EXHIBIT A

US007310374B2

(12) **United States Patent**
Wang et al.

(10) **Patent No.: US 7,310,374 B2**
(45) **Date of Patent: Dec. 18, 2007**

(54) **MACROBLOCK LEVEL ADAPTIVE FRAME/FIELD CODING FOR DIGITAL VIDEO CONTENT**

(75) Inventors: **Limin Wang**, San Diego, CA (US); **Rajeev Gandhi**, San Diego, CA (US); **Krit Panusopone**, San Diego, CA (US); **Ajay Luthra**, San Diego, CA (US)

(73) Assignee: **General Instrument Corporation**, Horsham, PA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 489 days.

(21) Appl. No.: **11/027,265**

(22) Filed: **Dec. 30, 2004**

(65) **Prior Publication Data**

US 2005/0117650 A1 Jun. 2, 2005

Related U.S. Application Data

(62) Division of application No. 10/301,290, filed on Nov. 20, 2002, now Pat. No. 6,980,596.

(60) Provisional application No. 60/398,161, filed on Jul. 23, 2002, provisional application No. 60/395,734, filed on Jul. 12, 2002, provisional application No. 60/333,921, filed on Nov. 27, 2001.

(51) **Int. Cl.**
H04B 1/66 (2006.01)

(52) **U.S. Cl.** **375/240.16**; 375/240.24; 375/240.25; 375/240.13; 375/240.02; 375/240.26; 382/238; 382/233; 382/236; 382/235; 382/239

(58) **Field of Classification Search** 375/240.16, 375/240.24, 240.25, 240.13, 240.02, 240.26; 382/238, 233, 236, 235, 239
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,437,119	A	3/1984	Matsumoto et al.	
5,412,428	A	5/1995	Tahara	
5,504,530	A *	4/1996	Obikane et al.	375/240.14
5,801,778	A	9/1998	Ju	
6,094,225	A	7/2000	Han	
6,192,148	B1	2/2001	Lin	
6,404,813	B1	6/2002	Haskell et al.	

OTHER PUBLICATIONS

"Core Experiment on Interlaced Video Coding", Peter Borgwart, VideoTele.com—A Tektronix Company, Study Group 16, Question 6.

"Adaptive field/frame block coding experiment proposal", Interested Parties for the Study of Interlaced Video Coding with H.26L,Video Coding Experts Group, Study Group 16.

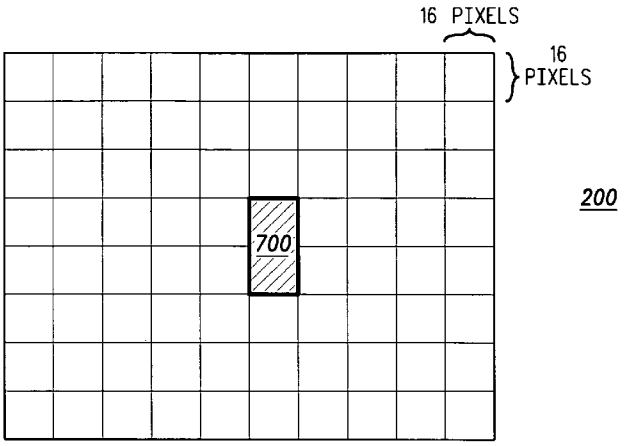
(Continued)

Primary Examiner—Shawn S. An
(74) *Attorney, Agent, or Firm*—Larry T. Cullen

(57) **ABSTRACT**

A method and system of encoding and decoding digital video content. The digital video content comprises a stream of pictures which can each be intra, predicted, or bi-predicted pictures. Each of the pictures comprises macroblocks that can be further divided into smaller blocks. The method entails encoding and decoding each of the smaller blocks in each picture in said stream of pictures in either frame mode or in field mode.

19 Claims, 8 Drawing Sheets



OTHER PUBLICATIONS

P. Borgwardt: "Core Experiment on Interfaced Video Coding VCEG-N85" ITU-Telecommunications Standardization Sector ITU-T Q.6/SG16 Video Coding Expert Group (VCEG) 24-27 Sep. 2001, pp. 1-10, XP002257037 Santa Barbara, CA USA

"H.26L Test Model Long Term Number 8 (TML-8) Drafto" ITU-T Telecommunicatoin Standardization Sector of ITU, Geneva, CH, Apr. 2, 2001, pp. 1-54, XP001089814 p. 9, paragraph 1 - p. 10, paragraph 6.1.2.2.1 p p. 40.

P. Borgwardt: "Handling Interlaced Video in H.26L VCEG-N57" ITU-Telecommunications Standardization Sector ITU Q.6/SG16 Video Coding Expert Group (VCEG), Sep. 24-27, 2001, pp. 1-3, XP002257142 Santa Barbara, CA USA

M. Gallant et al: High Rate, High Resolution Video Using H.26L VCEG-N84 ITU-Telecommunications Standardization Secotr ITY Q.6/SG16 Video Coding Expert GRoup (VCEG), Sep. 24-27, 2001, pp. 1-7, XP002257143 Santa Barbara, CA USA; p. 1; p. 3.

"Adaptive Field/From Block Coding Experiment Proposal". Interested Parties for the Study of Interfaced Video Coding With H.26L, Video Coding Experts Group, Study Group 16.

* cited by examiner

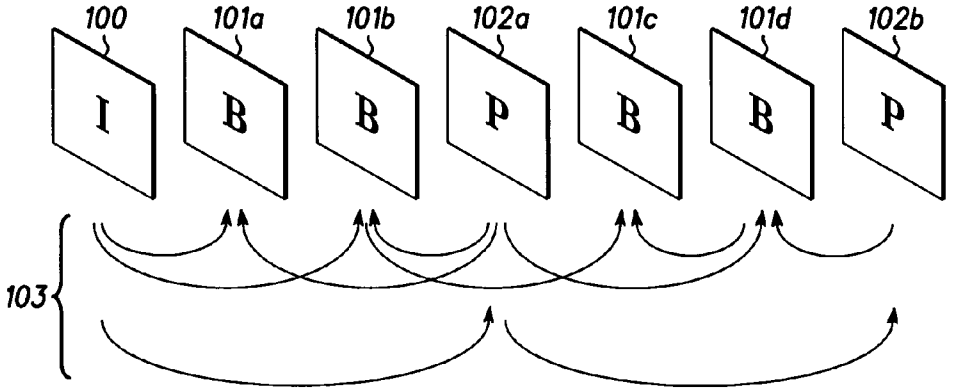


FIG. 1

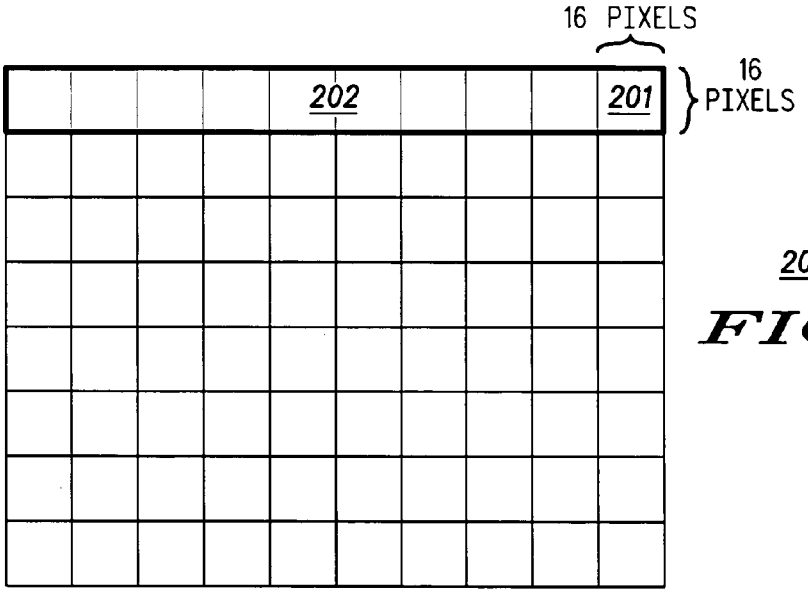


FIG. 2

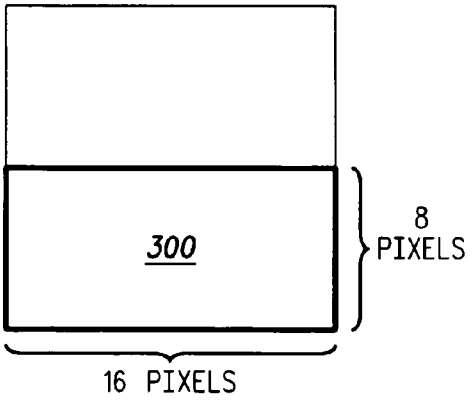


FIG. 3A

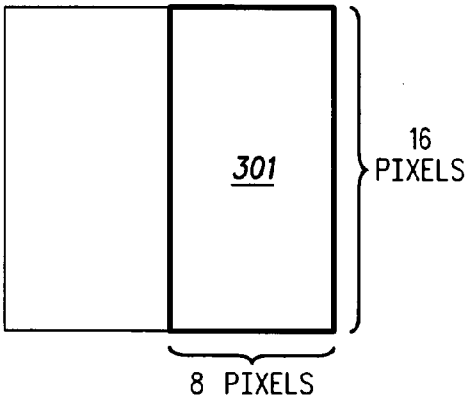


FIG. 3B

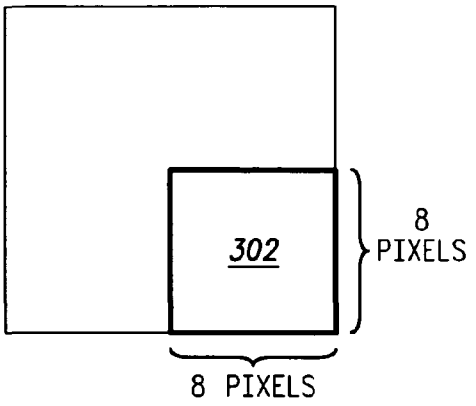


FIG. 3C

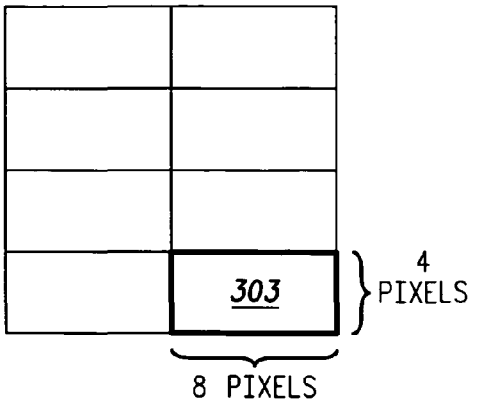


FIG. 3D

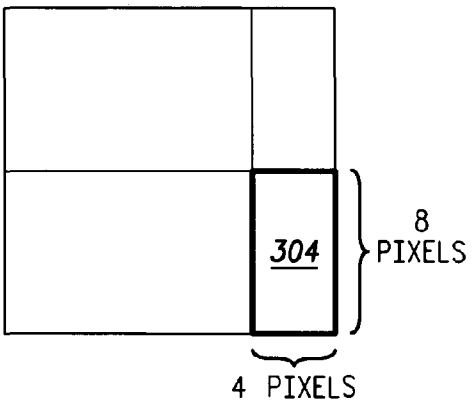


FIG. 3E

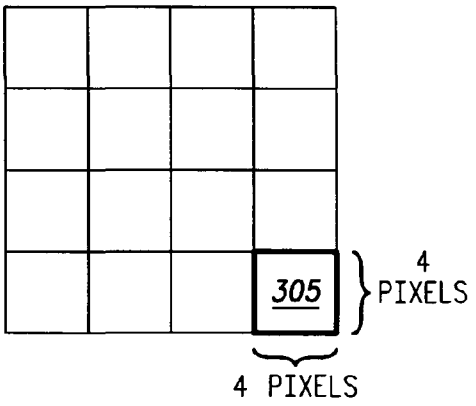


FIG. 3F

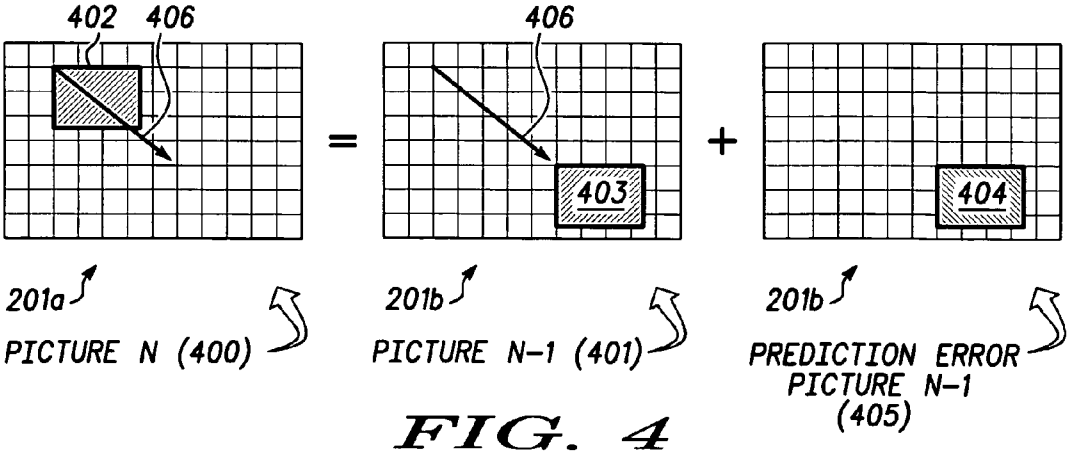
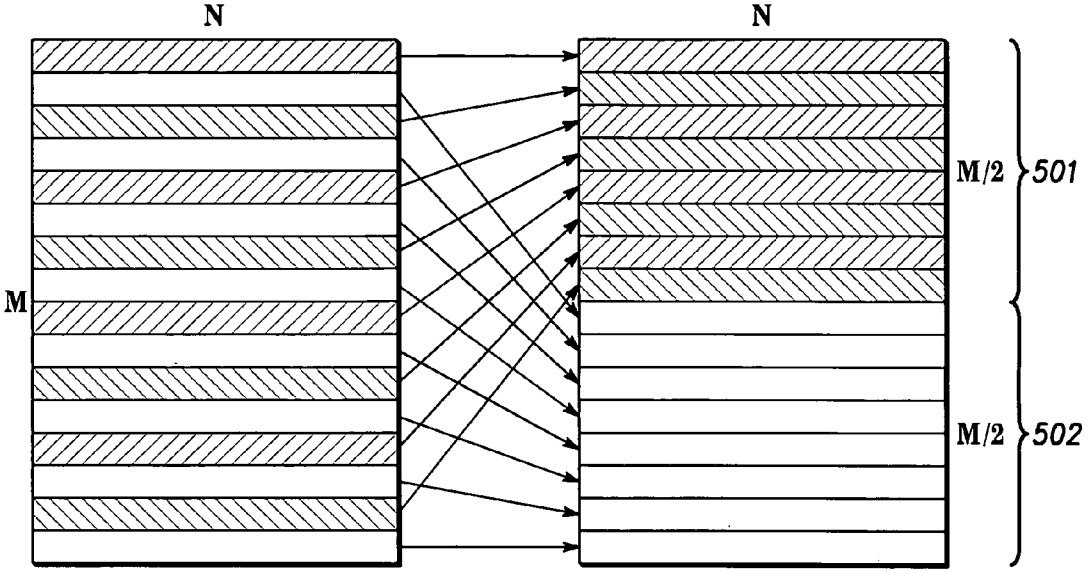


FIG. 4



500 FIG. 5

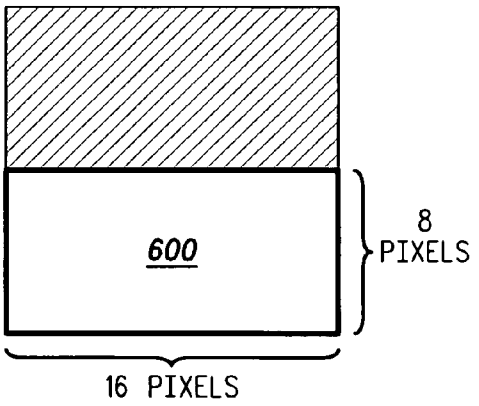


FIG. 6A

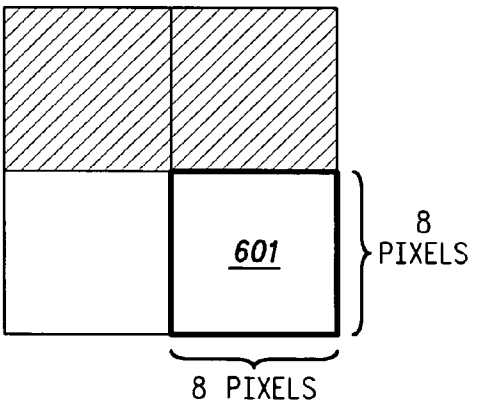


FIG. 6B

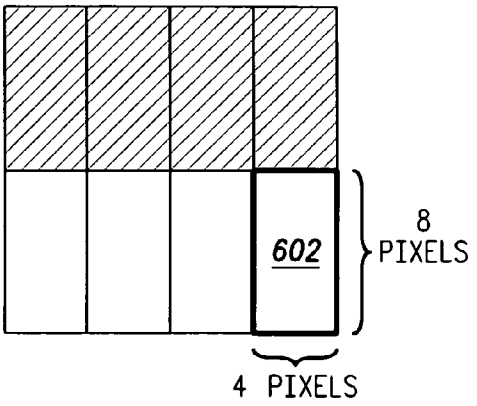


FIG. 6C

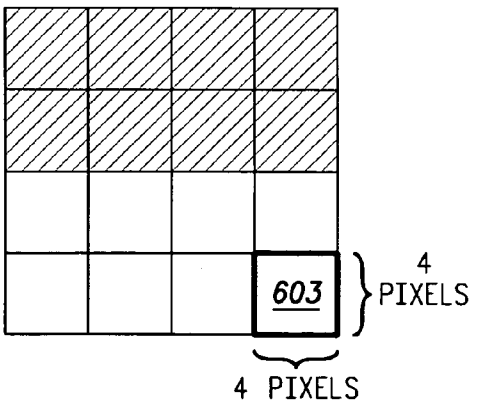


FIG. 6D

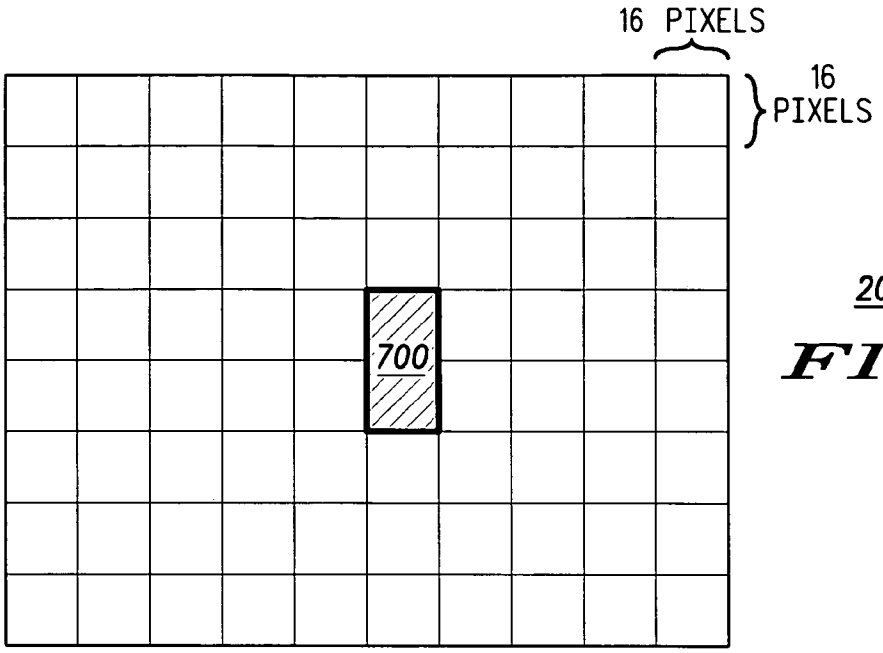


FIG. 7

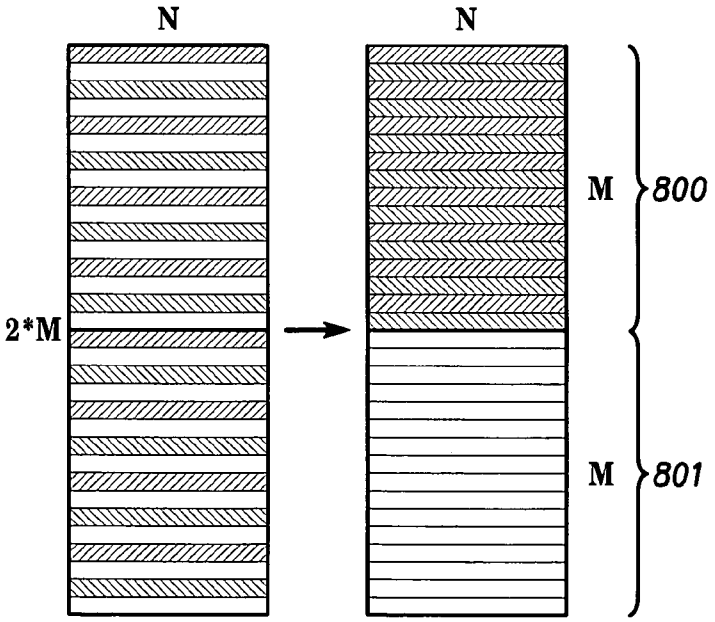


FIG. 8

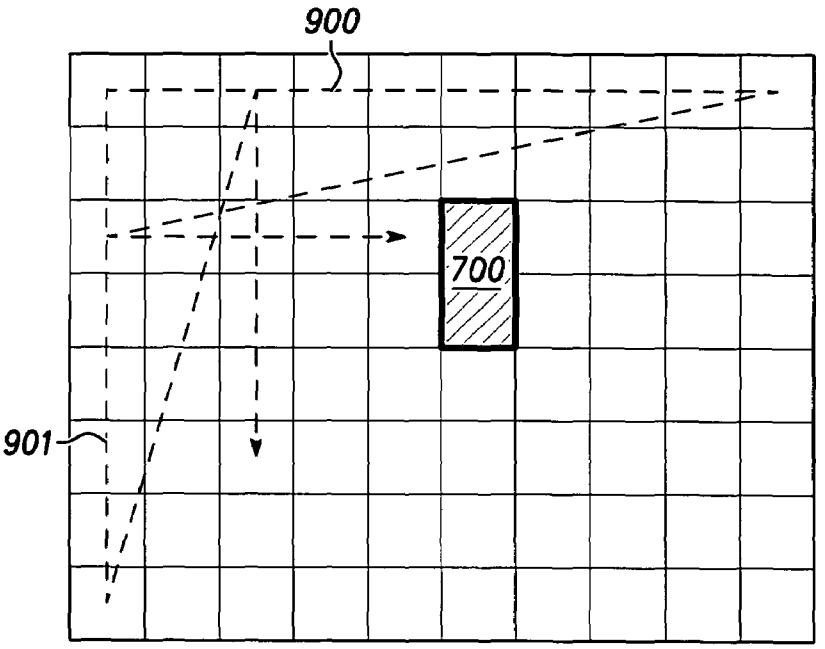


FIG. 9

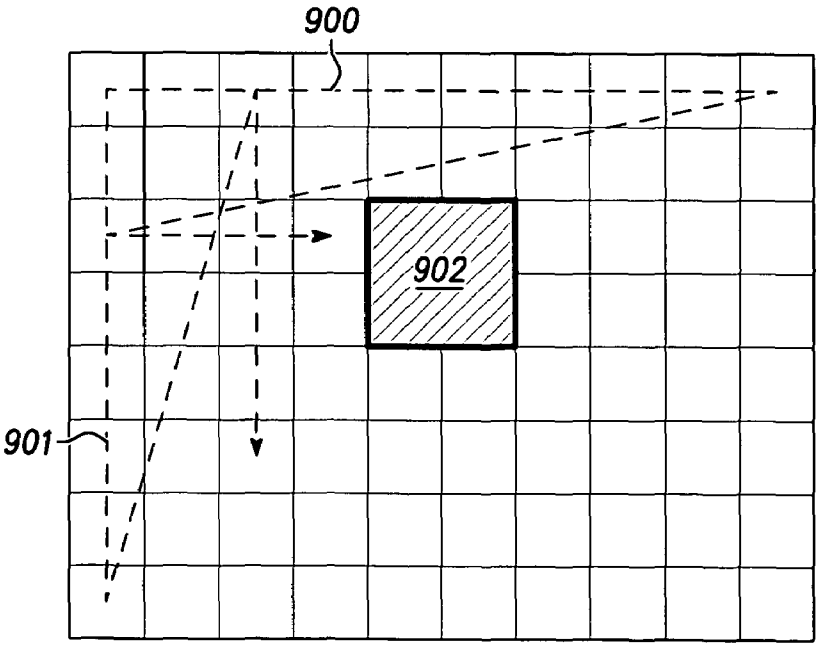


FIG. 10

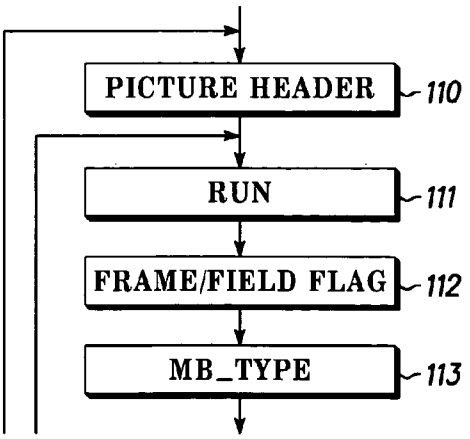


FIG. 11

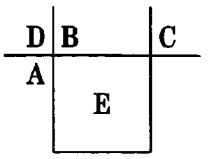


FIG. 12

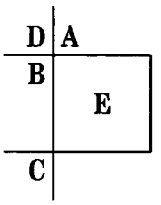


FIG. 13

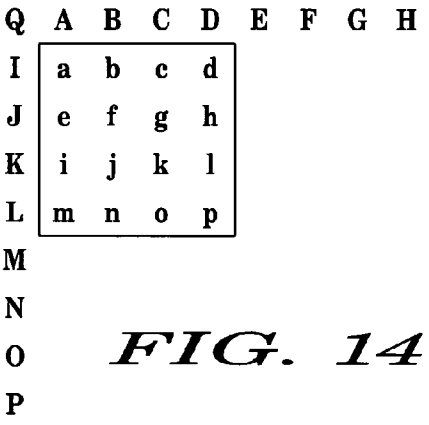


FIG. 14

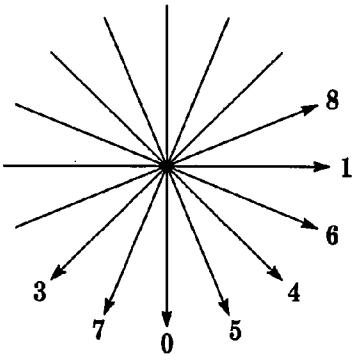


FIG. 15

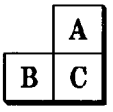


FIG. 16A

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

FIG. 16B

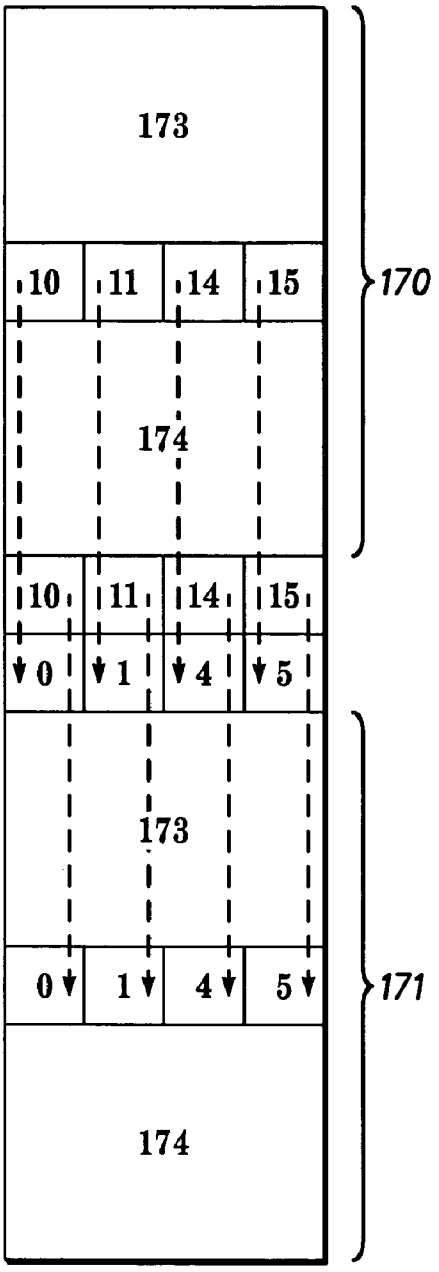


FIG. 17A

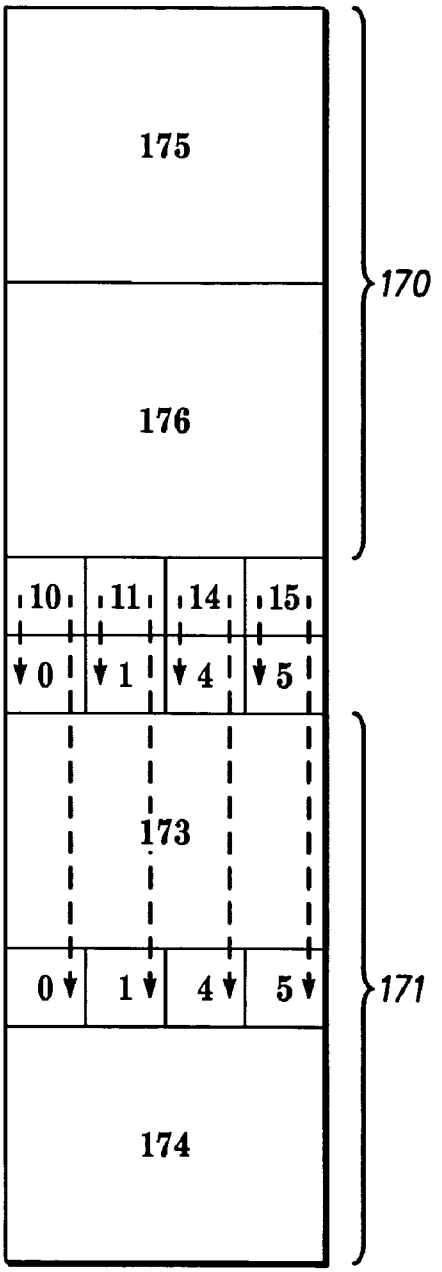


FIG. 17B

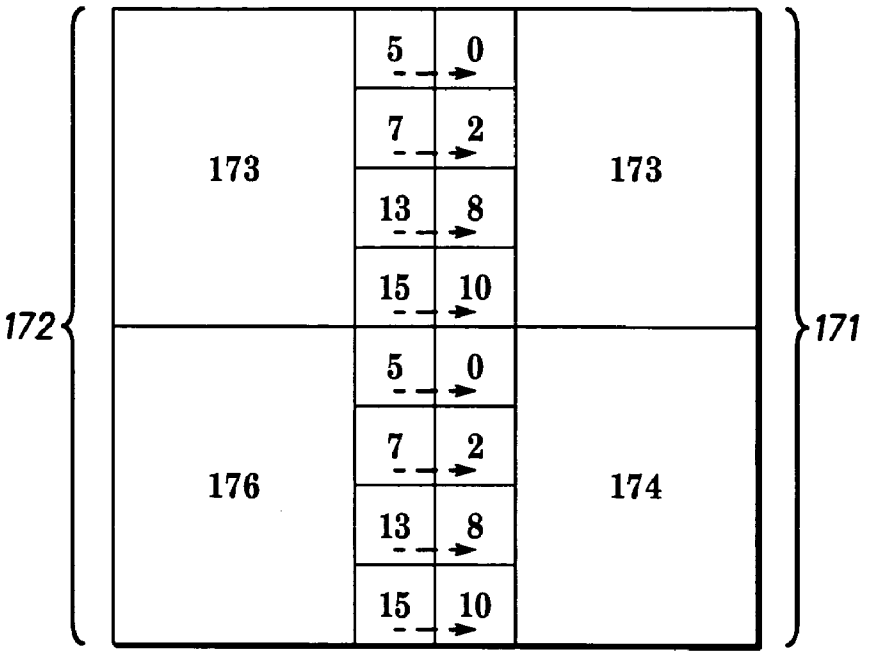


FIG. 17C

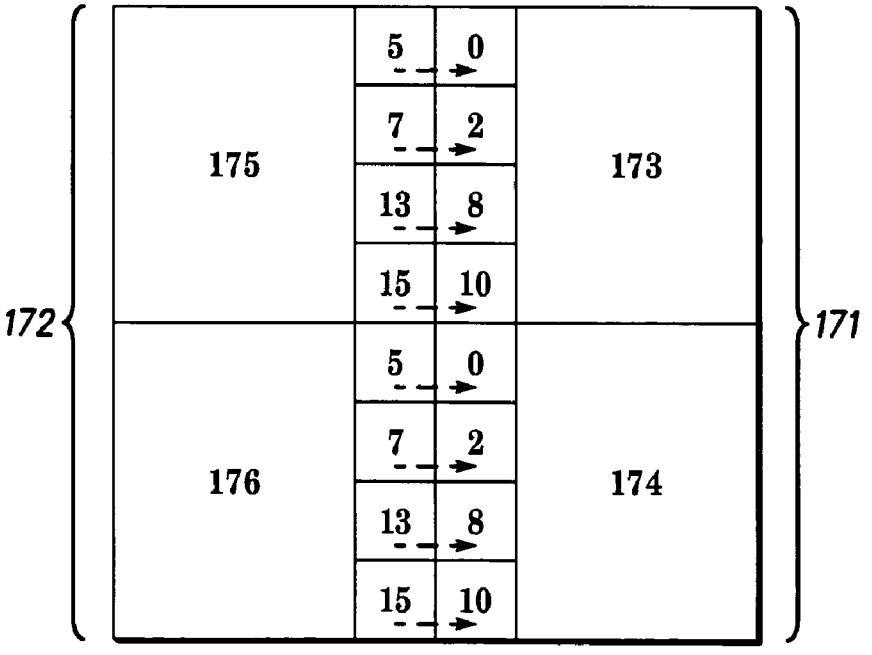


FIG. 17D

MACROBLOCK LEVEL ADAPTIVE FRAME/FIELD CODING FOR DIGITAL VIDEO CONTENT

The present application claims priority under 35 U.S.C. §119(e) from the following previously filed Provisional Patent Applications: Ser. No. 60/333,921, filed Nov. 27, 2001; Ser. No. 60/395,734, filed Jul. 12, 2002; Ser. No. 60/398,161, filed Jul. 23, 2002; all of which are herein incorporated by reference. This application is also a Divisional of U.S. patent application Ser. No. 10/301,290 filed on Nov. 20, 2002 now U.S. Pat. No. 6,980,596, which is herein incorporated by reference.

TECHNICAL FIELD

The present invention relates to encoding and decoding of digital video content. More specifically, the present invention relates to frame mode and field mode encoding of digital video content at a macroblock level as used in the MPEG-4 Part 10 AVC/H.264 standard video coding standard.

BACKGROUND

Video compression is used in many current and emerging products. It is at the heart of digital television set-top boxes (STBs), digital satellite systems (DSSs), high definition television (HDTV) decoders, digital versatile disk (DVD) players, video conferencing, Internet video and multimedia content, and other digital video applications. Without video compression, digital video content can be extremely large, making it difficult or even impossible for the digital video content to be efficiently stored, transmitted, or viewed.

The digital video content comprises a stream of pictures that can be displayed as an image on a television receiver, computer monitor, or some other electronic device capable of displaying digital video content. A picture that is displayed in time before a particular picture is in the "backward direction" in relation to the particular picture. Likewise, a picture that is displayed in time after a particular picture is in the "forward direction" in relation to the particular picture.

Video compression is accomplished in a video encoding, or coding, process in which each picture is encoded as either a frame or as two fields. Each frame comprises a number of lines of spatial information. For example, a typical frame contains 480 horizontal lines. Each field contains half the number of lines in the frame. For example, if the frame comprises 480 horizontal lines, each field comprises 240 horizontal lines. In a typical configuration, one of the fields comprises the odd numbered lines in the frame and the other field comprises the even numbered lines in the frame. The field that comprises the odd numbered lines will be referred to as the "top" field hereafter and in the appended claims, unless otherwise specifically denoted. Likewise, the field that comprises the even numbered lines will be referred to as the "bottom" field hereafter and in the appended claims, unless otherwise specifically denoted. The two fields can be interlaced together to form an interlaced frame.

The general idea behind video coding is to remove data from the digital video content that is "non-essential." The decreased amount of data then requires less bandwidth for broadcast or transmission. After the compressed video data has been transmitted, it must be decoded, or decompressed. In this process, the transmitted video data is processed to generate approximation data that is substituted into the video data to replace the "non-essential" data that was removed in the coding process.

Video coding transforms the digital video content into a compressed form that can be stored using less space and transmitted using less bandwidth than uncompressed digital video content. It does so by taking advantage of temporal and spatial redundancies in the pictures of the video content. The digital video content can be stored in a storage medium such as a hard drive, DVD, or some other non-volatile storage unit.

There are numerous video coding methods that compress the digital video content. Consequently, video coding standards have been developed to standardize the various video coding methods so that the compressed digital video content is rendered in formats that a majority of video encoders and decoders can recognize. For example, the Motion Picture Experts Group (MPEG) and International Telecommunication Union (ITU-T) have developed video coding standards that are in wide use. Examples of these standards include the MPEG-1, MPEG-2, MPEG-4, ITU-T H261, and ITU-T H263 standards.

Most modern video coding standards, such as those developed by MPEG and ITU-T, are based in part on a temporal prediction with motion compensation (MC) algorithm. Temporal prediction with motion compensation is used to remove temporal redundancy between successive pictures in a digital video broadcast.

The temporal prediction with motion compensation algorithm typically utilizes one or two reference pictures to encode a particular picture. A reference picture is a picture that has already been encoded. By comparing the particular picture that is to be encoded with one of the reference pictures, the temporal prediction with motion compensation algorithm can take advantage of the temporal redundancy that exists between the reference picture and the particular picture that is to be encoded and encode the picture with a higher amount of compression than if the picture were encoded without using the temporal prediction with motion compensation algorithm. One of the reference pictures may be in the backward direction in relation to the particular picture that is to be encoded. The other reference picture is in the forward direction in relation to the particular picture that is to be encoded.

However, as the demand for higher resolutions, more complex graphical content, and faster transmission time increases, so does the need for better video compression methods. To this end, a new video coding standard is currently being developed jointly by ISO and ITU-T. This new video coding standard is called the MPEG-4 Advanced Video Coding (AVC)/H.264 standard.

SUMMARY OF THE INVENTION

In one of many possible embodiments, the present invention provides a method of encoding, decoding, and bitstream generation of digital video content. The digital video content comprises a stream of pictures which can each be intra, predicted, or bi-predicted pictures. Each of the pictures comprises macroblocks that can be further divided into smaller blocks. The method entails encoding and decoding each of the macroblocks in each picture in said stream of pictures in either frame mode or in field mode.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings illustrate various embodiments of the present invention and are a part of the specification. Together with the following description, the drawings demonstrate and explain the principles of the present

invention. The illustrated embodiments are examples of the present invention and do not limit the scope of the invention.

FIG. **1** illustrates an exemplary sequence of three types of pictures that can be used to implement the present invention, as defined by an exemplary video coding standard such as the MPEG-4 Part 10 AVC/H.264 standard.

FIG. **2** shows that each picture is preferably divided into slices containing macroblocks according to an embodiment of the present invention.

FIG. **3***a* shows that a macroblock can be further divided into a block size of 16 by 8 pixels according to an embodiment of the present invention.

FIG. **3***b* shows that a macroblock can be further divided into a block size of 8 by 16 pixels according to an embodiment of the present invention.

FIG. **3***c* shows that a macroblock can be further divided into a block size of 8 by 8 pixels according to an embodiment of the present invention.

FIG. **3***d* shows that a macroblock can be further divided into a block size of 8 by 4 pixels according to an embodiment of the present invention.

FIG. **3***e* shows that a macroblock can be further divided into a block size of 4 by 8 pixels according to an embodiment of the present invention.

FIG. **3***f* shows that a macroblock can be further divided into a block size of 4 by 4 pixels according to an embodiment of the present invention.

FIG. **4** shows a picture construction example using temporal prediction with motion compensation that illustrates an embodiment of the present invention.

FIG. **5** shows that a macroblock is split into a top field and a bottom field if it is to be encoded in field mode.

FIG. **6***a* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 16 by 8 pixels according to an embodiment of the present invention.

FIG. **6***b* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 8 by 8 pixels according to an embodiment of the present invention.

FIG. **6***c* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 4 by 8 pixels according to an embodiment of the present invention.

FIG. **6***d* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 4 by 4 pixels according to an embodiment of the present invention.

FIG. **7** illustrates an exemplary pair of macroblocks that can be used in AFF coding on a pair of macroblocks according to an embodiment of the present invention.

FIG. **8** shows that a pair of macroblocks that is to be encoded in field mode is first split into one top field 16 by 16 pixel block and one bottom field 16 by 16 pixel block.

FIG. **9** shows two possible scanning paths in AFF coding of pairs of macroblocks.

FIG. **10** illustrates another embodiment of the present invention which extends the concept of AFF coding on a pair of macroblocks to AFF coding to a group of four or more neighboring macroblocks.

FIG. **11** shows some of the information included in the bitstream which contains information pertinent to each macroblock within a stream.

FIG. **12** shows a block that is to be encoded and its neighboring blocks and will be used to explain various preferable methods of calculating the PMV of a block in a macroblock.

FIG. **13** shows an alternate definition of neighboring blocks if the scanning path is a vertical scanning path.

FIG. **14** shows that each pixel value is predicted from neighboring blocks' pixel values according to an embodiment of the present invention.

FIG. **15** shows different prediction directions for intra_4×4 coding.

FIGS. **16***a-b* illustrate that the chosen intra-prediction mode (intra_pred_mode) of a 4 by 4 pixel block is highly correlated with the prediction modes of adjacent blocks.

FIGS. **17***a-d* show neighboring blocks definitions in relation to a current macroblock pair that is to be encoded.

Throughout the drawings, identical reference numbers designate similar, but not necessarily identical, elements.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

The present invention provides a method of adaptive frame/field (AFF) coding of digital video content comprising a stream of pictures or slices of a picture at a macroblock level. The present invention extends the concept of picture level AFF to macroblocks. In AFF coding at a picture level, each picture in a stream of pictures that is to be encoded is encoded in either frame mode or in field mode, regardless of the frame or field coding mode of other pictures that are to be coded. If a picture is encoded in frame mode, the two fields that make up an interlaced frame are coded jointly. Conversely, if a picture is encoded in field mode, the two fields that make up an interlaced frame are coded separately. The encoder determines which type of coding, frame mode coding or field mode coding, is more advantageous for each picture and chooses that type of encoding for the picture. The exact method of choosing between frame mode and field mode is not critical to the present invention and will not be detailed herein.

As noted above, the MPEG-4 Part 10 AVC/H.264 standard is a new standard for encoding and compressing digital video content. The documents establishing the MPEG-4 Part 10 AVC/H.264 standard are hereby incorporated by reference, including "Joint Final Committee Draft (JFCD) of Joint Video Specification" issued by the Joint Video Team (JVT) on Aug. 10, 2002. (ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC). The JVT consists of experts from ISO or MPEG and ITU-T. Due to the public nature of the MPEG-4 Part 10 AVC/H.264 standard, the present specification will not attempt to document all the existing aspects of MPEG-4 Part 10 AVC/H.264 video coding, relying instead on the incorporated specifications of the standard.

Although this method of AFF encoding is compatible with and will be explained using the MPEG-4 Part 10 AVC/H.264 standard guidelines, it can be modified and used as best serves a particular standard or application.

Using the drawings, the preferred embodiments of the present invention will now be explained.

FIG. **1** illustrates an exemplary sequence of three types of pictures that can be used to implement the present invention, as defined by an exemplary video coding standard such as the MPEG-4 Part 10 AVC/H.264 standard. As previously mentioned, the encoder encodes the pictures and the decoder decodes the pictures. The encoder or decoder can be a processor, application specific integrated circuit (ASIC), field programmable gate array (FPGA), coder/decoder (CODEC), digital signal processor (DSP), or some other electronic device that is capable of encoding the stream of pictures. However, as used hereafter and in the appended claims, unless otherwise specifically denoted, the term "encoder" will be used to refer expansively to all electronic devices that encode digital video content comprising a

stream of pictures. The term "decoder" will be used to refer expansively to all electronic devices that decode digital video content comprising a stream of pictures.

As shown in FIG. **1**, there are preferably three types of pictures that can be used in the video coding method. Three types of pictures are defined to support random access to stored digital video content while exploring the maximum redundancy reduction using temporal prediction with motion compensation. The three types of pictures are intra (I) pictures (**100**), predicted (P) pictures (**102***a,b*), and bi-predicted (B) pictures (**101***a-d*). An I picture (**100**) provides an access point for random access to stored digital video content and can be encoded only with slight compression. Intra pictures (**100**) are encoded without referring to reference pictures.

A predicted picture (**102***a,b*) is encoded using an I, P, or B picture that has already been encoded as a reference picture. The reference picture can be in either the forward or backward temporal direction in relation to the P picture that is being encoded. The predicted pictures (**102***a,b*) can be encoded with more compression than the intra pictures (**100**).

A bi-predicted picture (**101***a-d*) is encoded using two temporal reference pictures: a forward reference picture and a backward reference picture. The forward reference picture is sometimes called a past reference picture and the backward reference picture is sometimes called a future reference picture. An embodiment of the present invention is that the forward reference picture and backward reference picture can be in the same temporal direction in relation to the B picture that is being encoded. Bi-predicted pictures (**101***a-d*) can be encoded with the most compression out of the three picture types.

Reference relationships (**103**) between the three picture types are illustrated in FIG. **1**. For example, the P picture (**102***a*) can be encoded using the encoded I picture (**100**) as its reference picture. The B pictures (**101***a-d*) can be encoded using the encoded I picture (**100**) or the encoded P picture (**102***a*) as its reference pictures, as shown in FIG. **1**. Under the principles of an embodiment of the present invention, encoded B pictures (**101***a-d*) can also be used as reference pictures for other B pictures that are to be encoded. For example, the B picture (**10***c*) of FIG. **1** is shown with two other B pictures (**101***b* and **101***d*) as its reference pictures.

The number and particular order of the I (**100**), B (**101***a-d*), and P (**102***a,b*) pictures shown in FIG. **1** are given as an exemplary configuration of pictures, but are not necessary to implement the present invention. Any number of I, B, and P pictures can be used in any order to best serve a particular application. The MPEG-4 Part 10 AVC/H.264 standard does not impose any limit to the number of B pictures between two reference pictures nor does it limit the number of pictures between two I pictures.

FIG. **2** shows that each picture (**200**) is preferably divided into slices (**202**). A slice (**202**) comprises a group of macroblocks (**201**). A macroblock (**201**) is a rectangular group of pixels. As shown in FIG. **2**, a preferable macroblock (**201**) size is 16 by 16 pixels.

FIGS. **3***a-f* show that a macroblock can be further divided into smaller sized blocks. For example, as shown in FIGS. **3***a-f*, a macroblock can be further divided into block sizes of 16 by 8 pixels (FIG. **3***a*; **300**), 8 by 16 pixels (FIG. **3***b*; **301**), 8 by 8 pixels (FIG. **3***c*; **302**), 8 by 4 pixels (FIG. **3***d*; **303**), 4 by 8 pixels (FIG. **3***e*; **304**), or 4 by 4 pixels (FIG. **3***f*; **305**). These smaller block sizes are preferable in some applications that use the temporal prediction with motion compensation algorithm.

FIG. **4** shows a picture construction example using temporal prediction with motion compensation that illustrates an embodiment of the present invention. Temporal prediction with motion compensation assumes that a current picture, picture N (**400**), can be locally modeled as a translation of another picture, picture N−1 (**401**). The picture N−1 (**401**) is the reference picture for the encoding of picture N (**400**) and can be in the forward or backwards temporal direction in relation to picture N (**400**).

