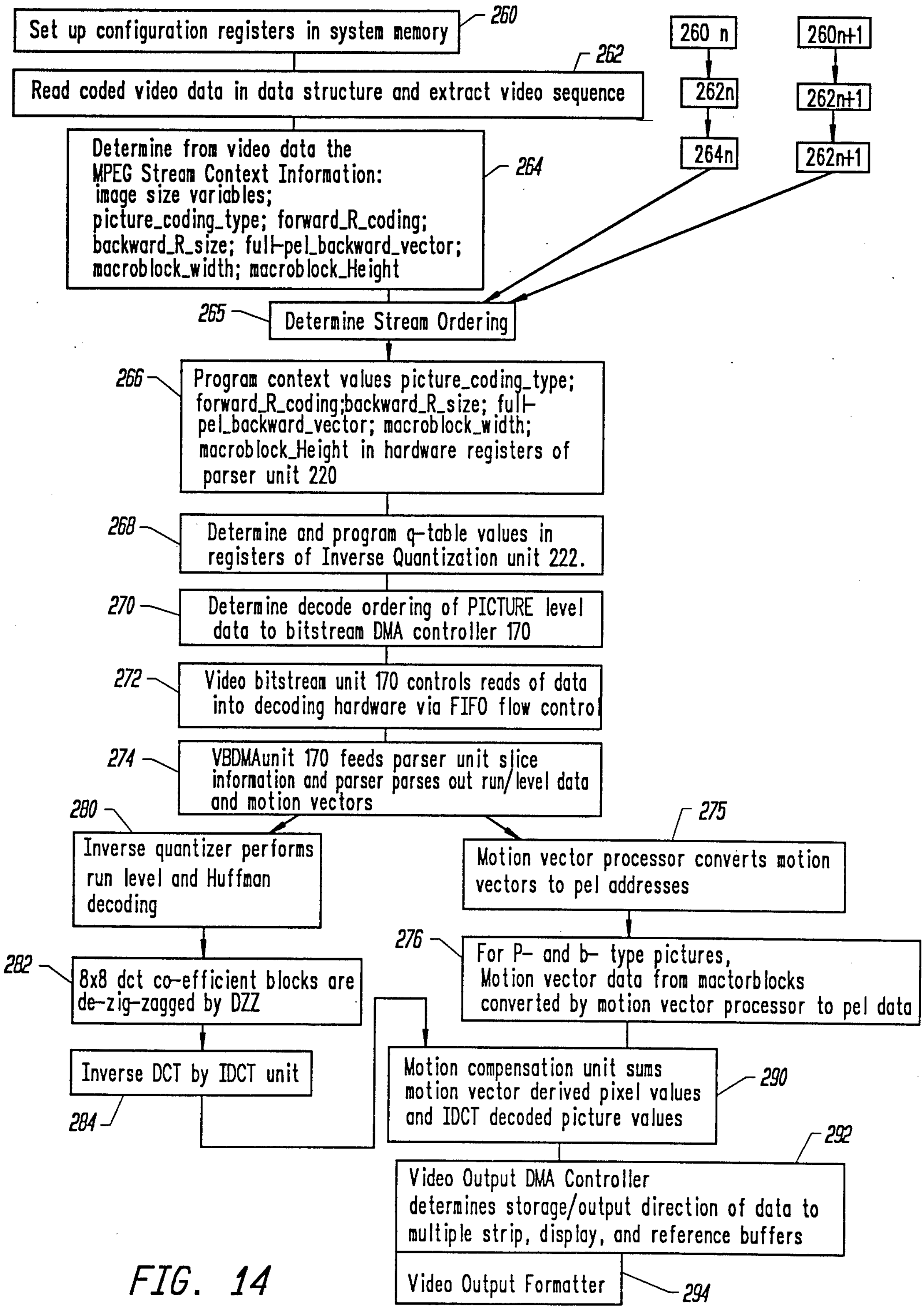

Set up configuration registers in system memory
260
Read coded video data in data structure and extract video sequence
262
Determine from video data the MPEG Stream Context Information: image size variables; picture_coding_type; forward_R_coding; backward_R_size; full-pel_backward_vector; macroblock_width; macroblock_Height
264
260 n
262n
264n
260n+1
262n+1
262n+1
265
Determine Stream Ordering
266
Program context values picture_coding_type; forward_R_coding;backward_R_size; full-pel_backward_vector; macroblock_width; macroblock_Height in hardware registers of parser unit 220
268
Determine and program q-table values in registers of Inverse Quantization unit 222.
270
Determine decode ordering of PICTURE level data to bitstream DMA controller 170
272
Video bitstream unit 170 controls reads of data into decoding hardware via FIFO flow control
274
VBDMAunit 170 feeds parser unit slice information and parser parses out run/level data and motion vectors
280
Inverse quantizer performs run level and Huffman decoding
275
Motion vector processor converts motion vectors to pel addresses
282
8x8 dct co-efficient blocks are de-zig-zagged by DZZ
276
For P- and b- type pictures, Motion vector data from mactorblocks converted by motion vector processor to pel data
Inverse DCT by IDCT unit
284
Motion compensation unit sums motion vector derived pixel values and IDCT decoded picture values
290
292
Video Output DMA Controller determines storage/output direction of data to multiple strip, display, and reference buffers
Video Output Formatter
294

FIG. 14

Bitstream Requirements

Frame Type	I	P	B	B	P	B	B	I	B	B	–
Sequence #	1	4	2	3	7	5	6	10	8	9	–
MPEG out	R0	R1	CSC	CSC	R0	CSC	CSC	R1	CSC	CSC	–
CSC in	MPEG	R0	MPEG	MPEG	R1	MPEG	MPEG	R0	MPEG	MPEG	–
CSC out	D0	D1	D0	D1	D0	D1	D0	D1	D0	D1	–
[Reference 0]	X	1	1	1	1	7	7	7	7	7	–
[Reference 1]	X	X	4	4	4	4	4	4	10	10	–
[Display 0]	X	1	1	2	2	4	4	6	6	8	8
[Display 1]	X	X	1	1	3	3	5	5	7	7	9
Display	X	X	D1	D0	D1	D0	D1	D0	D1	D0	D1
[Display]	x	x	1	2	3	4	5	6	7	8	9

FIG. 15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/06510

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :H04N 7/12
US CL :364/514R
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/514R; 348/403,416,420,426; 382/166,232,236,246,250

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

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

APS, DIALOG

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US, A, 5,379,356 (PURCELL ET AL.) 03 January 1995, col. 7, lines 51-63, cols. 17-18, col. 28, line 28, and the claims.	1, 11, 20, 27-31, 44, 56, 65, 71, 72
Y	US, A, 5,379,351 (FANDRIANTO ET AL.) 03 January 1995, col. 13, lines 20-23.	2-4, 9-10, 21-22, 24-26.

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

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be part of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 JUNE 1996	22 JUL 1996

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	E. Todd Voeltz
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9646

Form PCT/ISA/210 (second sheet)(July 1992)★