As shown in FIG. **4**, each picture is preferably divided into slices containing macroblocks (**201***a,b*). The picture N−1 (**401**) contains an image (**403**) that is to be shown in picture N (**400**). The image (**403**) will be in a different temporal position in picture N (**402**) than it is in picture N−1 (**401**), as shown in FIG. **4**. The image content of each macroblock (**201***b*) of picture N (**400**) is predicted from the image content of each corresponding macroblock (**201***a*) of picture N−1 (**401**) by estimating the required amount of temporal motion of the image content of each macroblock (**201***a*) of picture N−1 (**401**) for the image (**403**) to move to its new temporal position (**402**) in picture N (**400**). Instead of the original image (**402**) being encoded, the difference (**404**) between the image (**402**) and its prediction (**403**) is actually encoded and transmitted.

For each image (**402**) in picture N (**400**), the temporal prediction can often be described by motion vectors that represent the amount of temporal motion required for the image (**403**) to move to a new temporal position in the picture N (**402**). The motion vectors (**406**) used for the temporal prediction with motion compensation need to be encoded and transmitted.

FIG. **4** shows that the image (**402**) in picture N (**400**) can be represented by the difference (**404**) between the image and its prediction and the associated motion vectors (**406**). The exact method of encoding using the motion vectors can vary as best serves a particular application and can be easily implemented by someone who is skilled in the art.

To understand macroblock level AFF coding, a brief overview of picture level AFF coding of a stream of pictures will now be given. A frame of an interlaced sequence contains two fields, the top field and the bottom field, which are interleaved and separated in time by a field period. The field period is half the time of a frame period. In picture level AFF coding, the two fields of an interlaced frame can be coded jointly or separately. If they are coded jointly, frame mode coding is used. Conversely, if the two fields are coded separately, field mode coding is used.

Fixed frame/field coding, on the other hand, codes all the pictures in a stream of pictures in one mode only. That mode can be frame mode or it can be field mode. Picture level AFF is preferable to fixed frame/field coding in many applications because it allows the encoder to chose which mode, frame mode or field mode, to encode each picture in the stream of pictures based on the contents of the digital video material. AFF coding results in better compression than does fixed frame/field coding in many applications.

An embodiment of the present invention is that AFF coding can be performed on smaller portions of a picture. This small portion can be a macroblock, a pair of macroblocks, or a group of macroblocks. Each macroblock, pair of macroblocks, or group of macroblocks or slice is encoded in frame mode or in field mode, regardless of how the other macroblocks in the picture are encoded. AFF coding in each of the three cases will be described in detail below.

In the first case, AFF coding is performed on a single macroblock. If the macroblock is to be encoded in frame mode, the two fields in the macroblock are encoded jointly.

Once encoded as a frame, the macroblock can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if the macroblock is to be encoded in field mode, the macroblock (**500**) is split into a top field (**501**) and a bottom field (**502**), as shown in FIG. **5**. The two fields are then coded separately. In FIG. **5**, the macroblock has M rows of pixels and N columns of pixels. A preferable value of N and M is 16, making the macroblock (**500**) a 16 by 16 pixel macroblock. As shown in FIG. **5**, every other row of pixels is shaded. The shaded areas represent the rows of pixels in the top field of the macroblock (**500**) and the unshaded areas represent the rows of pixels in the bottom field of the macroblock (**500**).

As shown in FIGS. **6***a-d*, a macroblock that is encoded in field mode can be divided into four additional blocks. A block is required to have a single parity. The single parity requirement is that a block cannot comprise both top and bottom fields. Rather, it must contain a single parity of field. Thus, as shown in FIGS. **6***a-d*, a field mode macroblock can be divided into blocks of 16 by 8 pixels (FIG. **6***a*; **600**), 8 by 8 pixels (FIG. **6***b*; **601**), 4 by 8 pixels (FIG. **6***c*; **602**), and 4 by 4 pixels (FIG. **6***d*; **603**). FIGS. **6***a-d* shows that each block contains fields of a single parity.

AFF coding on macroblock pairs will now be explained. AFF coding on macroblock pairs will be occasionally referred to as pair based AFF coding. A comparison of the block sizes in FIGS. **6***a-d* and in FIGS. **3***a-f* show that a macroblock encoded in field mode can be divided into fewer block patterns than can a macroblock encoded in frame mode. The block sizes of 16 by 16 pixels, 8 by 16 pixels, and 8 by 4 pixels are not available for a macroblock encoded in field mode because of the single parity requirement. This implies that the performance of single macroblock based AFF may not be good for some sequences or applications that strongly favor field mode coding. In order to guarantee the performance of field mode macroblock coding, it is preferable in some applications for macroblocks that are coded in field mode to have the same block sizes as macroblocks that are coded in frame mode. This can be achieved by performing AFF coding on macroblock pairs instead of on single macroblocks.

FIG. **7** illustrates an exemplary pair of macroblocks (**700**) that can be used in AFF coding on a pair of macroblocks according to an embodiment of the present invention. If the pair of macroblocks (**700**) is to be encoded in frame mode, the pair is coded as two frame-based macroblocks. In each macroblock, the two fields in each of the macroblocks are encoded jointly. Once encoded as frames, the macroblocks can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if the pair of macroblocks (**700**) is to be encoded in field mode, it is first split into one top field 16 by 16 pixel block (**800**) and one bottom field 16 by 16 pixel block (**801**), as shown in FIG. **8**. The two fields are then coded separately. In FIG. **8**, each macroblock in the pair of macroblocks (**700**) has N=16 columns of pixels and M=16 rows of pixels. Thus, the dimensions of the pair of macroblocks (**700**) is 16 by 32 pixels. As shown in FIG. **8**, every other row of pixels is shaded. The shaded areas represent the rows of pixels in the top field of the macroblocks and the unshaded areas represent the rows of pixels in the bottom field of the macroblocks. The top field block (**800**) and the bottom field block (**801**) can now be divided into one of the possible block sizes of FIGS. **3***a-f*.

According to an embodiment of the present invention, in the AFF coding of pairs of macroblocks (**700**), there are two possible scanning paths. A scanning path determines the order in which the pairs of macroblocks of a picture are encoded. FIG. **9** shows the two possible scanning paths in AFF coding of pairs of macroblocks (**700**). One of the scanning paths is a horizontal scanning path (**900**). In the horizontal scanning path (**900**), the macroblock pairs (**700**) of a picture (**200**) are coded from left to right and from top to bottom, as shown in FIG. **9**. The other scanning path is a vertical scanning path (**901**). In the vertical scanning path (**901**), the macroblock pairs (**700**) of a picture (**200**) are coded from top to bottom and from left to right, as shown in FIG. **9**. For frame mode coding, the top macroblock of a macroblock pair (**700**) is coded first, followed by the bottom macroblock. For field mode coding, the top field macroblock of a macroblock pair is coded first followed by the bottom field macroblock.

Another embodiment of the present invention extends the concept of AFF coding on a pair of macroblocks to AFF coding on a group of four or more neighboring macroblocks (**902**), as shown in FIG. **10**. AFF coding on a group of macroblocks will be occasionally referred to as group based AFF coding. The same scanning paths, horizontal (**900**) and vertical (**901**), as are used in the scanning of macroblock pairs are used in the scanning of groups of neighboring macroblocks (**902**). Although the example shown in FIG. **10** shows a group of four macroblocks, the group can be more than four macroblocks.

If the group of macroblocks (**902**) is to be encoded in frame mode, the group coded as four frame-based macroblocks. In each macroblock, the two fields in each of the macroblocks are encoded jointly. Once encoded as frames, the macroblocks can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if a group of four macroblocks (**902**), for example, is to be encoded in field mode, it is first split into one top field 32 by 16 pixel block and one bottom field 32 by 16 pixel block. The two fields are then coded separately. The top field block and the bottom field block can now be divided into macroblocks. Each macroblock is further divided into one of the possible block sizes of FIGS. **3***a-f*. Because this process is similar to that of FIG. **8**, a separate figure is not provided to illustrate this embodiment.

In AFF coding at the macroblock level, a frame/field flag bit is preferably included in a picture's bitstream to indicate which mode, frame mode or field mode, is used in the encoding of each macroblock. The bitstream includes information pertinent to each macroblock within a stream, as shown in FIG. **11**. For example, the bitstream can include a picture header (**110**), run information (**111**), and macroblock type (**113**) information. The frame/field flag (**112**) is preferably included before each macroblock in the bitstream if AFF is performed on each individual macroblock. If the AFF is performed on pairs of macroblocks, the frame/field flag (**112**) is preferably included before each pair of macroblock in the bitstream. Finally, if the AFF is performed on a group of macroblocks, the frame/field flag (**112**) is preferably included before each group of macroblocks in the bitstream. One embodiment is that the frame/field flag (**112**) bit is a 0 if frame mode is to be used and a 1 if field coding is to be used. Another embodiment is that the frame/field flag (**112**) bit is a 1 if frame mode is to be used and a 0 if field coding is to be used.

Another embodiment of the present invention entails a method of determining the size of blocks into which the

encoder divides a macroblock in macroblock level AFF. A preferable, but not exclusive, method for determining the ideal block size is sum absolute difference (SAD) with or without bias or rate distortion (RD) basis. For example, SAD checks the performance of the possible block sizes and chooses the ideal block size based on its results. The exact method of using SAD with or without bias or RD basis can be easily be performed by someone skilled in the art.

According to an embodiment of the present invention, each frame and field based macroblock in macroblock level AFF can be intra coded or inter coded. In intra coding, the macroblock is encoded without temporally referring to other macroblocks. On the other hand, in inter coding, temporal prediction with motion compensation is used to code the macroblocks.

If inter coding is used, a block with a size of 16 by 16 pixels, 16 by 8 pixels, 8 by 16 pixels, or 8 by 8 pixels can have its own reference pictures. The block can either be a frame or field based macroblock. The MPEG-4 Part 10 AVC/H.264 standard allows multiple reference pictures instead of just two reference pictures. The use of multiple reference pictures improves the performance of the temporal prediction with motion compensation algorithm by allowing the encoder to find a block in the reference picture that most closely matches the block that is to be encoded. By using the block in the reference picture in the coding process that most closely matches the block that is to be encoded, the greatest amount of compression is possible in the encoding of the picture. The reference pictures are stored in frame and field buffers and are assigned reference frame numbers and reference field numbers based on the temporal distance they are away from the current picture that is being encoded. The closer the reference picture is to the current picture that is being stored, the more likely the reference picture will be selected. For field mode coding, the reference pictures for a block can be any top or bottom field of any of the reference pictures in the reference frame or field buffers.

Each block in a frame or field based macroblock can have its own motion vectors. The motion vectors are spatially predictive coded. According to an embodiment of the present invention, in inter coding, prediction motion vectors (PMV) are also calculated for each block. The algebraic difference between a block's PMVs and its associated motion vectors is then calculated and encoded. This generates the compressed bits for motion vectors.

FIG. **12** will be used to explain various preferable methods of calculating the PMV of a block in a macroblock. A current block, E, in FIG. **12** is to be inter coded as well as its neighboring blocks A, B, C, and D. E will refer hereafter to a current block and A, B, C, and D will refer hereafter to E's neighboring blocks, unless otherwise denoted. Block E's PMV is derived from the motion vectors of its neighboring blocks. These neighboring blocks in the example of FIG. **12** are A, B, C, and D. One preferable method of calculating the PMV for block E is to calculate either the median of the motion vectors of blocks A, B, C, and D, the average of these motion vectors, or the weighted average of these motion vectors. Each of the blocks A through E can be in either frame or field mode.

Another preferable method of calculating the PMV for block E is to use a yes/no method. Under the principles of the yes/no method, a block has to be in the same frame or field coding mode as block E in order to have its motion vector included in the calculation of the PMV for E. For example, if block E in FIG. **12** is in frame mode, block A must also be in frame mode to have its motion vector included in the calculation of the PMV for block E. If one of E's neighboring blocks does not have the same coding mode as does block E, its motion vectors are not used in the calculation of block E's PMV.

The "always method" can also be used to calculate the PMV for block E. In the always method, blocks A, B, C, and D are always used in calculating the PMV for block E, regardless of their frame or field coding mode. If E is in frame mode and a neighboring block is in field mode, the vertical component of the neighboring block is multiplied by 2 before being included in the PMV calculation for block E. If E is in field mode and a neighboring block is in frame mode, the vertical component of the neighboring block is divided by 2 before being included in the PMV calculation for block E.

The "selective method" can also be used to calculate the PMV for block E if the macroblock has been encoded using pair based AFF encoding or group based AFF encoding. In the selective method, a frame-based block has a frame-based motion vector pointing to a reference frame. The block is also assigned a field-based motion vector pointing to a reference field. The field-based motion vector is the frame-based motion vector of the block with the vertical motion vector component divided by two. The reference field number is the reference frame number multiplied by two. A field-based block has a field-based motion vector pointing to a reference field. The block is also assigned a frame-based motion vector pointing to a reference frame. The frame-based motion vector is the field-based motion vector of the block with the vertical motion vector component multiplied by two. The reference frame number is the reference field number divided by two.

The derivation of a block's PMV using the selective method will now be explained using FIG. **12** as a reference. In macroblock pair based AFF, each block in a macroblock is associated with a companion block that resides in the same geometric location within the second macroblock of the macroblock pair. In FIG. **12**, each of block E's neighboring blocks (A, B, C, and D) may or may not be in the same frame or field coding mode as block E. Hence, the following rules apply:

If E is in frame mode and a neighboring block is in frame mode, the true frame-based motion vector of the neighboring block is used for E's PMV.

If E is in frame mode and a neighboring block is in field mode, the following rules apply in calculating E's PMV. If the neighboring block (e.g.; block A) and its companion field-based block have the same reference field, the average of the assigned frame-based motion vectors of the two blocks is used for the calculation of E's PMV. The reference frame number used for the PMV calculation is the reference field number of the neighboring block divided by two. However, if the neighboring block and its companion field block have different reference fields, then the neighboring block cannot be used in the calculation of E's PMV.

If E is in field mode and a neighboring block is in frame mode, the following rules apply in calculating E's PMV. If the neighboring block (e.g.; block A) and its companion frame-based block have the same reference frame, the average of the assigned field-based motion vectors of the two blocks is used for the calculation of E's PMV. The reference field number used for the PMV calculation is the reference frame number of the neighboring block multiplied by two. However, if the neighboring block and its companion field block have different reference frames, then the neighboring block cannot be used in the calculation of E's PMV.

If E is in field mode and a neighboring block is in field mode, the true field-based motion vector of the neighboring block is used in the calculation of E's PMV.

An alternate preferable option can be used in the selective method to calculate a block's PMV. In FIG. **12**, each of block E's neighboring blocks (A, B, C, and D) may or may not be in the same frame or field coding mode as block E. Hence, the following rules apply for this alternate preferable option of the selective method:

If E is in frame mode and a neighboring block is in frame mode, the true frame-based motion vector of the neighboring block is used for E's PMV.

If E is in frame mode and a neighboring block is in field mode, the weighted average of the assigned frame-based motion vectors of the neighboring block and its companion field-based block is used for the calculation of E's PMV. The weighting factors are based upon the reference field numbers of the neighboring block and its companion block.

If E is in field mode, and a neighboring block is in frame mode, the weighted average of the assigned field-based motion vectors of the neighboring block and its companion frame-based block is used for the calculation of E's PMV. The weighting factors are based upon the reference frame numbers of the neighboring block and its companion block.

If E is in field mode and a neighboring block is in field mode, the true field-based motion vector of the neighboring block is used in the calculation of E's PMV.

Another preferable method of computing a block's PMV is the "alt selective method." This method can be used in single macroblock AFF coding, pair based macroblock AFF coding, or group based AFF coding. In this method, each block is assigned a horizontal and a vertical index number, which represents the horizontal and vertical coordinates of the block. Each block is also assigned a horizontal and vertical field coordinate. A block's horizontal field coordinate is same as its horizontal coordinate. For a block in a top field macroblock, the vertical field coordinate is half of vertical coordinate of the block and is assigned top field polarity. For a block in the bottom field macroblock, the vertical field coordinate of the block is obtained by subtracting 4 from the vertical coordinate of the block and dividing the result by 2. The block is also assigned bottom field polarity. The result of assigning different field polarities to two blocks is that there are now two blocks with the same horizontal and vertical field coordinates but with differing field polarities. Thus, given the coordinates of a block, the field coordinates and its field polarity can be computed and vice versa.

The alt selective method will now be explained in detail using FIG. **12** as a reference. The PMV of block E is to be computed. Let bx represent the horizontal size of block E divided by 4, which is the size of a block in this example. The PMVs for E are obtained as follows depending on whether E is in frame/field mode.

Let block E be in frame mode and let (x,y) represent the horizontal and vertical coordinates respectively of E. The neighboring blocks of E are defined in the following manner. A is the block whose coordinates are (x−1,y). B is the block whose coordinates are (x,y−1). D is the block whose coordinates are (x−1,y−1). C is the block whose coordinates are (x+bx+1,y−1). If either A, B, C or D is in field mode then its vertical motion vector is multiplied by 2 before being used for prediction and its reference frame number is computed by dividing its reference field by 2.

Now, let block E be in top or bottom field mode and let (xf,yf) represent the horizontal and vertical field coordinates respectively of E. In this case, the neighbors of E are defined as follows. A is the block whose field coordinates are (xf−1,yf) and has same polarity as E. B is the block whose field coordinates are (xf,yf−1) and has same polarity as E. D is the block whose field coordinates are (xf−1,yf−1) and has same polarity as E. C is the block whose field coordinates are (xf+bx+1,yf) and has same polarity as E. If either A, B, C or D is in frame mode then its vertical motion vector is divided by 2 before being used for prediction and its reference field is computed by multiplying its reference frame by 2.

In all of the above methods for determining the PMV of a block, a horizontal scanning path was assumed. However, the scanning path can also be a vertical scanning path. In this case, the neighboring blocks of the current block, E, are defined as shown in FIG. **13**. A vertical scanning path is preferable in some applications because the information on all the neighboring blocks is available for the calculation of the PMV for the current block E.

Another embodiment of the present invention is directional segmentation prediction. In directional segmentation prediction, 16 by 8 pixel blocks and 8 by 16 pixel blocks have rules that apply to their PMV calculations only. These rules apply in all PMV calculation methods for these block sizes. The rules will now be explained in detail in connection with FIG. **12**. In each of these rules, a current block E is to have its PMV calculated.

First, a 16 by 16 pixel block consists of an upper block and a lower block. The upper block contains the top 8 rows of 16 pixels. The lower block contains the bottom 8 rows of 16 pixels. In the following description, block E of FIG. **12** is a 16 by 8 pixel block. For the upper block having a 16 by 8 pixel block, block B is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV. For the lower block having a 16 by 8 pixel block, block A is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV.

A 16 by 16 pixel block is divided into a right and left block. Both right and left blocks are 8 by 16 pixels. In the following description, block E of FIG. **12** is a 8 by 16 pixel block. For the left block, block A is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV. For the right block, block C is used to predict block E's PMV if it has the same referenced picture as block E. Otherwise median prediction is used to predict block E's PMV.

For both 16 by 8 pixel blocks and 8 by 16 blocks, A, B, or C can be in different encoding modes (frame or field) than the current block E. The following rules apply for both block sizes. If E is in frame mode, and A, B, or C is in field mode, the reference frame number of A, B, or C is computed by dividing its reference field by 2. If E is in field mode, and A, B, or C is in frame mode, the reference field number of A, B, or C is computed by multiplying its reference frame by 2.

According to another embodiment of the present invention, a macroblock in a P picture can be skipped in AFF coding. If a macroblock is skipped, its data is not transmitted in the encoding of the picture. A skipped macroblock in a P picture is reconstructed by copying the co-located macroblock in the most recently coded reference picture. The co-located macroblock is defined as the one with motion compensation using PMV as defined above or without motion vectors. The following rules apply for skipped macroblocks in a P picture. If AFF coding is performed per macroblock, a skipped macroblock is in frame mode. If AFF

coding is performed on macroblock pairs and if both macroblocks are skipped, then they are in frame mode. However, if only one of the macroblocks in a macroblock pair is skipped, its frame or field coding mode is the same as the non-skipped macroblock in the same macroblock pair. If AFF coding is performed on a group of macroblocks and if the entire group of macroblocks is skipped, then all the macroblocks are in frame mode. If there is at least one macroblock that is not skipped, then the skipped macroblocks in the same group are in the same frame or field coding mode as the non-skipped macroblock.

An alternate method for skipped macroblocks is as follows. If a macroblock pair is skipped, its frame and field coding mode follows its neighboring macroblock pair to the left. If the left neighboring macroblock pair is not available, its coding mode follows its neighboring macroblock pair to the top. If neither the left nor top neighboring macroblock pairs are available, the skipped macroblock is set to frame mode.

Another embodiment of the present invention is direct mode macroblock coding for B pictures. In direct mode coding, a B picture has two motion vectors, forward and backward motion vectors. Each motion vector points to a reference picture. Both the forward and backward motion vectors can point in the same temporal direction. For direct mode macroblock coding in B pictures, the forward and backward motion vectors of a block are calculated from the co-located block in the backward reference picture. The co-located block in the backward reference picture can be frame mode or field mode coded. The following rules apply in direct mode macroblock coding for B picture.

If the co-located block is in frame mode and if the current direct mode macroblock is also in frame mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block. The forward reference frame is the one used by the co-located block. The backward reference frame is the same frame where the co-located block resides.

If the co-located block is in frame mode and if the current direct mode macroblock is in field mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block's motion vector with vertical component divided by two. The forward reference field is the same parity field of the reference frame used by the co-located block. The backward reference field is the same parity field of the backward reference frame where the co-located block resides.

If the co-located block is in field mode and if the current direct mode macroblock is also in field mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block of the same field parity. The forward reference field is the field used by the co-located block. The backward reference field is the same field where the co-located block resides.

If the co-located block is in field mode and if the current direct mode macroblock is in frame mode, the two associated motion vectors of the block in the direct mode macroblock are calculated from the co-located block's motion vector with vertical component multiplied by two. The forward reference frame is the frame one of whose fields is used by the co-located block. The backward reference field is the frame in one of whose fields the co-located block resides.

An alternate option is to force the direct mode block to be in the same frame or field coding mode as the co-located block. In this case, if the co-located block for a direct mode block is in frame mode, the direct mode block is in frame mode as well. The two frame-based motion vectors of the direct mode block are derived from the frame-based forward motion vector of the co-located block. The forward reference frame is used by the co-located block. The backward reference frame is where the co-located block resides.

However, if the co-located block for a block in direct mode is in field mode, the direct mode block is also in field mode. The two field-based motion vectors of the direct mode block are derived from the field-based forward motion vector of the co-located block. The forward reference field is used by the co-located block. The backward reference field is where the co-located block resides.

A macroblock in a B picture can also be skipped in AFF coding according to another embodiment of the present invention. A skipped macroblock in a B picture is reconstructed as a regular direct mode macroblock without any coded transform coefficient information. For skipped macroblocks in a B picture, the following rules apply. If AFF coding is performed per macroblock, a skipped macroblock is either in frame mode or in the frame or field coding mode of the co-located block in its backward reference picture. If AFF coding is performed on macroblock pairs and if both macroblocks are skipped, then they are in frame mode or in the frame or field coding mode of the co-located macroblock pair in the its backward reference picture. However, if only one of the macroblocks in a macroblock pair is skipped, its frame or field coding mode is the same as the non-skipped macroblock of the same macroblock pair. If AFF coding is performed on a group of macroblocks and if the entire group of macroblocks is skipped, then all the macroblocks are in frame mode or in the frame or field coding mode of the co-located group of macroblocks in the backward reference picture. If there is at least one macroblock that is not skipped, then the skipped macroblock in the same group are in the same frame or field coding mode as the non-skipped macroblock.

As previously mentioned, a block can be intra coded. Intra blocks are spatially predictive coded. There are two possible intra coding modes for a macroblock in macroblock level AFF coding. The first is intra_4×4 mode and the second is intra_16×16 mode. In both, each pixel's value is predicted using the real reconstructed pixel values from neighboring blocks. By predicting pixel values, more compression can be achieved. The intra_4×4 mode and the intra_16×16 modes will each be explained in more detail below.

For intra_4×4 mode, the predictions of the pixels in a 4 by 4 pixel block, as shown in FIG. **14**, are derived form its left and above pixels. In FIG. **14**, the 16 pixels in the 4 by 4 pixel block are labeled a through p. Also shown in FIG. **14** are the neighboring pixels A through P. The neighboring pixels are in capital letters. As shown in FIG. **15**, there are nine different prediction directions for intra_4×4 coding. They are vertical (**0**), horizontal (**1**), DC prediction (mode **2**), diagonal down/left (**3**), diagonal down/right (**4**), vertical-left (**5**), horizontal-down (**6**), vertical-right (**7**), and horizontal-up (**8**). DC prediction averages all the neighboring pixels together to predict a particular pixel value.

However, for intra_16×16 mode, there are four different prediction directions. Prediction directions are also referred to as prediction modes. These prediction directions are vertical prediction (**0**), horizontal prediction (**1**), DC prediction, and plane prediction. Plane prediction will not be explained.

An intra block and its neighboring blocks may be coded in frame or field mode. Intra prediction is performed on the reconstructed blocks. A reconstructed block can be represented in both frame and field mode, regardless of the actual

frame or field coding mode of the block. Since only the pixels of the reconstructed blocks are used for intra prediction, the following rules apply.

If a block of 4 by 4 pixels or 16 by 16 pixels is in frame mode, the neighboring pixels used in calculating the pixel value predictions of the block are in the frame structure. If a block of 4 by 4 pixels or 16 by 16 pixels is in field mode, the neighboring pixels used in calculating the pixel value prediction of the block are in field structure of the same field parity.

The chosen intra-prediction mode (intra_pred_mode) of a 4 by 4 pixel block is highly correlated with the prediction modes of adjacent blocks. This is illustrated in FIG. **16***a*. FIG. **16***a* shows that A and B are adjacent blocks to C. Block C's prediction mode is to be established. FIG. **16***b* shows the order of intra prediction information in the bitstream. When the prediction modes of A and B are known (including the case that A or B or both are outside the slice) the most probable prediction mode (most_probable_mode) of C is given. If one of the blocks A or B is "outside" the most probable prediction mode is equal DC prediction (mode **2**). Otherwise it is equal to the minimum of prediction modes used for blocks A and B. When an adjacent block is coded by 16×16 intra mode, prediction mode is DC prediction mode. When an adjacent block is coded a non-intra macroblock, prediction mode is "mode **2**: DC prediction" in the usual case and "outside" in the case of constrained intra update.

To signal a prediction mode number for a 4 by 4 block first parameter use_most_probable_mode is transmitted. This parameter is represented by 1 bit codeword and can take values 0 or 1. If use_most_probable_mode is equal to 1 the most probable mode is used. Otherwise an additional parameter remaining_mode_selector, which can take value from 0 to 7 is sent as 3 bit codeword. The codeword is a binary representation of remaining_mode_selector value. The prediction mode number is calculated as:

```
if (remaining_mode_selector<most_probable_mode)
intra_pred_mode=remaining_mode_selector;
else
intra_pred_mode=remaining_mode_selector+1;
```

The ordering of prediction modes assigned to blocks C is therefore the most probable mode followed by the remaining modes in the ascending order.

An embodiment of the present invention includes the following rules that apply to intra mode prediction for an intra-prediction mode of a 4 by 4 pixel block or an intra-prediction mode of a 16 by 16 pixel block. Block C and its neighboring blocks A and B can be in frame or field mode. One of the following rules shall apply. FIGS. **16***a-b* will be used in the following explanations of the rules.

Rule 1: A or B is used as the neighboring block of C only if A or B is in the same frame/field mode as C. Otherwise, A or B is considered as outside.

Rule 2: A and B are used as the neighboring blocks of C, regardless of their frame/field coding mode.

Rule 3: If C is coded in frame mode and has co-ordinates (x,y), then A is the block with co-ordinates (x,y−1) and B is the block with co-ordinates (x−1,y). Otherwise, if C is coded as field and has field co-ordinates (xf,yf) then A is the block whose field co-ordinates are (xf,yf−1) and has same field polarity as C and B is the block whose field co-ordinates are (xf−1,yf) and has same field polarity as C.

Rule 4: This rule applies to macroblock pairs only. In the case of decoding the prediction modes of blocks numbered **3**, **6**, **7**, **9**, **12**, **13**, **11**, **14** and **15** of FIG. **16***b*, the above and the left neighboring blocks are in the same macroblock as the current block. However, in the case of decoding the prediction modes of blocks numbered **1**, **4**, and **5**, the top block (block A) is in a different macroblock pair than the current macroblock pair. In the case of decoding the prediction mode of blocks numbered **2**, **8**, and **10**, the left block (block B) is in a different macroblock pair. In the case of decoding the prediction mode of the block numbered **0**, both the left and the above blocks are in different macroblock pairs. For a macroblock in field decoding mode the neighboring blocks of the blocks numbered **0**, **1**, **4**, **5**, **2**, **8**, and **10** shall be defined as follows:

If the above macroblock pair (**170**) is decoded in field mode, then for blocks number **0**, **1**, **4** and **5** in the top-field macroblock (**173**), blocks numbered **10**, **11**, **14** and **15** respectively in the top-field macroblock (**173**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**) as shown in FIG. **17***a*. For blocks number **0**, **1**, **4** and **5** in the bottom-field macroblock (**174**), blocks numbered **10**, **11**, **14** and **15** respectively in the bottom-field MB of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***a*.

However, if the above macroblock pair (**170**) is decoded in frame mode then for blocks number **0**, **1**, **4** and **5** in the top-field macroblock (**173**), blocks numbered **10**, **11**, **14** and **15** respectively in the bottom-frame macroblock (**176**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***b*. For blocks number **0**, **1**, **4** and **5** in the bottom-field macroblock (**174**), blocks numbered **10**, **11**, **14** and **15** respectively in the bottom-frame macroblock (**176**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown inn FIG. **17***b*.

If the left macroblock pair (**172**) is decoded in field mode, then for blocks number **0**, **2**, **8** and **10** in the top-field macroblock (**173**), blocks numbered **5**, **7**, **13** and **15** respectively in the top-field macroblock (**173**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**) as shown in FIG. **17***c*. For blocks number **0**, **2**, **8** and **10** in the bottom-field macroblock (**174**), blocks numbered **5**, **7**, **13** and **15** respectively in the bottom-field macroblock (**174**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***c*.

If the left macroblock pair (**172**) is decoded in frame mode, then for blocks number **0**, **2**, **8** and **10** in the top-field macroblock (**173**), the blocks numbered **5**, **7**, **13** and **15** respectively in the top-frame macroblock (**175**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***d*. For blocks number **0**, **2**, **8** and **10** in the bottom-field macroblock (**174**), blocks numbered **5**, **7**, **13** and **15** respectively in the bottom-frame macroblock (**176**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***d*.

For macroblock pairs on the upper boundary of a slice, if the left macroblock pair (**172**) is in frame decoding mode, then the intra mode prediction value used to predict a field macroblock shall be set to DC prediction.

The preceding descriptions of intra coding and intra mode prediction can be extended to adaptive block transforms.

Another embodiment of the present invention is that loop filtering is performed on the reconstructed blocks. A recon-

structed block can be represented in either frame or field structure, regardless of the frame/field coding mode of the block. Loop (deblock) filtering is a process of weighted averaging of the pixels of the neighboring blocks. FIG. **12** will be used to explain loop filtering. Assume E of FIG. **12** is a reconstructed block, and A, B, C and D are its neighboring reconstructed blocks, as shown in FIG. **12**, and they are all represented in frame structure. Since A, B, C, D and E can be either frame- or field-coded, the following rules apply:

Rule 1: If E is frame-coded, loop filtering is performed over the pixels of E and its neighboring blocks A B, C and D.

Rule 2: If E is field-coded, loop filtering is performed over the top-field and bottom-field pixels of E and its neighboring blocks A B, C and D, separately.

Another embodiment of the present invention is that padding is performed on the reconstructed frame by repeating the boundary pixels. Since the boundary blocks may be coded in frame or field mode, the following rules apply:

Rule 1: The pixels on the left or right vertical line of a boundary block are repeated, if necessary.

Rule 2: If a boundary block is in frame coding, the pixels on the top or bottom horizontal line of the boundary block are repeated.

Rule 3: if a boundary block is in field coding, the pixels on the two top or two bottom horizontal (two field) lines of the boundary block are repeated alternatively.

Another embodiment of the present invention is that two-dimensional transform coefficients are converted into one-dimensional series of coefficients before entropy coding. The scan path can be either zigzag or non-zigzag. The zigzag scanner is preferably for progressive sequences, but it may be also used for interlace sequences with slow motions. The non-zigzag scanners are preferably for interlace sequences. For macroblock AFF coding, the following options may be used:

Option 1: The zigzag scan is used for macroblocks in frame mode while the non-zigzag scanners are used for macroblocks in field coding.

Option 2: The zigzag scan is used for macroblocks in both frame and field modes.

Option 3: The non-zigzag scan is used for macroblocks in both frame and field modes.

The preceding description has been presented only to illustrate and describe embodiments of invention. It is not intended to be exhaustive or to limit the invention to any precise form disclosed. Many modifications and variations are possible in light of the above teaching.

The foregoing embodiments were chosen and described in order to illustrate principles of the invention and some practical applications. The preceding description enables others skilled in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims.

What is claimed is:

1. A method of encoding a picture in an image sequence, comprising:

dividing said picture into a plurality of smaller portions, wherein each of said smaller portions has a size that is larger than one macroblock;

selectively encoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode; and

selectively encoding at least one block within at least one of said plurality of smaller portions at a time in inter coding mode.

2. The method of claim **1**, wherein at least one motion vector is computed for said at least one block within at least one of said plurality of smaller portions.

3. The method of claim **2**, wherein said at least one motion vector is spatially predictive coded for a current block of said plurality of smaller portions.

4. An apparatus of encoding a picture in an image sequence, comprising:

means for dividing said picture into a plurality of smaller portions, wherein each of said smaller portions has a size that is larger than one macroblock;

means for selectively encoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode; and

means for selectively encoding at least one block within at least one of said plurality of smaller portions at a time in inter coding mode.

5. The apparatus of claim **4**, wherein at least one motion vector is computed for said at least one block within at least one of said plurality of smaller portions.

6. The apparatus of claim **5**, wherein said at least one motion vector is spatially predictive coded for a current block of said plurality of smaller portions.

7. A computer-readable medium encoded with computer executable instructions, the computer executable instructions including instructions which, when executed by a processor, cause the processor to perform the steps of a method of encoding a picture in an image sequence, comprising the steps of:

dividing said picture into a plurality of smaller portions, wherein each of said smaller portions has a size that is larger than one macroblock;

selectively encoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode; and

selectively encoding at least one block within at least one of said plurality of smaller portions at a time in inter coding mode.

8. A method of decoding an encoded picture having a plurality of smaller portions from a bitstream, comprising:

decoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode, wherein each of said smaller portions has a size that is larger than one macroblock, wherein at least one block within said at least one of said plurality of smaller portions at a time is encoded in inter coding mode; and

using said plurality of decoded smaller portions to construct a decoded picture.

9. The method of claim **8**, wherein at least one motion vector is received for said at least one block within at least one of said plurality of smaller portions.

10. The method of claim **9**, wherein said at least one motion vector is spatially predictive coded for a current block of said plurality of smaller portions.

11. The method of claim **10**, wherein said at least one motion vector is spatially predictive coded from a plurality of motion vectors associated with a plurality of neighboring blocks relative to said current block.

12. The method of claim **11**, wherein said motion vectors associated with said plurality of neighboring blocks relative to said current block are derived to generate at least one

prediction motion vector (PMV), wherein said at least one prediction motion vector (PMV) and a difference value received in the bitstream are used to derive said at least one motion vector of said current block.

13. The method of claim **12**, wherein said at least one PMV is calculated in accordance with directional segmentation prediction.

14. An apparatus for decoding an encoded picture from a bitstream, comprising:

means for decoding at least one of a plurality of smaller portions at a time of the encoded picture that is encoded in frame coding mode and at least one of said plurality of smaller portions at a time of the encoded picture in field coding mode, wherein each of said smaller portions has a size that is larger than one macroblock, wherein at least one block within at least one of said plurality of smaller portions at a time is encoded in inter coding mode; and

means for using said plurality of decoded smaller portions to construct a decoded picture.

15. The apparatus of claim **14**, wherein at least one motion vector is received for said at least one block within at least one of said plurality of smaller portions.

16. The apparatus of claim **15**, wherein said at least one motion vector is spatially predictive coded for a current block of said plurality of smaller portions.

17. The apparatus of claim **16**, wherein said at least one motion vector is spatially predictive coded from a plurality of motion vectors associated with a plurality of neighboring blocks relative to said current block.

18. The apparatus of claim **17**, wherein said motion vectors associated with said plurality of neighboring blocks relative to said current block are used to generate at least one prediction motion vector (PMV), where a difference between said at least one PMV and said at least one motion vector of said current block is calculated and encoded.

19. A bitstream comprising:

a picture that has been divided into a plurality of smaller portions, wherein at least one of said plurality of smaller portions at a time is encoded in frame coding mode and at least one of said plurality of smaller portions at a time is encoded in field coding mode, wherein each of said smaller portions has a size that is larger than one macroblock, and wherein at least one block within at least one of said plurality of smaller portions at a time is encoded in inter coding mode.

* * * * *

EXHIBIT B

US007310375B2

(12) **United States Patent**
Wang et al.

(10) **Patent No.: US 7,310,375 B2**
(45) **Date of Patent: Dec. 18, 2007**

(54) **MACROBLOCK LEVEL ADAPTIVE FRAME/FIELD CODING FOR DIGITAL VIDEO CONTENT**

(75) Inventors: **Limin Wang**, San Diego, CA (US); **Rajeev Gandhi**, San Diego, CA (US); **Krit Panusopone**, San Diego, CA (US); **Ajay Luthra**, San Diego, CA (US)

(73) Assignee: **General Instrument Corporation**, Horsham, PA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 520 days.

(21) Appl. No.: **11/027,626**

(22) Filed: **Dec. 30, 2004**

(65) **Prior Publication Data**

US 2005/0111550 A1 May 26, 2005

Related U.S. Application Data

(62) Division of application No. 10/301,290, filed on Nov. 20, 2002, now Pat. No. 6,980,596.

(60) Provisional application No. 60/395,734, filed on Jul. 12, 2002, provisional application No. 60/333,921, filed on Nov. 27, 2001, provisional application No. 60/398,161, filed on Jul. 23, 2002.

(51) **Int. Cl.**
H04B 1/66 (2006.01)

(52) **U.S. Cl.** **375/240.16**; 375/240.24; 375/240.25; 375/240.13; 375/240.02; 375/240.26; 382/238; 382/233; 382/236; 382/235; 382/239

(58) **Field of Classification Search** 375/240.16, 375/240.24, 240.25, 240.13, 240.02, 240.26; 382/238, 233, 236, 235, 239
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,437,119	A		3/1984	Matsumoto et al.
5,412,428	A		5/1995	Tahara
5,504,530	A	*	4/1996	Obikane et al. 375/240.14
5,801,778	A		9/1998	Ju
6,094,225	A		7/2000	Han
6,192,148	B1		2/2001	Lin
6,404,813	B1		6/2002	Haskell et al.

OTHER PUBLICATIONS

P. Borgwardt: "Core Experiment on Interlaced Video Coding VCEG-N85" ITU-Telecommunications Standardization Sector ITU-T Q.6/SG16 Video Coding Expert Group (VCEG) Sep. 24-27, 2001, pp. 1-10, XP002257037 Santa Barbara, CA USA.

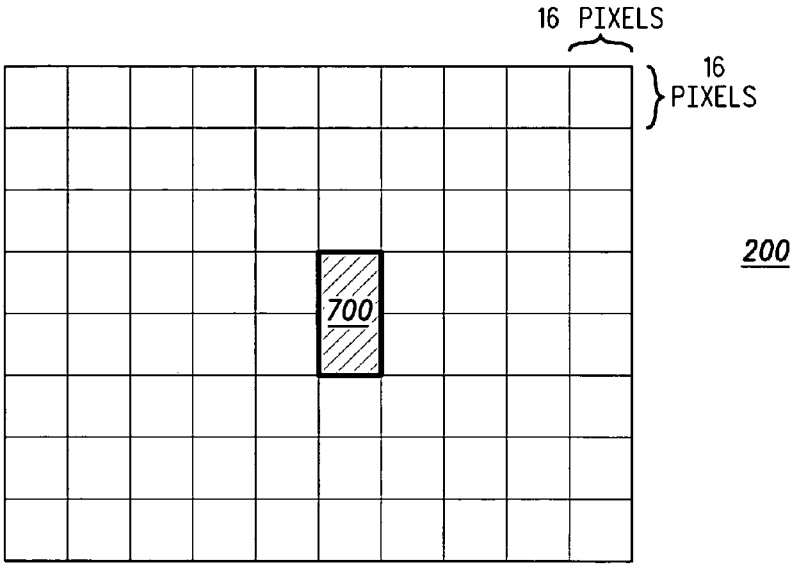
(Continued)

Primary Examiner—Shawn S. An
(74) *Attorney, Agent, or Firm*—Larry T. Cullen

(57) **ABSTRACT**

A method and system of encoding and decoding digital video content. The digital video content comprises a stream of pictures which can each be intra, predicted, or bi-predicted pictures. Each of the pictures comprises macroblocks that can be further divided into smaller blocks. The method entails encoding and decoding each of the smaller blocks in each picture in said stream of pictures in either frame mode or in field mode.

18 Claims, 8 Drawing Sheets



OTHER PUBLICATIONS

"H.26L Test Model Long Term No. 8 (TML-8) DRAFTO" ITU-T Telecommunicatoin Standardization Sector of ITU, Geneva, CH, Apr. 2, 2001, pp. 1-54, XP001089814 p. 9, paragraph 1—p. 10, paragraph 6.1.2.2.1 p p. 40.

P. Borgwardt: "Handling Interlaced Video in H.26L VCEG-N57" ITU-Telecommunications Standardization Sector ITU Q.6/SG16 Video Coding Expert Group (VCEG), Sep. 24-27, 2001, pp. 1-3, XP002257142 Santa Barbara, CA USA.

M. Gallant et al: High Rate, High Resolution Video Using H.26L VCEG-N84 ITU-Telecommunications Standardization Secotr ITU Q.6/SG16 Video Coding Expert Group (VCEG), Sep. 24-27, 2001, pp. 1-7, XP002257143 Santa Barbara, CA USA; p. 1; p. 3.

"Adaptive Field/From Block Coding Experiment Proposal". Interested Parties for the Study of Interfaced Video Coding With H.26L, Video Coding Experts Group, Study Group 16.

* cited by examiner

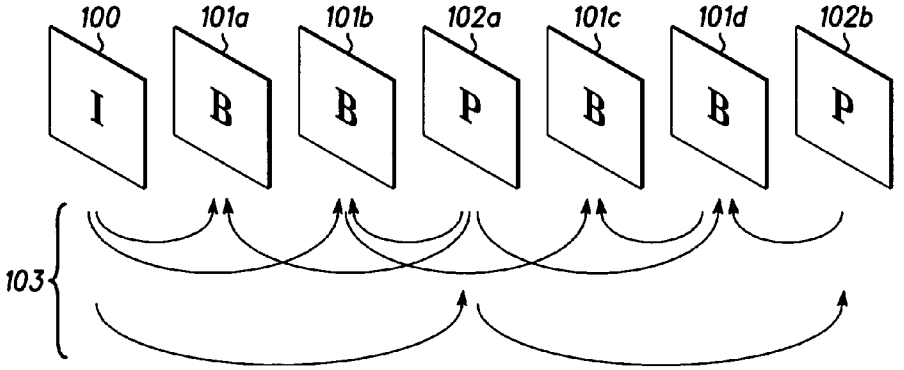


FIG. 1

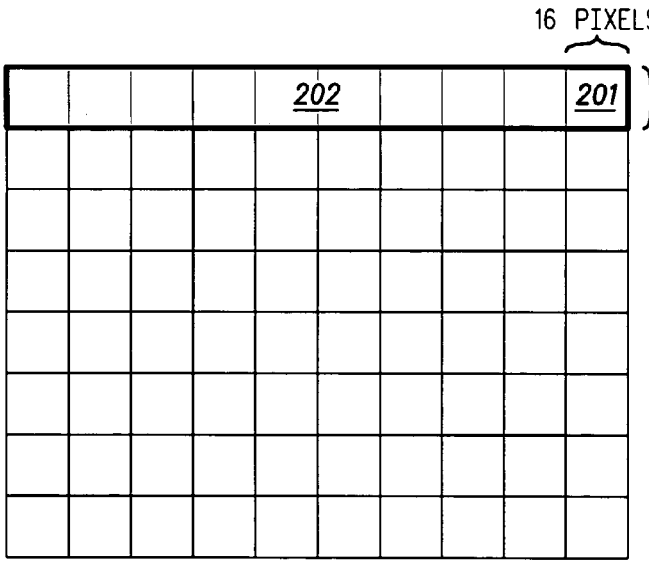


FIG. 2

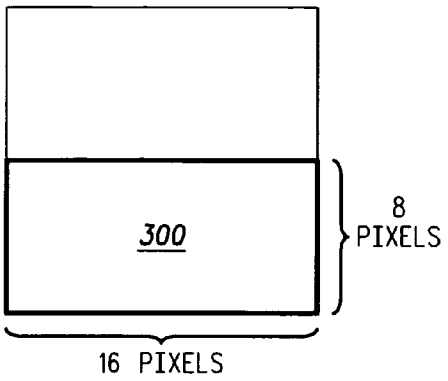


FIG. 3A

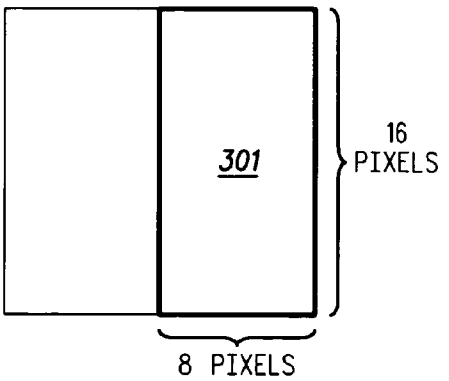


FIG. 3B

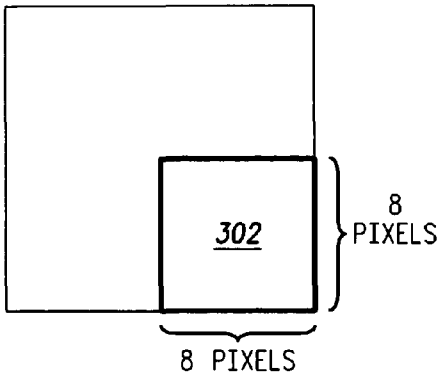


FIG. 3C

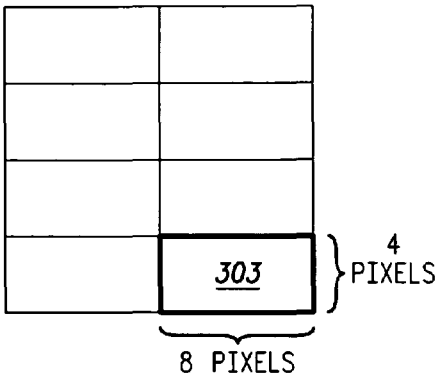


FIG. 3D

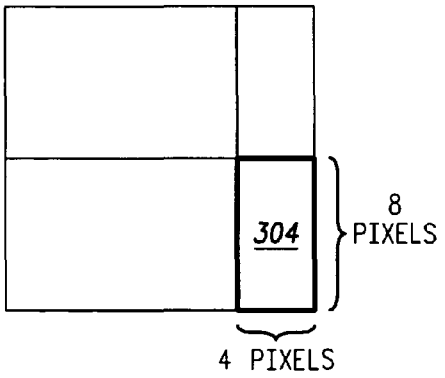


FIG. 3E

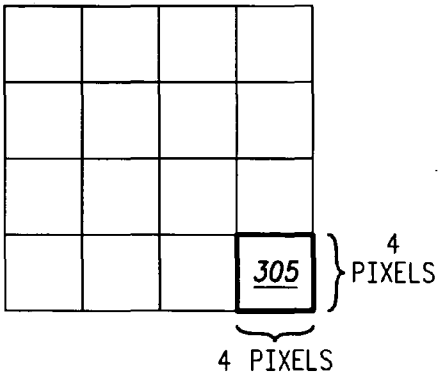


FIG. 3F

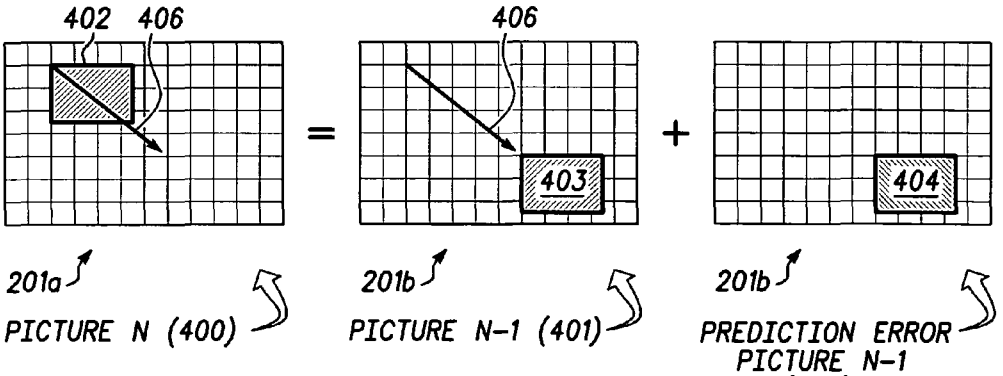
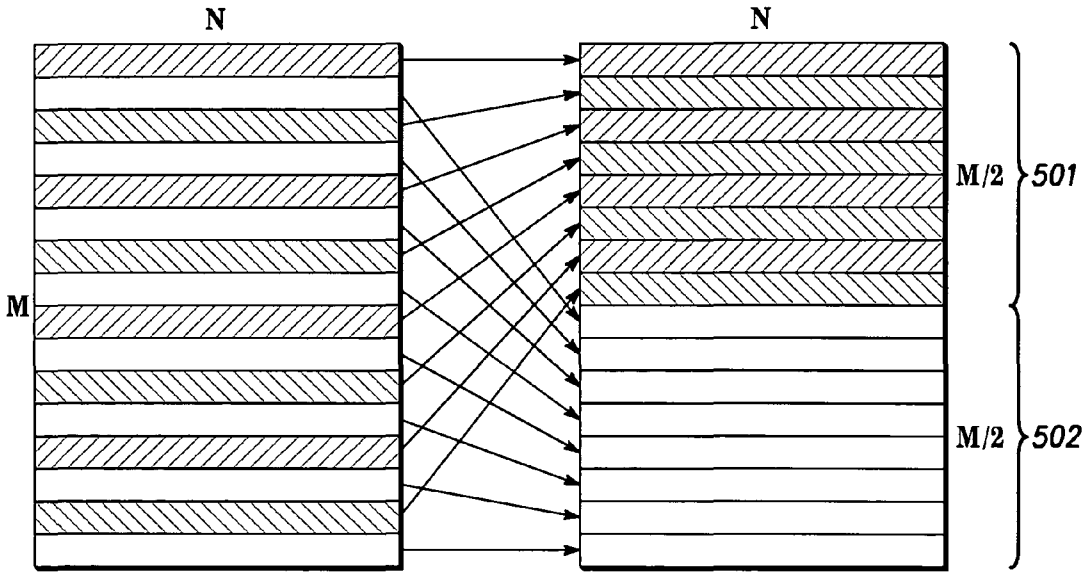


FIG. 4



500 FIG. 5

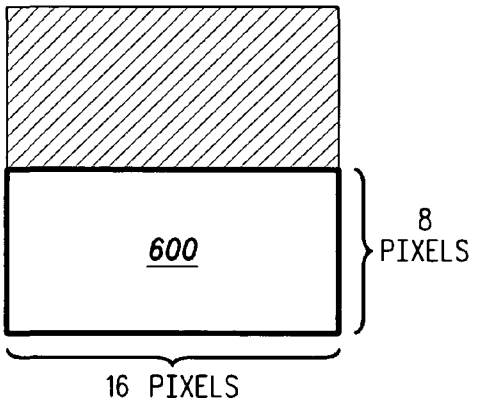


FIG. 6A

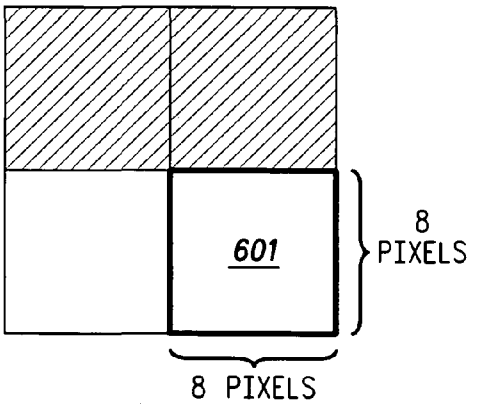


FIG. 6B

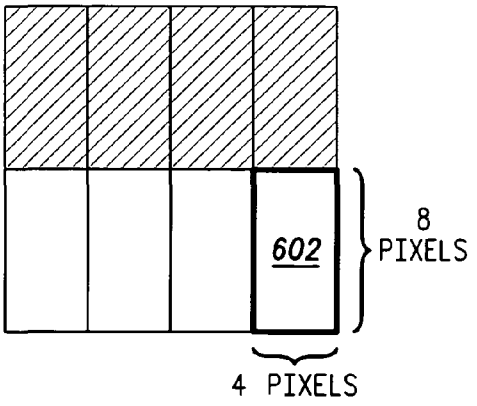


FIG. 6C

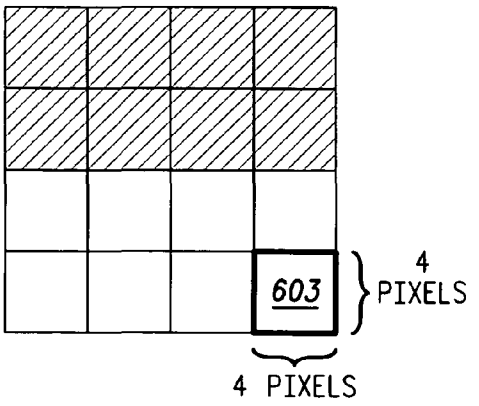


FIG. 6D

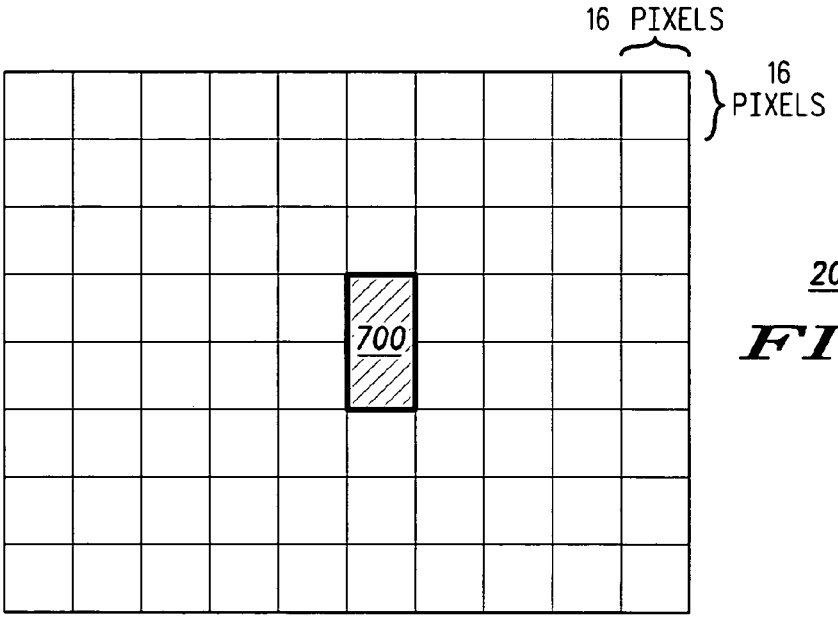


FIG. 7

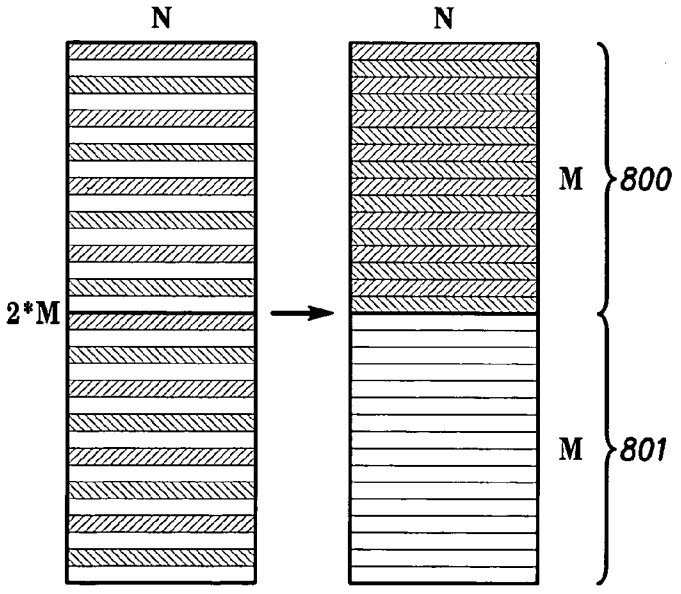


FIG. 8

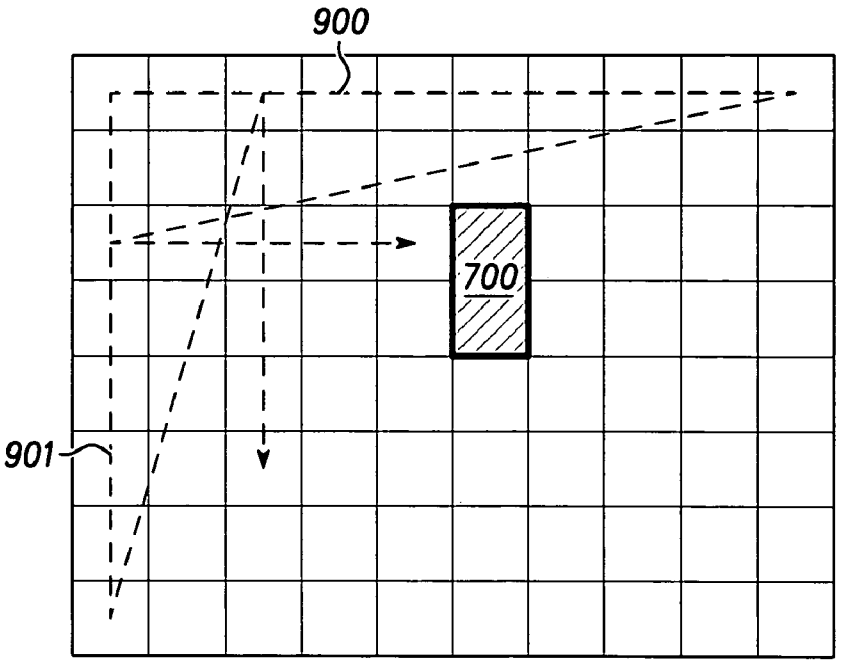


FIG. 9

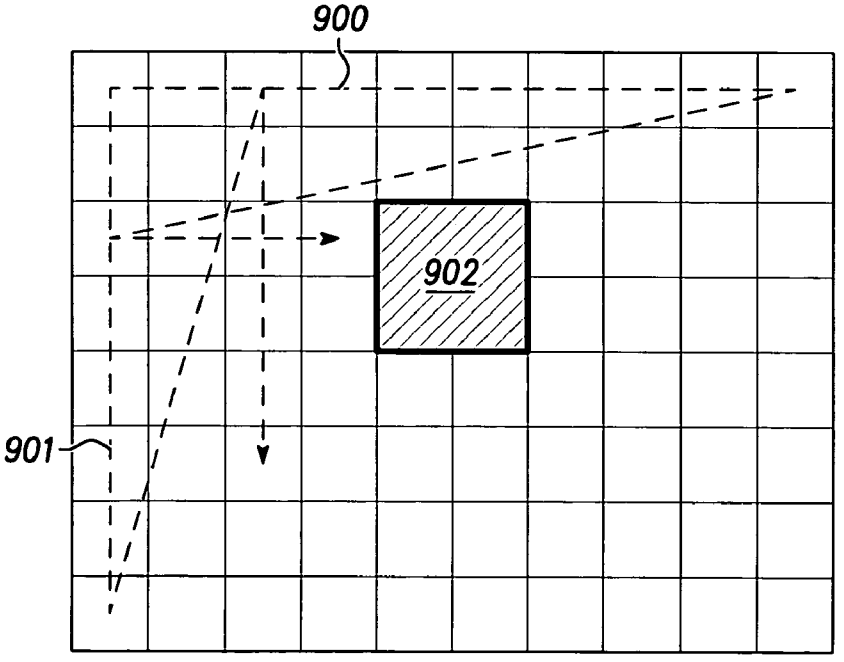


FIG. 10

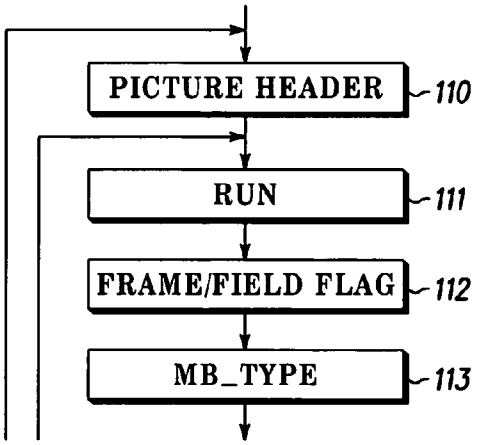


FIG. 11

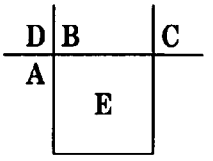


FIG. 12

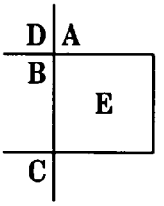


FIG. 13

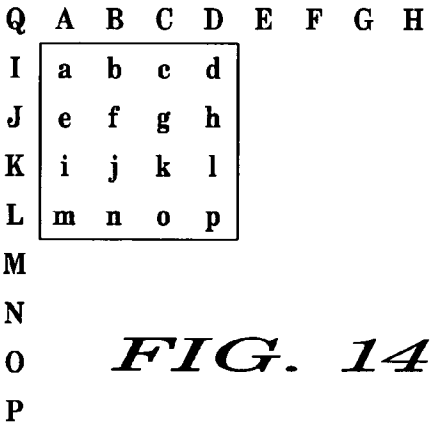


FIG. 14

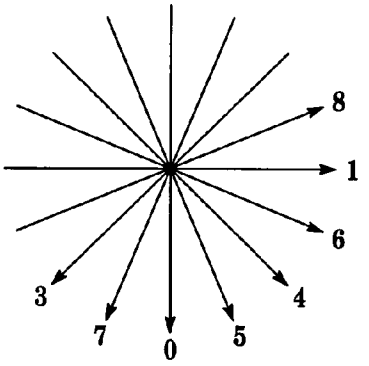


FIG. 15

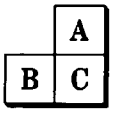


FIG. 16A

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

FIG. 16B

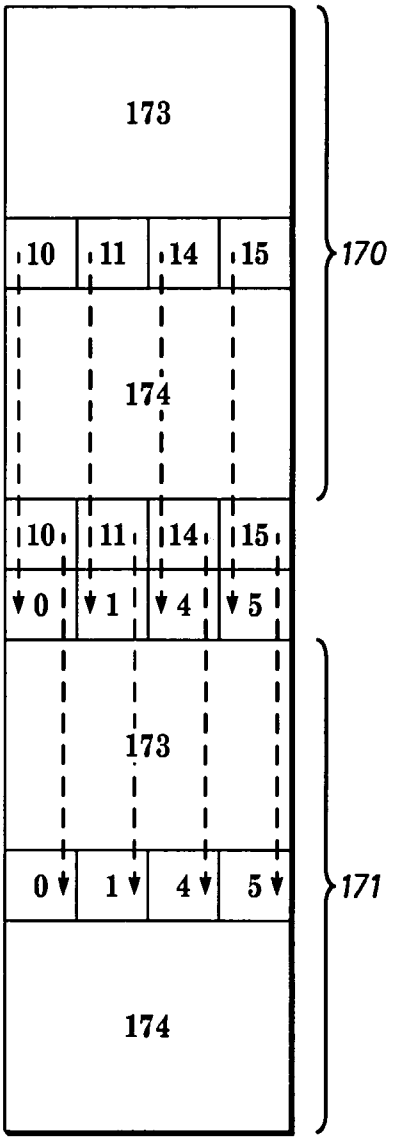


FIG. 17A

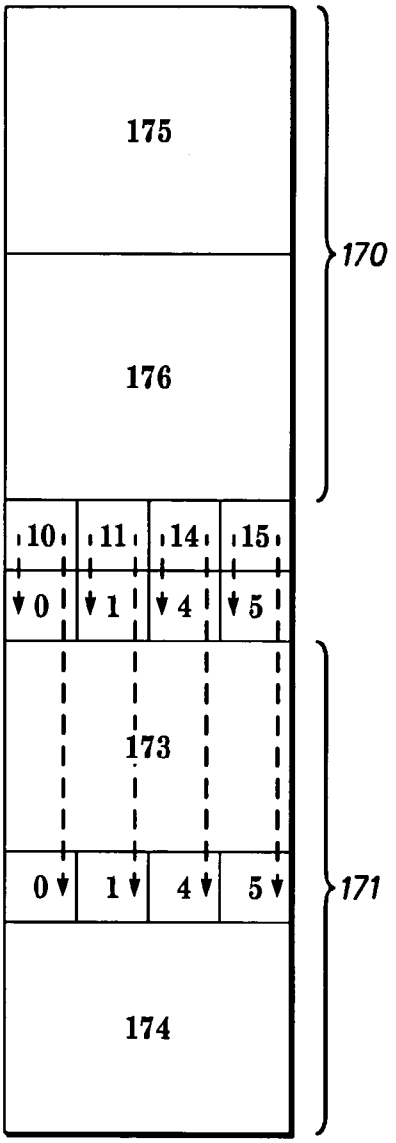


FIG. 17B

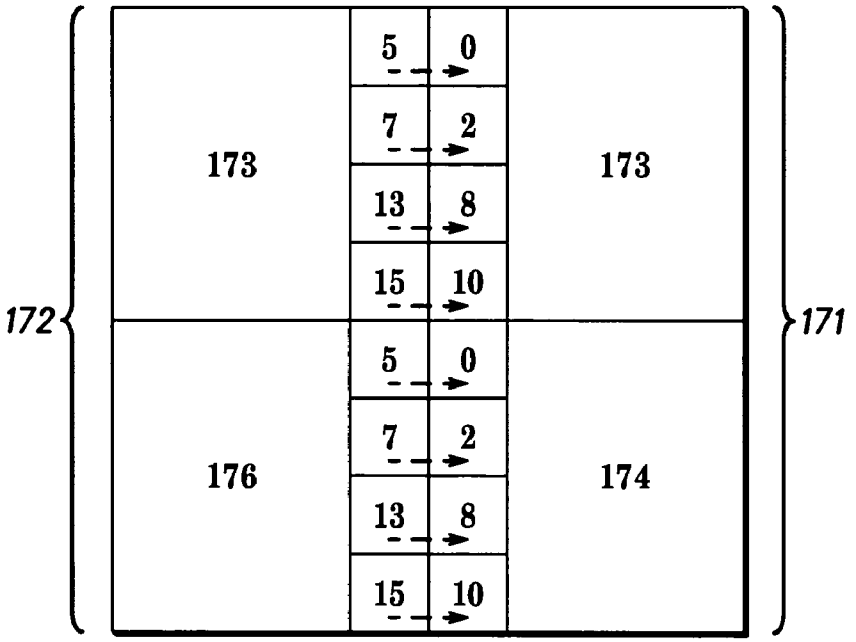


FIG. 17C

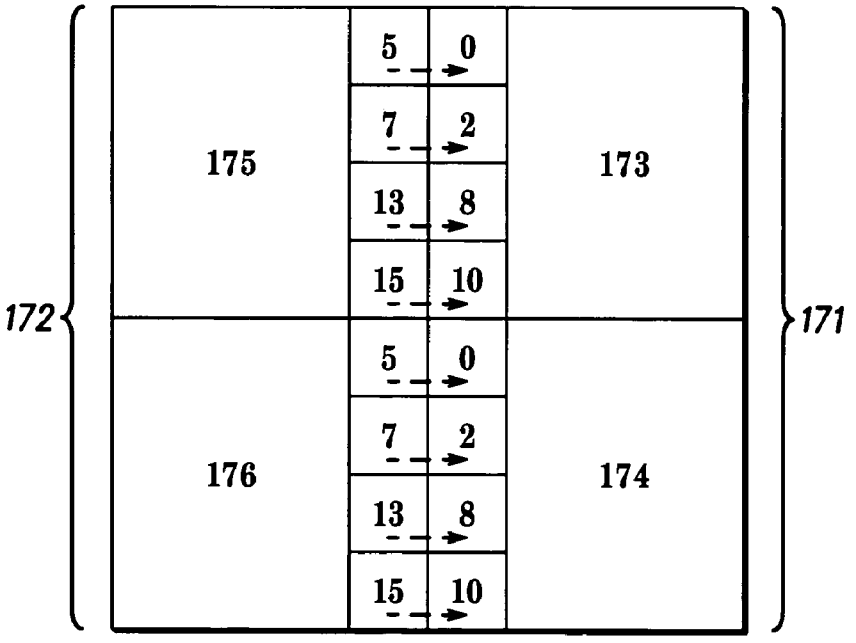


FIG. 17D

MACROBLOCK LEVEL ADAPTIVE FRAME/FIELD CODING FOR DIGITAL VIDEO CONTENT

The present application claims priority under 35 U.S.C. §119(e) from the following previously filed Provisional Patent Applications: Ser. No. 60/333,921, filed Nov. 27, 2001; Ser. No. 60/395,734, filed Jul. 12, 2002; Ser. No. 60/398,161, filed Jul. 23, 2002; all of which are herein incorporated by reference. This application is also a Divisional of U.S. patent application Ser. No. 10/301,290 filed on Nov. 20, 2002 now U.S. Pat. No. 6,980,596, which is herein incorporated by reference.

TECHNICAL FIELD

The present invention relates to encoding and decoding of digital video content. More specifically, the present invention relates to frame mode and field mode encoding of digital video content at a macroblock level as used in the MPEG-4 Part **10** AVC/H.264 standard video coding standard.

BACKGROUND

Video compression is used in many current and emerging products. It is at the heart of digital television set-top boxes (STBs), digital satellite systems (DSSs), high definition television (HDTV) decoders, digital versatile disk (DVD) players, video conferencing, Internet video and multimedia content, and other digital video applications. Without video compression, digital video content can be extremely large, making it difficult or even impossible for the digital video content to be efficiently stored, transmitted, or viewed.

The digital video content comprises a stream of pictures that can be displayed as an image on a television receiver, computer monitor, or some other electronic device capable of displaying digital video content. A picture that is displayed in time before a particular picture is in the "backward direction" in relation to the particular picture. Likewise, a picture that is displayed in time after a particular picture is in the "forward direction" in relation to the particular picture.

Video compression is accomplished in a video encoding, or coding, process in which each picture is encoded as either a frame or as two fields. Each frame comprises a number of lines of spatial information. For example, a typical frame contains 480 horizontal lines. Each field contains half the number of lines in the frame. For example, if the frame comprises 480 horizontal lines, each field comprises 240 horizontal lines. In a typical configuration, one of the fields comprises the odd numbered lines in the frame and the other field comprises the even numbered lines in the frame. The field that comprises the odd numbered lines will be referred to as the "top" field hereafter and in the appended claims, unless otherwise specifically denoted. Likewise, the field that comprises the even numbered lines will be referred to as the "bottom" field hereafter and in the appended claims, unless otherwise specifically denoted. The two fields can be interlaced together to form an interlaced frame.

The general idea behind video coding is to remove data from the digital video content that is "non-essential." The decreased amount of data then requires less bandwidth for broadcast or transmission. After the compressed video data has been transmitted, it must be decoded, or decompressed. In this process, the transmitted video data is processed to generate approximation data that is substituted into the video data to replace the "non-essential" data that was removed in the coding process.

Video coding transforms the digital video content into a compressed form that can be stored using less space and transmitted using less bandwidth than uncompressed digital video content. It does so by taking advantage of temporal and spatial redundancies in the pictures of the video content. The digital video content can be stored in a storage medium such as a hard drive, DVD, or some other non-volatile storage unit.

There are numerous video coding methods that compress the digital video content. Consequently, video coding standards have been developed to standardize the various video coding methods so that the compressed digital video content is rendered in formats that a majority of video encoders and decoders can recognize. For example, the Motion Picture Experts Group (MPEG) and International Telecommunication Union (ITU-T) have developed video coding standards that are in wide use. Examples of these standards include the MPEG-1, MPEG-2, MPEG-4, ITU-T H261, and ITU-T H263 standards.

Most modern video coding standards, such as those developed by MPEG and ITU-T, are based in part on a temporal prediction with motion compensation (MC) algorithm. Temporal prediction with motion compensation is used to remove temporal redundancy between successive pictures in a digital video broadcast.

The temporal prediction with motion compensation algorithm typically utilizes one or two reference pictures to encode a particular picture. A reference picture is a picture that has already been encoded. By comparing the particular picture that is to be encoded with one of the reference pictures, the temporal prediction with motion compensation algorithm can take advantage of the temporal redundancy that exists between the reference picture and the particular picture that is to be encoded and encode the picture with a higher amount of compression than if the picture were encoded without using the temporal prediction with motion compensation algorithm. One of the reference pictures may be in the backward direction in relation to the particular picture that is to be encoded. The other reference picture is in the forward direction in relation to the particular picture that is to be encoded.

However, as the demand for higher resolutions, more complex graphical content, and faster transmission time increases, so does the need for better video compression methods. To this end, a new video coding standard is currently being developed jointly by ISO and ITU-T. This new video coding standard is called the MPEG-4 Advanced Video Coding (AVC)/H.264 standard.

SUMMARY OF THE INVENTION

In one of many possible embodiments, the present invention provides a method of encoding, decoding, and bitstream generation of digital video content. The digital video content comprises a stream of pictures which can each be intra, predicted, or bi-predicted pictures. Each of the pictures comprises macroblocks that can be further divided into smaller blocks. The method entails encoding and decoding each of the macroblocks in each picture in said stream of pictures in either frame mode or in field mode.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings illustrate various embodiments of the present invention and are a part of the specification. Together with the following description, the drawings demonstrate and explain the principles of the present

invention. The illustrated embodiments are examples of the present invention and do not limit the scope of the invention.

FIG. **1** illustrates an exemplary sequence of three types of pictures that can be used to implement the present invention, as defined by an exemplary video coding standard such as the MPEG-4 Part **10** AVC/H.264 standard.

FIG. **2** shows that each picture is preferably divided into slices containing macroblocks according to an embodiment of the present invention.

FIG. **3***a* shows that a macroblock can be further divided into a block size of 16 by 8 pixels according to an embodiment of the present invention.

FIG. **3***b* shows that a macroblock can be further divided into a block size of 8 by 16 pixels according to an embodiment of the present invention.

FIG. **3***c* shows that a macroblock can be further divided into a block size of 8 by 8 pixels according to an embodiment of the present invention.

FIG. **3***d* shows that a macroblock can be further divided into a block size of 8 by 4 pixels according to an embodiment of the present invention.

FIG. **3***e* shows that a macroblock can be further divided into a block size of 4 by 8 pixels according to an embodiment of the present invention.

FIG. **3***f* shows that a macroblock can be further divided into a block size of 4 by 4 pixels according to an embodiment of the present invention.

FIG. **4** shows a picture construction example using temporal prediction with motion compensation that illustrates an embodiment of the present invention.

FIG. **5** shows that a macroblock is split into a top field and a bottom field if it is to be encoded in field mode.

FIG. **6***a* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 16 by 8 pixels according to an embodiment of the present invention.

FIG. **6***b* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 8 by 8 pixels according to an embodiment of the present invention.

FIG. **6***c* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 4 by 8 pixels according to an embodiment of the present invention.

FIG. **6***d* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 4 by 4 pixels according to an embodiment of the present invention.

FIG. **7** illustrates an exemplary pair of macroblocks that can be used in AFF coding on a pair of macroblocks according to an embodiment of the present invention.

FIG. **8** shows that a pair of macroblocks that is to be encoded in field mode is first split into one top field **16** by 16 pixel block and one bottom field 16 by 16 pixel block.

FIG. **9** shows two possible scanning paths in AFF coding of pairs of macroblocks.

FIG. **10** illustrates another embodiment of the present invention which extends the concept of AFF coding on a pair of macroblocks to AFF coding to a group of four or more neighboring macroblocks.

FIG. **11** shows some of the information included in the bitstream which contains information pertinent to each macroblock within a stream.

FIG. **12** shows a block that is to be encoded and its neighboring blocks and will be used to explain various preferable methods of calculating the PMV of a block in a macroblock.

FIG. **13** shows an alternate definition of neighboring blocks if the scanning path is a vertical scanning path.

FIG. **14** shows that each pixel value is predicted from neighboring blocks pixel values according to an embodiment of the present invention.

FIG. **15** shows different prediction directions for intra_4×4 coding.

FIGS. **16***a-b* illustrate that the chosen intra-prediction mode (intra_pred_mode) of a 4 by 4 pixel block is highly correlated with the prediction modes of adjacent blocks.

FIGS. **17***a-d* show neighboring blocks definitions in relation to a current macroblock pair that is to be encoded.

Throughout the drawings, identical reference numbers designate similar, but not necessarily identical, elements.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

The present invention provides a method of adaptive frame/field (AFF) coding of digital video content comprising a stream of pictures or slices of a picture at a macroblock level. The present invention extends the concept of picture level AFF to macroblocks. In AFF coding at a picture level, each picture in a stream of pictures that is to be encoded is encoded in either frame mode or in field mode, regardless of the frame or field coding mode of other pictures that are to be coded. If a picture is encoded in frame mode, the two fields that make up an interlaced frame are coded jointly. Conversely, if a picture is encoded in field mode, the two fields that make up an interlaced frame are coded separately. The encoder determines which type of coding, frame mode coding or field mode coding, is more advantageous for each picture and chooses that type of encoding for the picture. The exact method of choosing between frame mode and field mode is not critical to the present invention and will not be detailed herein.

As noted above, the MPEG-4 Part **10** AVC/H.264 standard is a new standard for encoding and compressing digital video content. The documents establishing the MPEG-4 Part **10** AVC/H.264 standard are hereby incorporated by reference, including "Joint Final Committee Draft (JFCD) of Joint Video Specification" issued by the Joint Video Team (JVT) on Aug. 10, 2002. (ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC). The JVT consists of experts from ISO or MPEG and ITU-T. Due to the public nature of the MPEG-4 Part **10** AVC/H.264 standard, the present specification will not attempt to document all the existing aspects of MPEG-4 Part **10** AVC/H.264 video coding, relying instead on the incorporated specifications of the standard.

Although this method of AFF encoding is compatible with and will be explained using the MPEG-4 Part **10** AVC/H.264 standard guidelines, it can be modified and used as best serves a particular standard or application.

Using the drawings, the preferred embodiments of the present invention will now be explained.

FIG. **1** illustrates an exemplary sequence of three types of pictures that can be used to implement the present invention, as defined by an exemplary video coding standard such as the MPEG-4 Part **10** AVC/H.264 standard. As previously mentioned, the encoder encodes the pictures and the decoder decodes the pictures. The encoder or decoder can be a processor, application specific integrated circuit (ASIC), field programmable gate array (FPGA), coder/decoder (CODEC), digital signal processor (DSP), or some other electronic device that is capable of encoding the stream of pictures. However, as used hereafter and in the appended claims, unless otherwise specifically denoted, the term "encoder" will be used to refer expansively to all electronic devices that encode digital video content comprising a

stream of pictures. The term "decoder" will be used to refer expansively to all electronic devices that decode digital video content comprising a stream of pictures.

As shown in FIG. **1**, there are preferably three types of pictures that can be used in the video coding method. Three types of pictures are defined to support random access to stored digital video content while exploring the maximum redundancy reduction using temporal prediction with motion compensation. The three types of pictures are intra (I) pictures (**100**), predicted (P) pictures (**102***a,b*), and bi-predicted (B) pictures (**101***a-d*). An I picture (**100**) provides an access point for random access to stored digital video content and can be encoded only-with slight compression. Intra pictures (**100**) are encoded without referring to reference pictures.

A predicted picture (**102***a,b*) is encoded using an I, P, or B picture that has already been encoded as a reference picture. The reference picture can be in either the forward or backward temporal direction in relation to the P picture that is being encoded. The predicted pictures (**102***a,b*) can be encoded with more compression than the intra pictures (**100**).

A bi-predicted picture (**101***a-d*) is encoded using two temporal reference pictures: a forward reference picture and a backward reference picture. The forward reference picture is sometimes called a past reference picture and the backward reference picture is sometimes called a future reference picture. An embodiment of the present invention is that the forward reference picture and backward reference picture can be in the same temporal direction in relation to the B picture that is being encoded. Bi-predicted pictures (**101***a-d*) can be encoded with the most compression out of the three picture types.

Reference relationships (**103**) between the three picture types are illustrated in FIG. **1**. For example, the P picture (**102***a*) can be encoded using the encoded I picture (**100**) as its reference picture. The B pictures (**101***a-d*) can be encoded using the encoded I picture (**100**) or the encoded P picture (**102***a*) as its reference pictures, as shown in FIG. **1**. Under the principles of an embodiment of the present invention, encoded B pictures (**101***a-d*) can also be used as reference pictures for other B pictures that are to be encoded. For example, the B picture (**101***c*) of FIG. **1** is shown with two other B pictures (**101***b* and **101***d*) as its reference pictures.

The number and particular order of the I (**100**), B (**101***a-d*), and P (**102***a,b*) pictures shown in FIG. **1** are given as an exemplary configuration of pictures, but are not necessary to implement the present invention. Any number of I, B, and P pictures can be used in any order to best serve a particular application. The MPEG-4 Part **10** AVC/H.264 standard does not impose any limit to the number of B pictures between two reference pictures nor does it limit the number of pictures between two I pictures.

FIG. **2** shows that each picture (**200**) is preferably divided into slices (**202**). A slice (**202**) comprises a group of macroblocks (**201**). A macroblock (**201**) is a rectangular group of pixels. As shown in FIG. **2**, a preferable macroblock (**201**) size is 16 by 16 pixels.

FIGS. **3***a-f* show that a macroblock can be further divided into smaller sized blocks. For example, as shown in FIGS. **3***a-f*, a macroblock can be further divided into block sizes of 16 by 8 pixels (FIG. **3***a*; **300**), 8 by 16 pixels (FIG. **3***b*; **301**), 8 by 8 pixels (FIG. **3***c*; **302**), 8 by 4 pixels (FIG. **3***d*; **303**), 4 by 8 pixels (FIG. **3***e*; **304**), or 4 by 4 pixels (FIG. **3***f*; **305**). These smaller block sizes are preferable in some applications that use the temporal prediction with motion compensation algorithm.

FIG. **4** shows a picture construction example using temporal prediction with motion compensation that illustrates an embodiment of the present invention. Temporal prediction with motion compensation assumes that a current picture, picture N (**400**), can be locally modeled as a translation of another picture, picture N-**1** (**401**). The picture N-**1** (**401**) is the reference picture for the encoding of picture N (**400**) and can be in the forward or backwards temporal direction in relation to picture N (**400**).

As shown in FIG. **4**, each picture is preferably divided into slices containing macroblocks (**201***a,b*). The picture N-**1** (**401**) contains an image (**403**) that is to be shown in picture N (**400**). The image (**403**) will be in a different temporal position in picture N (**402**) than it is in picture N-**1** (**401**), as shown in FIG. **4**. The image content of each macroblock (**201***b*) of picture N (**400**) is predicted from the image content of each corresponding macroblock (**201***a*) of picture N-**1** (**401**) by estimating the required amount of temporal motion of the image content of each macroblock (**201***a*) of picture N-**1** (**401**) for the image (**403**) to move to its new temporal position (**402**) in picture N (**400**). Instead of the original image (**402**) being encoded, the difference (**404**) between the image (**402**) and its prediction (**403**) is actually encoded and transmitted.

For each image (**402**) in picture N (**400**), the temporal prediction can often be described by motion vectors that represent the amount of temporal motion required for the image (**403**) to move to a new temporal position in the picture N (**402**). The motion vectors (**406**) used for the temporal prediction with motion compensation need to be encoded and transmitted.

FIG. **4** shows that the image (**402**) in picture N (**400**) can be represented by the difference (**404**) between the image and its prediction and the associated motion vectors (**406**). The exact method of encoding using the motion vectors can vary as best serves a particular application and can be easily implemented by someone who is skilled in the art.

To understand macroblock level AFF coding, a brief overview of picture level AFF coding of a stream of pictures will now be given. A frame of an interlaced sequence contains two fields, the top field and the bottom field, which are interleaved and separated in time by a field period. The field period is half the time of a frame period. In picture level AFF coding, the two fields of an interlaced frame can be coded jointly or separately. If they are coded jointly, frame mode coding is used. Conversely, if the two fields are coded separately, field mode coding is used.

Fixed frame/field coding, on the other hand, codes all the pictures in a stream of pictures in one mode only. That mode can be frame mode or it can be field mode. Picture level AFF is preferable to fixed frame/field coding in many applications because it allows the encoder to chose which mode, frame mode or field mode, to encode each picture in the stream of pictures based on the contents of the digital video material. AFF coding results in better compression than does fixed frame/field coding in many applications.

An embodiment of the present invention is that AFF coding can be performed on smaller portions of a picture. This small portion can be a macroblock, a pair of macroblocks, or a group of macroblocks. Each macroblock, pair of macroblocks, or group of macroblocks or slice is encoded in frame mode or in field mode, regardless of how the other macroblocks in the picture are encoded. AFF coding in each of the three cases will be described in detail below.

In the first case, AFF coding is performed on a single macroblock. If the macroblock is to be encoded in frame mode, the two fields in the macroblock are encoded jointly. Once encoded as a frame, the macroblock can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if the macroblock is to be encoded in field mode, the macroblock (**500**) is split into a top field (**501**) and a bottom field (**502**), as shown in FIG. **5**. The two fields are then coded separately. In FIG. **5**, the macroblock has M rows of pixels and N columns of pixels. A preferable value of N and M is 16, making the macroblock (**500**) a 16 by 16 pixel macroblock. As shown in FIG. **5**, every other row of pixels is shaded. The shaded areas represent the rows of pixels in the top field of the macroblock (**500**) and the unshaded areas represent the rows of pixels in the bottom field of the macroblock (**500**).

As shown in FIGS. **6***a-d*, a macroblock that is encoded in field mode can be divided into four additional blocks. A block is required to have a single parity. The single parity requirement is that a block cannot comprise both top and bottom fields. Rather, it must contain a single parity of field. Thus, as shown in FIGS. **6***a-d*, a field mode macroblock can be divided into blocks of 16 by 8 pixels (FIG. **6***a*; **600**), 8 by 8 pixels (FIG. **6***b*; **601**), 4 by 8 pixels (FIG. **6***c*; **602**), and 4 by 4 pixels (FIG. **6***d*; **603**). FIGS. **6***a-d* shows that each block contains fields of a single parity.

AFF coding on macroblock pairs will now be explained. AFF coding on macroblock pairs will be occasionally referred to as pair based AFF coding. A comparison of the block sizes in FIGS. **6***a-d* and in FIGS. **3***a-f* show that a macroblock encoded in field mode can be divided into fewer block patterns than can a macroblock encoded in frame mode. The block sizes of 16 by 16 pixels, 8 by 16 pixels, and 8 by 4 pixels are not available for a macroblock encoded in field mode because of the single parity requirement. This implies that the performance of single macroblock based AFF may not be good for some sequences or applications that strongly favor field mode coding. In order to guarantee the performance of field mode macroblock coding, it is preferable in some applications for macroblocks that are coded in field mode to have the same block sizes as macroblocks that are coded in frame mode. This can be achieved by performing AFF coding on macroblock pairs instead of on single macroblocks.

FIG. **7** illustrates an exemplary pair of macroblocks (**700**) that can be used in AFF coding on a pair of macroblocks according to an embodiment of the present invention. If the pair of macroblocks (**700**) is to be encoded in frame mode, the pair is coded as two frame-based macroblocks. In each macroblock, the two fields in each of the macroblocks are encoded jointly. Once encoded as frames, the macroblocks can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if the pair of macroblocks (**700**) is to be encoded in field mode, it is first split into one top field 16 by 16 pixel block (**800**) and one bottom field 16 by 16 pixel block (**801**), as shown in FIG. **8**. The two fields are then coded separately. In FIG. **8**, each macroblock in the pair of macroblocks (**700**) has N=16 columns of pixels and M=16 rows of pixels. Thus, the dimensions-of the pair of macroblocks (**700**) is 16 by 32 pixels. As shown in FIG. **8**, every other row of pixels is shaded. The shaded areas represent the rows of pixels in the top field of the macroblocks and the unshaded areas represent the rows of pixels in the bottom field of the macroblocks. The top field block (**800**) and the bottom field block (**801**) can now be divided into one of the possible block sizes of FIGS. **3***a-f*.

According to an embodiment of the present invention, in the AFF coding of pairs of macroblocks (**700**), there are two possible scanning paths. A scanning path determines the order in which the pairs of macroblocks of a picture are encoded. FIG. **9** shows the two possible scanning paths in AFF coding of pairs of macroblocks (**700**). One of the scanning paths is a horizontal scanning path (**900**). In the horizontal scanning path (**900**), the macroblock pairs (**700**) of a picture (**200**) are coded from left to right and from top to bottom, as shown in FIG. **9**. The other scanning path is a vertical scanning path (**901**). In the vertical scanning path (**901**), the macroblock pairs (**700**) of a picture (**200**) are coded from top to bottom and from left to right, as shown in FIG. **9**. For frame mode coding, the top macroblock of a macroblock pair (**700**) is coded first, followed by the bottom macroblock. For field mode coding, the top field macroblock of a macroblock pair is coded first followed by the bottom field macroblock.

Another embodiment of the present invention extends the concept of AFF coding on a pair of macroblocks to AFF coding on a group of four or more neighboring macroblocks (**902**), as shown in FIG. **10**. AFF coding on a group of macroblocks will be occasionally referred to as group based AFF coding. The same scanning paths, horizontal (**900**) and vertical (**901**), as are used in the scanning of macroblock pairs are used in the scanning of groups of neighboring macroblocks (**902**). Although the example shown in FIG. **10** shows a group of four macroblocks, the group can be more than four macroblocks.

If the group of macroblocks (**902**) is to be encoded in frame mode, the group coded as four frame-based macroblocks. In each macroblock, the two fields in each of the macroblocks are encoded jointly. Once encoded as frames, the macroblocks can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if a group of four macroblocks (**902**), for example, is to be encoded in field mode, it is first split into one top field 32 by 16 pixel block and one bottom field 32 by 16 pixel block. The two fields are then coded separately. The top field block and the bottom field block can now be divided into macroblocks. Each macroblock is further divided into one of the possible block sizes of FIGS. **3***a-f*. Because this process is similar to that of FIG. **8**, a separate figure is not provided to illustrate this embodiment.

In AFF coding at the macroblock level, a frame/field flag bit is preferably included in a picture's bitstream to indicate which mode, frame mode or field mode, is used in the encoding of each macroblock. The bitstream includes information pertinent to each macroblock within a stream, as shown in FIG. **11**. For example, the bitstream can include a picture header (**110**), run information (**111**), and macroblock type (**113**) information. The frame/field flag (**112**) is preferably included before each macroblock in the bitstream if AFF is performed on each individual macroblock. If the AFF is performed on pairs of macroblocks, the frame/field flag (**112**) is preferably included before each pair of macroblock in the bitstream. Finally, if the AFF is performed on a group of macroblocks, the frame/field flag (**112**) is preferably included before each group of macroblocks in the bitstream. One embodiment is that the frame/field flag (**112**) bit is a 0 if frame mode is to be used and a 1 if field coding is to be used. Another embodiment is that the frame/field flag (**112**) bit is a 1 if frame mode is to be used and a 0 if field coding is to be used.

Another embodiment of the present invention entails a method of determining the size of blocks into which the encoder divides a macroblock in macroblock level AFF. A preferable, but not exclusive, method for determining the ideal block size is sum absolute difference (SAD) with or without bias or rate distortion (RD) basis. For example, SAD checks the performance of the possible block sizes and chooses the ideal block size based on its results. The exact method of using SAD with or without bias or RD basis can be easily be performed by someone skilled in the art.

According to an embodiment of the present invention, each frame and field based macroblock in macroblock level AFF can be intra coded or inter coded. In intra coding, the macroblock is encoded without temporally referring to other macroblocks. On the other hand, in inter coding, temporal prediction with motion compensation is used to code the macroblocks.

If inter coding is used, a block with a size of 16 by 16 pixels, 16 by 8 pixels, 8 by 16 pixels, or 8 by 8 pixels can have its own reference pictures. The block can either be a frame or field based macroblock. The MPEG-4 Part **10** AVC/H.264 standard allows multiple reference pictures instead of just two reference pictures. The use of multiple reference pictures improves the performance of the temporal prediction with motion compensation algorithm by allowing the encoder to find a block in the reference picture that most closely matches the block that is to be encoded. By using the block in the reference picture in the coding process that most closely matches the block that is to be encoded, the greatest amount of compression is possible in the encoding of the picture. The reference pictures are stored in frame and field buffers and are assigned reference frame numbers and reference field numbers based on the temporal distance they are away from the current picture that is being encoded. The closer the reference picture is to the current picture that is being stored, the more likely the reference picture will be selected. For field mode coding, the reference pictures for a block can be any top or bottom field of any of the reference pictures in the reference frame or field buffers.

Each block in a frame or field based macroblock can have its own motion vectors. The motion vectors are spatially predictive coded. According to an embodiment of the present invention, in inter coding, prediction motion vectors (PMV) are also calculated for each block. The algebraic difference between a block's PMVs and its associated motion vectors is then calculated and encoded. This generates the compressed bits for motion vectors.

FIG. **12** will be used to explain various preferable methods of calculating the PMV of a block in a macroblock. A current block, E, in FIG. **12** is to be inter coded as well as its neighboring blocks A, B, C, and D. E will refer hereafter to a current block and A, B, C, and D will refer hereafter to E's neighboring blocks, unless otherwise denoted. Block E's PMV is derived from the motion vectors of its neighboring blocks. These neighboring blocks in the example of FIG. **12** are A, B, C, and D. One preferable method of calculating the PMV for block E is to calculate either the median of the motion vectors of blocks A, B, C, and D, the average of these motion vectors, or the weighted average of these motion vectors. Each of the blocks A through E can be in either frame or field mode.

Another preferable method of calculating the PMV for block E is to use a yes/no method. Under the principles of the yes/no method, a block has to be in the same frame or field coding mode as block E in order to have its motion vector included in the calculation of the PMV for E. For example, if block E in FIG. **12** is in frame mode, block A must also be in frame mode to have its motion vector included in the calculation of the PMV for block E. If one of E's neighboring blocks does not have the same coding mode as does block E, its motion vectors are not used in the calculation of block E's PMV.

The "always method" can also be used to calculate the PMV for block E. In the always method, blocks A, B, C, and D are always used in calculating the PMV for block E, regardless of their frame or field coding mode. If E is in frame mode and a neighboring block is in field mode, the vertical component of the neighboring block is multiplied by 2 before being included in the PMV calculation for block E. If E is in field mode and a neighboring block is in frame mode, the vertical component of the neighboring block is divided by 2 before being included in the PMV calculation for block E.

The "selective method" can also be used to calculate the PMV for block E if the macroblock has been encoded using pair based AFF encoding or group based AFF encoding. In the selective method, a frame-based block has a frame-based motion vector pointing to a reference frame. The block is also assigned a field-based motion vector pointing to a reference field. The field-based motion vector is the frame-based motion vector of the block with the vertical motion vector component divided by two. The reference field number is the reference frame number multiplied by two. A field-based block has a field-based motion vector pointing to a reference field. The block is also assigned a frame-based motion vector pointing to a reference frame. The frame-based motion vector is the field-based motion vector of the block with the vertical motion vector component multiplied by two. The reference frame number is the reference field number divided by two.

The derivation of a block's PMV using the selective method will now be explained using FIG. **12** as a reference. In macroblock pair based AFF, each block in a macroblock is associated with a companion block that resides in the same geometric location within the second macroblock of the macroblock pair. In FIG. **12**, each of block E's neighboring blocks (A, B, C, and D) may or may not be in the same frame or field coding mode as block E. Hence, the following rules apply:

If E is in frame mode and a neighboring block is in frame mode, the true frame-based motion vector of the neighboring block is used for E's PMV.

If E is in frame mode and a neighboring block is in field mode, the following rules apply in calculating E's PMV. If the neighboring block (e.g.; block A) and its companion field-based block have the same reference field, the average of the assigned frame-based motion vectors of the two blocks is used for the calculation of E's PMV. The reference frame number used for the PMV calculation is the reference field number of the neighboring block divided by two. However, if the neighboring block and its companion field block have different reference fields, then the neighboring block cannot be used in the calculation of E's PMV.

If E is in field mode and a neighboring block is in frame mode, the following rules apply in calculating E's PMV. If the neighboring block (e.g.; block A) and its companion frame-based block have the same reference frame, the average of the assigned field-based motion vectors of the two blocks is used for the calculation of E's PMV. The reference field number used for the PMV calculation is the reference frame number of the neighboring block multiplied by two. However, if the neighboring block and its compan-

ion field block have different reference frames, then the neighboring block cannot be used in the calculation of E's PMV.

If E is in field mode and a neighboring block is in field mode, the true field-based motion vector of the neighboring block is used in the calculation of E's PMV.

An alternate preferable option can be used in the selective method to calculate a block's PMV. In FIG. **12**, each of block E's neighboring blocks (A, B, C, and D) may or may not be in the same frame or field coding mode as block E. Hence, the following rules apply for this alternate preferable option of the selective method:

If E is in frame mode and a neighboring block is in frame mode, the true frame-based motion vector of the neighboring block is used for E's PMV.

If E is in frame mode and a neighboring block is in field mode, the weighted average of the assigned frame-based motion vectors of the neighboring block and its companion field-based block is used forte calculation of E's PMV. The weighting factors are based upon the reference field numbers of to neighboring block and its companion block.

If E is in field mode, and a neighboring block is in frame mode, the weighted average of the assigned field-based motion vectors of the neighboring block and its companion frame-based block is used for the calculation of E's PMV. The weighting factors are based upon the reference frame numbers of the neighboring block and its companion block.

If E is in field mode and a neighboring block is in field mode, the true field-based motion vector of the neighboring block is used in the calculation of E's PMV.

Another preferable method of computing a block's PMV is the "alt selective method." This method can be used in single macroblock AFF coding, pair based macroblock AFF coding, or group based AFF coding. In this method, each block is assigned a horizontal and a vertical index number, which represents the horizontal and vertical coordinates of the block. Each block is also assigned a horizontal and vertical field coordinate. A block's horizontal field coordinate is same as its horizontal coordinate. For a block in a top field macroblock, the vertical field coordinate is half of vertical coordinate of the block and is assigned top field polarity. For a block in the bottom field macroblock, the vertical field coordinate of the block is obtained by subtracting 4 from the vertical coordinate of the block and dividing the result by 2. The block is also assigned bottom field polarity. The result of assigning different field polarities to two blocks is that there are now two blocks with the same horizontal and vertical field coordinates but with differing field polarities. Thus, given the coordinates of a block, the field coordinates and its field polarity can be computed and vice versa.

The alt selective method will now be explained in detail using FIG. **12** as a reference. The PMV of block E is to be computed. Let bx represent the horizontal size of block E divided by 4, which is the size of a block in this example. The PMVs for E are obtained as follows depending on whether E is in frame/field mode.

Let block E be in frame mode and let (x,y) represent the horizontal and vertical coordinates respectively of E. The neighboring blocks of E are defined in the following manner. A is the block whose coordinates are (x−1,y). B is the block whose coordinates are (x,y−1). D is the block whose coordinates are (x−1,y−1). C is the block whose coordinates are (x+bx+1,y−1). If either A, B, C or D is in field mode then its vertical motion vector is multiplied by 2 before being used for prediction and its reference frame number is computed by dividing its reference field by 2.

Now, let black E be in top or bottom field mode and let (xf,yf) represent the horizontal and vertical field coordinates respectively of E. In this case, the neighbors of E are defined as follows. A is the block whose field coordinates are (xf−1,yf) and has sonic polarity as E. B is the block whose field coordinates are (xf,yf−1) and has same polarity as E. D is the block whose field coordinates are (xf−1,yf−1) and has same polarity as E. C is the block whose field coordinates are (xf+bx+1,yf) and has same polarity as E. If either A,B,C or D is in frame mode then its vertical motion vector is divided by 2 before being used for prediction and its reference field is computed by multiplying its reference frame by 2.

In all of the above methods for determining the PMV of a block, a horizontal scanning path was assumed. However, the scanning path can also be a vertical scanning path. In this case, the neighboring blocks of the current block, E, are defined as shown in FIG. **13**. A vertical scanning path is preferable in some applications because the information on all the neighboring blocks is available for the calculation of the PMV for the current block E.

Another embodiment of the present invention is directional segmentation prediction. In directional segmentation prediction, 16 by 8 pixel blocks and 8 by 16 pixel blocks have rules that apply to their PMV calculations only. These rules apply in all PMV calculation methods for these block sizes. The rules will now be explained in detail in connection with FIG. **12**. In each of these rules, a current block E is to have its PMV calculated.

First, a 16 by 16 pixel block consists of an upper block and a lower block. The upper block contains the top 8 rows of 16 pixels. The lower block contains the bottom 8 rows of 16 pixels. In the following description, block E of FIG. **12** is a 16 by 8 pixel block. For the upper block having a 16 by 8 pixel block, block B is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV. For the lower block having a 16 by 8 pixel block, block A is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV.

A 16 by 16 pixel block is divided into a right and left block. Both right and left blocks are 8 by 16 pixels. In the following description, block E of FIG. **12** is a 8 by 16 pixel block. For the left block, block A is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV. For the right block, block C is used to predict block E's PMV if it has the same referenced picture as block E. Otherwise median prediction is used to predict block E's PMV.

For both 16 by 8 pixel blocks and 8 by 16 blocks, A, B, or C can be in different encoding modes (frame or field) than the current block E. The following rules apply for both block sizes. If E is in frame mode, and A, B, or C is in field mode, the reference frame number of A, B, or C is computed by dividing its reference field by 2. If E is in field mode, and A, B, or C is in frame mode, the reference field number of A, B, or C is computed by multiplying its reference frame by 2.

According to another embodiment of the present invention, a macroblock in a P picture can be skipped in AFF coding. If a macroblock is skipped, its data is not transmitted in the encoding of the picture. A skipped macroblock in a P picture is reconstructed by copying the co-located macroblock in the most recently coded reference picture. The co-located macroblock is defined as the one with motion compensation using PMV as defined above or without

motion vectors. The following rules apply for skipped macroblocks in a P picture. If AFF coding is performed per macroblock, a skipped macroblock is in frame mode. If AFF coding is performed on macroblock pairs and if both macroblocks are skipped, then they are in frame mode. However, if only one of the macroblocks in a macroblock pair is skipped, its frame or field coding mode is the same as the non-skipped macroblock in the same macroblock pair. If AFF coding is performed on a group of macroblocks and if the entire group of macroblocks is skipped, then all the macroblocks are in frame mode. If there is at least one macroblock that is not skipped, then the skipped macroblocks in the same group are in the same frame or field coding mode as the non-skipped macroblock.

An alternate method for skipped macroblocks is as follows. If a macroblock pair is skipped, its frame and field coding mode follows its neighboring macroblock pair to the left. If the left neighboring macroblock pair is not available, its coding mode follows its neighboring macroblock pair to the top. If neither the left nor top neighboring macroblock pairs are available, the skipped macroblock is set to frame mode.

Another embodiment of the present invention is direct mode macroblock coding for B pictures. In direct mode coding, a B picture has two motion vectors, forward and backward motion vectors. Each motion vector points to a reference picture. Both the forward and backward motion vectors can point in the same temporal direction. For direct mode macroblock coding in B pictures, the forward and backward motion vectors of a block are calculated from the co-located block in the backward reference picture. The co-located block in the backward reference picture can be frame mode or field mode coded. The following rules apply in direct mode macroblock coding for B picture.

If the co-located block is in frame mode and if the current direct mode macroblock is also in frame mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block. The forward reference frame is the one used by the co-located block. The backward reference frame is the same frame where the co-located block resides.

If the co-located block is in frame mode and if the current direct mode macroblock is in field mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block's motion vector with vertical component divided by two. The forward reference field is the same parity field of the reference frame used by the co-located block. The backward reference field is the same parity field of the backward reference frame where the co-located block resides.

If the co-located block is in field mode and if the current direct mode macroblock is also in field mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block of the same field parity. The forward reference field is the field used by the co-located block. The backward reference field is the same field where the co-located block resides.

If the co-located block is in field mode and if the current direct mode macroblock is in frame mode, the two associated motion vectors of the block in the direct mode macroblock are calculated from the co-located block's motion vector with vertical component multiplied by two. The forward reference frame is the frame one of whose fields is used by the co-located block. The backward reference field is the frame in one of whose fields the co-located block resides.

An alternate option is to force the direct mode block to be in the same frame or field coding mode as the co-located block. In this case, if the co-located block for a direct mode block is in frame mode, the direct mode block is in frame mode as well. The two frame-based motion vectors of the direct mode block are derived from the frame-based forward motion vector of the co-located block. The forward reference frame is used by the co-located block. The backward reference frame is where the co-located block resides.

However, if the co-located block for a block in direct mode is in field mode, the direct mode block is also in field mode. The two field-based motion vectors of the direct mode block are derived from the field-based forward motion vector of the co-located block. The forward reference field is used by the co-located block. The backward reference field is where the co-located block resides.

A macroblock in a B picture can also be skipped in AFF coding according to another embodiment of the present invention. A skipped macroblock in a B picture is reconstructed as a regular direct mode macroblock without any coded transform coefficient information. For skipped macroblocks in a B picture, the following rules apply. If AFF coding is performed per macroblock, a skipped macroblock is either in frame mode or in the frame or field coding mode of the co-located block in its backward reference picture. If AFF coding is performed on macroblock pairs and if both macroblocks are skipped, then they are in frame mode or in the frame or field coding mode of the co-located macroblock pair in the its backward reference picture. However, if only one of the macroblocks in a macroblock pair is skipped, its frame or field coding mode is the same as the non-skipped macroblock of the same macroblock pair. If AFF coding is performed on a group of macroblocks and if the entire group of macroblocks is skipped, then all the macroblocks are in frame mode or in the frame or field coding mode of the co-located group of macroblocks in the backward reference picture. If there is at least one macroblock that is not skipped, then the skipped macroblock in the same group are in the same frame or field coding mode as the non-skipped macroblock.

As previously mentioned, a block can be intra coded. Intra blocks are spatially predictive coded. There are two possible intra coding modes for a macroblock in macroblock level AFF coding. The first is intra_4×4 mode and the second is intra_16×16 mode. In both, each pixel's value is predicted using the real reconstructed pixel values from neighboring blocks. By predicting pixel values, more compression can be achieved. The intra_4×4 mode and the intra_16×16 modes will each be explained in more detail below.

For intra_4×4 mode, the predictions of the pixels in a 4 by 4 pixel block, as shown in FIG. **14**, are derived form its left and above pixels. In FIG. **14**, the 16 pixels in the 4 by 4 pixel block are labeled a through p. Also shown in FIG. **14** are the neighboring pixels A through P. The neighboring pixels are in capital letters. As shown in FIG. **15**, there are nine different prediction directions for intra_4×4 coding. They are vertical (**0**), horizontal (**1**), DC prediction (mode **2**), diagonal down/left (**3**), diagonal down/right (**4**), vertical-left (**5**), horizontal-down (**6**), vertical-right (**7**), and horizontal-up (**8**). DC prediction averages all the neighboring pixels together to predict a particular pixel value.

However, for intra_16×16 mode, there are four different prediction directions. Prediction directions are also referred to as prediction modes. These prediction directions are vertical prediction (**0**), horizontal prediction (**1**), DC prediction, and plane prediction. Plane prediction will not be explained.

An intra block and its neighboring blocks may be coded in frame or field mode. Intra prediction is performed on the reconstructed blocks. A reconstructed block can be represented in both frame and field mode, regardless of the actual frame or field coding mode of the block. Since only the pixels of the reconstructed blocks are used for intra prediction, the following rules apply.

If a block of 4 by 4 pixels or 16 by 16 pixels is in frame mode, the neighboring pixels used in calculating the pixel value predictions of the block are in the frame structure. If a block of 4 by 4 pixels or 16 by 16 pixels is in field mode, the neighboring pixels used in calculating the pixel value prediction of the block are in field structure of the same field parity.

The chosen intra-prediction mode (intra_pred_mode) of a 4 by 4 pixel block is highly correlated with the prediction modes of adjacent blocks. This is illustrated in FIGS. **16***a*. FIG. **16***a* shows that A and B are adjacent blocks to C. Block C's prediction mode is to be established. FIG. **16***b* shows the order of intra prediction information in the bitstream. When the prediction modes of A and B are known (including the case that A or B or both are outside the slice) the most probable prediction mode (most_probable_mode) of C is given. If one of the blocks A or B is "outside" the most probable prediction mode is equal DC prediction (mode **2**). Otherwise it is equal to the minimum of prediction modes used for blocks A and B. When an adjacent block is coded by 16×16 intra mode, prediction mode is DC prediction mode. When an adjacent block is coded a non-intra macroblock, prediction mode is "mode **2**: DC prediction" in the usual case and "outside" in the case of constrained intra update.

To signal a prediction mode number for a 4 by 4 block first parameter use_most_probable_mode is transmitted. This parameter is represented by 1 bit codeword and can take values 0 or 1. If use_most_probable_mode is equal to 1 the most probable mode is used. Otherwise an additional parameter remaining_mode_selector, which can take value from 0 to 7 is sent as 3 bit codeword. The codeword is a binary representation of remaining_mode_selector value. The prediction mode number is calculated as:

```
if (remaining_mode_selector<most_probable_mode)
intra_pred_mode=remaining_mode_selector;
else
intra_pred_mode=remaining_mode_selector+1;
```

The ordering of prediction modes assigned to blocks C is therefore the most probable mode followed by the remaining modes in the ascending order.

An embodiment of the present invention includes the following rules that apply to intra mode prediction for an intra-prediction mode of a 4 by 4 pixel block or an intra-prediction mode of a 16 by 16 pixel block. Block C and its neighboring blocks A and B can be in frame or field mode. One of the following rules shall apply. FIGS. **16***a-b* will be used in the following explanations of the rules.

Rule 1: A or B is used as the neighboring block of C only if A or B is in the same frame/field mode as C. Otherwise, A or B is considered as outside.

Rule 2: A and B are used as the neighboring blocks of C, regardless of their frame/field coding mode.

Rule 3: If C is coded in frame mode and has co-ordinates (x,y), then A is the block with co-ordinates (x,y−1) and B is the block with co-ordinates (x−1,y). Otherwise, if C is coded as field and has field co-ordinates (xf,yf) then A is the block whose field co-ordinates are (xf,yf−1) and has same field polarity as C and B is the block whose field co-ordinates are (xf−1,yf) and has same field polarity as C.

Rule 4: This rule applies to macroblock pairs only. In the case of decoding the prediction modes of blocks numbered **3**, **6**, **7**, **9**, **12**, **13**, **11**, **14** and **15** of FIG. **16***b*, the above and the left neighboring blocks are in the same macroblock as the current block. However, in the case of decoding the prediction modes of blocks numbered **1**, **4**, and **5**, the top block (block A) is in a different macroblock pair than the current macroblock pair. In the case of decoding the prediction mode of blocks numbered **2**, **8**, and **10**, the left block (block B) is in a different macroblock pair. In the case of decoding the prediction mode of the block numbered **0**, both the left and the above blocks are in different macroblock pairs. For a macroblock in field decoding mode the neighboring blocks of the blocks numbered **0**, **1**, **4**, **5**, **2**, **8**, and **10** shall be defined as follows:

If the above macroblock pair (**170**) is decoded in field mode, then for blocks number **0**, **1**, **4** and **5** in the top-field macroblock (**173**), blocks numbered **10**, **11**, **14** and **15** respectively in the top-field macroblock (**173**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**) as shown in FIG. **17***a*. For blocks number **0**, **1**, **4** and **5** in the bottom-field macroblock (**174**), blocks numbered **10**, **11**, **14** and **15** respectively in the bottom-field MB of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***a*.

However, if the above macroblock pair (**170**) is decoded in frame mode then for blocks number **0**, **1**, **4** and **5** in the top-field macroblock (**173**), blocks numbered **10**, **11**, **14** and **15** respectively in the bottom-frame macroblock (**176**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***b*. For blocks number **0**, **1**, **4** and **5** in the bottom-field macroblock (**174**), blocks numbered **10**, **11**, **14** and **15** respectively in the bottom-frame macroblock (**176**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown inn FIG. **17***b*.

If the left macroblock pair (**172**) is decoded in field mode, then for blocks number **0**, **2**, **8** and **10** in the top-field macroblock (**173**), blocks numbered **5**, **7**, **13** and **15** respectively in the top-field macroblock (**173**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**) as shown in FIG. **17***c*. For blocks number **0**, **2**, **8** and **10** in the bottom-field macroblock (**174**), blocks numbered **5**, **7**, **13** and **15** respectively in the bottom-field macroblock (**174**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***c*.

If the left macroblock pair (**172**) is decoded in frame mode, then for blocks number **0**, **2**, **8** and **10** in the top-field macroblock (**173**), the blocks numbered **5**, **7**, **13** and **15** respectively in the top-frame macroblock (**175**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***d*. For blocks number **0**, **2**, **8** and **10** in the bottom-field macroblock (**174**), blocks numbered **5**, **7**, **13** and **15** respectively in the bottom-frame macroblock (**176**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***d*.

For macroblock pairs on the upper boundary of a slice, if the left macroblock pair (**172**) is in frame decoding mode, then the intra mode prediction value used to predict a field macroblock shall be set to DC prediction.

The preceding descriptions of intra coding and intra mode prediction can be extended to adaptive block transforms.

Another embodiment of the present invention is that loop filtering is performed on the reconstructed blocks. A reconstructed block can be represented in either frame or field structure, regardless of the frame/field coding mode of the block. Loop (deblock) filtering is a process of weighted averaging of the pixels of the neighboring blocks. FIG. **12** will be used to explain loop filtering. Assume E of FIG. **12** is a reconstructed block, and A, B, C and D are its neighboring reconstructed blocks, as shown in FIG. **12**, and they are all represented in frame structure. Since A, B, C, D and E can be either frame- or field-coded, the following rules apply:

Rule 1: If E is frame-coded, loop filtering is performed over the pixels of E and its neighboring blocks A B, C and D.

Rule 2: If E is field-coded, loop filtering is performed over the top-field and bottom-field pixels of E and its neighboring blocks A B, C and D, separately.

Another embodiment of the present invention is that padding is performed on the reconstructed frame by repeating the boundary pixels. Since the boundary blocks may be coded in frame or field mode, the following rules apply:

Rule 1: The pixels on the left or right vertical line of a boundary block are repeated, if necessary.

Rule 2: If a boundary block is in frame coding, the pixels on the top or bottom horizontal line of the boundary block are repeated.

Rule 3: if a boundary block is in field coding, the pixels on the two top or two bottom horizontal (two field) lines of the boundary block are repeated alternatively.

Another embodiment of the present invention is that two-dimensional transform coefficients are converted into one-dimensional series of coefficients before entropy coding. The scan path can be either zigzag or non-zigzag. The zigzag scanner is preferably for progressive sequences, but it may be also used for interlace sequences with slow motions. The non-zigzag scanners are preferably for interlace sequences. For macroblock AFF coding, the following options may be used:

Option 1: The zigzag scan is used for macroblocks in frame mode while the non-zigzag scanners are used for macroblocks in field coding.

Option 2: The zigzag scan is used for macroblocks in both frame and field modes.

Option 3: The non-zigzag scan is used for macroblocks in both frame and field modes.

The preceding description has been presented only to illustrate and describe embodiments of invention. It is not intended to be exhaustive or to limit the invention to any precise form disclosed. Many modifications and variations are possible in light of the above teaching.

The foregoing embodiments were chosen and described in order to illustrate principles of the invention and some practical applications. The preceding description enables others skilled in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims.

What is claimed is:

1. A method of encoding a picture in an image sequence, comprising:

dividing-said picture into a plurality of smaller portions, wherein each of said smaller portions has a size that is larger than one macroblock;

selectively encoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode; and

selectively encoding at least one block within said at least one of said plurality of smaller portions at a time in intra coding mode.

2. The method of claim **1**, wherein said intra coding mode employs spatially predictive coding for a current block in accordance with a plurality of neighboring blocks to said current block.

3. An apparatus of encoding a picture in an image sequence, comprising:

means for dividing said picture into a plurality of smaller portions, wherein each of said smaller portions has a size that is larger than one macroblock;

means for selectively encoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode; and

means for selectively encoding at least one block within at least one of said plurality of smaller portions at a time in intra coding mode.

4. The apparatus of claim **3**, wherein said intra coding mode employs spatially predictive coding for a current block in accordance with a plurality of neighboring blocks to said current block.

5. A computer-readable medium encoded with computer executable instructions, the computer executable instructions including instructions which, when executed by a processor, cause the processor to perform the steps of-a method of encoding a picture in an image sequence, comprising the steps of:

dividing said picture into a plurality of smaller portions, wherein each of said smaller portions has a size that is larger than one macroblock;

selectively encoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode; and

selectively encoding at least one block within at least one of said plurality of smaller portions at a time in intra coding mode.

6. A method of decoding an encoded picture having a plurality of smaller portions from a bitstream, comprising:

selectively decoding at least one of a plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode, wherein each of said smaller portions has a size that is larger than one macroblock, wherein at least one block within said at least one of said plurality of smaller portions is encoded in intra coding mode at a time; and

using said plurality of decoded smaller portions to construct a decoded picture.

7. The method of claim **6**, wherein said intra coding mode employs spatially predictive coding for a current block in accordance with a plurality of neighboring blocks to said current block.

8. The method of claim **7**, wherein for said current block, said neighboring blocks comprises at least one of a neighboring block that is left of said current block to be encoded and a neighboring block that is above said current block to be encoded.

9. The method of claim **8**, wherein one of a plurality of prediction directions is deemed to be a most probable mode for said current block.

10. The method of claim 9, further comprising: receiving at least one codeword in said bitstream, wherein said at least one codeword indicates if said most probable prediction coding mode is used.

11. The method of claim 9, wherein said most probable prediction mode for a current block is selected in accordance with a neighboring block that is left of said current block to be encoded and a neighboring block that is above said current block to be encoded, wherein if one of said neighboring blocks is outside a slice, then said most probable prediction mode for said current block is DC prediction, and wherein if both of said neighboring blocks are inside said slice, then said most probable prediction mode for said current block is selected in accordance with a minimum of prediction modes used for said left neighboring block and said above neighboring block.

12. The method claim 6, wherein a size of said current block is selected in accordance with any block size defined in adaptive block transforms.

13. An apparatus for decoding an encoded picture from a bitstream, comprising:

means for selectively decoding at least one of a plurality of smaller portions at a time of the encoded picture that is encoded in frame coding mode and at least one of said plurality of smaller portions at a time of the encoded picture in field coding mode, wherein each of said smaller portions has a size that is larger than one macroblock, wherein at least one block within at least one of said plurality of smaller portions is encoded in intra coding mode at a time; and

means for using said plurality of decoded smaller portions to construct a decoded picture.

14. The apparatus of claim 13, wherein said intra coding mode employs spatially predictive coding for a current block in accordance with a plurality of neighboring blocks to said current block.

15. The apparatus claim 14, wherein a size of said current block is selected in accordance with any block size defined in adaptive block transforms.

16. The apparatus of claim 14, wherein for said current block, said neighboring blocks comprises at least one of a neighboring block that is left of said current block to be encoded and a neighboring block that is above said current block to be encoded.

17. A computer-readable medium encoded with computer executable instructions, the computer executable instructions including instructions which, when executed by a processor, cause the processor to perform the steps of a method for decoding an encoded picture having a plurality of smaller portions from a bitstream, comprising the steps of:

selectively decoding at least one of said plurality of smaller portions at a time in frame coding mode and at least one of said plurality of smaller portions at a time in field coding mode, wherein each of said smaller portions has a size that is larger than one macroblock, wherein at least one block within said at least one of said plurality of smaller portions is encoded in intra coding mode at a time; and

using said plurality of decoded smaller portions to construct a decoded picture.

18. A bitstream comprising:

a picture that has been divided into a plurality of smaller portions, wherein at least one of said plurality of smaller portions at a time is encoded in frame coding mode and at least one of said plurality of smaller portions at a time is encoded in field coding mode, wherein each of said smaller portions has a size that is larger than one macroblock, and wherein at least one block within at least one of said plurality of smaller portions is encoded in intra coding mode at a time.

* * * * *

EXHIBIT C

US007310376B2

(12) **United States Patent**
Wang et al.

(10) **Patent No.: US 7,310,376 B2**
(45) **Date of Patent: Dec. 18, 2007**

(54) **MACROBLOCK LEVEL ADAPTIVE FRAME/FIELD CODING FOR DIGITAL VIDEO CONTENT**

(75) Inventors: **Limin Wang**, San Diego, CA (US); **Rajeev Gandhi**, San Diego, CA (US); **Krit Panusopone**, San Diego, CA (US); **Ajay Luthra**, San Diego, CA (US)

(73) Assignee: **General Instrument Corporation**, Horsham, PA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 100 days.

(21) Appl. No.: **11/026,394**

(22) Filed: **Dec. 30, 2004**

(65) **Prior Publication Data**

US 2005/0123043 A1 Jun. 9, 2005

Related U.S. Application Data

(62) Division of application No. 10/301,290, filed on Nov. 20, 2002, now Pat. No. 6,980,596.

(60) Provisional application No. 60/398,161, filed on Jul. 23, 2002, provisional application No. 60/395,734, filed on Jul. 12, 2002, provisional application No. 60/333,921, filed on Nov. 27, 2001.

(51) **Int. Cl.**
H04B 1/66 (2006.01)

(52) **U.S. Cl.** **375/240.24**; 375/240.25; 375/240.02; 375/240.26; 382/233; 382/235; 382/239

(58) **Field of Classification Search** 375/240.24, 375/240.25, 240.23, 240.02, 240.26; 382/233, 382/235, 239, 246, 245; 348/206
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,437,119	A	3/1984	Matsumoto et al.	
5,412,428	A	5/1995	Tahara	
5,504,530	A *	4/1996	Obikane et al.	375/240.14
5,801,778	A	9/1998	Ju	
5,825,419	A *	10/1998	Mishima et al.	375/240.15
6,094,225	A	7/2000	Han	
6,192,148	B1	2/2001	Lin	
6,404,813	B1	6/2002	Haskell et al.	

OTHER PUBLICATIONS

"Core Experiment on Interlaced Video Coding", Peter Borgwart, VideoTele.com—A Tektronix Company, Study Group 16, Question 6.

"Adaptive field/frame block coding experiment proposal", Interested Parties for the Study of Interlaced Video Coding with H.26L,Video Coding Experts Group, Study Group 16.

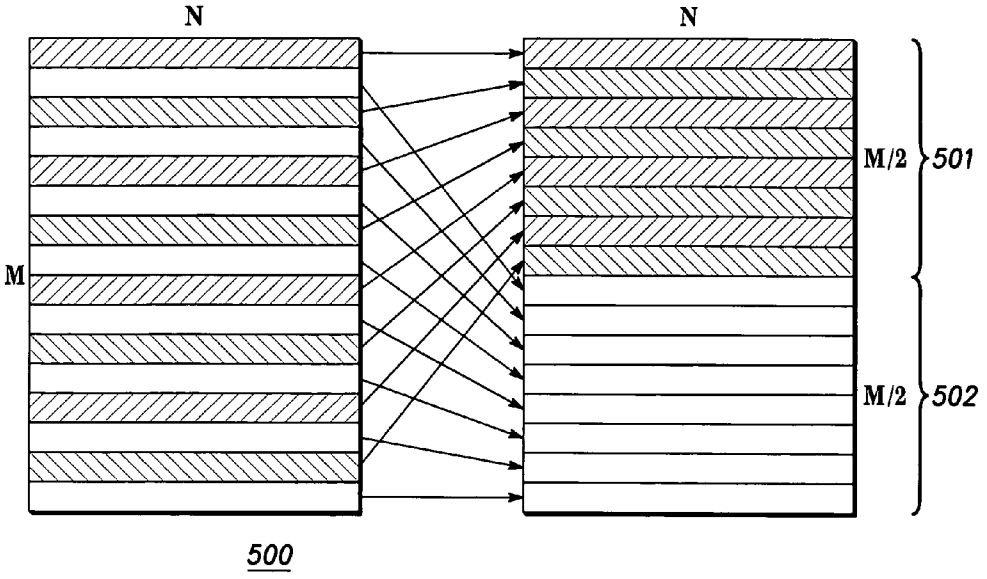
(Continued)

Primary Examiner—Shawn S. An
(74) *Attorney, Agent, or Firm*—Larry T. Cullen

(57) **ABSTRACT**

A method and system of encoding and decoding digital video content. The digital video content comprises a stream of pictures which can each be intra, predicted, or bi-predicted pictures. Each of the pictures comprises macroblocks that can be further divided into smaller blocks. The method entails encoding and decoding each of the smaller blocks in each picture in said stream of pictures in either frame mode or in field mode.

31 Claims, 8 Drawing Sheets



OTHER PUBLICATIONS

P.Borgwardt: "Core Experiment on Interlaced Video Coding VCEG-N85" ITU-Telecommunications Standardization Sector ITU-T Q.6/SG16 Video Coding Expert Group (VCEG) 24-27 Sep. 2001, pp. 1-10, XP002257037 Santa Barbara, CA USA.

"H.26L Test Model Long Term Number 8 (TML-8) Drafto" ITU-T Telecommunication Standardization Sector of ITU, Geneva, CH, 2 Apr. 2001. (Apr. 2, 2001), pp.1-54, XP001089814 p. 9, paragraph 1 - p. 10, paragraph 6.1.2.2.1 p p. 40.

P. Borgwardt: "Handling Interlaced Video in H.26l VCEG-N57" ITU-Telecommunications Standardization Sector ITU Q.6/SG16 Video Coding Expert Group (VCEG), 24-27 Sep. 2001, pp. 1-3, XP002257142 Santa Barbara, CA USA.

M. Gallant et al: High Rate, High Resolution Video Using H.26L VCEG-N84 ITU-Telecommunications Standarization Secotr ITU Q.6/SG16 Video Coding Expert Group (VCEG), 24-27 Sep. 2001, pp. 1-7, XP002257143 Santa Barbara, CA USA; p. 1; p. 3.

"Adaptive Field/From Block Coding Experiment Proposal". Interested Parties for the Study of Interfaced Video Coding with H.26L, Video Coding Experts Group, Study Group 16.

* cited by examiner

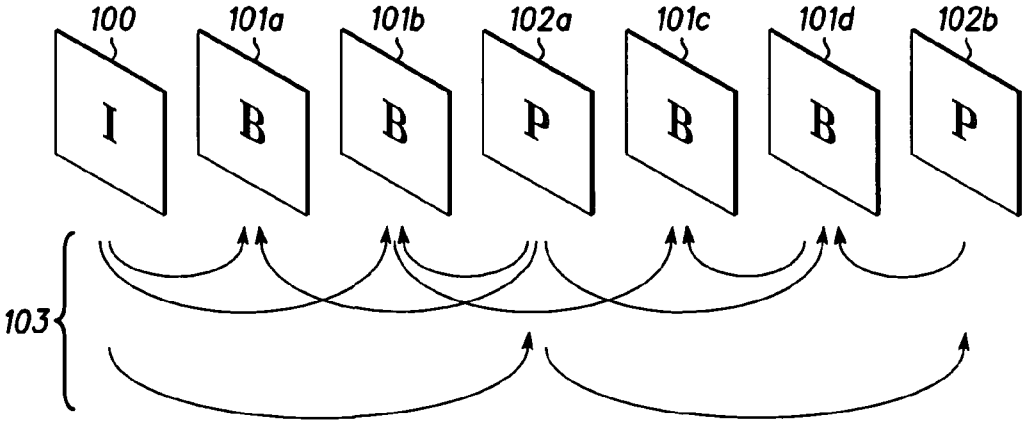


FIG. 1

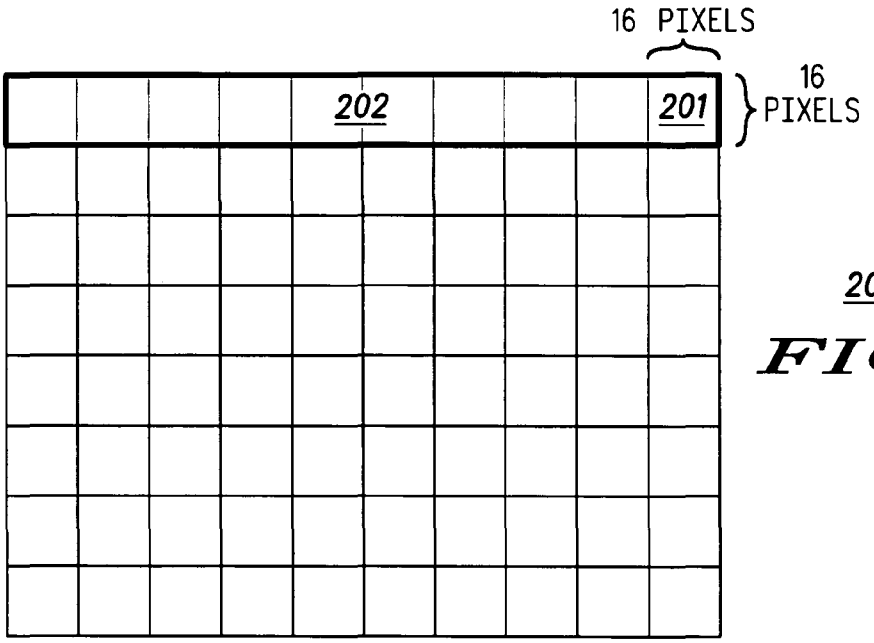


FIG. 2

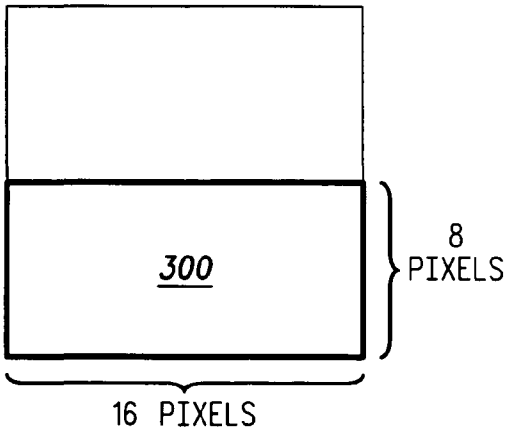


FIG. 3A

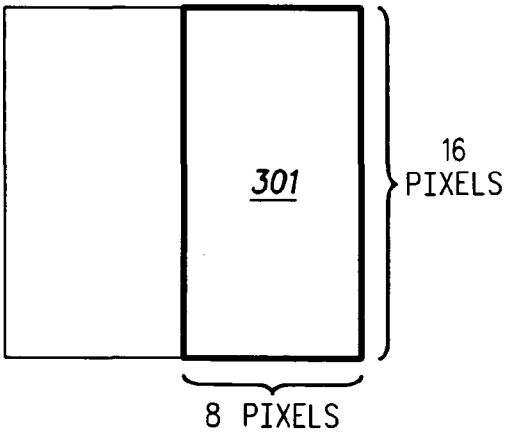


FIG. 3B

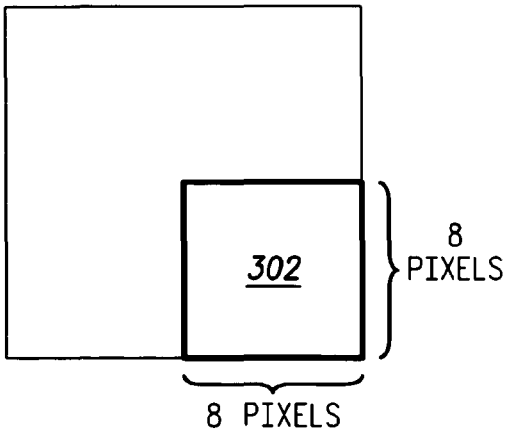


FIG. 3C

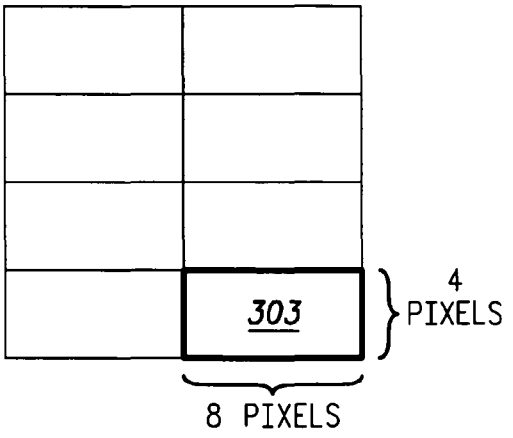


FIG. 3D

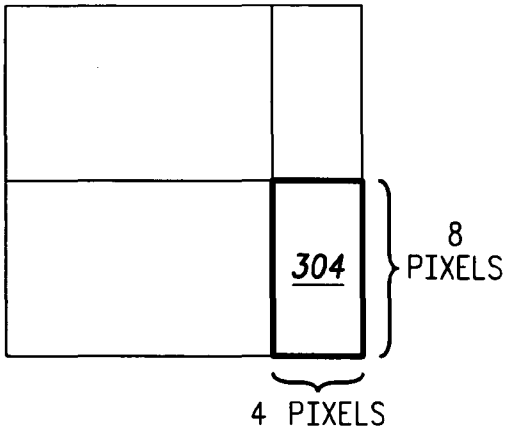


FIG. 3E

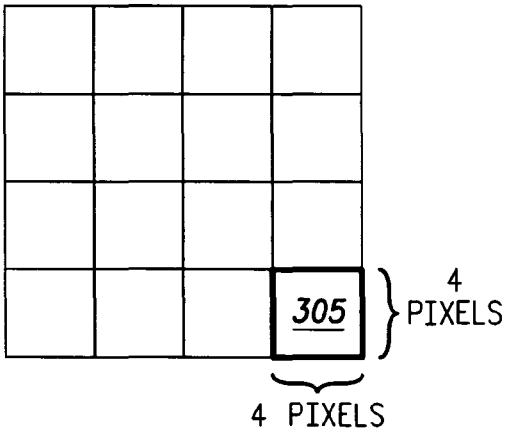


FIG. 3F

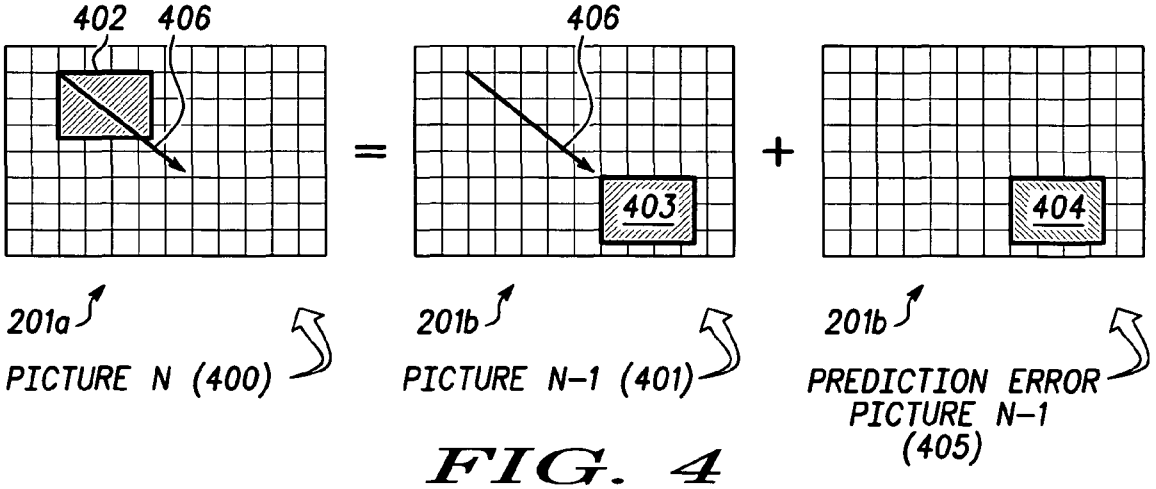
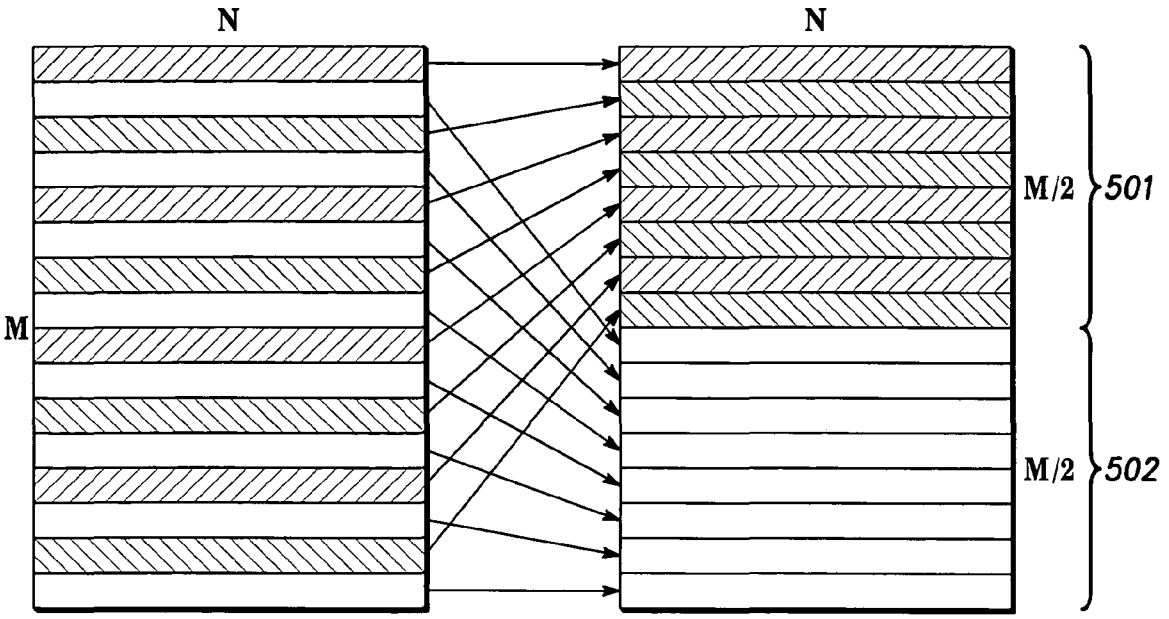


FIG. 4



500 FIG. 5

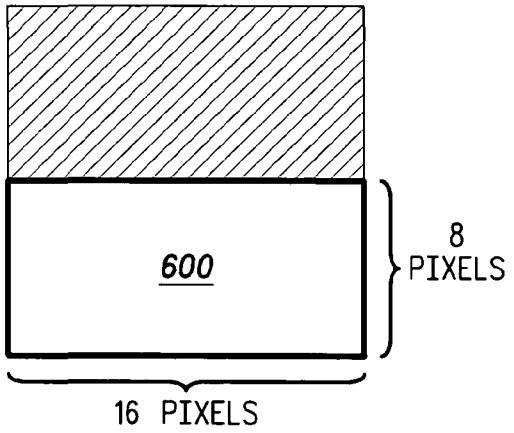


FIG. 6A

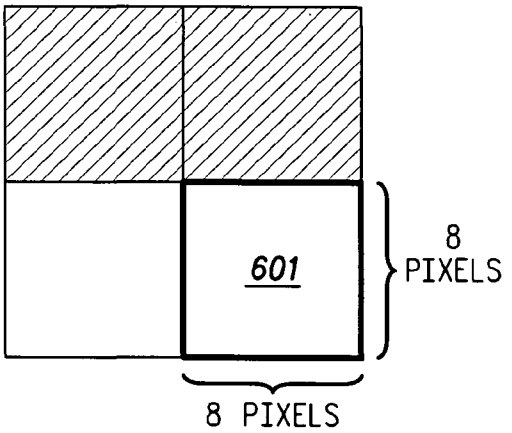


FIG. 6B

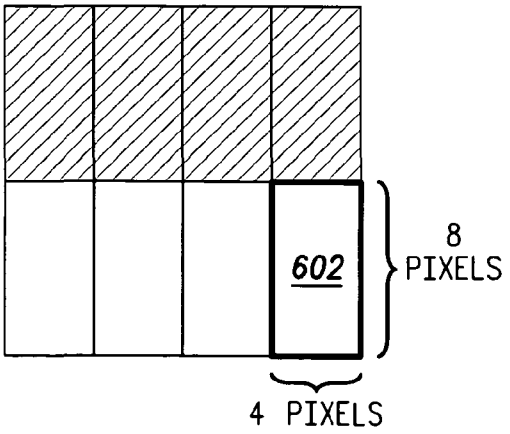


FIG. 6C

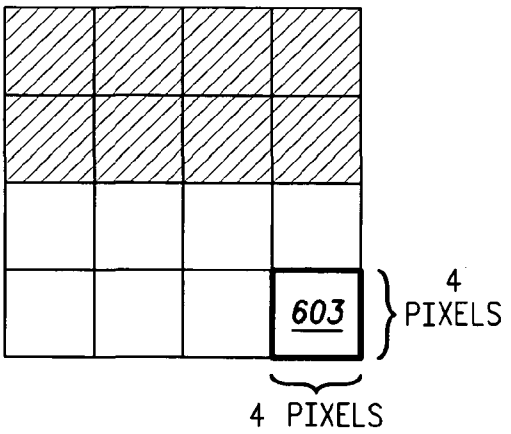


FIG. 6D

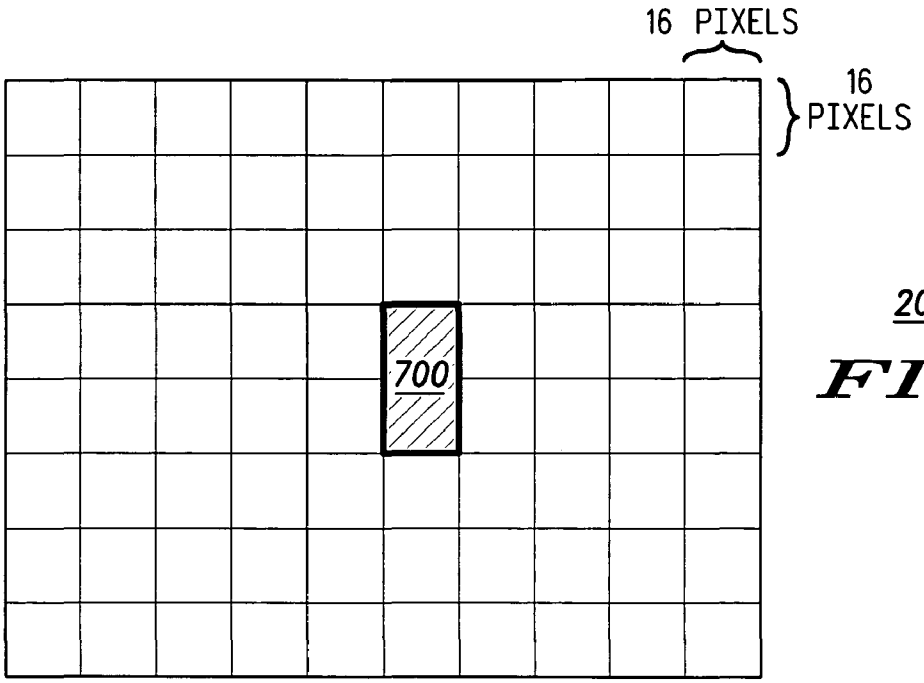


FIG. 7

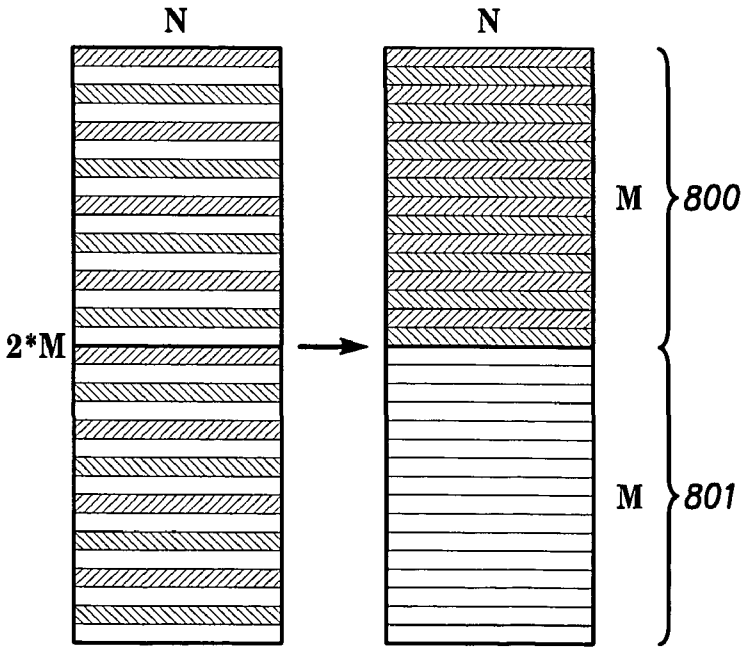


FIG. 8

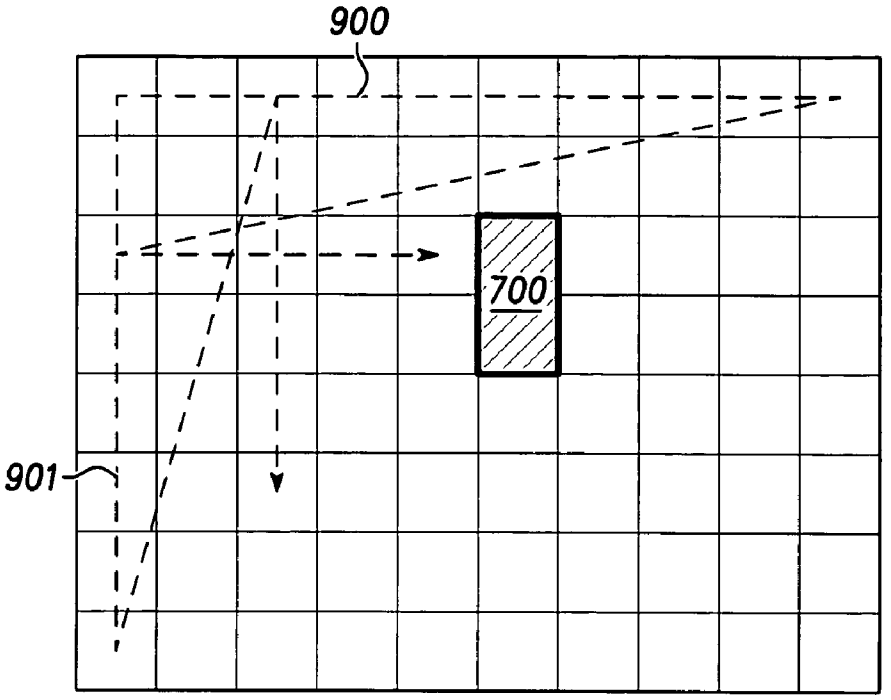


FIG. 9

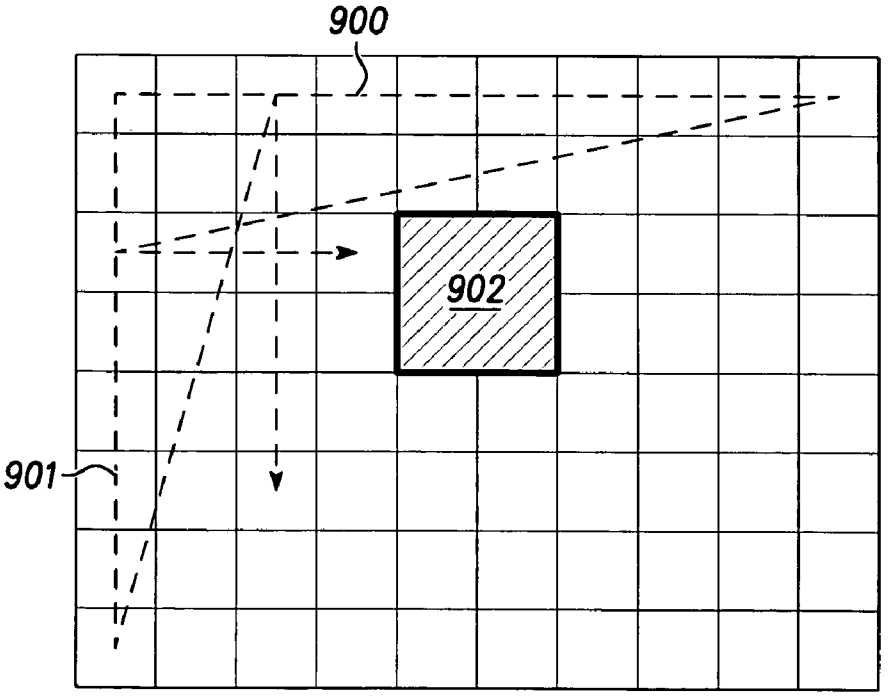


FIG. 10

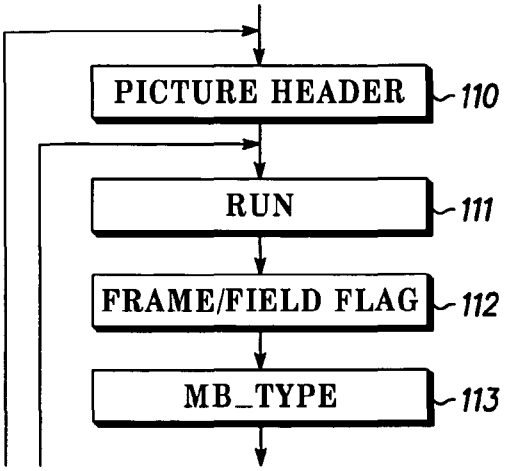


FIG. 11

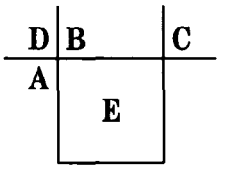


FIG. 12

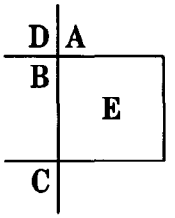


FIG. 13

Q	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				
M								
N								
O								
P								

FIG. 14

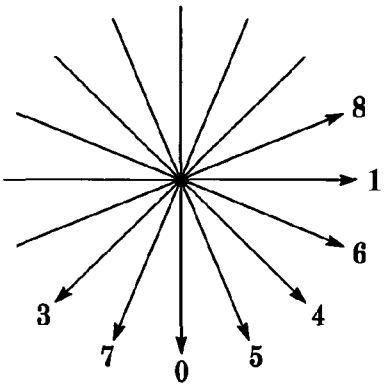


FIG. 15

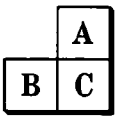


FIG. 16A

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

FIG. 16B

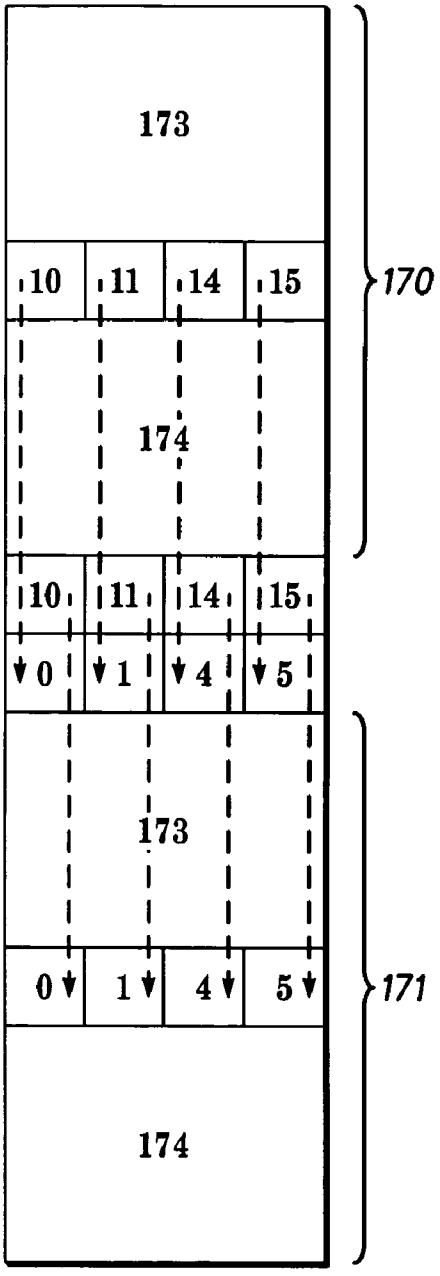


FIG. 17A

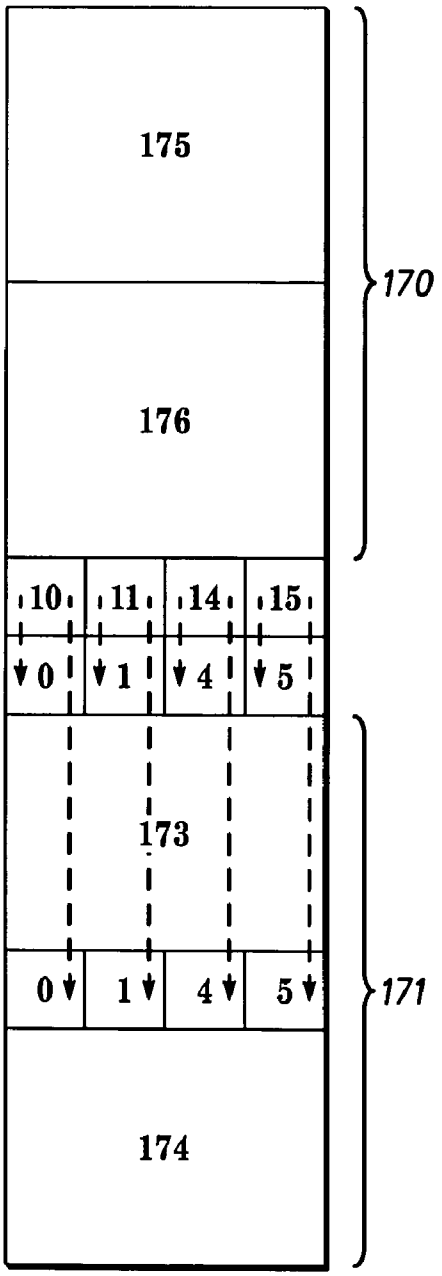


FIG. 17B

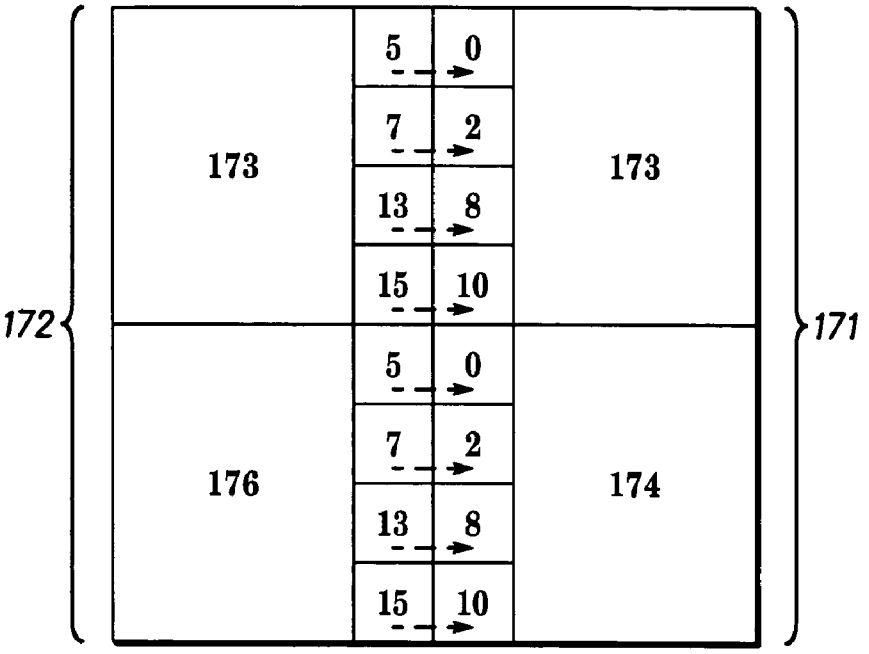


FIG. 17C

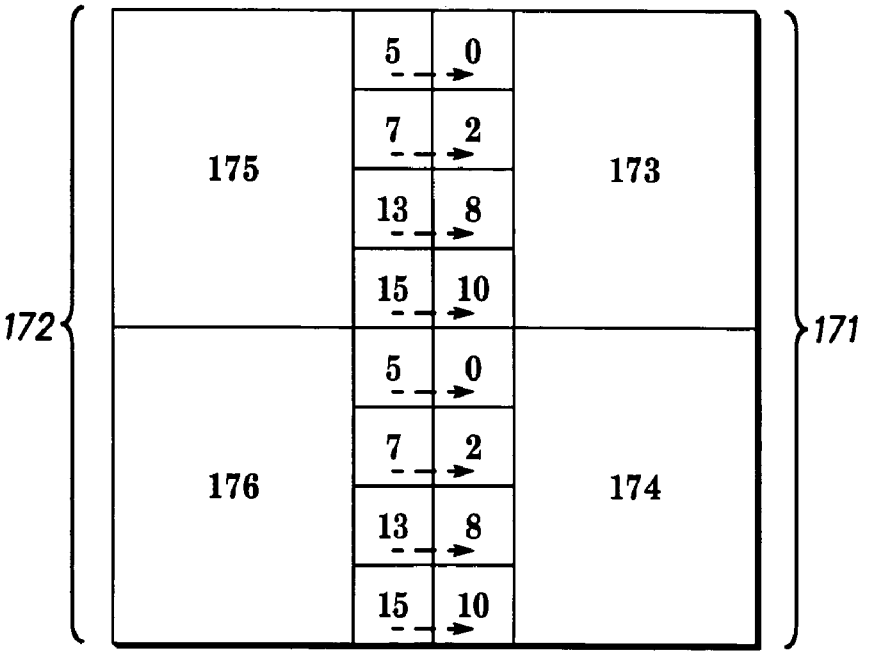


FIG. 17D

MACROBLOCK LEVEL ADAPTIVE FRAME/FIELD CODING FOR DIGITAL VIDEO CONTENT

The present application claims priority under 35 U.S.C. §119(e) from the following previously filed Provisional patent applications: Ser. No. 60/333,921, filed Nov. 27, 2001; Ser. No. 60/395,734, filed Jul. 12, 2002; Ser. No. 60/398,161, filed Jul. 23, 2002; all of which are herein incorporated by reference. This application is also a Divisional of U.S. patent application Ser. No. 10/301,290 filed on Nov. 20, 2002 now U.S. Pat. No. 6,980,596, which is herein incorporated by reference.

TECHNICAL FIELD

The present invention relates to encoding and decoding of digital video content. More specifically, the present invention relates to frame mode and field mode encoding of digital video content at a macroblock level as used in the MPEG-4 Part 10 AVC/H.264 standard video coding standard.

BACKGROUND

Video compression is used in many current and emerging products. It is at the heart of digital television set-top boxes (STBs), digital satellite systems (DSSs), high definition television (HDTV) decoders, digital versatile disk (DVD) players, video conferencing, Internet video and multimedia content, and other digital video applications. Without video compression, digital video content can be extremely large, making it difficult or even impossible for the digital video content to be efficiently stored, transmitted, or viewed.

The digital video content comprises a stream of pictures that can be displayed as an image on a television receiver, computer monitor, or some other electronic device capable of displaying digital video content. A picture that is displayed in time before a particular picture is in the "backward direction" in relation to the particular picture. Likewise, a picture that is displayed in time after a particular picture is in the "forward direction" in relation to the particular picture.

Video compression is accomplished in a video encoding, or coding, process in which each picture is encoded as either a frame or as two fields. Each frame comprises a number of lines of spatial information. For example, a typical frame contains 480 horizontal lines. Each field contains half the number of lines in the frame. For example, if the frame comprises 480 horizontal lines, each field comprises 240 horizontal lines. In a typical configuration, one of the fields comprises the odd numbered lines in the frame and the other field comprises the even numbered lines in the frame. The field that comprises the odd numbered lines will be referred to as the "top" field hereafter and in the appended claims, unless otherwise specifically denoted. Likewise, the field that comprises the even numbered lines will be referred to as the "bottom" field hereafter and in the appended claims, unless otherwise specifically denoted. The two fields can be interlaced together to form an interlaced frame.

The general idea behind video coding is to remove data from the digital video content that is "non-essential." The decreased amount of data then requires less bandwidth for broadcast or transmission. After the compressed video data has been transmitted, it must be decoded, or decompressed. In this process, the transmitted video data is processed to generate approximation data that is substituted into the video data to replace the "non-essential" data that was removed in the coding process.

Video coding transforms the digital video content into a compressed form that can be stored using less space and transmitted using less bandwidth than uncompressed digital video content. It does so by taking advantage of temporal and spatial redundancies in the pictures of the video content. The digital video content can be stored in a storage medium such as a hard drive, DVD, or some other non-volatile storage unit.

There are numerous video coding methods that compress the digital video content. Consequently, video coding standards have been developed to standardize the various video coding methods so that the compressed digital video content is rendered in formats that a majority of video encoders and decoders can recognize. For example, the Motion Picture Experts Group (MPEG) and International Telecommunication Union (ITU-T) have developed video coding standards that are in wide use. Examples of these standards include the MPEG-1, MPEG-2, MPEG-4, ITU-T H261, and ITU-T H263 standards.

Most modern video coding standards, such as those developed by MPEG and ITU-T, are based in part on a temporal prediction with motion compensation (MC) algorithm. Temporal prediction with motion compensation is used to remove temporal redundancy between successive pictures in a digital video broadcast.

The temporal prediction with motion compensation algorithm typically utilizes one or two reference pictures to encode a particular picture. A reference picture is a picture that has already been encoded. By comparing the particular picture that is to be encoded with one of the reference pictures, the temporal prediction with motion compensation algorithm can take advantage of the temporal redundancy that exists between the reference picture and the particular picture that is to be encoded and encode the picture with a higher amount of compression than if the picture were encoded without using the temporal prediction with motion compensation algorithm. One of the reference pictures may be in the backward direction in relation to the particular picture that is to be encoded. The other reference picture is in the forward direction in relation to the particular picture that is to be encoded.

However, as the demand for higher resolutions, more complex graphical content, and faster transmission time increases, so does the need for better video compression methods. To this end, a new video coding standard is currently being developed jointly by ISO and ITU-T. This new video coding standard is called the MPEG-4 Advanced Video Coding (AVC)/H.264 standard.

SUMMARY OF THE INVENTION

In one of many possible embodiments, the present invention provides a method of encoding, decoding, and bitstream generation of digital video content. The digital video content comprises a stream of pictures which can each be intra, predicted, or bi-predicted pictures. Each of the pictures comprises macroblocks that can be further divided into smaller blocks. The method entails encoding and decoding each of the macroblocks in each picture in said stream of pictures in either frame mode or in field mode.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings illustrate various embodiments of the present invention and are a part of the specification. Together with the following description, the drawings demonstrate and explain the principles of the present

invention. The illustrated embodiments are examples of the present invention and do not limit the scope of the invention.

FIG. **1** illustrates an exemplary sequence of three types of pictures that can be used to implement the present invention, as defined by an exemplary video coding standard such as the MPEG-4 Part 10 AVC/H.264 standard.

FIG. **2** shows that each picture is preferably divided into slices containing macroblocks according to an embodiment of the present invention.

FIG. **3***a* shows that a macroblock can be further divided into a block size of 16 by 8 pixels according to an embodiment of the present invention.

FIG. **3***b* shows that a macroblock can be further divided into a block size of 8 by 16 pixels according to an embodiment of the present invention.

FIG. **3***c* shows that a macroblock can be further divided into a block size of 8 by 8 pixels according to an embodiment of the present invention.

FIG. **3***d* shows that a macroblock can be further divided into a block size of 8 by 4 pixels according to an embodiment of the present invention.

FIG. **3***e* shows that a macroblock can be further divided into a block size of 4 by 8 pixels according to an embodiment of the present invention.

FIG. **3***f* shows that a macroblock can be further divided into a block size of 4 by 4 pixels according to an embodiment of the present invention.

FIG. **4** shows a picture construction example using temporal prediction with motion compensation that illustrates an embodiment of the present invention.

FIG. **5** shows that a macroblock is split into a top field and a bottom field if it is to be encoded in field mode.

FIG. **6***a* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 16 by 8 pixels according to an embodiment of the present invention.

FIG. **6***b* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 8 by 8 pixels according to an embodiment of the present invention.

FIG. **6***c* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 4 by 8 pixels according to an embodiment of the present invention.

FIG. **6***d* shows that a macroblock that is encoded in field mode can be divided into a block with a size of 4 by 4 pixels according to an embodiment of the present invention.

FIG. **7** illustrates an exemplary pair of macroblocks that can be used in AFF coding on a pair of macroblocks according to an embodiment of the present invention.

FIG. **8** shows that a pair of macroblocks that is to be encoded in field mode is first split into one top field 16 by 16 pixel block and one bottom field 16 by 16 pixel block.

FIG. **9** shows two possible scanning paths in AFF coding of pairs of macroblocks.

FIG. **10** illustrates another embodiment of the present invention which extends the concept of AFF coding on a pair of macroblocks to AFF coding to a group of four or more neighboring macroblocks.

FIG. **11** shows some of the information included in the bitstream which contains information pertinent to each macroblock within a stream.

FIG. **12** shows a block that is to be encoded and its neighboring blocks and will be used to explain various preferable methods of calculating the PMV of a block in a macroblock.

FIG. **13** shows an alternate definition of neighboring blocks if the scanning path is a vertical scanning path.

FIG. **14** shows that each pixel value is predicted from neighboring blocks' pixel values according to an embodiment of the present invention.

FIG. **15** shows different prediction directions for intra_4×4 coding.

FIGS. **16***a-b* illustrate that the chosen intra-prediction mode (intra_pred_mode) of a 4 by 4 pixel block is highly correlated with the prediction modes of adjacent blocks.

FIGS. **17***a-d* show neighboring blocks definitions in relation to a current macroblock pair that is to be encoded.

Throughout the drawings, identical reference numbers designate similar, but not necessarily identical, elements.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

The present invention provides a method of adaptive frame/field (AFF) coding of digital video content comprising a stream of pictures or slices of a picture at a macroblock level. The present invention extends the concept of picture level AFF to macroblocks. In AFF coding at a picture level, each picture in a stream of pictures that is to be encoded is encoded in either frame mode or in field mode, regardless of the frame or field coding mode of other pictures that are to be coded. If a picture is encoded in frame mode, the two fields that make up an interlaced frame are coded jointly. Conversely, if a picture is encoded in field mode, the two fields that make up an interlaced frame are coded separately. The encoder determines which type of coding, frame mode coding or field mode coding, is more advantageous for each picture and chooses that type of encoding for the picture. The exact method of choosing between frame mode and field mode is not critical to the present invention and will not be detailed herein.

As noted above, the MPEG-4 Part 10 AVC/H.264 standard is a new standard for encoding and compressing digital video content. The documents establishing the MPEG-4 Part 10 AVC/H.264 standard are hereby incorporated by reference, including "Joint Final Committee Draft (JFCD) of Joint Video Specification" issued by the Joint Video Team (JVT) on Aug. 10, 2002. (ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC). The JVT consists of experts from ISO or MPEG and ITU-T. Due to the public nature of the MPEG-4 Part 10 AVC/H.264 standard, the present specification will not attempt to document all the existing aspects of MPEG-4 Part 10 AVC/H.264 video coding, relying instead on the incorporated specifications of the standard.

Although this method of AFF encoding is compatible with and will be explained using the MPEG-4 Part 10 AVC/H.264 standard guidelines, it can be modified and used as best serves a particular standard or application.

Using the drawings, the preferred embodiments of the present invention will now be explained.

FIG. **1** illustrates an exemplary sequence of three types of pictures that can be used to implement the present invention, as defined by an exemplary video coding standard such as the MPEG-4 Part 10 AVC/H.264 standard. As previously mentioned, the encoder encodes the pictures and the decoder decodes the pictures. The encoder or decoder can be a processor, application specific integrated circuit (ASIC), field programmable gate array (FPGA), coder/decoder (CODEC), digital signal processor (DSP), or some other electronic device that is capable of encoding the stream of pictures. However, as used hereafter and in the appended claims, unless otherwise specifically denoted, the term "encoder" will be used to refer expansively to all electronic devices that encode digital video content comprising a

stream of pictures. The term "decoder" will be used to refer expansively to all electronic devices that decode digital video content comprising a stream of pictures.

As shown in FIG. **1**, there are preferably three types of pictures that can be used in the video coding method. Three types of pictures are defined to support random access to stored digital video content while exploring the maximum redundancy reduction using temporal prediction with motion compensation. The three types of pictures are intra (I) pictures (**100**), predicted (P) pictures (**102***a,b*), and bi-predicted (B) pictures (**101***a-d*). An I picture (**100**) provides an access point for random access to stored digital video content and can be encoded only with slight compression. Intra pictures (**100**) are encoded without referring to reference pictures.

A predicted picture (**102***a,b*) is encoded using an I, P, or B picture that has already been encoded as a reference picture. The reference picture can be in either the forward or backward temporal direction in relation to the P picture that is being encoded. The predicted pictures (**102***a,b*) can be encoded with more compression than the intra pictures (**100**).

A bi-predicted picture (**101***a-d*) is encoded using two temporal reference pictures: a forward reference picture and a backward reference picture. The forward reference picture is sometimes called a past reference picture and the backward reference picture is sometimes called a future reference picture. An embodiment of the present invention is that the forward reference picture and backward reference picture can be in the same temporal direction in relation to the B picture that is being encoded. Bi-predicted pictures (**101***a-d*) can be encoded with the most compression out of the three picture types.

Reference relationships (**103**) between the three picture types are illustrated in FIG. **1**. For example, the P picture (**102***a*) can be encoded using the encoded I picture (**100**) as its reference picture. The B pictures (**101***a-d*) can be encoded using the encoded I picture (**100**) or the encoded P picture (**102***a*) as its reference pictures, as shown in FIG. **1**. Under the principles of an embodiment of the present invention, encoded B pictures (**101***a-d*) can also be used as reference pictures for other B pictures that are to be encoded. For example, the B picture (**101***c*) of FIG. **1** is shown with two other B pictures (**101***b* and **101***d*) as its reference pictures.

The number and particular order of the I (**100**), B (**101***a-d*), and P (**102***a,b*) pictures shown in FIG. **1** are given as an exemplary configuration of pictures, but are not necessary to implement the present invention. Any number of I, B, and P pictures can be used in any order to best serve a particular application. The MPEG-4 Part 10 AVC/H.264 standard does not impose any limit to the number of B pictures between two reference pictures nor does it limit the number of pictures between two I pictures.

FIG. **2** shows that each picture (**200**) is preferably divided into slices (**202**). A slice (**202**) comprises a group of macroblocks (**201**). A macroblock (**201**) is a rectangular group of pixels. As shown in FIG. **2**, a preferable macroblock (**201**) size is 16 by 16 pixels.

FIGS. **3***a-f* show that a macroblock can be further divided into smaller sized blocks. For example, as shown in FIGS. **3***a-f*, a macroblock can be further divided into block sizes of 16 by 8 pixels (FIG. **3***a*; **300**), 8 by 16 pixels (FIG. **3***b*; **301**), 8 by 8 pixels (FIG. **3***c*; **302**), 8 by 4 pixels (FIG. **3***d*; **303**), 4 by 8 pixels (FIG. **3***e*; **304**), or 4 by 4 pixels (FIG. **3***f*; **305**). These smaller block sizes are preferable in some applications that use the temporal prediction with motion compensation algorithm.

FIG. **4** shows a picture construction example using temporal prediction with motion compensation that illustrates an embodiment of the present invention. Temporal prediction with motion compensation assumes that a current picture, picture N (**400**), can be locally modeled as a translation of another picture, picture N-**1** (**401**). The picture N-**1** (**401**) is the reference picture for the encoding of picture N (**400**) and can be in the forward or backwards temporal direction in relation to picture N (**400**).

As shown in FIG. **4**, each picture is preferably divided into slices containing macroblocks (**201***a,b*). The picture N-**1** (**401**) contains an image (**403**) that is to be shown in picture N (**400**). The image (**403**) will be in a different temporal position in picture N (**402**) than it is in picture N-**1** (**401**), as shown in FIG. **4**. The image content of each macroblock (**201***b*) of picture N (**400**) is predicted from the image content of each corresponding macroblock (**201***a*) of picture N-**1** (**401**) by estimating the required amount of temporal motion of the image content of each macroblock (**201***a*) of picture N-**1** (**401**) for the image (**403**) to move to its new temporal position (**402**) in picture N (**400**). Instead of the original image (**402**) being encoded, the difference (**404**) between the image (**402**) and its prediction (**403**) is actually encoded and transmitted.

For each image (**402**) in picture N (**400**), the temporal prediction can often be described by motion vectors that represent the amount of temporal motion required for the image (**403**) to move to a new temporal position in the picture N (**402**). The motion vectors (**406**) used for the temporal prediction with motion compensation need to be encoded and transmitted.

FIG. **4** shows that the image (**402**) in picture N (**400**) can be represented by the difference (**404**) between the image and its prediction and the associated motion vectors (**406**). The exact method of encoding using the motion vectors can vary as best serves a particular application and can be easily implemented by someone who is skilled in the art.

To understand macroblock level AFF coding, a brief overview of picture level AFF coding of a stream of pictures will now be given. A frame of an interlaced sequence contains two fields, the top field and the bottom field, which are interleaved and separated in time by a field period. The field period is half the time of a frame period. In picture level AFF coding, the two fields of an interlaced frame can be coded jointly or separately. If they are coded jointly, frame mode coding is used. Conversely, if the two fields are coded separately, field mode coding is used.

Fixed frame/field coding, on the other hand, codes all the pictures in a stream of pictures in one mode only. That mode can be frame mode or it can be field mode. Picture level AFF is preferable to fixed frame/field coding in many applications because it allows the encoder to chose which mode, frame mode or field mode, to encode each picture in the stream of pictures based on the contents of the digital video material. AFF coding results in better compression than does fixed frame/field coding in many applications.

An embodiment of the present invention is that AFF coding can be performed on smaller portions of a picture. This small portion can be a macroblock, a pair of macroblocks, or a group of macroblocks. Each macroblock, pair of macroblocks, or group of macroblocks or slice is encoded in frame mode or in field mode, regardless of how the other macroblocks in the picture are encoded. AFF coding in each of the three cases will be described in detail below.

In the first case, AFF coding is performed on a single macroblock. If the macroblock is to be encoded in frame mode, the two fields in the macroblock are encoded jointly. Once encoded as a frame, the macroblock can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if the macroblock is to be encoded in field mode, the macroblock (**500**) is split into a top field (**501**) and a bottom field (**502**), as shown in FIG. **5**. The two fields are then coded separately. In FIG. **5**, the macroblock has M rows of pixels and N columns of pixels. A preferable value of N and M is 16, making the macroblock (**500**) a 16 by 16 pixel macroblock. As shown in FIG. **5**, every other row of pixels is shaded. The shaded areas represent the rows of pixels in the top field of the macroblock (**500**) and the unshaded areas represent the rows of pixels in the bottom field of the macroblock (**500**).

As shown in FIGS. **6***a-d*, a macroblock that is encoded in field mode can be divided into four additional blocks. A block is required to have a single parity. The single parity requirement is that a block cannot comprise both top and bottom fields. Rather, it must contain a single parity of field. Thus, as shown in FIGS. **6***a-d*, a field mode macroblock can be divided into blocks of 16 by 8 pixels (FIG. **6***a*; **600**), 8 by 8 pixels (FIG. **6***b*; **601**), 4 by 8 pixels (FIG. **6***c*; **602**), and 4 by 4 pixels (FIG. **6***d*; **603**). FIGS. **6***a-d* shows that each block contains fields of a single parity.

AFF coding on macroblock pairs will now be explained. AFF coding on macroblock pairs will be occasionally referred to as pair based AFF coding. A comparison of the block sizes in FIGS. **6***a-d* and in FIGS. **3***a-f* show that a macroblock encoded in field mode can be divided into fewer block patterns than can a macroblock encoded in frame mode. The block sizes of 16 by 16 pixels, 8 by 16 pixels, and 8 by 4 pixels are not available for a macroblock encoded in field mode because of the single parity requirement. This implies that the performance of single macroblock based AFF may not be good for some sequences or applications that strongly favor field mode coding. In order to guarantee the performance of field mode macroblock coding, it is preferable in some applications for macroblocks that are coded in field mode to have the same block sizes as macroblocks that are coded in frame mode. This can be achieved by performing AFF coding on macroblock pairs instead of on single macroblocks.

FIG. **7** illustrates an exemplary pair of macroblocks (**700**) that can be used in AFF coding on a pair of macroblocks according to an embodiment of the present invention. If the pair of macroblocks (**700**) is to be encoded in frame mode, the pair is coded as two frame-based macroblocks. In each macroblock, the two fields in each of the macroblocks are encoded jointly. Once encoded as frames, the macroblocks can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if the pair of macroblocks (**700**) is to be encoded in field mode, it is first split into one top field 16 by 16 pixel block (**800**) and one bottom field 16 by 16 pixel block (**801**), as shown in FIG. **8**. The two fields are then coded separately. In FIG. **8**, each macroblock in the pair of macroblocks (**700**) has N=16 columns of pixels and M=16 rows of pixels. Thus, the dimensions of the pair of macroblocks (**700**) is 16 by 32 pixels. As shown in FIG. **8**, every other row of pixels is shaded. The shaded areas represent the rows of pixels in the top field of the macroblocks and the unshaded areas represent the rows of pixels in the bottom field of the macroblocks. The top field block (**800**) and the bottom field block (**801**) can now be divided into one of the possible block sizes of FIGS. **3***a-f*.

According to an embodiment of the present invention, in the AFF coding of pairs of macroblocks (**700**), there are two possible scanning paths. A scanning path determines the order in which the pairs of macroblocks of a picture are encoded. FIG. **9** shows the two possible scanning paths in AFF coding of pairs of macroblocks (**700**). One of the scanning paths is a horizontal scanning path (**900**). In the horizontal scanning path (**900**), the macroblock pairs (**700**) of a picture (**200**) are coded from left to right and from top to bottom, as shown in FIG. **9**. The other scanning path is a vertical scanning path (**901**). In the vertical scanning path (**901**), the macroblock pairs (**700**) of a picture (**200**) are coded from top to bottom and from left to right, as shown in FIG. **9**. For frame mode coding, the top macroblock of a macroblock pair (**700**) is coded first, followed by the bottom macroblock. For field mode coding, the top field macroblock of a macroblock pair is coded first followed by the bottom field macroblock.

Another embodiment of the present invention extends the concept of AFF coding on a pair of macroblocks to AFF coding on a group of four or more neighboring macroblocks (**902**), as shown in FIG. **10**. AFF coding on a group of macroblocks will be occasionally referred to as group based AFF coding. The same scanning paths, horizontal (**900**) and vertical (**901**), as are used in the scanning of macroblock pairs are used in the scanning of groups of neighboring macroblocks (**902**). Although the example shown in FIG. **10** shows a group of four macroblocks, the group can be more than four macroblocks.

If the group of macroblocks (**902**) is to be encoded in frame mode, the group coded as four frame-based macroblocks. In each macroblock, the two fields in each of the macroblocks are encoded jointly. Once encoded as frames, the macroblocks can be further divided into the smaller blocks of FIGS. **3***a-f* for use in the temporal prediction with motion compensation algorithm.

However, if a group of four macroblocks (**902**), for example, is to be encoded in field mode, it is first split into one top field 32 by 16 pixel block and one bottom field 32 by 16 pixel block. The two fields are then coded separately. The top field block and the bottom field block can now be divided into macroblocks. Each macroblock is further divided into one of the possible block sizes of FIGS. **3***a-f*. Because this process is similar to that of FIG. **8**, a separate figure is not provided to illustrate this embodiment.

In AFF coding at the macroblock level, a frame/field flag bit is preferably included in a picture's bitstream to indicate which mode, frame mode or field mode, is used in the encoding of each macroblock. The bitstream includes information pertinent to each macroblock within a stream, as shown in FIG. **11**. For example, the bitstream can include a picture header (**110**), run information (**111**), and macroblock type (**113**) information. The frame/field flag (**112**) is preferably included before each macroblock in the bitstream if AFF is performed on each individual macroblock. If the AFF is performed on pairs of macroblocks, the frame/field flag (**112**) is preferably included before each pair of macroblock in the bitstream. Finally, if the AFF is performed on a group of macroblocks, the frame/field flag (**112**) is preferably included before each group of macroblocks in the bitstream. One embodiment is that the frame/field flag (**112**) bit is a 0 if frame mode is to be used and a 1 if field coding is to be used. Another embodiment is that the frame/field flag (**112**) bit is a 1 if frame mode is to be used and a 0 if field coding is to be used.

Another embodiment of the present invention entails a method of determining the size of blocks into which the encoder divides a macroblock in macroblock level AFF. A preferable, but not exclusive, method for determining the ideal block size is sum absolute difference (SAD) with or without bias or rate distortion (RD) basis. For example, SAD checks the performance of the possible block sizes and chooses the ideal block size based on its results. The exact method of using SAD with or without bias or RD basis can be easily be performed by someone skilled in the art.

According to an embodiment of the present invention, each frame and field based macroblock in macroblock level AFF can be intra coded or inter coded. In intra coding, the macroblock is encoded without temporally referring to other macroblocks. On the other hand, in inter coding, temporal prediction with motion compensation is used to code the macroblocks.

If inter coding is used, a block with a size of 16 by 16 pixels, 16 by 8 pixels, 8 by 16 pixels, or 8 by 8 pixels can have its own reference pictures. The block can either be a frame or field based macroblock. The MPEG-4 Part 10 AVC/H.264 standard allows multiple reference pictures instead of just two reference pictures. The use of multiple reference pictures improves the performance of the temporal prediction with motion compensation algorithm by allowing the encoder to find a block in the reference picture that most closely matches the block that is to be encoded. By using the block in the reference picture in the coding process that most closely matches the block that is to be encoded, the greatest amount of compression is possible in the encoding of the picture. The reference pictures are stored in frame and field buffers and are assigned reference frame numbers and reference field numbers based on the temporal distance they are away from the current picture that is being encoded. The closer the reference picture is to the current picture that is being stored, the more likely the reference picture will be selected. For field mode coding, the reference pictures for a block can be any top or bottom field of any of the reference pictures in the reference frame or field buffers.

Each block in a frame or field based macroblock can have its own motion vectors. The motion vectors are spatially predictive coded. According to an embodiment of the present invention, in inter coding, prediction motion vectors (PMV) are also calculated for each block. The algebraic difference between a block's PMVs and its associated motion vectors is then calculated and encoded. This generates the compressed bits for motion vectors.

FIG. **12** will be used to explain various preferable methods of calculating the PMV of a block in a macroblock. A current block, E, in FIG. **12** is to be inter coded as well as its neighboring blocks A, B, C, and D. E will refer hereafter to a current block and A, B, C, and D will refer hereafter to E's neighboring blocks, unless otherwise denoted. Block E's PMV is derived from the motion vectors of its neighboring blocks. These neighboring blocks in the example of FIG. **12** are A, B, C, and D. One preferable method of calculating the PMV for block E is to calculate either the median of the motion vectors of blocks A, B, C, and D, the average of these motion vectors, or the weighted average of these motion vectors. Each of the blocks A through E can be in either frame or field mode.

Another preferable method of calculating the PMV for block E is to use a yes/no method. Under the principles of the yes/no method, a block has to be in the same frame or field coding mode as block E in order to have its motion vector included in the calculation of the PMV for E. For example, if block E in FIG. **12** is in frame mode, block A must also be in frame mode to have its motion vector included in the calculation of the PMV for block E. If one of E's neighboring blocks does not have the same coding mode as does block E, its motion vectors are not used in the calculation of block E's PMV.

The "always method" can also be used to calculate the PMV for block E. In the always method, blocks A, B, C, and D are always used in calculating the PMV for block E, regardless of their frame or field coding mode. If E is in frame mode and a neighboring block is in field mode, the vertical component of the neighboring block is multiplied by 2 before being included in the PMV calculation for block E. If E is in field mode and a neighboring block is in frame mode, the vertical component of the neighboring block is divided by 2 before being included in the PMV calculation for block E.

The "selective method" can also be used to calculate the PMV for block E if the macroblock has been encoded using pair based AFF encoding or group based AFF encoding. In the selective method, a frame-based block has a frame-based motion vector pointing to a reference frame. The block is also assigned a field-based motion vector pointing to a reference field. The field-based motion vector is the frame-based motion vector of the block with the vertical motion vector component divided by two. The reference field number is the reference frame number multiplied by two. A field-based block has a field-based motion vector pointing to a reference field. The block is also assigned a frame-based motion vector pointing to a reference frame. The frame-based motion vector is the field-based motion vector of the block with the vertical motion vector component multiplied by two. The reference frame number is the reference field number divided by two.

The derivation of a block's PMV using the selective method will now be explained using FIG. **12** as a reference. In macroblock pair based AFF, each block in a macroblock is associated with a companion block that resides in the same geometric location within the second macroblock of the macroblock pair. In FIG. **12**, each of block E's neighboring blocks (A, B, C, and D) may or may not be in the same frame or field coding mode as block E. Hence, the following rules apply:

If E is in frame mode and a neighboring block is in frame mode, the true frame-based motion vector of the neighboring block is used for E's PMV.

If E is in frame mode and a neighboring block is in field mode, the following rules apply in calculating B's PMV. If the neighboring block (e.g.; black A) and its companion field-based block have the same reference field, the average of the assigned frame-based motion vectors of the two blocks is used for the calculation of E's PMV. The reference frame number used for the PMV calculation is the reference field number of the neighboring block divided by two. However, if the neighboring block and its companion field block have different reference fields, then the neighboring block cannot be used in the calculation of E's PMV.

If E is in field mode and a neighboring block is in frame mode, the following rules apply in calculating E's PMV. If the neighboring block (e.g.; block A) and its companion frame-based block have the same reference frame, the average of the assigned field-based motion vectors of the two blocks is used for the calculation of E's PMV. The reference field number used for the PMV calculation is the reference frame number of the neighboring block multiplied by two. However, if the neighboring block and its compan-

ion field block have different reference frames, then the neighboring block cannot be used in the calculation of E's PMV.

If E is in field mode and a neighboring block is in field mode, the true field-based motion vector of the neighboring block is used in the calculation of E's PMV.

An alternate preferable option can be used in the selective method to calculate a block's PMV. In FIG. **12**, each of block E's neighboring blocks (A, B, C, and D) may or may not be in the same frame or field coding mode as block E. Hence, the following rules apply for this alternate preferable option of the selective method:

If E is in frame mode and a neighboring block is in frame mode, the true frame-based motion vector of the neighboring block is used for E's PMV.

If E is in frame mode and a neighboring block is in field mode, the weighted average of the assigned frame-based motion vectors of the neighboring block and its companion field-based block is used for the calculation of E's PMV. The weighting factors are based upon the reference field numbers of the neighboring block and its companion block.

If E is in field mode, and a neighboring block is in frame mode, the weighted average of the assigned field-based motion vectors of the neighboring block and its companion frame-based block is used for the calculation of E's PMV. The weighting factors are based upon the reference frame numbers of the neighboring block and its companion block.

If E is in field mode and a neighboring block is in field mode, the true field-based motion vector of the neighboring block is used in the calculation of E's PMV.

Another preferable method of computing a block's PMV is the "alt selective method." This method can be used in single macroblock AFF coding, pair based macroblock AFF coding, or group based AFF coding. In this method, each block is assigned a horizontal and a vertical index number, which represents the horizontal and vertical coordinates of the block. Each block is also assigned a horizontal and vertical field coordinate. A block's horizontal field coordinate is same as its horizontal coordinate. For a block in a top field macroblock, the vertical field coordinate is half of vertical coordinate of the block and is assigned top field polarity. For a block in the bottom field macroblock, the vertical field coordinate of the block is obtained by subtracting 4 from the vertical coordinate of the block and dividing the result by 2. The block is also assigned bottom field polarity. The result of assigning different field polarities to two blocks is that there are now two blocks with the same horizontal and vertical field coordinates but with differing field polarities. Thus, given the coordinates of a block, the field coordinates and its field polarity can be computed and vice versa.

The alt selective method will now be explained in detail using FIG. **12** as a reference. The PMV of block E is to be computed. Let bx represent the horizontal size of block E divided by 4, which is the size of a block in this example. The PMVs for E are obtained as follows depending on whether E is in frame/field mode.

Let block E be in frame mode and let (x,y) represent the horizontal and vertical coordinates respectively of E. The neighboring blocks of E are defined in the following manner. A is the block whose coordinates are (x−1,y). B is the block whose coordinates are (x,y−1). D is the block whose coordinates are (x−1,y−1). C is the block whose coordinates are (x+bx+1,y−1). If either A, B, C or D is in field mode then its vertical motion vector is multiplied by 2 before being used for prediction and its reference frame number is computed by dividing its reference field by 2.

Now, let block E be in top or bottom field mode and let (xf,yf) represent the horizontal and vertical field coordinates respectively of E. In this case, the neighbors of E are defined as follows. A is the block whose field coordinates are (xf−1,yf) and has same polarity as E. B is the block whose field coordinates are (xf,yf−1) and has same polarity as E. D is the block whose field coordinates are (xf−1,yf−1) and has same polarity as E. C is the block whose field coordinates arc (xf+bx+1,yf) and has same polarity as B. If either A,B,C or D is in frame mode then its vertical motion vector is divided by 2 before being used for prediction and its reference field is computed by multiplying its reference frame by 2.

In all of the above methods for determining the PMV of a block, a horizontal scanning path was assumed. However, the scanning path can also be a vertical scanning path. In this case, the neighboring blocks of the current block, E, are defined as shown in FIG. **13**. A vertical scanning path is preferable in some applications because the information on all the neighboring blocks is available for the calculation of the PMV for the current block E.

Another embodiment of the present invention is directional segmentation prediction. In directional segmentation prediction, 16 by 8 pixel blocks and 8 by 16 pixel blocks have rules that apply to their PMV calculations only. These rules apply in all PMV calculation methods for these block sizes. The rules will now be explained in detail in connection with FIG. **12**. In each of these rules, a current block E is to have its PMV calculated.

First, a 16 by pixel block consists of an upper block and a lower block. The upper block contains the top 8 rows of 16 pixels. The lower block contains the bottom 8 rows of 16 pixels. In the following description, block E of FIG. **12** is a 16 by 8 pixel block. For the upper block having a 16 by 8 pixel block, block B is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV. For the lower block having a 16 by 8 pixel block, block A is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV.

A 16 by 16 pixel block is divided into a right and left block. Both right and left blocks are 8 by 16 pixels. In the following description, block E of FIG. **12** is a 8 by 16 pixel block. For the left block, block A is used to predict block E's PMV if it has the same reference picture as block E. Otherwise, median prediction is used to predict block E's PMV For the right block, block C is used to predict block E's PMV if it has the same referenced picture as block E. Otherwise median prediction is used to predict block E's PMV.

For both 16 by 8 pixel blocks and 8 by 16 blocks, A, B, or C can be in different encoding modes (frame or field) than the current block E. The following rules apply for both block sizes. If E is in frame mode, and A, B, or C is in field mode, the reference frame number of A, B, or C is computed by dividing its reference field by 2. If E is in field mode, and A, B, or C is in frame mode, the reference field number of A, B, or C is computed by multiplying its reference frame by 2.

According to another embodiment of the present invention, a macroblock in a P picture can be skipped in AFF coding. If a macroblock is skipped, its data is not transmitted in the encoding of the picture. A skipped macroblock in a P picture is reconstructed by copying the co-located macroblock in the most recently coded reference picture. The co-located macroblock is defined as the one with motion compensation using PMV as defined above or without

motion vectors. The following rules apply for skipped macroblocks in a P picture. If AFF coding is performed per macroblock, a skipped macroblock is in frame mode. If AFF coding is performed on macroblock pairs and if both macroblocks are skipped, then they are in frame mode. However, if only one of the macroblocks in a macroblock pair is skipped, its frame or field coding mode is the same as the non-skipped macroblock in the same macroblock pair. If AFF coding is performed on a group of macroblocks and if the entire group of macroblocks is skipped, then all the macroblocks are in frame mode. If there is at least one macroblock that is not skipped, then the skipped macroblocks in the same group are in the same frame or field coding mode as the non-skipped macroblock.

An alternate method for skipped macroblocks is as follows. If a macroblock pair is skipped, its frame and field coding mode follows its neighboring macroblock pair to the left. If the left neighboring macroblock pair is not available, its coding mode follows its neighboring macroblock pair to the top. If neither the left nor top neighboring macroblock pairs are available, the skipped macroblock is set to frame mode.

Another embodiment of the present invention is direct mode macroblock coding for B pictures. In direct mode coding, a B picture has two motion vectors, forward and backward motion vectors. Each motion vector points to a reference picture. Both the forward and backward motion vectors can point in the same temporal direction. For direct mode macroblock coding in B pictures, the forward and backward motion vectors of a block are calculated from the co-located block in the backward reference picture. The co-located block in the backward reference picture can be frame mode or field mode coded. The following rules apply in direct mode macroblock coding for B picture.

If the co-located block is in frame mode and if the current direct mode macroblock is also in frame mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block. The forward reference frame is the one used by the co-located block. The backward reference frame is the same frame where the co-located block resides.

If the co-located block is in frame mode and if the current direct mode macroblock is in field mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block's motion vector with vertical component divided by two. The forward reference field is the same parity field of the reference frame used by the co-located block. The backward reference field is the same parity field of the backward reference frame where the co-located block resides.

If the co-located block is in field mode and if the current direct mode macroblock is also in field mode, the two associated motion vectors of a block in the direct mode macroblock are calculated from the co-located block of the same field parity. The forward reference field is the field used by the co-located block. The backward reference field is the same field where the co-located block resides.

If the co-located block is in field mode and if the current direct mode macroblock is in frame mode, the two associated motion vectors of the block in the direct mode macroblock are calculated from the co-located block's motion vector with vertical component multiplied by two. The forward reference frame is the frame one of whose fields is used by the co-located block. The backward reference field is the frame in one of whose fields the co-located block resides.

An alternate option is to force the direct mode block to be in the same frame or field coding mode as the co-located block. In this case, if the co-located block for a direct mode block is in frame mode, the direct mode block is in frame mode as well. The two frame-based motion vectors of the direct mode block are derived from the frame-based forward motion vector of the co-located block. The forward reference frame is used by the co-located block. The backward reference frame is where the co-located block resides.

However, if the co-located block for a block in direct mode is in field mode, the direct mode block is also in field mode. The two field-based motion vectors of the direct mode block are derived from the field-based forward motion vector of the co-located block. The forward reference field is used by the co-located block. The backward reference field is where the co-located block resides.

A macroblock in a B picture can also be skipped in AFF coding according to another embodiment of the present invention. A skipped macroblock in a B picture is reconstructed as a regular direct mode macroblock without any coded transform coefficient information. For skipped macroblocks in a B picture, the following rules apply. If AFF coding is performed per macroblock, a skipped macroblock is either in frame mode or in the frame or field coding mode of the co-located block in its backward reference picture. If AFF coding is performed on macroblock pairs and if both macroblocks are skipped, then they are in frame mode or in the frame or field coding mode of the co-located macroblock pair in the its backward reference picture. However, if only one of the macroblocks in a macroblock pair is skipped, its frame or field coding mode is the same as the non-skipped macroblock of the same macroblock pair. If AFF coding is performed on a group of macroblocks and if the entire group of macroblocks is skipped, then all the macroblocks are in frame mode or in the frame or field coding mode of the co-located group of macroblocks in the backward reference picture. If there is at least one macroblock that is not skipped, then the skipped macroblock in the same group are in the same frame or field coding mode as the non-skipped macroblock.

As previously mentioned, a block can be intra coded. Intra blocks are spatially predictive coded. There are two possible intra coding modes for a macroblock in macroblock level AFF coding. The first is intra_4×4 mode and the second is intra_16×16 mode. In both, each pixel's value is predicted using the real reconstructed pixel values from neighboring blocks. By predicting pixel values, more compression can be achieved. The intra_4×4 mode and the intra_16×16 modes will each be explained in more detail below.

For intra_4×4 mode, the predictions of the pixels in a 4 by 4 pixel block, as shown in FIG. **14**, are derived form its left and above pixels. In FIG. **14**, the 16 pixels in the 4 by 4 pixel block are labeled a through p. Also shown in FIG. **14** are the neighboring pixels A through P. The neighboring pixels are in capital letters. As shown in FIG. **15**, there are nine different prediction directions for intra_4×4 coding. They are vertical (**0**), horizontal (**1**), DC prediction (mode 2), diagonal down/left (**3**), diagonal down/right (**4**), vertical-left (**5**), horizontal-down (**6**), vertical-right (**7**), and horizontal-up (**8**). DC prediction averages all the neighboring pixels together to predict a particular pixel value.

However, for intra_16×16 mode, there are four different prediction directions. Prediction directions are also referred to as prediction modes. These prediction directions are vertical prediction (**0**), horizontal prediction (**1**), DC prediction, and plane prediction. Plane prediction will not be explained.

An intra block and its neighboring blocks may be coded in frame or field mode. Intra prediction is performed on the reconstructed blocks. A reconstructed block can be represented in both frame and field mode, regardless of the actual frame or field coding mode of the block. Since only the pixels of the reconstructed blocks are used for intra prediction, the following rules apply.

If a block of 4 by 4 pixels or 16 by 16 pixels is in frame mode, the neighboring pixels used in calculating the pixel value predictions of the block are in the frame structure. If a block of 4 by 4 pixels or 16 by 16 pixels is in field mode, the neighboring pixels used in calculating the pixel value prediction of the block are in field structure of the same field parity.

The chosen intra-prediction mode (intra_pred_mode) of a 4 by 4 pixel block is highly correlated with the prediction modes of adjacent blocks. This is illustrated in FIG. **16***a*. FIG. **16***a* shows that A and B are adjacent blocks to C. Block C's prediction mode is to be established. FIG. **16***b* shows the order of intra prediction information in the bitstream. When the prediction modes of A and B are known (including the case that A or B or both are outside the slice) the most probable prediction mode (most_probable_mode) of C is given. If one of the blocks A or B is "outside" the most_probable prediction mode is equal DC prediction (mode 2). Otherwise it is equal to the minimum of prediction modes used for blocks A and B. When an adjacent block is coded by 16×16 intra mode, prediction mode is DC prediction mode. When an adjacent block is coded a non-intra macroblock, prediction mode is "mode 2: DC prediction" in the usual case and "outside" in the case of constrained intra update.

To signal a prediction mode number for a 4 by 4 block first parameter use_most_probable_mode is transmitted. This parameter is represented by 1 bit codeword and can take values 0 or 1. If use_most_probable_mode is equal to 1 the most probable mode is used. Otherwise an additional parameter remaining_mode_selector, which can take value from 0 to 7 is sent as 3 bit codeword. The codeword is a binary representation of remaining_mode_selector value. The prediction mode number is calculated as:

```
if (remaining_mode_selector<most_probable mode)
intra_pred_mode=remaining_mode_selector;
else
intra_pred_mode=remaining_mode_selector+1;
```

The ordering of prediction modes assigned to blocks C is therefore the most probable mode followed by the remaining modes in the ascending order.

An embodiment of the present invention includes the following rules that apply to intra mode prediction for an intra-prediction mode of a 4 by 4 pixel block or an intra-prediction mode of a 16 by 16 pixel block. Block C and its neighboring blocks A and B can be in frame or field mode. One of the following rules shall apply. FIGS. **16***a-b* will be used in the following explanations of the rules.

Rule 1: A or B is used as the neighboring block of C only if A or B is in the same frame/field mode as C. Otherwise, A or B is considered as outside.

Rule 2: A and B are used as the neighboring blocks of C, regardless of their frame/field coding mode.

Rule 3: If C is coded in frame mode and has co-ordinates (x,y), then A is the block with co-ordinates (x,y-1) and B is the block with co-ordinates (x-1,y). Otherwise, if C is coded as field and has field co-ordinates (xf,yf) then A is the block whose field co-ordinates are (xf,yf-1) and has same field polarity as C and B is the block whose field co-ordinates are (xf-**1**,yf) and has same field polarity as C.

Rule 4: This rule applies to macroblock pairs only. In the case of decoding the prediction modes of blocks numbered 3, 6, 7, 9, 12, 13, 11, 14 and 15 of FIG. **16***b*, the above and the left neighboring blocks are in the same macroblock as the current block. However, in the case of decoding the prediction modes of blocks numbered 1, 4, and 5, the top block (block A) is in a different macroblock pair than the current macroblock pair. In the case of decoding the prediction mode of blocks numbered 2, 8, and 10, the left block (block B) is in a different macroblock pair. In the case of decoding the prediction mode of the block numbered 0, both the left and the above blocks are in different macroblock pairs. For a macroblock in field decoding mode the neighboring blocks of the blocks numbered 0, 1, 4, 5, 2, 8, and 10 shall be defined as follows:

If the above macroblock pair (**170**) is decoded in field mode, then for blocks number 0, 1, 4 and 5 in the top-field macroblock (**173**), blocks numbered 10, 11, 14 and 15 respectively in the top-field macroblock (**173**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**) as shown in FIG. **17***a*. For blocks number 0, 1, 4 and 5 in the bottom-field macroblock (**174**), blocks numbered 10, 11, 14 and 15 respectively in the bottom-field MB of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***a*.

However, if the above macroblock pair (**170**) is decoded in frame mode then for blocks number 0, 1, 4 and 5 in the top-field macroblock (**173**), blocks numbered 10, 11, 14 and 15 respectively in the bottom-frame macroblock (**176**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***b*. For blocks number 0, 1, 4 and 5 in the bottom-field macroblock (**174**), blocks numbered 10, 11, 14 and 15 respectively in the bottom-frame macroblock (**176**) of the above macroblock pair (**170**) shall be considered as the above neighboring blocks to the current macroblock pair (**171**), as shown inn FIG. **17***b*.

If the left macroblock pair (**172**) is decoded in field mode, then for blocks number 0, 2, 8 and 10 in the top-field macroblock (**173**), blocks numbered 5, 7, 13 and 15 respectively in the top-field macroblock (**173**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**) as shown in FIG. **17***c*. For blocks number 0, 2, 8 and 10 in the bottom-field macroblock (**174**), blocks numbered 5, 7, 13 and 15 respectively in the bottom-field macroblock (**174**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***c*.

If the left macroblock pair (**172**) is decoded in frame mode, then for blocks number 0, 2, 8 and 10 in the top-field macroblock (**173**), the blocks numbered 5, 7, 13 and 15 respectively in the top-frame macroblock (**175**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***d*. For blocks number 0, 2, 8 and 10 in the bottom-field macroblock (**174**), blocks numbered 5, 7, 13 and 15 respectively in the bottom-frame macroblock (**176**) of the left macroblock pair (**172**) shall be considered as the left neighboring blocks to the current macroblock pair (**171**), as shown in FIG. **17***d*.

For macroblock pairs on the upper boundary of a slice, if the left macroblock pair (**172**) is in frame decoding mode, then the intra mode prediction value used to predict a field macroblock shall be set to DC prediction.

The preceding descriptions of intra coding and intra mode prediction can be extended to adaptive block transforms.

Another embodiment of the present invention is that loop filtering is performed on the reconstructed blocks. A reconstructed block can be represented in either frame or field structure, regardless of the frame/field coding mode of the block. Loop (deblock) filtering is a process of weightcd averaging of the pixels of the neighboring blocks. FIG. **12** will be used to explain loop filtering. Assume E of FIG. **12** is a reconstructed block, and A, B, C and D are its neighboring reconstructed blocks, as shown in FIG. **12**, and they are all represented in frame structure. Since A, B, C, D and B can be either frame- or field-coded, the following rules apply:

Rule 1: If E is frame-coded, loop filtering is performed over the pixels of E and its neighboring blocks A B, C and D.

Rule 2: If E is field-coded, loop filtering is performed over the top-field and bottom-field pixels of E and its neighboring blocks A B, C and D, separately.

Another embodiment of the present invention is that padding is performed on the reconstructed frame by repeating the boundary pixels. Since the boundary blocks may be coded in frame or field mode, the following rules apply:

Rule 1: The pixels on the left or right vertical line of a boundary block are repeated, if necessary.

Rule 2: If a boundary block is in frame coding, the pixels on the top or bottom horizontal line of the boundary block are repeated.

Rule 3: if a boundary block is in field coding, the pixels on the two top or two bottom horizontal (two field) lines of the boundary block are repeated alternatively.

Another embodiment of the present invention is that two-dimensional transform coefficients are converted into one-dimensional series of coefficients before entropy coding. The scan path can be either zigzag or non-zigzag. The zigzag scanner is preferably for progressive sequences, but it may be also used for interlace sequences with slow motions. The non-zigzag scanners are preferably for interlace sequences. For macroblock AFF coding, the following options may be used:

Option 1: The zigzag scan is used for macroblocks in frame mode while the non-zigzag scanners are used for macroblocks in field coding.

Option 2: The zigzag scan is used for macroblocks in both frame and field modes.

Option 3: The non-zigzag scan is used for macroblocks in both frame and field modes.

The preceding description has been presented only to illustrate and describe embodiments of invention. It is not intended to be exhaustive or to limit the invention to any precise form disclosed. Many modifications and variations are possible in light of the above teaching.

The foregoing embodiments were chosen and described in order to illustrate principles of the invention and some practical applications. The preceding description enables others skilled in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims.

What is claimed is:

1. A method of encoding a picture in an image sequence, comprising:

dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks;

generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks; and

selectively encoding at least one of said processing blocks at a time in frame coding mode and at least one of said processing blocks at a time in field coding mode,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

2. The method of claim **1**, wherein said processing block includes a pair of macroblocks.

3. The method of claim **2**, wherein each pair of macroblocks of said image is encoded from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of macroblocks of said image is encoded from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

4. The method of claim **2**, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

5. The method of claim **2**, further comprising:

splitting said pair of macroblocks into a top field block and a bottom field block when said pair of macroblocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

6. The method of claim **1**, wherein said processing block includes a group of at least four macroblocks blocks.

7. An apparatus for encoding a picture in an image sequence, comprising:

means for dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks; and

means for generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks;

means for selectively encoding at least one of said processing blocks at a time in frame coding mode and at least one of said processing blocks at a time in field coding mode,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

8. The apparatus of claim **7** wherein said processing block includes a pair of macroblocks.

9. The apparatus of claim **8**, wherein each pair of macroblocks of said image is encoded from left to right and from top to bottom if said encoding is performed in said horizontal scanning path, and

wherein each pair of macroblocks of said image is encoded from top to bottom and from left to right if said encoding is performed in said vertical scanning path.

10. The apparatus of claim **8**, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is encoded prior to said bottom block in said frame coding mode.

11. The appartus of claim **8**, wherein said pair of macroblocks are split into a top field block and a bottom field block when said pair of macroblocks are encoded in said field coding mode, and where said top field block is encoded prior to said bottom field block.

12. The apparatus of claim **7**, wherein said processing block includes a group of at least four macroblocks blocks.

13. A computer-readable medium encoded with computer executable instructions, the computer executable instruc-

tions including instructions which, when executed by a processor, cause the processor to perform the steps of a method for encoding a picture in an image sequence, comprising the steps of:

dividing said picture into a plurality of macroblocks, each macroblock containing a plurality of blocks;

generating a plurality of processing blocks, each processing block being generated by grouping said plurality of macroblocks as a processing block, said plurality of macroblocks including a pair of macroblocks or a group of macroblocks; and

selectively encoding at least one of said processing blocks at a time in frame coding mode and at least one of said processing blocks at a time in field coding mode,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

14. A method of decoding an encoded picture having a plurality of processing blocks, each processing block containing macroblocks, each macroblock containing a plurality of blocks, from a bitstream, comprising:

decoding at least one of a plurality of processing blocks at a time, wherein each of said plurality of processing blocks includes a pair of macroblocks or a group of macroblocks, in frame coding mode and at least one of said plurality of processing blocks at a time in field coding mode, wherein said decoding is applied to a pair of blocks, or a group of blocks,

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

using said plurality of decoded processing blocks to construct a decoded picture.

15. The method of claim **14**, wherein said processing blocks include a pair of macroblocks.

16. The method of claim **14**, wherein said decoding is applied to a group of at least four macroblocks.

17. The method of claim **16**, further comprising: joining said group of at least four macroblocks into a top field block and a bottom field block when said group of at least four macroblocks is decoded in said field coding mode.

18. The method of claim **15**, wherein each pair of macroblocks of said image is decoded from left to right and from top to bottom if said decoding is performed in said horizontal scanning path, and

wherein each pair of macroblocks of said image is decoded from top to bottom and from left to right if said decoding is performed in said vertical scanning path.

19. The method of claim **15**, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is decoded prior to said bottom block in said frame coding mode.

20. The method of claim **15**, wherein said pair of macroblocks is represented by a top field block and a bottom field block in said field coding mode, the method further comprising:

decoding said top field block and said bottom field block, and

joining said top field block and said bottom field block into said pair of macroblocks.

21. The method of claim **14**, wherein said processing block includes a group of at least four macroblocks blocks.

22. An apparatus for decoding an encoded picture from a bitstream, comprising:

means for decoding at least one of a plurality of processing blocks at a time, each processing block containing a pair of macroblocks or a group of macroblocks, each macroblock containing a plurality of blocks, from said encoded picture that is encoded in frame coding mode and at least one of said plurality of processing blocks at a time that is encoded in field coding mode,

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

means for using said plurality of decoded processing blocks to construct a decoded picture.

23. The apparatus of claim **22**, wherein said processing blocks include a pair of macroblocks.

24. The apparatus of claim **22**, wherein said decoding is applied to a group of at least four macroblocks.

25. The apparatus of claim **24**, further comprising: means for joining said group of at least four macroblocks into a top field block and a bottom field block when said group of at least four macroblocks is decoded in said field coding mode.

26. The apparatus of claim **23**, wherein each pair of macroblocks of said image is decoded from left to right and from top to bottom if said decoding is performed in said horizontal scanning path, and

wherein each pair of macroblocks of said image is decoded from top to bottom and from left to right if said decoding is performed in said vertical scanning path.

27. The apparatus of claim **23**, wherein said pair of macroblocks comprises a top block and a bottom block, where said top block is decoded prior to said bottom block in said frame coding mode.

28. The apparatus of claim **23**, wherein said pair of macroblocks is represented by a top field block and a bottom field block in said field coding mode, the method further comprising:

decoding said top field block and said bottom field block, and

joining said top field block and said bottom field block into said pair of macroblocks.

29. The apparatus of claim **22**, wherein said processing block includes a group of at least four macroblocks blocks.

30. A computer-readable medium encoded with computer executable instructions, the computer executable instructions including instructions which, when executed by a processor, cause the processor to perform the steps of a method for encoding a picture in an image sequence, comprising the steps of:

decoding at least one of a plurality of processing blocks at a time, each processing block containing a pair of macroblocks or a group of macroblocks, each macroblock containing a plurality of blocks, from said encoded picture that is encoded in frame coding mode and at least one of said plurality of processing blocks at a time that is encoded in field coding mode,

wherein said decoding is performed in a horizontal scanning path or a vertical scanning path; and

using said plurality of decoded processing blocks to construct a decoded picture.

31. A bitstream comprising:

a picture that has been divided into a plurality of processing blocks, each processing block containing a pair of macroblocks or a group of macroblocks, each macroblock containing a plurality of blocks,

wherein at least one of said plurality of processing blocks from said picture is encoded in frame coding mode at a time and at least one of said plurality of processing blocks is encoded in field coding mode at a time,

wherein said encoding is performed in a horizontal scanning path or a vertical scanning path.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,310,376 B2
APPLICATION NO. : 11/026394
DATED : December 18, 2007
INVENTOR(S) : Wang et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

PAGE 2:
Title Page item 56, Under "Other Publications", in Column 2, Line 4: Please delete "Secotr" and replace with --Sector--

COLUMN 10:
Line 49: Please delete "B's" and replace with --E's--
Line 50: Please delete "black" and replace with --block--

COLUMN 12:
Line 8: Please delete "arc" and replace with --are--
Line 9: Please delete "B." and replace with --E.--
Line 29: Please delete "by" and replace with --by 16--

COLUMN 17:
Line 7: Please delete "weightcd" and replace with --weighted--
Line 13: Please delete "B" and replace with --E--

COLUMN 18:
Line 59: ln Claim 11, please delete "appartus" and replace with --apparatus--

Signed and Sealed this

Twenty-fifth Day of August, 2009

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

EXHIBIT D

EM8470 EM8475
EM8471 EM8476

MPEG-4 Decoder for Set-top, DVD and Streaming Applications

Description

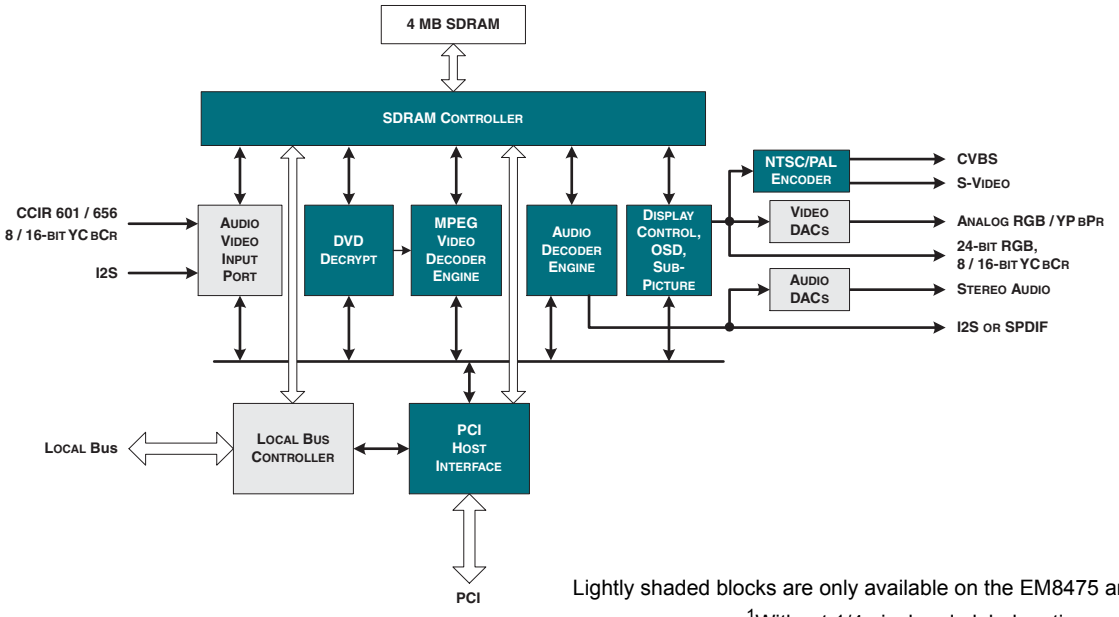
The EM847x family is a single-chip MPEG audio/video decoder that supports DVD-Video, Superbit™ DVD, SVCD, VCD and audio CD media formats. Video decoding support includes MPEG-1, MPEG-2 MP@ML and MPEG-4 advanced simple video profile Level 5[1]. Audio decoding support includes Dolby® Digital, MPEG-1 Layers 1 and 2, MPEG-4 high quality profile Level 2, and 16-bit linear PCM. DVD-Video support includes hardware CSS decryption, 16:9 and 4:3 aspect ratios, letterboxing, 3:2 pull-down, multiple angles and sub-picture.

Based on the company's award-winning REALmagic® Video Streaming Technology, the EM847x family provides highly-integrated solutions for high-quality decoding of MPEG-1, MPEG-2 and MPEG-4. Positioned as a cost-effective solution for streaming video clients, advanced digital set-top boxes and next-generation interactive DVD players, the EM847x enables manufacturers to easily incorporate streaming video, progressive DVD playback and video-on-demand into their products.

Common Features

- Pin and functionally compatible with EM8400 (EM8470 and EM8471 only)
- Compatible with MPEG-4 Client Players from Envivio and iVAST and a wide variety of MPEG-2 video servers
- Supports ISMA and MPEG-2 streaming formats, IP multicasting
- Supports DVD-Video, Superbit DVD, SVCD, VCD, audio CD media formats
- MPEG-1, MPEG-2 MP@ML, MPEG-4 Advanced Simple Profile Level 5[1] video decoding
- Dolby Digital, MPEG-1 Layers 1 and 2, MPEG-4 High Quality Profile Level 2, and linear PCM audio decoding

Block Diagram



Lightly shaded blocks are only available on the EM8475 and EM8476

[1]Without 1/4 pixel and global motion compensation

Sigma Designs, Inc.
355 Fairview Way • Milpitas, CA, USA 95035 • Tel: 408.262.9003 • Fax: 408.957.9740
www.sigmadesigns.com • sales@sigmadesigns.com

Bitstream Demultiplexing

The host processor performs MPEG-4 file demultiplexing and system decoding (except audio and video decoding), MPEG-2 transport and program stream and MPEG-1 system stream demultiplexing. The EM847x operates as a PCI bus master while audio and video elementary streams are transferred from host system memory to EM847x memory by the EM847x's DMA engine.

Software drivers for the x86, available from Sigma Designs, include MPEG-4 audio decode, MPEG-2 transport and program stream demultiplex, MPEG-1 system stream demultiplex and DVD-Video/SVCD/VCD stream demultiplex and navigation. Supported operating systems for our x86-based reference designs include Windows® XP Embedded, Windows CE .NET and Linux. For other operating systems and hardware platforms, our Professional Services Group can assist in supporting your requirements.

DVD-Video Decryption

The EM847x includes decryption logic and supports DVD-Video CSS procedural specifications. It also fully supports DVD-Video control features including up to 8 language sound tracks, 32 subtitle settings, letterbox, pan and scan, multi-angles and 3:2 pull-down.

MPEG Video Decoding

The MPEG video decoder engine contains the following modules used for MPEG-1, MPEG-2 and MPEG-4 video decoding:

- MPEG-1 & 2 Huffman Decoder
- MPEG-4 Huffman Decoder
- Motion Compensation
- IDCT
- Inverse Quantizer
- AC / DC Predictor

The Huffman Decoder receives bitstream data from DRAM and works in collaboration with the internal RISC processor to decode bitstream elements down to the block level. It then decodes transformed coefficients and sends them to the Inverse Quantizer. The Inverse Quantizer mainly performs two consecutive multiplications on the transformed coefficients (quantizer scale and quantizer matrix) and sends them to the IDCT. The IDCT converts them into pixels (or pixel differences for non-intra pictures), which are sent to the Motion Compensation block. The Motion Compensation block loads decoded macroblocks from DRAM (for prediction), adds the pixel differences and stores the result back to DRAM.

Common Features

- Programmable display output with improved scaling up to 1920x1080 resolution, interlaced or progressive, up to 120 Hz refresh
- 2-, 4-, 7- or 8-bit OSD with optional run-length compression, alpha blending and flicker filtering
- 80 MHz digital video output interface: 8- or 16-bit CCIR 601 / 656 YCbCr
- Improved NTSC/PAL composite and s-video outputs with optional Macrovision v7.1 protection and VBI data support (10-bit DACs)
- Digital audio output interface: I2S for PCM or S/PDIF (IEC 60958) for PCM, compressed Dolby Digital and compressed DTS
- Improved PCI v2.1 bus master / slave interface supports both read and write operations
- 2.5V core with 3.3V I/O (5V tolerant) for low power operation
- Package: 208 PQFP for EM8470 and EM8471 (pin compatible with EM8400), 328 BGA for EM8475 and EM8476

EM8475 / EM8476 Features

- Digital audio input interface: I2S
- 80 MHz digital video input interface: 8- or 16-bit CCIR 601 / 656 YCbCr supporting resolutions up to 1920x1080i
- 80 MHz 24-bit digital RGB output interface for DVI with HDCP support
- Interlaced or progressive analog RGB or YPbPr video outputs with optional Macrovision AGC v1.03 protection and VBI data support (10-bit DACs)
- Stereo audio DACs
- Flexible Local Bus interface

Applications

- Consumer products needing PVR and playback of video-on-demand (VOD), streaming video, progressive DVD
- Set-top boxes, media/home gateways, video end points, convergence appliances

MPEG-1 and MPEG-2 Huffman Decoder

The MPEG Video Decoder engine uses the Huffman Decoder as its front end. It can decode either a MPEG-1 or MPEG-2 formatted data stream. The Huffman Decoder is commanded directly from the RISC processor. It extracts fixed-length, variable-length or start codes from the bitstream and returns the value to the RISC processor or passes it directly to the Inverse Quantizer. The Huffman Decoder can also do tasks such as decode the macroblock increment address, get the macroblock type, get coded block pattern, get motion vector codes and decode a delta motion vector.

MPEG-4 Huffman Decoder

The MPEG-4 Huffman Decoder is the front-end to the MPEG-4 engine. It provides specialized instructions to the RISC processor to enable the decoding of Simple Profile MPEG-4 streams. These instructions are of three different kinds: get or view a fixed-length bit field, get an individual variable length code and get the DCT coefficients for the whole macroblock. Additionally, the Huffman Decoder provides support for the error-resiliency features of MPEG-4. It can be used to read data-partitioned Video Object Planes, with or without Reversible Variable Length Codes (RVLCs). It cannot, however, read RVLCs backwards.

AC/DC Predictor

If the input stream is MPEG-1 or MPEG-2, the AC/DC predictor passes through. Otherwise, it receives data from the zigzag through a 19-bit stream interface. A predictor is added to the incoming AC/DC coefficients. The result is saturated and output to the inverse quantizer through a similar interface. This result will also be used as future predictors. For prediction, this block must save an entire macroblock line of informations. This module supports MPEG-4 simple profile.

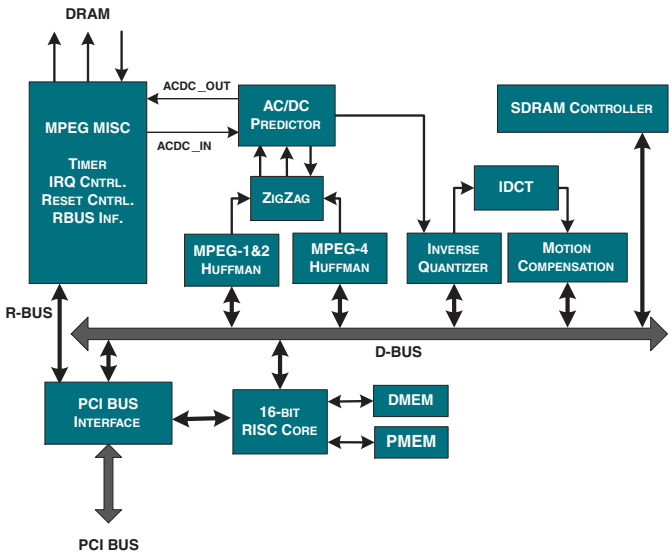
Inverse Quantizer

The Inverse Quantizer block resides between the AC/DC predictor block and the inverse DCT block. Its primary function is to receive coefficients from the AC/DC predictor, scale them and send them to the IDCT. It supports the MPEG-1, MPEG-2 MP@ML and MPEG-4 simple profile.

Inverse DCT

The IDCT (Inverse Discrete Cosine Transform) module is a hardware implementation of the DCT/IDCT of an 8x8 pixel block used in MPEG compression/ decompression. The DCT transforms a block of 8x8 pixels into a block of 8x8 transformed coefficients. The IDCT transforms a block of 8x8 transformed coefficients back into a block of 8x8 pixels.

Video Decoder Block Diagram



Motion Compensation Module

The Motion Compensation Module performs all the motion compensation tasks required to decode MPEG-1, MPEG-2 and MPEG-4 bitstreams. This includes predicting the image block for the picture being decoded, using pixels from previously decoded pictures.

Audio Input and Output Interfaces

The audio decoding block supports the following audio bitstream formats:

- Dolby Digital with conformance to Group A (20-bit)
- MPEG-1 Layers 1 and 2
- MPEG-4 CELP and Low Complexity AAC (decoded using software on x86 host CPU)
- 16-bit linear PCM data
- Compressed Dolby Digital and DTS® digital output via S/PDIF

I2S Digital Audio Output

The I2S serial audio output block receives either the 2-channel down-mixed decoded Dolby Digital audio, decoded MPEG-1 audio, decoded MPEG-4 audio or PCM audio data. It then converts this data into a serial bitstream compatible with the I2S specification. The 256x F_s serial clock is generated by an internal digital PLL or an external clock source may be used.

S/PDIF Digital Audio Output

In addition to receiving the same audio data as the I2S digital audio output block, the S/PDIF output block can receive compressed DTS and compressed Dolby Digital audio data. It then converts this data into a serial bitstream compatible with the S/PDIF specification.

Analog Stereo Audio Output

The EM8475 and EM8476 include two on-chip audio DACs that receive the same 2-channel information as the I2S digital output and convert it to analog.

I2S Digital Audio Input

The EM8475 and EM8476 also support an I2S digital audio input. This audio data may be output onto the I2S or S/PDIF outputs.

Video Display Controller

The display controller reads picture data from DRAM and displays it with proper format, timing and synchronization signals. This is a real-time process driven by the video clock.

The display controller operates in one of four modes:

- Master mode -- the display controller generates HSYNC and VSYNC from an internal or external video clock up to 80 MHz
- Slave mode -- the display controller receives HSYNC and VSYNC from an internal or external video clock up to 80 MHz

The video display timing can be set for interlaced or non-interlaced (progressive) video output up to 120 Hz.

Sub-Pictures

Sub-pictures are compressed bit maps overlaid on decoded MPEG video which can be scrolled up and down and faded in and out. The area, content, color and contrast in every video field can be changed. These modifications produce special effects such as highlighting.

OSD (On-Screen Display)

The OSD enables simple full screen graphical menus to be displayed and blended with the MPEG decoded video and sub-picture. It supports 4 palletized color depths: 4 colors (2 bits per pixel), 16 colors (4 bits per pixel), 128 colors (7 bits per pixel) and 256 colors (8 bits per pixel). The bit map can be compressed using Run-Length Coding (RLC) in 2-, 4- and 7-bit per pixel modes. A 256x32 color look-up table (CLUT) is provided to convert the 2-, 4-, 7- or 8-bit code into a 24-bit YCbCr color and 16 levels of alpha blending. The Highlight function is supported in 2-, 4- and 7-bit per pixel modes.

The OSD supports programmable 3-line flicker filtering to improve the graphics quality on interlaced displays.

Letterbox Display

Letterbox mode provides vertical downscaling; 16:9 pictures can be displayed in a letterbox fashion on a traditional 4:3 display.

Pan and Scan Display

Pan and scan mode expands the video image to 16:9. A section of the image can be displayed at full height on a 4:3 TV display.

Alpha Blending

Alpha blending provides two layers of blending: sub-picture over the MPEG video and OSD over both the sub-picture and MPEG video. Up to 16 levels of blending are available.

Video Input and Output Interfaces

Video output ports provide digital RGB or YCbCr outputs and include an integrated NTSC/PAL video encoder and video DACs for composite, s-video and component analog outputs.

VBI Support

NTSC closed captioning on lines 21 and 284 is supported. In addition, NTSC widescreen signaling (WSS) and copy generation management (CGMS-A) on lines 20 and 283, as defined by IEC 61880, is supported. This information specifies the aspect ratio of the current program and also provides copy protection capabilities.

PAL widescreen signaling (WSS) on line 23, as defined by ETSI EN300284 and ITU-R BT.1119, is supported. This information specifies the aspect ratio of the current program.

Digital Video Output

The digital video output, controlled by the video display controller, supports 8- / 16-bit CCIR 601 / 656 4:2:2 YCbCr data. An additional 24-bit RGB output mode is supported on the EM8475 and EM8476. It may be operated as a timing master or slave at rates up to 80 MHz (160 MBps) and resolutions up to 720p or 1080i.

The 24-bit RGB output mode supports HDCP (High-bandwidth Digital Content Protection) when interfaced to a DVI transmitter chip.

TV Encoder - Composite and S-Video Output

A high-quality NTSC/PAL encoder (with optional Macrovision v7.1 protection) that supports the NTSC-M, NTSC-J, PAL-B/D/G/H/I, PAL-60 and PAL-M baseband video standards is available. It features three 10-bit video DACs to generate simultaneous composite and s-video outputs. The analog video outputs are capable of driving a doubly-terminated 75-ohm load.

Component Analog Output

The EM8475 and EM8476 also include component YPbPr or RGB analog outputs, with optional Macrovision AGC v1.03 protection in 480p YPbPr mode. This component analog output can be programmed to be either RGB or YPbPr video data, interlaced or progressive, from NTSC / PAL resolution up to 1920x1080. The analog video outputs are capable of driving a doubly-terminated 75-ohm load.

When generating analog RGB or YPbPr video, the composite video output is always present; the s-video output is not available.

Supported analog formats are:

- RGB and composite video for SCART support
- RGB with sync on green
- SMPTE GBR
- Betacam YUV
- M-II YUV
- SMPTE YPbPr

If copy protection is used during DVD-Video playback, the resolution on the EM8475 analog YPbPr / RGB outputs may not be higher than standard definition (720x480 or 720x576) unless driving a VGA monitor. If the output is configured to be progressive YPbPr, only 480p video may be generated.

Digital Video Input

The video input port, available only on the EM8475 and EM8476, is designed to capture video data in 8- / 16-bit CCIR 601 / 656 YCbCr video data formats. The capture port must operate in slave mode, synchronized to the input clock and sync signals. The capture port is designed to support video rates up to 80 MHz (160 MBps) and resolutions up to 720p or 1080i.

This input may be converted to progressive using either field merging or scan line interpolation.

Note that only one video stream can be displayed at a time - either MPEG video or input video.

Local Bus Controller Interface

The Local Bus Controller (LBC) interface, supported on the EM8475 and EM8476, is a versatile block, designed to interface the DRAM controller and PCI interface to an external device, such as a non-PCI MPEG encoder chip. The LBC is designed to handle DMA transfers (FIFO read/write from the DRAM) and single register read/write from the external device to or from the PCI interface. The same DMA transfers and register read/write operations can be done from the external device to or from the DRAM controller within the EM8475 and EM8476.

The LBC interface is compatible with the 16-bit multiplexed address / data format (Intel format), 16-bit data and 8-bit address format (generic format) and 16-bit compressed data format. It also directly supports the Stream Machine SM2210 / 2288 MPEG codecs.

PCI Host Interface

The PCI interface is the main conduit between the EM847x and the host processor. It is designed for both master reads and writes with programmable burst length and is PCI v2.1 compliant. The PCI interface supports 3.3V and 5V operation and ACPI power management from the PCI v2.2 specification.

Additional Software Available

VOD Client Software (MPEG-2)

VOD client software for the x86 is available that supports select MPEG-2 video servers, including:

- Cisco's IPTV family
- Concurrent's MediaHawk
- InfoValue's MediaQuick
- Kasenna's MediaBase
- nCUBE's NVS
- Streaming21's Media Server
- Thirdspace's OVS (Oracle Video Server)

For the latest list, please check with your local sales representative.

VOD Client Software (ISMA, RTSP)

VOD client software for the x86 is available that supports RTSP-based ISMA streaming.

IP Multicasting Client Software

IP multicasting client software for the x86 is available.

MPEG-4 Client System Software

Third party x86 client software is available that supports the EM847x, including Envivio and iVAST. For the latest list, please check with your local sales representative.

Memory Requirements

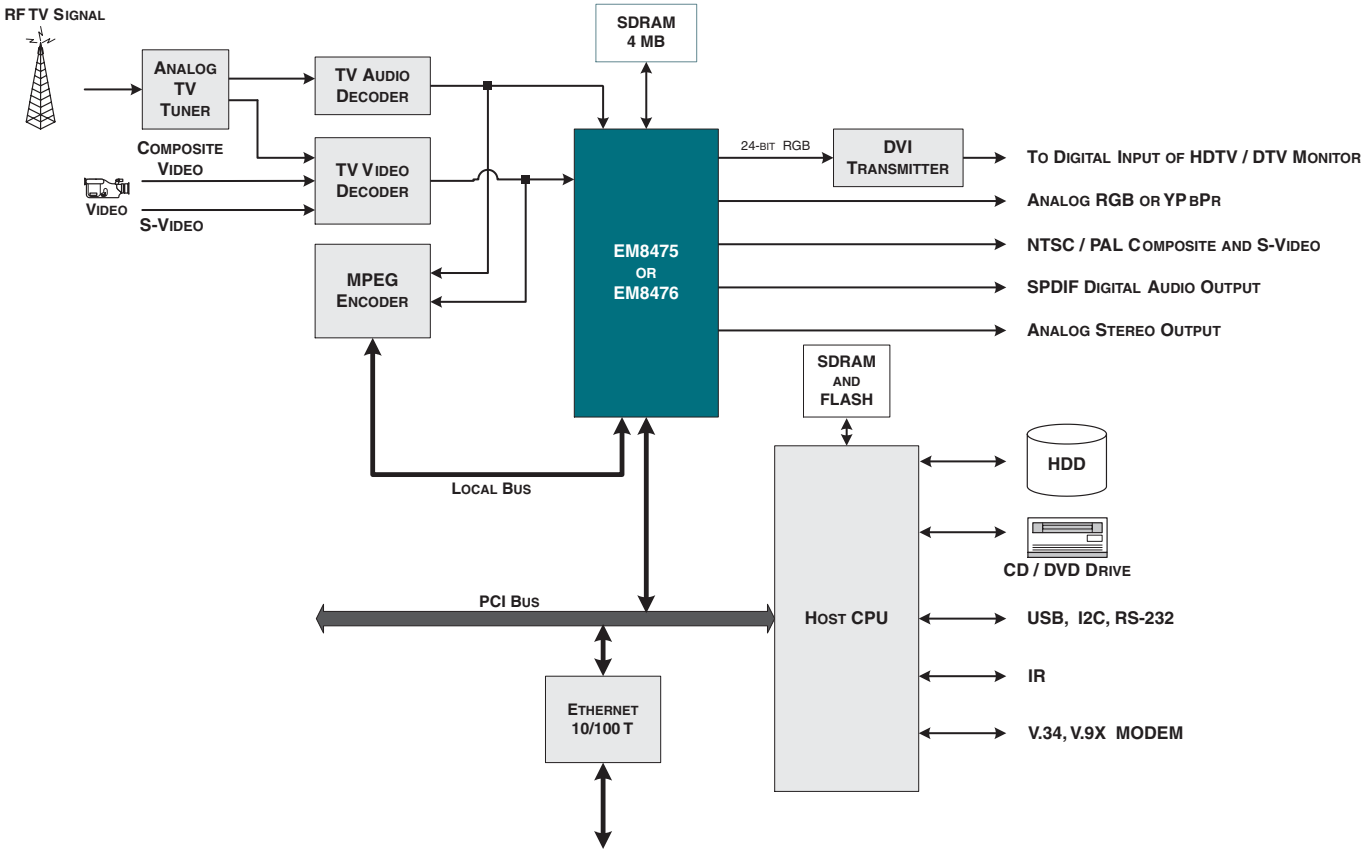
The EM847x requires a minimum of 4 MB (1Mb x 32) of 100 MHz SDRAM.

Sigma Designs, Inc.
355 Fairview Way • Milpitas, CA, USA 95035 • Tel: 408.262.9003 • Fax: 408.957.9740
www.sigmadesigns.com • sales@sigmadesigns.com

Application Example: Set-top Box with PVR, DVD-Video Playback, DVI Output

The set-top box example below supports PVR, DVD-Video playback, and DVI output capability. System integration requires very little external logic since the EM847x provides most of the features including:

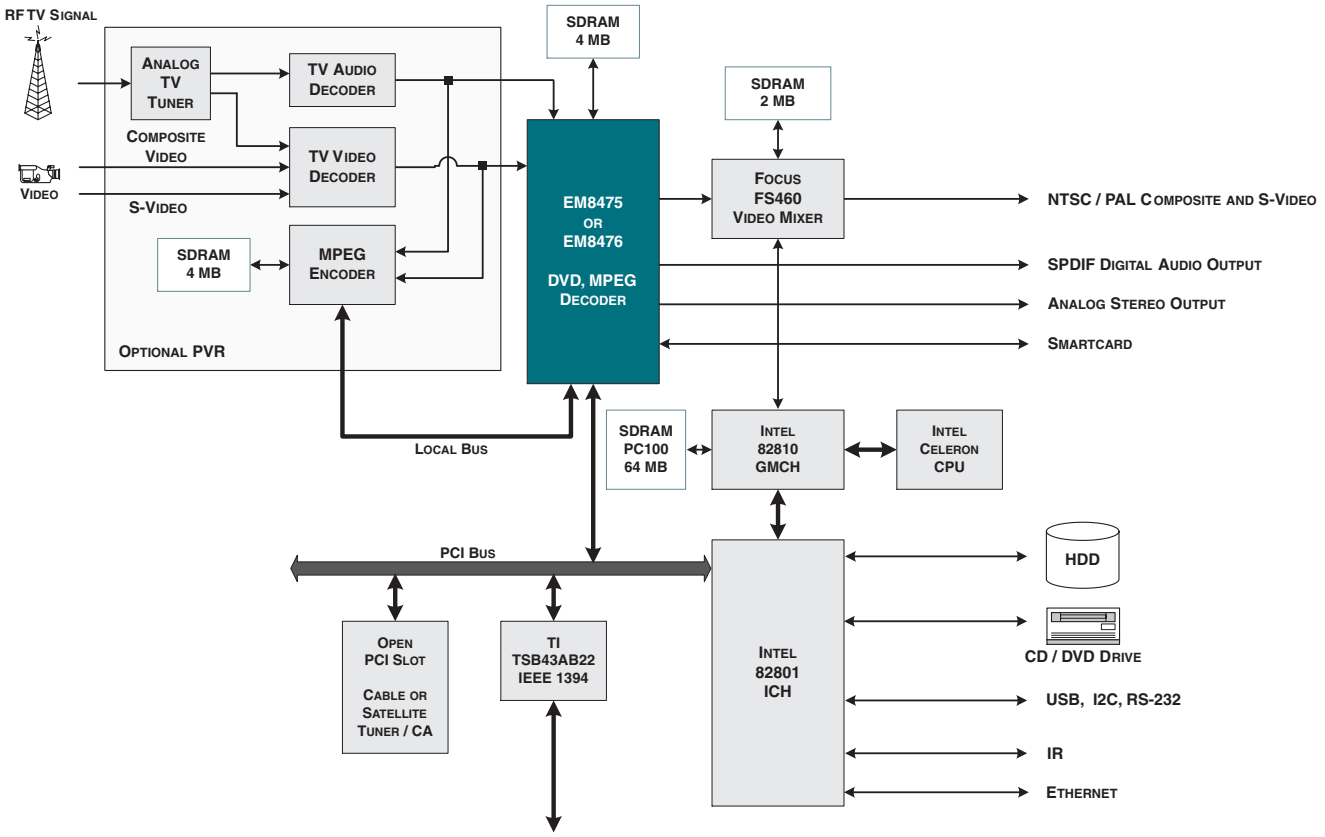
- Streaming video or video-on-demand (VOD) playback
- Progressive DVD-Video playback
- Interlaced or progressive YPbPr or RGB video outputs
- NTSC / PAL composite and s-video outputs
- Analog and S/PDIF audio outputs



Application Example: Gateway

The home gateway example below also supports PVR and DVD-Video playback. System integration requires very little external logic since the EM847x provides most of the features including:

- Streaming video or video-on-demand (VOD) playback
- Progressive DVD-Video playback
- Analog and S/PDIF audio outputs



Hardware vs. Software MPEG Decoding

Use hardware-based MPEG decoding when:

- Delivery of audio and video to a television is the primary function of the system. Standard consumer-oriented connectors for audio and video must be used, instead of PC-oriented break-out cables and mini jacks. (Example: set-top box.)
- Home theater quality audio and video (such as 6-channel audio analog outputs, component YPbPr or RGB analog video outputs and a SPDIF digital audio output) are required. Although specifications for audio and video solutions for the PC market look good on paper, the products themselves rarely meet consumer expectations.
- A relatively slow CPU, such as an embedded processor, is used to lower system costs. Slower CPUs also reduce power consumption and eliminate noisy fans. Decoding full-screen MPEG-4 requires a 1 GHz CPU, which is currently far too costly for low-cost set-top boxes to employ.
- Multiple video streams must be decoded simultaneously to support picture-in-picture (PIP).
- Ancillary data processing, such as copy protection, closed captioning (multiple languages), teletext (multiple languages), v-chip (multiple regions) and widescreen signaling must be performed for each active video stream displayed. Wide-screen processing must be able to handle anamorphic and letterboxed content for either 4:3 or 16:9 televisions. These are typically overlooked in software-based solutions.
- Alpha blending is required when overlaying text and graphics (such as closed captioning, teletext, EPG, OSD, etc.) over video to avoid artifacts (such as flicker) when displayed on a television. Such artifacts are caused by the limited-bandwidth video connection between the set-top box and television. It is typically implemented using anti-aliased fonts and graphics.
- High quality video scaling is required for large television screen sizes. Artifacts not noticeable on a small VGA monitor are magnified on a large television.

Use software-based MPEG decoding when:

- MPEG decoding is a secondary, rather than a primary function of the system. (Example: PC.)
- MPEG decoding is used in a single-tasking environment. Since MPEG decoding requires substantial CPU resources, using other applications usually affects playback quality.
- A fast CPU is already available on the platform.
- Only standard stereo audio and video (composite or VGA) outputs are needed.

General Specifications

Media Formats

- DVD-Video, Superbit DVD, SVCD (IEC 62107-2000), VCD 1.x, VCD 2.0, audio CD
- 15 Mbps sustained input data rate

Streaming Formats

- MPEG-2, ISMA
- 15 Mbps sustained input data rate

Video Decoding Standards

- MPEG-1, MPEG-2 MP@ML
- MPEG-4 Advanced Simple Profile Level 5[1]. Rectangular shape video decoding up to CCIR 601 resolution (720x480 or 720x576), support for B pictures, data partitioning support for error resiliency, up to 4 objects decoded in CIF resolution and compositing with text and graphics.
- Error concealment
- DVD-Video and Superbit DVD
  - CSS decryption
  - 16:9 and 4:3 playback, letterbox, 3:2 pull-down
  - Multiple angles and sub-picture
  - DVD-Video navigation software

Video Processing

- Brightness, color, contrast controls
- Advanced scaling up to 1920x1080 pixels
- Interlaced or non-interlaced output up to 120 Hz
- 2-, 4-, 7- or 8-bit OSD with optional run-length compression, alpha blending and flicker filtering

Video Interfaces

- 80 MHz 8- / 16-bit CCIR 601 / 656 YCbCr video input port supports capture resolutions up to 1920x1080i (EM8475 and EM8476 only)
- 80 MHz 8- / 16-bit CCIR 656 / 601 YCbCr video output port
- 80 MHz 24-bit RGB video output port for DVI with HDCP support (EM8475 and EM8476 only)
- NTSC/PAL composite and s-video analog outputs with optional Macrovision v7.1 protection (10-bit DACs)
- Analog RGB / YPbPr (SDTV or HDTV resolution, interlaced or progressive) with optional Macrovision AGC v1.03 protection in 480p output mode (10-bit DACs, EM8475 and EM8476 only)

Audio Decoding Standards

- 16-bit linear PCM
- MPEG-1 Layers 1 and 2
- MPEG-4 High Quality Profile Level 2
  - CELP and low-complexity AAC
- Dolby Digital down-mixed to 2 channels

Audio Interfaces

- I2S serial digital output for PCM or S/PDIF (IEC 60958) serial digital output for PCM, compressed Dolby Digital and compressed DTS
- I2S serial digital input (EM8475 and EM8476 only)
- On-chip stereo audio DACs (100 dB SNR, EM8475 and EM8476 only)

[1]Without 1/4 pixel and global motion compensation

General Specifications (continued)

Host Interface

- PCI v2.1 with programmable burst length

Local Bus Interface

- PCI-to-Local Bus bridge
- 16-bit multiplexed address / data (Intel mode)
- 16-bit data and 8-bit address (generic mode)
- 16-bit data compressed
- Supports Stream Machine SM2210 / SM2288 MPEG encoder

Power Management

- 2.5V core with 3.3V I/O (5V tolerant)
- Typical power dissipation:
  - Analog video outputs off: 1.2W
  - Analog video outputs on: 1.4W

Package

- 208-pin PQFP for EM8470 and EM8471 (pin compatible with EM8400)
- 328-pin BGA for EM8475 and EM8476

Product Selection Guide

Feature	EM8470[1]	EM8471[2]	EM8475[1]	EM8476[2]
Media Formats				
DVD-Video, Superbit DVD, SVCD, VCD 1.x, VCD 2.0, audio CD	yes	yes	yes	yes
Video Decoding				
MPEG-1, MPEG-2 MP@ML, MPEG-4 Advanced Simple Profile Level 5[3]	yes	yes	yes	yes
Audio Decoding				
Dolby Digital, MPEG-1 Layers 1 and 2	yes	yes	yes	yes
MPEG-4 High Quality Profile Level 2	yes	yes	yes	yes
Digital Video Inputs				
8- / 16-bit YCbCr (80 MHz)	-	-	yes	yes
Digital Video Outputs				
8- / 16-bit YCbCr (80 MHz)	yes	yes	yes	yes
24-bit RGB for DVI / HDCP support (80 MHz)	-	-	yes	yes
NTSC/PAL Composite and S-video Outputs	yes	yes	yes	yes
Macrovision v7.1 protection	yes	-	yes	-
YPbPr or RGB Analog Video Outputs	-	-	yes	yes
Macrovision AGC v1.03 protection for 480p output mode	-	-	yes	-
Audio Inputs and Outputs				
Stereo analog output	-	-	yes	yes
S/PDIF / I2S digital output	yes	yes	yes	yes
I2S digital input	-	-	yes	yes
Local Bus Interface	-	-	yes	yes
Package	208 PQFP	208 PQFP	328 BGA	328 BGA
Power Supply	2.5v / 3.3v	2.5v / 3.3v	2.5v / 3.3v	2.5v / 3.3v

[1]May be sampled or sold only to companies that are both Macrovision and Dolby licensed.
[2]May be sampled or sold only to companies that are Dolby licensed.
[3]Without 1/4 pixel and global motion compensation.

Sales Offices

UNITED STATES
Sigma Designs, Inc.
355 Fairview Way
Milpitas, CA 95035
+1 (408) 262-9003
+1 (408) 957-9740 FAX

CHINA
Sigma Designs China
Room C1, 32F
Electronic Science & Technology Building (Phase 2)
30 Shennan Road Central
Shenzhen, PRC
Postcode 518031
+86 755 3683878
+86 755 3683873 FAX

EUROPE
Sigma Designs, Inc.
49, Rue des Moissonneurs
Brussels, Belgium 1040
+32 496.501234
+32 234.72260 FAX

HONG KONG
Sigma Designs (Asia) Ltd.
Unit 1516, Tower 1, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T.
Hong Kong
+852 2401 7388
+852 2610 2177 FAX

JAPAN
Sigma Designs Japan
4-16-8 Nakahara,
Mitaka-shi,
Tokyo 181-0005, Japan
+81 422 79 3067
+81 422 79 3067 FAX

KOREA
Sigma Designs Korea
801, Dongil Techno Tower B/D
Kuro-dong, Kuro-gu,
Seoul, Korea
+82 11 288 0406
+82 2 3281 2034 FAX

TAIWAN
Sigma Designs Taiwan
Far East World Center, C Tower
8F-8, No. 79, Sec. 1
Hsin Tai Wu Road
Hsichih, Taipei Hsien
Taiwan, R.O.C.
+886 2 2698 2066
+886 2 2698 2099 FAX

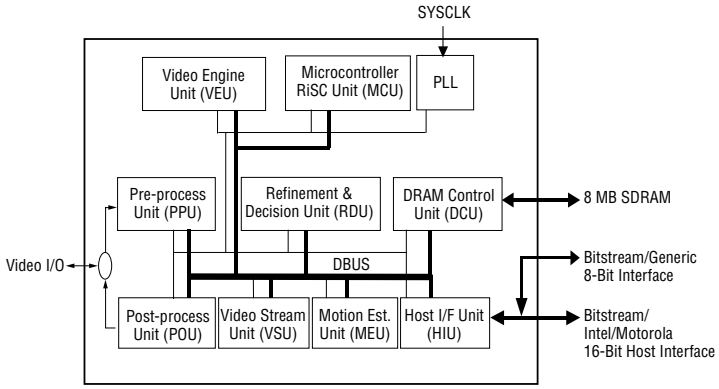
Revision Date: March 18, 2002

EXHIBIT E

MPEG-2 VIDEO Encoder/Decoder

CS92210 Features

- Single chip MPEG-2 Video CODEC
- Supports real time MPEG-1 encoding and decoding
- Supports real time MPEG-2 MP@ML, SP@ML, and MP@LL encoding and decoding
- Support for constant and one-pass variable bit rate
  - IPB-pictures, CBR or VBR to 15 Mbps
  - I-pictures only to 30 Mbps
- Proprietary high performance motion estimation
- Low external memory
  - 8 Mbytes for full D1 (720) NTSC/PAL pictures
- Provides complete video encoding and decoding (half-duplex) when combined with system function and supporting commodity devices
- Direct interface to video modulator & demodulator
- Supports multiple resolutions & scan rates
  - NTSC: (720-D1, 704-D1, 640-VGA, 544, 480-2/3D1, 352-1/2D1) x 480, or 352 x 240 (CIF), 320 x 240 (MPEG-1) or 176 x 112 (QCIF) at 30 or 29.97 Hz
  - PAL: (720-D1, 704-D1, 640-VGA, 544, 480-2/3D1, 352-1/2D1) x 576, or 352 x 288 (CIS/SIF), or 176 x 144 (QCIF) at 25 Hz
- Intel/Motorola 16-bit host interface
- 5 V I/O tolerance, 3.3 V and 1.8 V power supplies
- 0.65 watts @ 87.75 MHz average power consumption
- 256-pin PQFP package



The CS92210 is a real time MPEG-2 video encoder and decoder (CODEC) that fully complies with the ISO/IEC-13818 Main Profile@Main Level (MP@ML) format, Simple Profile@ Main Level (SP@ML), and Main Profile @ Low Level (MP@LL).

In encode mode, the CS92210 accepts digital video in ITU-R BT.601 (CCIR-601) or ITU-R BT.656 (CCIR-656) format. The input is filtered and then encoded to produce compressed bitstreams in MPEG-2 MP@ML syntax. In decode mode, CS92210 accepts an MPEG bitstream, decodes it, and then filters the video output to produce either ITU-R BT.601 or ITU-R BT.656 format digital video. Designed for flexibility, the CS92210's video interface supports multiple video formats, resolutions, and frame rates including NTSC, PAL, and FILM.

The CS92210's versatile time-stamp feature enables flexible muxing of audio and video elementary bit streams. Also, the CS92210 can encode and decode bitstreams in both the VCD and SVCD formats.

The algorithmic and architectural innovations of the CS92210 enable a high degree of integration while still providing exceptional video quality over the widest range of bit rates. Also, the CS92210 provides ease of system design by interfacing to a wide variety of commodity components such as Philips video decoders and encoders, Flash and SDRAM memories.

Technical Overview

The CS92210 is organized as a process pipeline that implements the MPEG-2 encoding and decoding algorithms.

The CS92210 provides application program control over a large number of encoding parameters such as I, P, B-picture cadence, GOP structure and decoder buffer sizes.

For communications applications, the CS92210 can match its output bit rate to the channel rate. This feature allows the host controller to make bit rate changes as needed to demonstrate better bandwidth utilization across multiple channels.

Internal rate control provides a high degree of flexibility in relation to the output bit rate, including the ability to generate variable bitrate compressed video stream in one pass. This makes it suitable for storage sensitive applications such as digital camcorders and personal video recorders (PVRs).

The CS92210 also has features geared toward MPEG-2 publishing and authoring systems.

Pre- and post-processing supports includes pre- and post-filtering and up and down chroma conversions. Other features include:

- DMA in either 8-b or 16-b modes
- Encodes/decodes full D1 to QCIF video resolutions
- Bit rates up to 15 Mb/s in either CBR or VBR modes
- Debugging and DMA monitoring control
- Asynchronous video and system clocks
- Support for commodity video NTSC/PAL encoders and decoders

VCD, Super-VCD Support

The CS92210 supports MPEG-1 and MPEG-2 video encoding at 1/2 (VCD), 2/3 (SVCD), and full D1 resolutions. In addition, the CS92210's versatile pre-processing features allow the input video to be either scaled or cropped to the desired encode size.

Interfaces

The CS92210 includes a 64-bit SDRAM memory interface, a video interface, 16-bit Motorola/Intel host interface, a generic 8-bit interface, and a serial EPROM /Flash memory interface.

Applications

The CS92210 can be used in a variety of applications:

- VCD, Super-VCD player and recorder
- DVD-recordable products
- Advanced set-top boxes
- Personal video recorder (time shifting)
- PC-based content creation/editing boards
- USB-based products for video capture and display

Cirrus Logic, Inc.
Corporate Headquarters

4210 S. Industrial Drive
Austin, TX 78744
USA
T (512) 445-7222
T (800) 888-5016
F (512) 912-3977
www.cirrus.com

WORLDWIDE DISTRIBUTION

United States:

Insight Electronics
9980 Huennekens
San Diego, California 92121
T (800) 677-6011
F (858) 450-8550
www.insight-electronics.com

Nu Horizons
70 Maxess Road
Melville, New York 11747
T (631) 396-5000
F (631) 396-5060
www.nuhorizons.com

Asia, Europe, and Japan:
Please access our website, www.cirrus.com, for your nearest local distributor.

WORLDWIDE SALES

Regional Offices:

United States
Cirrus Logic, Inc.
46831 Lakeview Blvd
Fremont, California 94538
T (510) 623-8300
F (510) 252-6020

Japan
Cirrus Logic K.K.
Aioi Sonpo, bldg 6F
5-6 Niban-cho, Chiyoda-ku
Tokyo, Japan
T 81-3-5226-7390
F 81-3-5226-7677

Asia
Cirrus Logic Intl. Ltd.
20F., Ocean Building
80 Shanghai Street
Kowloon, Hong Kong, China
T 852-2376-0801
T 852-2314-9920
F 852-2375-1202

Europe
Cirrus Logic UK
4-5 Anglers Court
33-44 Spittal Street
Marlow, Bucks SL71DB
England
T 44-0-1628-472-211
F 44-0-1628-486-114

EXHIBIT F

DATA SHEET

SAA7201
Integrated MPEG2 AVG decoder

Objective specification
Supersedes data of 1997 Jan 29
File under Integrated Circuits, IC02

2001 Mar 28

Philips Semiconductors

PHILIPS

FEATURES

General

- Uses single external Synchronous DRAM (SDRAM) organized as 1M × 16 interfacing at 81 MHz; compatible with the SDRAM 'lite' or 'PC'
- Fast external CPU interface; 16-bit data + 8-bit address
- Dedicated input for audio and video data in PES or ES format; data input rate: ≤9 Mbytes/s in byte mode; ≤20 Mbit/s in bit serial mode; audio and/or video data can also serve as input via CPU interface
- Single 27 MHz external clock for time base reference and internal processing; all required decoding and presentation clocks are generated internally
- Internal system time base at 90 kHz can be synchronized via CPU port
- Flexible memory allocation under control of the external CPU enables optimized partitioning of memory for different tasks
- Boundary scan (JTAG) plus external SDRAM self test implemented
- Supply voltage 3.3 V
- Package 160 QFP.

CPU relation

- 16-bit data, 8-bit address, or 16-bit multiplexed bus; Motorola and Intel mode supported
- Support for fast DMA transfer to either internal registers or external SDRAM
- Maximum sustained rate to the external SDRAM is 9 Mbytes/s.

MPEG2 system

- Parsing of MPEG2 PES and MPEG1 packet streams
- Double System Time Clock (STC) counters for discontinuity handling
- Time stamps or CPU controlled audio/video synchronization
- Support for seamless time base change (edition)
- Processing of errors flagged by channel decoding or demux section
- Support for retrieval of PES header and PES private data.

MPEG2 audio

- Decoding of 2 channel, layer I and II MPEG audio; support for mono, stereo, intensity stereo and dual channel mode
- Constant and variable bit rates up to 448 kbit/s
- Audio sampling frequencies: 48, 44.1, 32, 24, 22.05 and 16 kHz
- CRC error detection
- Selectable output channel in dual channel mode
- Independent volume control for both channels and programmable inter-channel crosstalk control through a baseband audio processing unit
- Storage ancillary data up to 54 bytes
- Dynamic range control at output
- Muting possibility via external controller; automatic muting in case of errors
- Generation of 'beeps' with programmable tone height, duration and amplitude
- Serial two channel digital audio output with 16, 18, 20 or 22 bits per sample, compatible with either I^2S or Japanese formats
- Serial SPDIF audio output
- Clock output 256 or 384 × f_s for external D/A converter
- Audio input buffer in external SDRAM with programmable size (default is 64 kbit)
- Programmable processing delay compensation
- Software controlled stop, pause, restricted skip, and restart functions.

MPEG2 video

- Decoding of MPEG2 video up to main level, main profile
- Nominal video input buffer size equals 2.6 Mbit for Video Main Profile and Main Level (MP@ML)
- Output picture format: CCIR-601 4 : 2 : 2 interlaced pictures; picture format 720 × 576 at 50 Hz or 720 × 480 at 60 Hz
- 3 : 2 pull-down supported with 24 and 30 Hz sequences
- Support of constant and variable bit rates up to 15 Mbit/s
- Output interface at 8-bit wide, 27 MHz UYVY multiplexed bus
- Horizontal and vertical pan and scan allows the extraction of a window from the coded picture

- Flexible horizontal continuous scaling from 0.5 up to 4 allows easy aspect ratio conversion including support for 2.21 : 1 aspect ratio movies
- Vertical scaling with fixed factors 0.5, 1 or 2 to support picture scaling and up-sampling
- Scaling of incoming pictures to 25% of their original size with anti-aliasing filtering to free screen space for graphics applications like electronic program guides
- Non-full screen MPEG pictures will be displayed in a box of which position and background colour are adjustable by the external CPU
- Video output may be slaved to internally (master) generated or externally (slave) supplied HV synchronization signals; the position of active video is programmable; MPEG timebase changes do not affected the display phase
- Video output direct connectable to SAA718X encoder family
- Various trick modes under control of external CPU:
  - Freeze I or P pictures; restart on I picture
  - Freeze on B pictures; restart at any moment
  - Scanning and decoding of I or I and P pictures
  - Single step mode
  - Repeat/Skip field for time base correction.

Graphics

- Graphics is region based and presented in boxes independent of video format
- Screen arrangement of boxes is determined by display list mechanism which allows for multiple boxes, background loading, fast switching, scrolling and fading of regions
- Support of 2, 4, 8 bits/pixel bit-maps in fixed bit-maps or coded in accordance to the DVB variable/run length standard for region bases graphics
- Optimized memory control in MPEG video decoding allows for storage of graphical bit-maps up to 1.2 Mbit in 50 Hz and 2.0 Mbit in 60 Hz systems
- VL/RL encoding enables full screen graphics at 8 bit/pixel in 50 Hz
- Fast CPU access enables full bit-map updates within a display field period
- Display colours are obtained via colour look-up tables; CLUT output is YUVT at 8-bit for each signal component thus enabling 16M different colours and 6-bit for T (transparency) which gives 64 mixing levels with video
- Bit-map table mechanism to specify a sub-set of entries if the CLUT is larger than required by the coded bit pattern; supported bit-map tables are 16 to 256, 4 to 256 and 4 to 16
- Graphics boxes may not overlap vertically; if 256 entry CLUT has to be down loaded, a vertical separation of 1 field line is mandatory
- Internal support for fast block moves in the external SDRAM during MPEG decoding
- Graphics mechanism can be used for signal generation in the vertical blanking interval; useful for teletext, wide screen signalling, closed caption etc.
- Support for a single down-loadable cursor of 1 kpixel with programmable shape; supported shapes are 8×128, 16×64, 32×32, 64×16 and 128×8
- Cursor colours are determined via a 4-entry CLUT with YUVT at 6, 4, 4 respectively 2 bits; mixing of cursor with video + graphics in 4 levels
- Cursor can be moved freely across the screen without overlapping restrictions.

GENERAL DESCRIPTION

The SAA7201 is an MPEG2 decoder which combines audio decoding and video decoding. Additionally to these basic MPEG functions it also provides means for enhanced graphics and/or on-screen display.

Due to an optimized architecture for audio and video decoding, maximum capacity in the external memory and processing power from the external CPU is available for the support for graphics.

QUICK REFERENCE DATA

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
V_{DD}	functional supply voltage	3.0	3.3	3.6	V
V_{CC}	pad supply voltage	3.0	3.3	3.6	V
$I_{DD(tot)}$	total supply current at V_{DD} = 3.3 V	–	tbf	–	mA
f_{CLK}	clock frequency	–	27.0	–	MHz
Δf_{CLK}	frequency deviation	-30×10^{-6}	–	$+30 \times 10^{-6}$	

ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
SAA7201H	QFP160	plastic quad flat package; 160 leads (lead length 1.95 mm); body 28 × 28 × 3.4 mm; high stand-off height	SOT322-4

BLOCK DIAGRAM

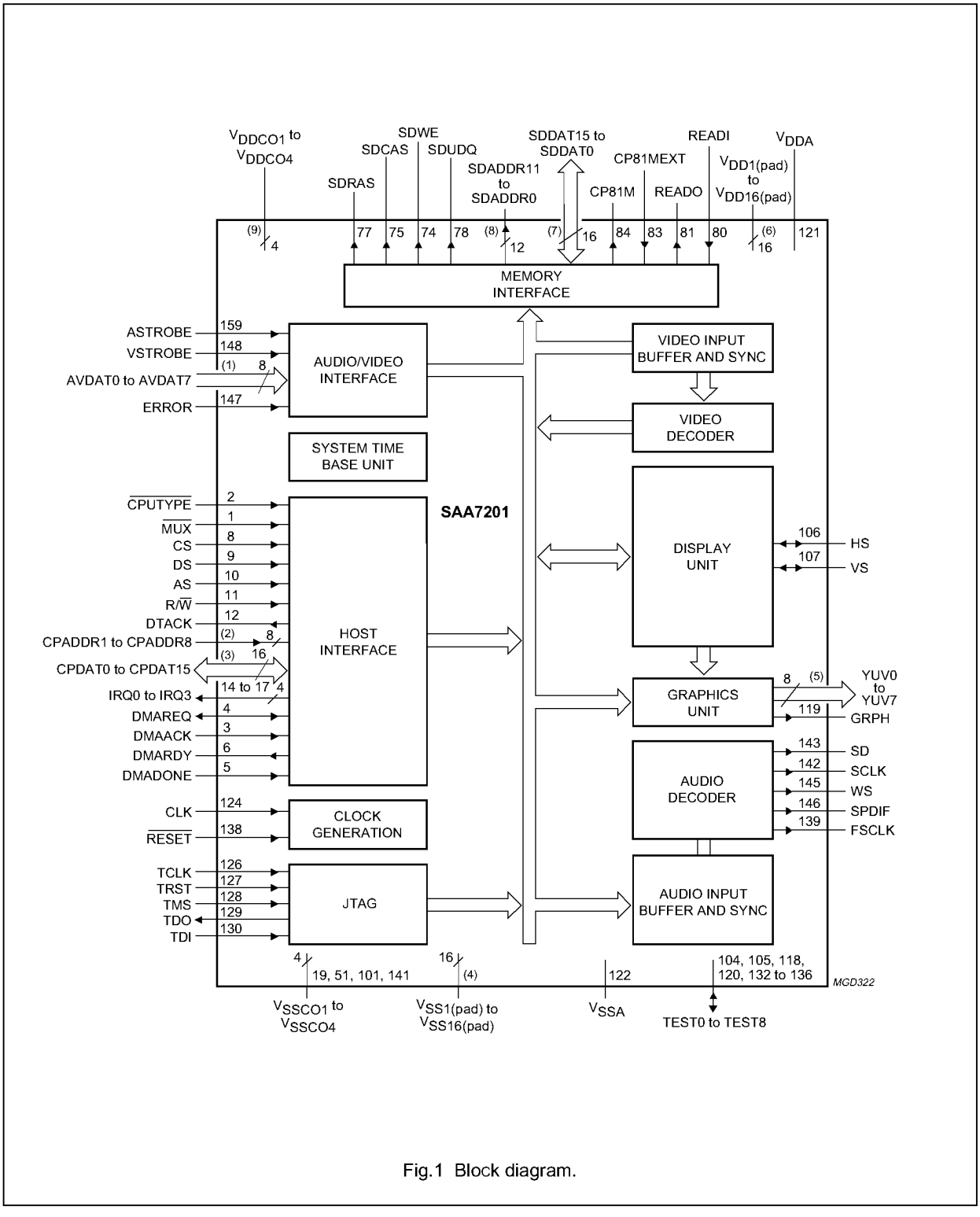


Fig.1 Block diagram.

PINNING

SYMBOL	PIN	DESCRIPTION	V	I/O
MUX	1	multiplexed/non-multiplexed (active LOW) bus input	5.0	I
CPU_TYPE	2	Intel/Motorola (active LOW) selection input	5.0	I
DMA_ACK	3	DMA acknowledge input	3.3	I
DMA_REQ	4	DMA request input and output	3.3	I/O
DMA_DONE	5	DMA end input	3.3	I
DMA_RDY	6	DMA ready output	3.3	O/Z
V_{SS1}	7	ground for pad ring	3.3	–
CS	8	chip select input	5.0	I
DS	9	data strobe input	5.0	I
AS	10	address strobe input	5.0	I
R/$\overline{W}$	11	read/write (active LOW) input	5.0	I
DTACK	12	data acknowledge output	5.0	O/Z
V_{DD1}	13	supply for pad ring	3.3	–
IRQ0	14	individually maskable interrupts	3.3	O/Z
IRQ1	15	individually maskable interrupts	3.3	O/Z
IRQ2	16	individually maskable interrupts	3.3	O/Z
IRQ3	17	individually maskable interrupts	3.3	O/Z
V_{SS2}	18	ground for pad ring	–	–
V_{SSCO1}	19	ground for core logic	–	–
V_{DDCO1}	20	supply for core logic	3.3	–
DATA0	21	CPU data interface	5.0	I/O
DATA1	22	CPU data interface	5.0	I/O
DATA2	23	CPU data interface	5.0	I/O
DATA3	24	CPU data interface	5.0	I/O
V_{DD2}	25	supply for pad ring	3.3	–
DATA4	26	CPU data interface	5.0	I/O
DATA5	27	CPU data interface	5.0	I/O
DATA6	28	CPU data interface	5.0	I/O
DATA7	29	CPU data interface	5.0	I/O
V_{SS3}	30	ground for pad ring	–	–
DATA8	31	CPU data interface	5.0	I/O
DATA9	32	CPU data interface	5.0	I/O
DATA10	33	CPU data interface	5.0	I/O
DATA11	34	CPU data interface	5.0	I/O
V_{DD3}	35	supply for pad ring	–	–
DATA12	36	CPU data interface	5.0	I/O
DATA13	37	CPU data interface	5.0	I/O
DATA14	38	CPU data interface	5.0	I/O
DATA15	39	CPU data interface	5.0	I/O
V_{SS4}	40	ground for pad ring	–	–

SYMBOL	PIN	DESCRIPTION	V	I/O
ADDRESS1	41	CPU address interface	5.0	I
ADDRESS2	42	CPU address interface	5.0	I
ADDRESS3	43	CPU address interface	5.0	I
ADDRESS4	44	CPU address interface	5.0	I
V_{DD4}	45	supply for pad ring	3.3	–
ADDRESS5	46	CPU address interface	5.0	I
ADDRESS6	47	CPU address interface	5.0	I
ADDRESS7	48	CPU address interface	5.0	I
ADDRESS8	49	CPU address interface	5.0	I
V_{SS5}	50	ground for pad ring	–	–
V_{SSCO2}	51	ground for core logic	–	–
V_{DDCO2}	52	supply for core logic	3.3	–
SDRAM_DATA0	53	memory data interface	3.3	I/O
SDRAM_DATA15	54	memory data interface	3.3	I/O
SDRAM_DATA1	55	memory data interface	3.3	I/O
V_{DD5}	56	supply for pad ring	3.3	–
SDRAM_DATA14	57	memory data interface	3.3	I/O
SDRAM_DATA2	58	memory data interface	3.3	I/O
SDRAM_DATA13	59	memory data interface	3.3	I/O
V_{SS6}	60	ground for pad ring	–	–
SDRAM_DATA3	61	memory data interface	3.3	I/O
SDRAM_DATA12	62	memory data interface	3.3	I/O
SDRAM_DATA4	63	memory data interface	3.3	I/O
V_{DD6}	64	supply for pad ring	3.3	–
SDRAM_DATA11	65	memory data interface	3.3	I/O
SDRAM_DATA5	66	memory data interface	3.3	I/O
SDRAM_DATA10	67	memory data interface	3.3	I/O
V_{SS7}	68	ground for pad ring	–	–
SDRAM_DATA6	69	memory data interface	3.3	I/O
SDRAM_DATA9	70	memory data interface	3.3	I/O
SDRAM_DATA7	71	memory data interface	3.3	I/O
V_{DD7}	72	supply for pad ring	3.3	–
SDRAM_DATA8	73	memory data interface	3.3	I/O
SDRAM_WE	74	SDRAM write enable output	3.3	O
SDRAM_CAS	75	SDRAM column address strobe output	3.3	O
V_{SS8}	76	ground for pad ring	–	–
SDRAM_RAS	77	SDRAM row address strobe output	3.3	O
SDRAM_UDQ	78	SDRAM write mask output	3.3	O
V_{DD8}	79	supply for pad ring	3.3	–
$READ_I$	80	read command input	3.3	I

SYMBOL	PIN	DESCRIPTION	V	I/O
$READ_O$	81	read command output	3.3	O
V_{SS9}	82	ground for pad ring	–	–
CP81MEXT	83	81 MHz clock return path input	3.3	I
CP81M	84	81 MHz memory clock output	3.3	O
V_{DD9}	85	supply for pad ring	3.3	–
SDRAM_ADDR8	86	memory address	3.3	O
SDRAM_ADDR9	87	memory address	3.3	O
SDRAM_ADDR11	88	memory address	3.3	O
V_{SS10}	89	ground for pad ring	–	–
SDRAM_ADDR7	90	memory address	3.3	O
SDRAM_ADDR10	91	memory address	3.3	O
SDRAM_ADDR6	92	memory address	3.3	O
V_{DD10}	93	supply for pad ring	3.3	–
SDRAM_ADDR0	94	memory address	3.3	O
SDRAM_ADDR5	95	memory address	3.3	O
SDRAM_ADDR1	96	memory address	3.3	O
V_{SS11}	97	ground for pad ring	–	–
SDRAM_ADDR4	98	memory address	3.3	O
SDRAM_ADDR2	99	memory address	3.3	O
SDRAM_ADDR3	100	memory address	3.3	O
V_{SSCO3}	101	ground for core logic	–	–
V_{DDCO3}	102	supply for core logic	3.3	–
V_{DD11}	103	supply for pad ring	3.3	–
TEST8	104	IC test interface	3.3	I/O
TEST7	105	IC test interface	3.3	I/O
HS	106	horizontal synchronization input and output	3.3	I/O
VS	107	vertical synchronization input and output	3.3	I/O
V_{SS12}	108	ground for pad ring	–	–
YUV0	109	YUV video output at 27 MHz	3.3	O/Z
YUV1	110	YUV video output at 27 MHz	3.3	O/Z
YUV2	111	YUV video output at 27 MHz	3.3	O/Z
YUV3	112	YUV video output at 27 MHz	3.3	O/Z
V_{DD12}	113	supply for pad ring	3.3	–
YUV4	114	YUV video output at 27 MHz	3.3	O/Z
YUV5	115	YUV video output at 27 MHz	3.3	O/Z
YUV6	116	YUV video output at 27 MHz	3.3	O/Z
YUV7	117	YUV video output at 27 MHz	3.3	O/Z
TEST6	118	IC test interface	3.3	I/O
GRPH	119	indicator for graphics information output	3.3	O
TEST5	120	IC test interface	3.3	I/O

SYMBOL	PIN	DESCRIPTION	V	I/O
V_{DDA}	121	supply for analogue blocks	3.3	–
V_{SSA}	122	ground for analogue blocks	–	–
V_{SS13}	123	ground for pad ring	–	–
CLK	124	27 MHz clock input	3.3	I
V_{SS14}	125	ground for pad ring	–	–
TCLK	126	boundary scan test clock input	3.3	I
TRST	127	boundary scan test reset input	3.3	I
TMS	128	boundary scan test mode select input	3.3	I
TD_O	129	boundary scan test data output	3.3	O
TD_I	130	boundary scan test data input	3.3	I
V_{DD13}	131	supply for pad ring	3.3	–
TEST4	132	IC test interface	3.3	I/O
TEST3	133	IC test interface	3.3	I/O
TEST2	134	IC test interface	3.3	I/O
TEST1	135	IC test interface	3.3	I/O
TEST0	136	IC test interface	3.3	I/O
V_{DD14}	137	supply for pad ring	3.3	–
$\overline{RESET}$	138	hard reset input (active LOW)	3.3	I
FSCLK	139	256 or 384 f_s (audio sampling) output	3.3	O/Z
V_{DDCO4}	140	supply for core logic	3.3	–
V_{SSCO4}	141	ground for core logic	–	–
SCLK	142	serial audio clock output	3.3	O/Z
SD	143	serial audio data output	3.3	O/Z
V_{SS15}	144	ground for pad ring	–	–
WS	145	word select output	3.3	O/Z
SPDIF	146	digital audio output	3.3	O/Z
ERROR	147	flag for bitstream error input	5.0	I
V_STROBE	148	video strobe input	5.0	I
V_{DD15}	149	supply for pad ring	3.3	–
AV_DATA0	150	MPEG input port for PES data	5.0	I
AV_DATA1	151	MPEG input port for PES data	5.0	I
AV_DATA2	152	MPEG input port for PES data	5.0	I
AV_DATA3	153	MPEG input port for PES data	5.0	I
V_{SS16}	154	ground for pad ring	–	–
AV_DATA4	155	MPEG input port for PES data	5.0	I
AV_DATA5	156	MPEG input port for PES data	5.0	I
AV_DATA6	157	MPEG input port for PES data	5.0	I
AV_DATA7	158	MPEG input port for PES data	5.0	I
A_STROBE	159	audio strobe input	5.0	I
V_{DD16}	160	supply for pad ring	3.3	–

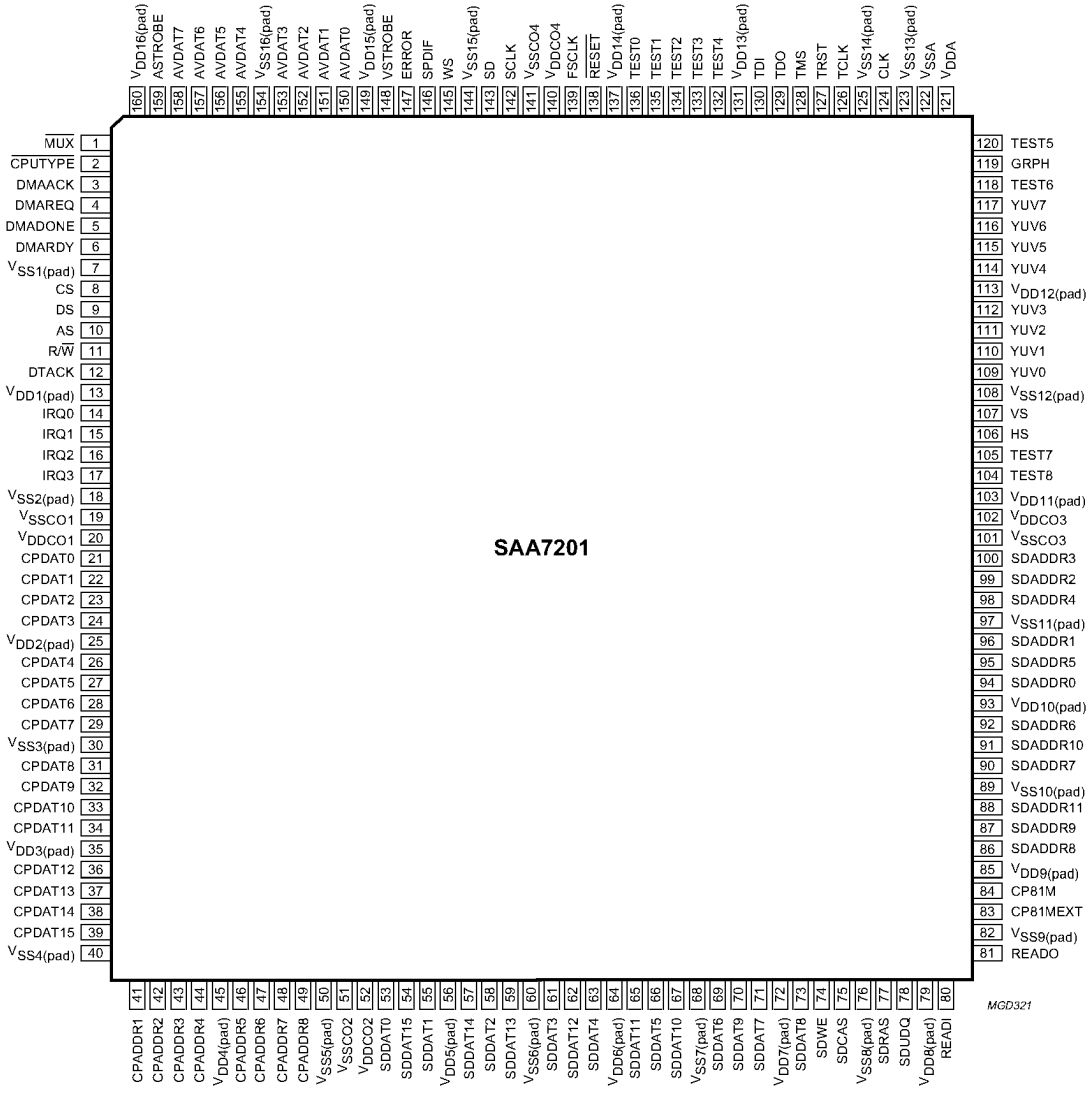


Fig.2 Pin configuration.

FUNCTIONAL DESCRIPTION

General

The SAA7201 is an MPEG2 decoder which combines audio decoding, video decoding and enhanced region based graphics. The decoder operates with a single 16 Mbit external synchronous dynamic random access memory (SDRAM) and runs from a single external 27 MHz clock. Due to the optimized memory control for MPEG2 decoding, more than 1 Mbit is available for graphics in 50 Hz systems.

MPEG2 data can be accepted up to 9 Mbytes/s through a dedicated byte wide interface. The data on this interface can be either in PES (Packetized Elementary Stream), MPEG1 packet or ES (Elementary Stream) format as described in Chapter “References”. Two additional strobe signals distinguish between audio and video data.

The internal video decoder is capable of decoding all MPEG compliant streams up to main level main profile as specified in Chapter “References”. The audio decoder implements 2 channel audio decoding according to the standards in Chapter “References”.

All real time audio/video decoding and synchronization tasks are performed autonomously, so the external microcontroller only needs to perform high-level tasks like initialization, status monitoring and trick mode control.

The main support task of the external microcontroller concerns the control of the graphical unit. This unit should be supplied with bit-maps, determining the contents of the graphical regions and by a simple set of instructions determining the appearance of the graphical data on the screen. Most graphical information should be stored in the external memory which implies multiple data transfers between CPU and the external memory. By performing these data transfers on a direct memory access (DMA) basis, full bit-maps can be transferred within one video frame period.

The video output, containing a mix of MPEG video and graphical data, is at a YUV multiplexed format which can be directly connected to an external composite video encoder. The audio output, containing a mix of MPEG audio and programmable ‘beeps’, is in a serial, I^2S or Japanese format which can be directly supplied to most commercially available up-sampling audio DA converters.

A functional block diagram of the decoder is given in Fig.1. Its application environment is depicted in Fig.24. In the following sections, a brief description of the individual internal blocks of the MPEG2 decoder will be given.

Audio/video interface

In a basic set-top box application the SAA7201 receives audio and video PES data in a byte wide format at rates up to 9 Mbytes/s. A timing diagram is shown in Fig.3. Next to the 8-bit wide data bus an audio and video strobe is expected at the input. Erroneous data may be flagged via the error indicator.

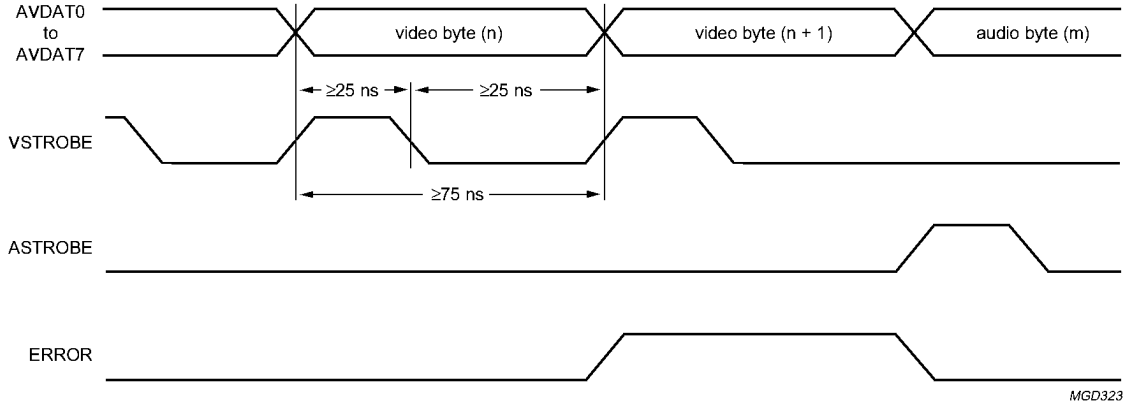


Fig.3 Timing diagram of parallel input mode.

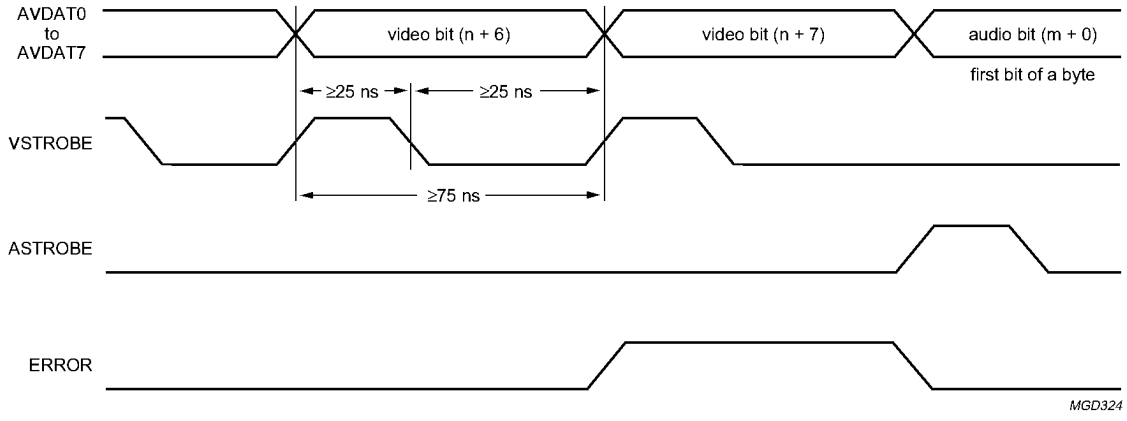


Fig.4 Timing diagram of serial input mode.

Alternatively data can be received in a 1-bit serial format at rates up to 20 Mbit/s. In this mode, data is input at the LSB input of the AV_DATA bus. Audio and video data must be input in multiples of 8 bits. The first bit after switching from audio to video (or the other way around) must be the first bit of a byte since this transition will be used for the internal bit-to-byte conversion.

Audio/video data can also be received via the CPU interface in 8 or 16-bit mode. The peak rate is 27 Mbytes/s in bursts of ≤128 bytes with a sustained rate up to 9 Mbytes/s. However, the MPEG bit rate is still limited to 15 Mbit/s for video and 448 kbit/s for audio.

Independent of the input mode all audio and video input data are stored sequentially in the audio or video input buffer area of the external memory. The audio and video data can be either in MPEG2 PES, MPEG1 packet or ES format.

Memory interface unit

The memory interface takes care of addressing and control of the 16-Mbit external SDRAM. The SDRAM should be either JEDEC compliant either the 'lite/PC' version.

Due to memory communication requirements this interface runs at 81 MHz. The SDRAM types used with the SAA7201 should be organized as 1M × 16, split internally in two banks, each having 2048 pages of 256 words of 16 bits.

The target SDRAM type is NEC µPD 4516161G5-A12-7FJ (83 MHz JEDEC version) or NEC µPD 4516421G5-A83-7FJ-PC (83 Mhz PC version).

Clock generation

The clock generation unit generates all the internal processing clocks, the clock for the system time base counter and the audio oversampling clock for the audio DAC. For this purpose a non-integer divider plus a PLL is implemented. In order to get reliable audio and video decoding the 27 MHz input clock should be locked externally to the MPEG time base.

Host interface system

The host interface system handles the communication between on one side the SAA7201 plus SDRAM and on the other side the external CPU. The interface consists of a 16-bit wide data bus plus 8 address lines. It is compatible with both Motorola's 68xxx and Intel's x86 family.
An optimized interface with the SAA7208 is also supported. Via this interface a fast direct access to a large number of internal status and data registers can be achieved. Moreover, the external SDRAM can be accessed via a specific register in combination with an internally implemented auto increment counter.
The access to the external SDRAM is guaranteed up to a sustained data rate of 9 Mbyte per second. However, in practice the achievable data rate can be much higher.

Next to the data and address lines, 4 interrupt lines are part of the host interface bus. Each interrupt line can monitor up to 32 internal events which all can be masked individually. Examples of internal events are audio/video bit stream information, decoder status, internal error conditions and input buffer occupation. The latter may be very useful in interactive applications to serve as input data request line.

System time base unit

The system time base unit serves as a timing master for all internal processes. It consists of two 24-bit wide System Time Clock (STC) counters, running at 90 kHz. The STCs will be used as internal synchronization reference for audio and video.The contents of the STC can be loaded by the external CPU which should insure that the phase of the SAA7201 internal STC is identical to the main system time clock in the system demultiplexer. The CPU should correct for possible latency problems.

Because two counters are implemented, the previous time base reference which might still be required as reference for some time in case of time base discontinuity, can be maintained. Thus all information for audio/video synchronization is available in the decoder chip and only minor support of the external controller is required.

The synchronization of graphics for e.g. subtitling, should be controlled by the external CPU.

Video input buffer and synchronization control

The size and position of the video input buffer in the external SDRAM is programmable. By default 2.6 Mbit/s are reserved for the video input buffer but in principle any other value can be programmed. The current fullness of the video input buffer can be monitored by the CPU and an internal interrupt will be generated is case of either near over- or near underflow.

Data retrieval from the input buffer can be controlled by DTS time stamps parsed from the PES or MPEG1 packet stream. For those frames where no DTS time stamp is present in the video bitstream a DTS is emulated by the SAA7201.

Obviously this emulation mode can also be used when the input stream is a video elementary stream (ES). The latter case should be handled by start and stop decode commands from the CPU.

The external CPU can select to retrieve the video PES header and/or video PES private data for further software processing.

Audio input buffer and synchronization control

The audio input buffer and synchronization control basically behaves identical to its video counter part. The default buffer size is 64 kbit in this case. Synchronization will be controlled by PTS time stamps in the audio Packetized Elementary Stream. Also in this case an PTS emulation or a free running start/stop controlled mode are supported.

Audio decoder

A functional block diagram for the audio decoding part is depicted in Fig.5.

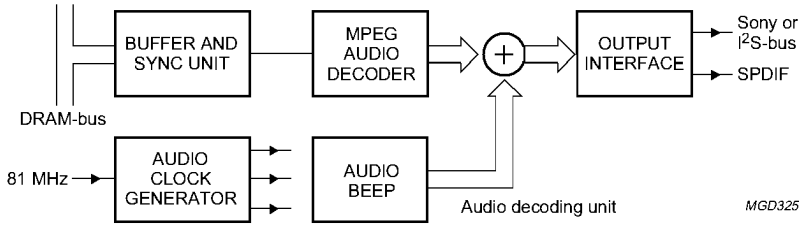


Fig.5 Audio decoding unit.

Audio decoding will be performed at a clock locked to the video decoding clock and only the output interface is running on the audio oversampling clock.

The audio decoder unit performs the decoding of the selected MPEG audio stream in a range from 8 up to 448 kbit/s in a fixed or variable bit rate format. Decoding is restricted to 2 channel, layer I, II MPEG audio at sampling frequency of 48.0, 44.1, 32.0, 24.0, 22.05 or 16.0 kHz.

The audio decoder support the stop, mute, and skip function to support insertion

Apart from basic MPEG processing the audio decoder core contain also:

- Support for: stop, mute and skip function.
- Fully parameterized dynamic range compression unit to decrease the dynamic range of the output signal on audio frame basis. Depending on the power level a programmable amplification and offset may be applied.
- Fully programmable base band audio processing unit to control the gain in both output channels independently and/or to mix both channels.
- MPEG de-emphasis filtering on the output data, thus avoiding the need of external analog de-emphasis filter circuitry.
- Storage buffer for the last 54 bytes of each audio frame. The CPU can retrieve eventual ancillary data from this buffer.

The output of the audio decoder unit can be mixed with square waveform audio signals which are generated by a beep generator. Programmable parameters for the beep generator are amplitude, frequency and duration.

The audio output interface module produces stereo base band output samples on two different outputs at the same time:

- Serial digital audio in I^2S-bus or in Japanese format in 16, 18, 20 or 22-bit
- SPDIF (Sony/Philips Digital Interface).

Any of the two outputs may be enabled or set to high impedance mode. The I^2S-bus format with 18-bit sample precision is shown in Fig.6.

The difference between I^2S-bus and the Japanese format is that I^2S-bus is MSB aligned whereas the Japanese format is LSB aligned.

The 1-bit serial interface SPDIF contains 64-bit per audio sample period. Complete frames must be transmitted at the audio sample rate. Not only left/right information but also validity flags, channel status, user data and parity information is contained in an SPDIF frame (see Chapter “References”).

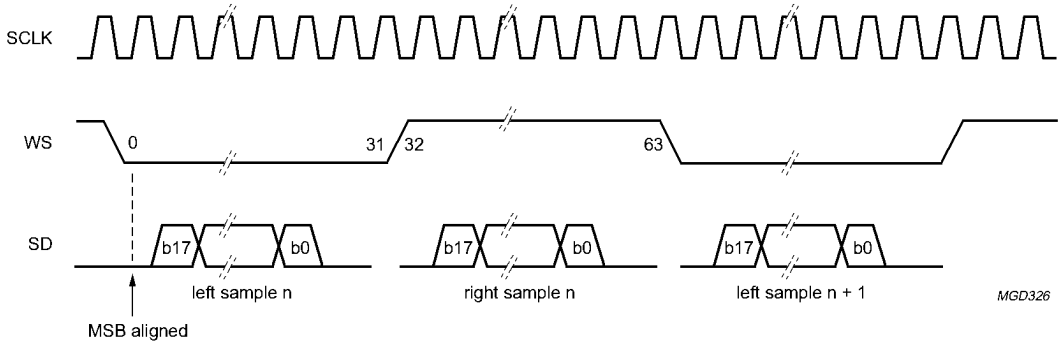


Fig.6 I^2S-bus format with 18-bit sample precision.

Video decoder

The video decoding unit provides all actions required for compliant decoding of MPEG2 main level, main profile coded video bit streams. The decoding process consists of fixed and variable length decoding, run length decoding, inverse quantization, inverse discrete cosine transformation, motion compensation and interpolation.

In general the arithmetic decoding result is stored as reference picture in the external memory.
Decoded B-frames are only stored for the conversion from the frame coded macro block (MB) to the scanning line format. In many cases a field storage is sufficient for this conversion but in some cases the user might decide to use a full frame storage to enable chroma frame up-conversion or full performance 3 : 2 pull-down in 60 Hz systems. Obviously when using less memory for the video decoding process more memory is available for non-video decoding tasks.

The Frame Buffer Management unit (FBM) manages the allocation of frame buffers in external SDRAM for both video decoding and display unit and can be programmed to use less memory in not fully MP@ML bitstreams: smaller pictures (e.g. 544 × 576), simple profile, etc.

Apart from decoding compliant MPEG video streams the decoder deals with some trick modes. Supported are field or frame freeze at I or P pictures or freeze field on B-pictures. In the latter case decoding will continue as a background process and the output can be restarted at any moment. When receiving non-compliant MPEG streams the decoder can be switched to a scanning mode in which only I or I + P frames are decoded while skipping all other pictures. In the single step mode, the decoder decodes just one frame and awaits a next step command.

The functional diagram of the video decoding unit is shown in Fig.7.

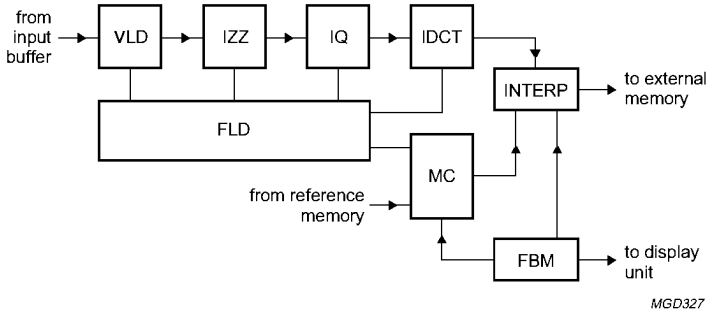


Fig.7 Video decoding unit.

Graphics unit

The SAA7201 incorporates the display support for pixel based graphics. Possible applications are the user interface, logos and subtitling. Graphical data should be grouped logically in regions and will be displayed in boxes at the screen.

The definition of each region in the decoder consists of four parts being a region descriptor, a top-field descriptor, a bottom-field descriptor and a table-data descriptor:

- The region descriptor contains information relevant for the full region like format, size, position and pointers to the other descriptors.
- The top-field and bottom-field descriptor contain a pixel based bit-map for the contents of that region for both fields independently. The bit-maps can be stored in either straight forward or in a compressed bit-map format.
- The table-data descriptor defines the tables to be used for the transformation of bit-maps to display colours.

All descriptors should be loaded under control of the external CPU in the external memory.

The appearance of graphical data at the display is determined by the assembly of region descriptors in a so called display list. An example of such a display list for the 4 regions example is shown in Fig.9.

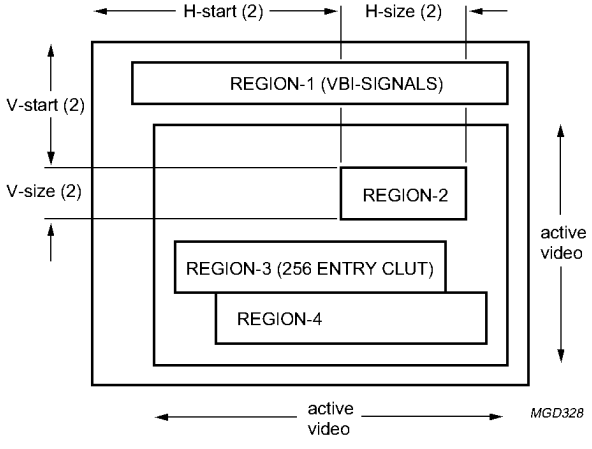


Fig.8 Graphics unit.

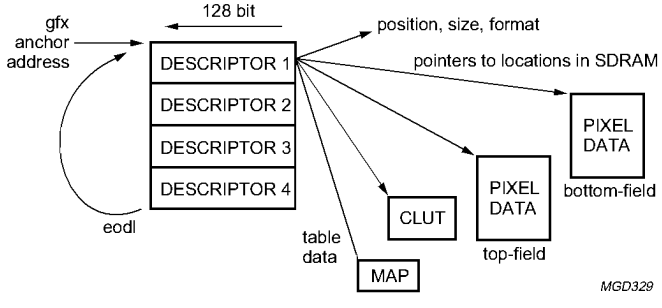


Fig.9 Display list.

Basically there is no restriction on the number of different regions but because regions may not vertically overlap the practical limit will be the number of lines within a field. However, one should realize that each region requires its own 128-bit region descriptor.

The display list will be scanned twice per frame, once for each display field. The region descriptors should be ordered properly in the external SDRAM, starting from the graphics anchor address. The last descriptor in the list must have the end of display list indicator set.

Multiple pixel bit-maps, CLUTs and map tables may be stored in the external memory but per region only two bit-maps (one for each fields) and two tables (CLUT + map table) may be used. Obviously bit-maps and tables may be shared by multiple regions.

Pixel data bit-maps can be described in 2, 4 or 8 bit/pixel in either a direct bit-map or coded in a one-dimensional (H) variable and run length encoded format according the pixel-data-sub-block syntax as specified in Chapter “References” and illustrated in Chapter “Appendix”. The actual coding format is specified in the region descriptor for each region thus allowing different coding schemes within a picture.

During display the 2, 4 and 8 bit/pixel bit-maps will be transformed, eventually with run length decoding, via a table look-up mechanism into a 4, 16 or 256 different YUV colours with 8-bit resolution for each component plus a factor T for mixing of graphics and MPEG video.

In order to obtain maximum flexibility two cascaded tables are active in this bit-map to pixel conversion as indicated in Fig.10.

The tables are retrieved from the external memory just before the region is going to be displayed. One table per region can be updated and for small tables this occurs during the horizontal blanking interval. However, updating a 256 entry CLUT may take about one line period which means that a spatial separation of one line with the previous region is mandatory in this case. If the required tables for a certain region are already stored in the local memory, the table down load action can be skipped.

Additionally some special bits can be set in the region descriptor.

- Transparency shift: this parameter overrules the pixel based transparency in order to support fading of the entire graphical region.
- Zoom: this parameter initiates horizontal pixel repetition. It should be noted that a copy of pixels in vertical direction can be achieved by pointing to a single bit-map for both fields.

Regions can also be defined in the vertical blanking interval. In combination with 8 bit/pixel coding, arbitrary test signals on 13.5 MHz grid can be programmed. Possible application areas are teletext, closed caption, wide screen signalling bits, Video Programming Signals (VPS) and Vertical Interval Test Signals (VITS).

As indicated above multiple regions can be specified in a display list which will be scanned sequentially every frame. In case of stationary graphics no updates of the display list are required, but the external CPU can update it dynamically to achieve scrolling and/or fading of one or more graphical boxes. The display list mechanism also allows for non real time transfer of large bit-maps by keeping that region out of the display list during loading.

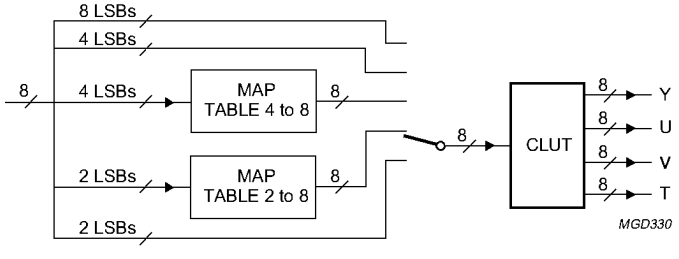


Fig.10 Bit-map to pixel conversion.

CURSOR PROCESSING

Additionally to the above defined graphics boxes one cursor can be activated on the screen. Since the cursor data is fully stored locally, no overlapping restriction apply to this box so the cursor can moved over the entire screen. The cursor can be as large as 1 kpixel with a 2 bits/pixel colour depth. Obviously data transfer can be done on DMA basis and need only be performed when a cursor is required or when its contents must be modified.

The cursor XY dimensions (where the Y dimension refers to frame lines) can be selected between 8 × 128, 16 × 64, 32 × 32, 64 × 16 and 128 × 8. On top of these shapes, a zoom with a factor 2 can be applied in both directions independently.

The cursor pixels will be translated via a 4-entry CLUT to YUV colours and a transparency factor T. The resolution of the YUV parameters is 6, 4, 4 bits respectively. The T parameter is coded in 2 bits to enable the mixing with video and graphics in 4 steps being 100% (cursor only), 50%, 25% and 0% (fully transparent cursor).

Display unit

Before feeding the MPEG decoded and graphical data to the output, a display unit re-formats the MPEG specific 4 : 2 : 0 format to CCIR-601 4 : 2 : 2 format and performs a mixing between video and graphics where required. The output picture can be up to 720 × 576 pixels at 50 Hz or 720 × 480 pixels at 60 Hz.

A schematic representation of this unit is shown in Fig.11.

- In a first step a selected window can be retrieved from the decoded MPEG data. This might be useful for e.g. pan and scan operations for aspect ratio conversion.
- In case the resulting number of pixels per line does not match the 720 pixels/line output format a horizontal scaler can be activated. This scaling unit can transform any number of bits below 720 to the required output format. Internally a poly-phase filter is used which performs a 64 phases interpolation. Not only up-conversion but also down-conversion is supported up to a scaling by a factor 2. Thus horizontal scaling can be performed in a range from 0.5 up to 64. In practice the maximum up-conversion factor will be less or equal to 4.
- In vertical direction the picture can be expanded or scaled down, in both cases by a factor 2. Expansion with a factor 2 might be relevant for the up-conversion of SIF resolution pictures to full screen. The factor 2 scaling, if combined with the appropriate horizontal scaling, results is ¼ picture thus freeing-up a large screen area for graphics. This might be very useful for electronic program guide applications. It should be noted that in case of picture compression an anti-aliasing filter can be activated.
- Shifting: when the resulting MPEG picture is smaller than the 720 × 576 (480) display format, this picture can be located anywhere on the display screen. Moreover, the non-covered area can be given any background colour.
- Clipping: the amplitude of the MPEG decoded and re-scaled video signal is kept within the range 16 to 235 for luminance and 16 to 240 for both chrominance components.

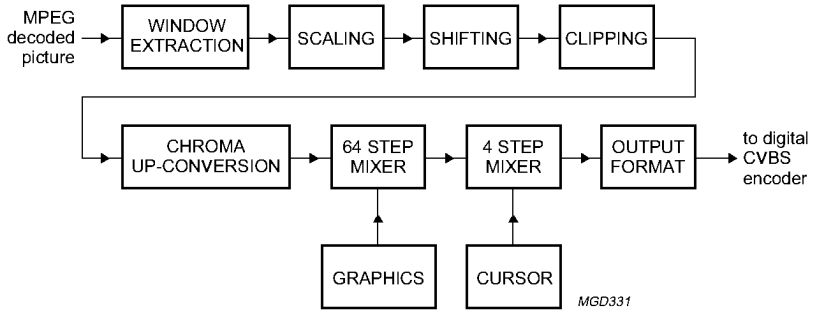


Fig.11 Display unit.

- The chroma up-conversion unit converts the MPEG 4 : 2 : 0 format into the at the output required 4 : 2 : 2 format. This vertical up-conversion is performed by a simple 8-phase interpolation between two adjacent lines.
- The mixer units combine MPEG video with graphics and cursor information in two steps. In a first step the MPEG decoded information is mixed with graphical information. Mixing can be done at pixel basis in 64 steps and is controlled by the internally implemented colour look up table. In a second step, the video plus graphics can be mixed in 4 steps with the internally generated cursor.
- The output formatting unit performs two main tasks, i.e. synchronization and formatting. Synchronization is characterized by three signals being horizontal (H), vertical (V) and field parity (FP), all having programmable length and polarity.

Since the decoder can operate in master or slave mode, the synchronization signals can be generated by the decoder or should be delivered by an external device. In both cases the length and polarity should be programmed internally.

The video output samples are supplied in a multiplexed YUV format to the output. Next to this byte wide YUV stream, which can directly be supplied to most commercially available composite video encoder ICs, three additional signals are delivered at the output. HREF indicates all active samples; CREF can flag any combination of pixels: U, V, Y_{odd} and/or Y_{even}; GRPH flags all the pixels inside a graphical box.

Additionally the full YUV bus can be set to a HIGH impedance state under control of the signal YUV_ENA. This might be useful for multiplexing the MPEG decoder output with any other signal source on static basis.

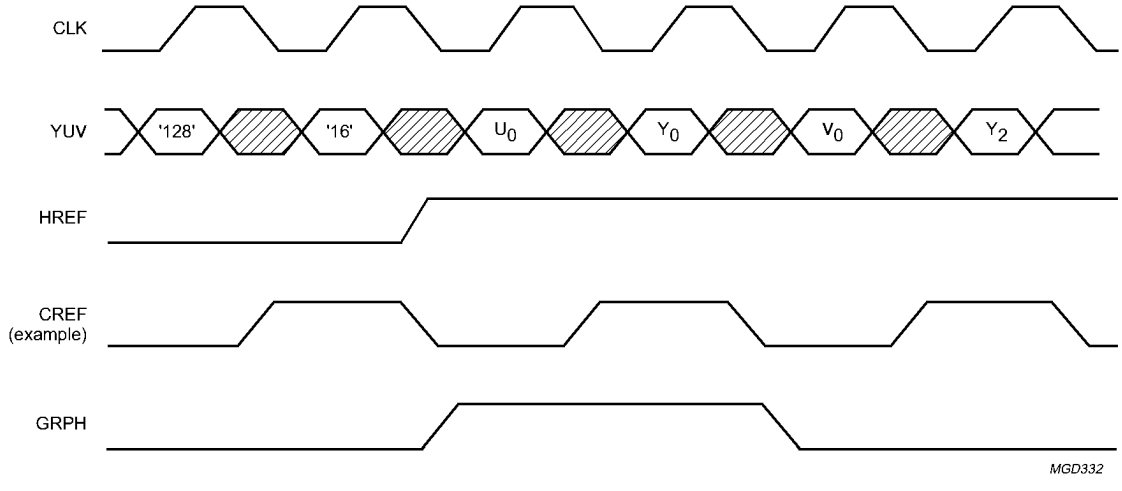


Fig.12 Timing diagram of graphics information output.

JTAG

The SAA7201 supports the standard Boundary Scan test instructions: bypass, extest, sample, intest, runbist, idcode.

Memory requirements

As indicated before the MPEG source decoder operates with 16 Mbit of external memory. Several processes require access to the external memory, mostly being the video decoding process. In normal main level, main profile video applications about 1.2 Mbit of memory space is free for non-video processes. In practice most of this capacity will be used for graphics. In combination with the internal variable length decoding, full screen graphics at 8-bit per pixel is feasible. Moreover, by having introduced a flexible memory allocation procedure the available memory capacity for graphics may be enlarged when decoding lower resolution MPEG pictures or when the input bit rate is less than 15 Mbit/s.

Obviously for graphics-only applications all 16 Mbit can be used for the storage of bit-maps and look-up tables.

In Table 1 an overview is given of the required memory capacity for some user defined modes.

In 50 Hz systems memory capacity can be saved by restricting the chroma vertical interpolation to field interpolation. This mode would only bring some extra chroma resolution in case the input stream contains progressive coded pictures.

In 60 Hz systems the reduction of storage capacity for B-frames to field capacity has not only consequences for the chroma vertical interpolation but also for the 3 : 2 pull-down operation mode. The operation repeat-first-field is not possible in all cases and a modified 3 : 2 pull-down is performed under the control of the SAA7201. The user may decide to use this modified 3 : 2 pull-down mode in order to have more memory available for OSD or graphics.

Table 1 Required memory capacity for some user defined modes

System	**50 Hz**			**60 Hz**		
Bit rate (R)	**15 Mbit/s**			**15 Mbit/s**		**9 Mbit/s**
Chroma interpolation	**frame**	**field**	**field**	**frame**	**field**	**field**
3 : 2 pull-down	**n.a.**	**n.a.**	**n.a.**	**full MPEG**	**modified**	**modified**
Picture format	**720 × 576**	**720 × 576**	**544 × 576**	**720 × 480**	**720 × 480**	**720 × 480**
Audio input buffer	64 kbit	64 kbit	64 kbit	64 kbit	64 kbit	64 kbit
Video input buffer	1835 kbit	1835 kbit	1835 kbit	1835 kbit	1835 kbit	1835 kbit
Video implementation buffer (R/P)	600 kbit	600 kbit	400 kbit	500 kbit	500 kbit	300 kbit
Slave synchronization buffer (R/2P)	300 kbit	300 kbit	200 kbit	250 kbit	250 kbit	150 kbit
Reference and decoded picture	13456 kbit	12719 kbit	9609 kbit	12441 kbit	10634 kbit	8042 kbit
Total for video + audio	16255 kbit	15518 kbit	12108 kbit	15090 kbit	13292 kbit	10391 kbit
Remains for OSDG (2^{24} = 16777 kbit)	522 kbit	1259 kbit	4669 kbit	1687 kbit	3485 kbit	6386 kbit

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 60134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{DD}	supply voltage (on all supply pins)		3.0	3.6	V
V_{max}	maximum voltage on all pins		0	5.5	V
P_{tot}	total power dissipation	T_{amb} = 25 °C	–	tbf	W
T_{stg}	storage temperature		–55	+150	°C
T_{amb}	operating ambient temperature		0	+70	°C

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	VALUE	UNIT
$R_{th\ j\text{-}a}$	thermal resistance from junction to ambient in free air	30	K/W

CHARACTERISTICS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
Supply					
V_{DD}	supply voltage	3.0	3.3	3.6	V
I_{DD}	supply current	–	tbf	–	mA
Inputs					
V_{IH}	HIGH level input voltage	2.0	–	V_{DD} + 2.0	V
V_{IL}	LOW level input voltage	–0.5	–	0.8	V
I_{LI}	leakage current	–	–	20	mA
C_i	input capacitance	0	–	10	pF
Outputs					
V_{OH}	HIGH level output voltage	2.4	–	–	V
V_{OL}	LOW level output voltage	–	–	0.4	V
CLK timing					
T_C	cycle time	37.036	37.037	37.038	ns
δ	duty factor	40	–	60	%
t_r	rise time	2	–	4	ns
t_f	fall time	2	–	4	ns
Input timing with respect to CLK rising edge					
t_{su}	set-up time	8	–	–	ns
t_h	hold time	0	–	–	ns
Timing (see Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 21 **and** 21**)**					
$t_{su(A-CS)}$	address/CS set-up time	20	–	–	ns
$t_{h(A-CS)}$	address/CS hold time	75	–	–	ns
$t_{su(D-W)}$	data write set-up time	20	–	–	ns
$t_{su(D-R)}$	data read set-up time	20	–	–	ns
$t_{rel(D)}$	data release time	0	–	10	ns
$t_{h(CT)}$	control signal hold time	0	–	–	ns
$t_{W(ACK)}$	acknowledge pulse width	25	–	–	ns
$t_{rel(ACK)}$	acknowledge release time	0	–	10	ns
$t_{d(ACK-R)}$	delay time for acknowledge read	96	–	125	ns
$t_{d(ACK-W)}$	delay time for acknowledge write	48	–	75	ns
$t_{W(RW)}$	write/read pulse width	25	–	–	ns
Output timing with respect to CLK rising edge					
t_h	hold time	3	–	t_d	ns
t_d	delay time	t_h	–	20	ns
C_L	load capacitance	10	–	30	pF

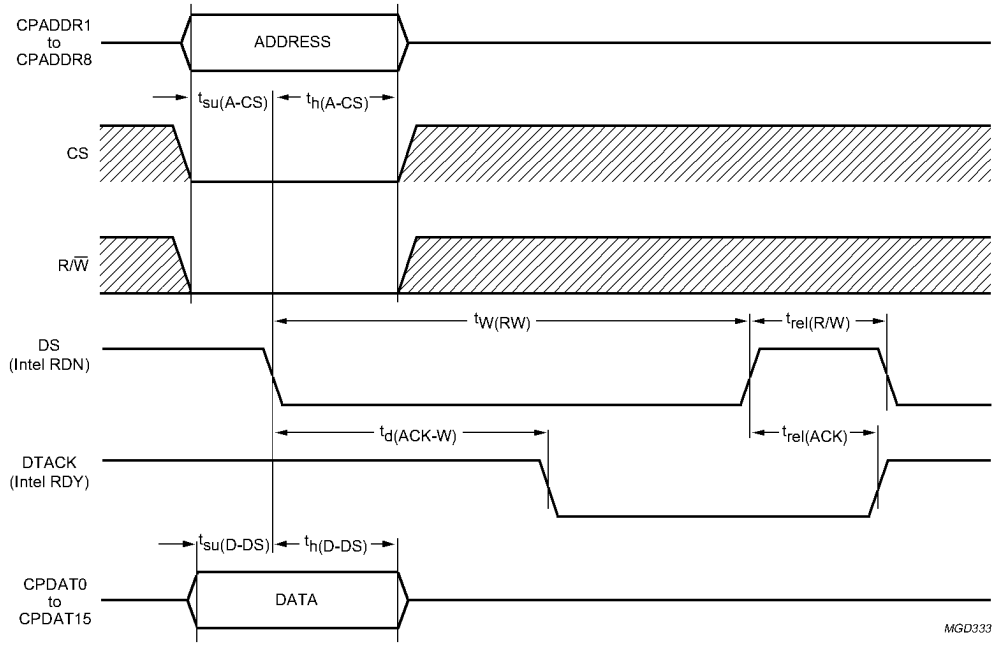


Fig.13 Motorola write timing (non-multiplexed).

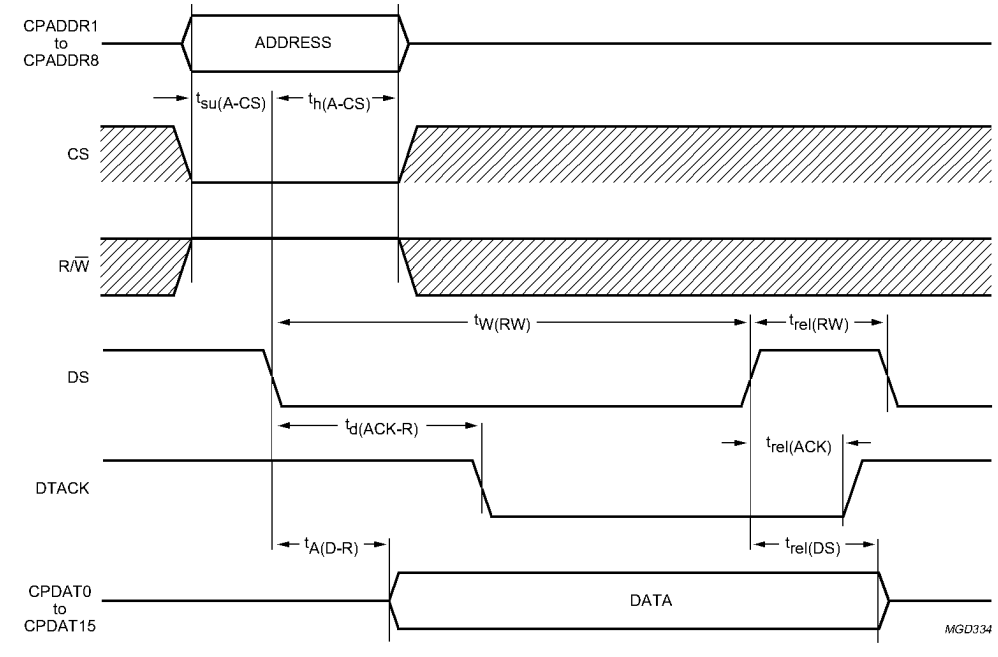


Fig.14 Motorola read timing (non-multiplexed).

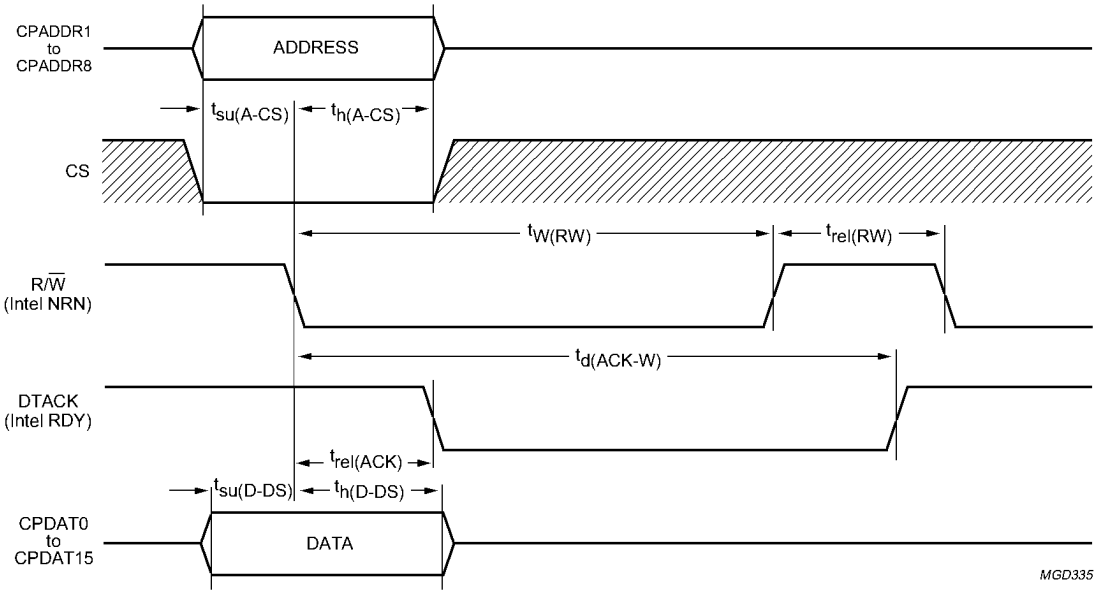


Fig.15 Intel write timing (non-multiplexed).

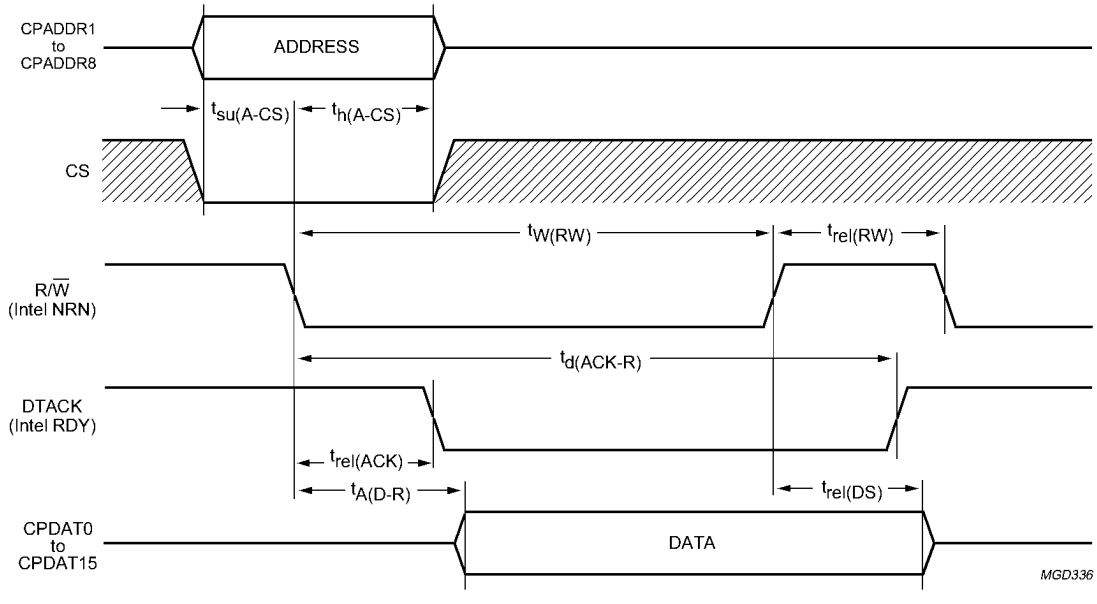


Fig.16 Intel read timing (non-multiplexed).

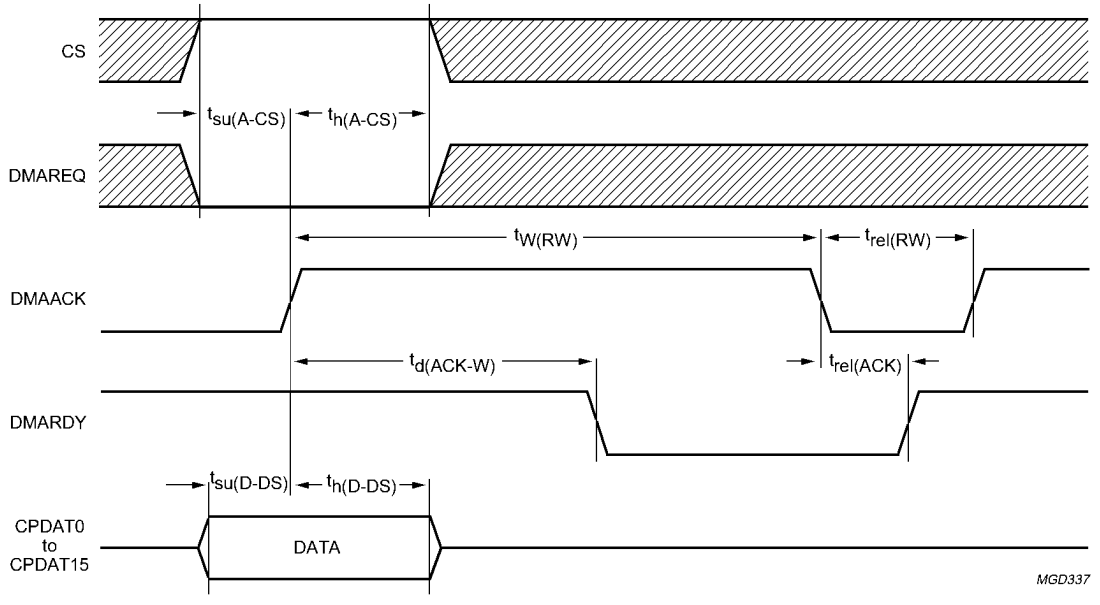


Fig.17 DMA read access.

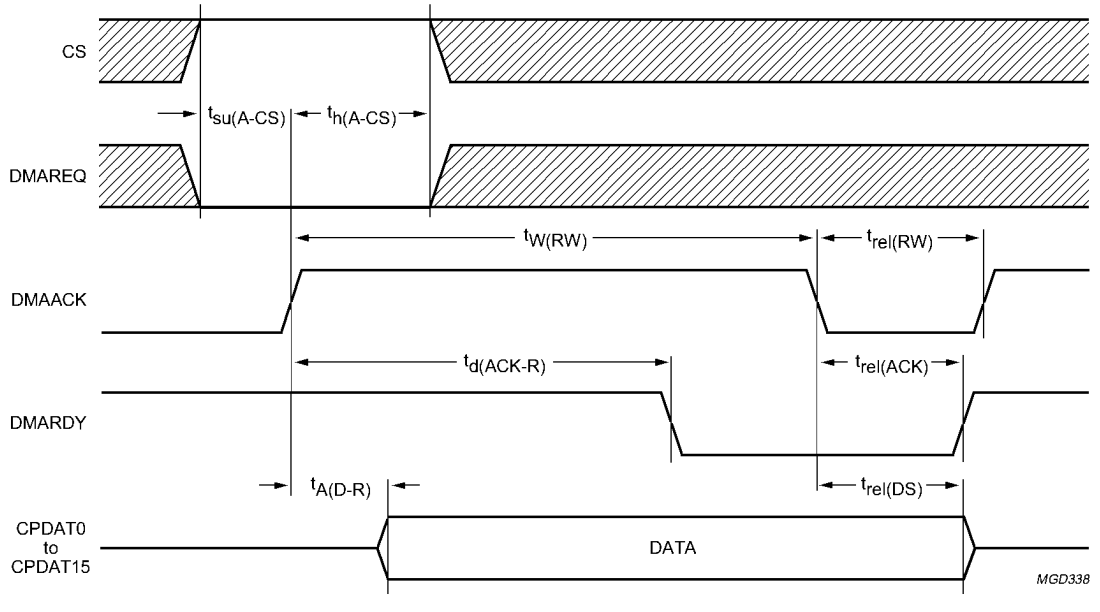


Fig.18 DMA write access.

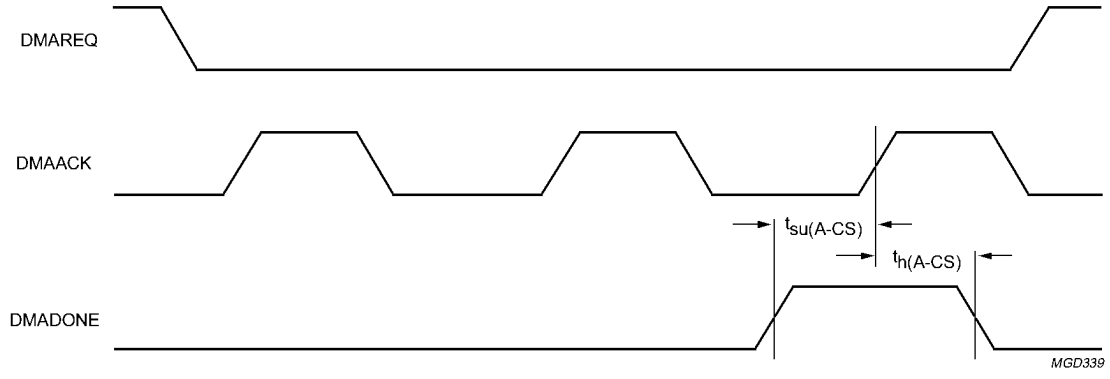


Fig.19 DMA_DONE timing.

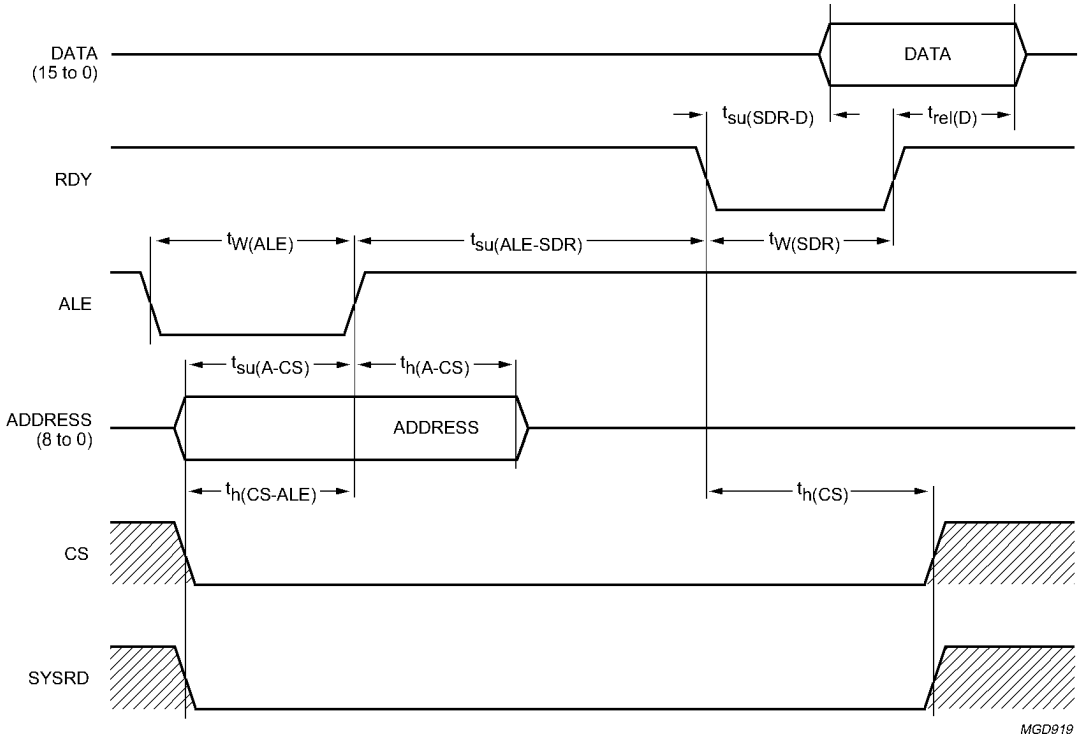


Fig.20 SAA7208 mode; read timing.

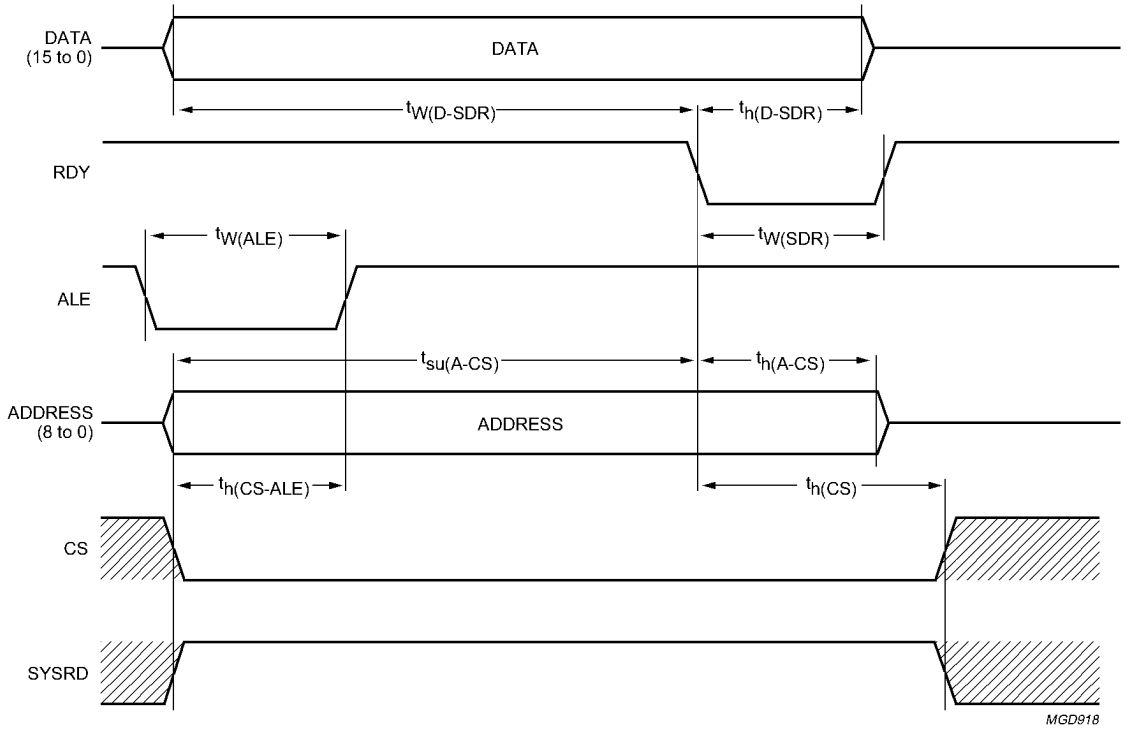


Fig.21 SAA7208 mode; write timing.

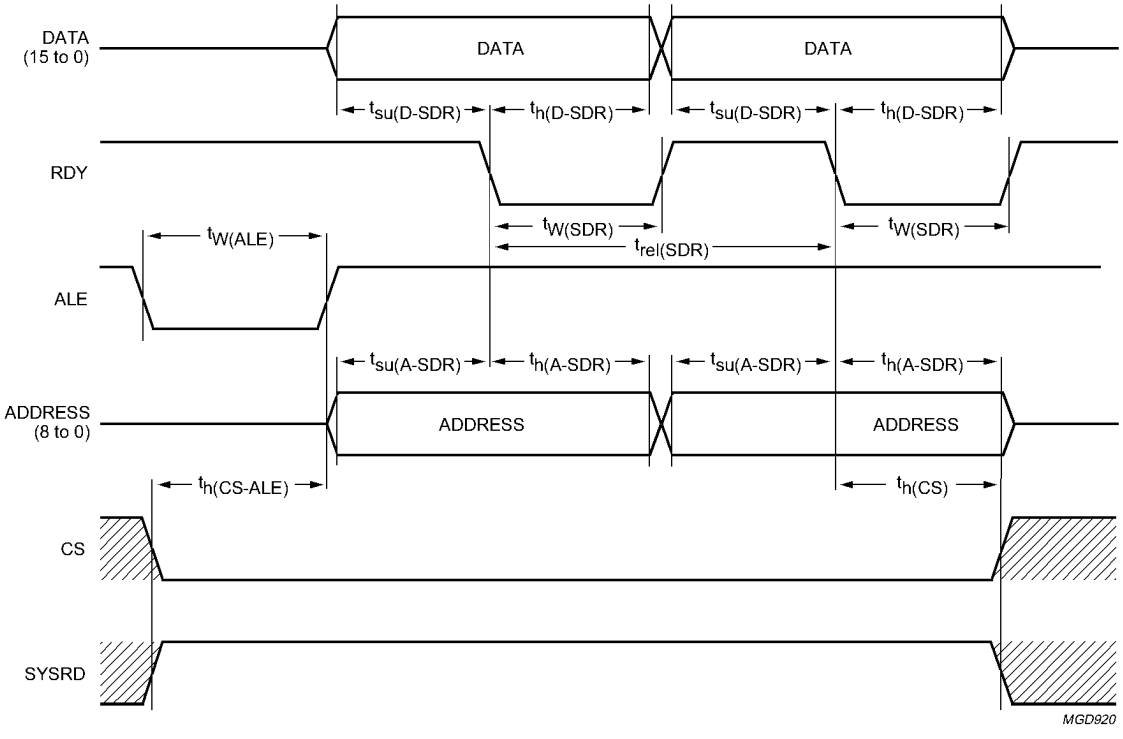


Fig.22 SAA7208 mode; burst write timing.

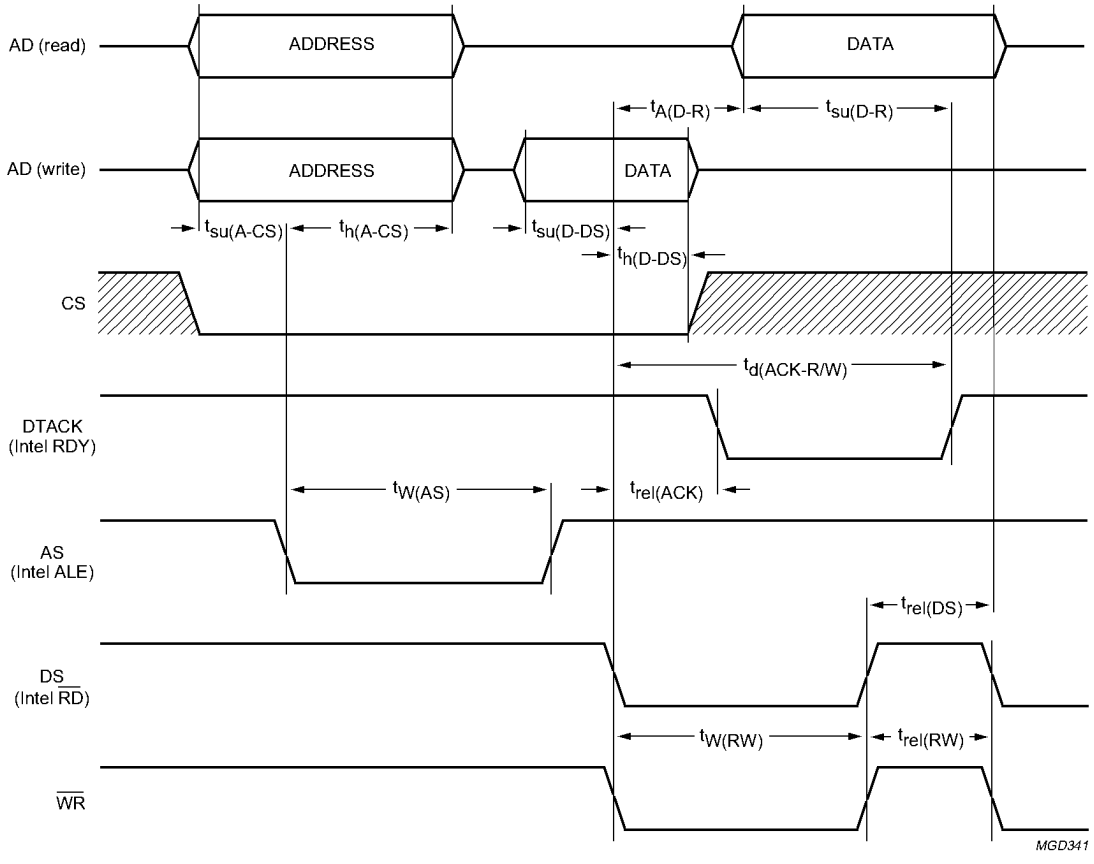


Fig.23 Intel read/write timing multiplexed.

REFERENCES

1. *MPEG ISO/IEC 11172-1 International standard; MPEG-1 systems.*
2. *MPEG ISO/IEC 13818-1 International standard; MPEG-2 systems.*
3. *MPEG ISO/IEC 11172-2 International standard; MPEG-1 Video.*
4. *MPEG ISO/IEC 13818-2 International standard; MPEG-2 Video.*
5. *MPEG ISO/IEC 11172-3 International standard; MPEG-1 Audio.*
6. *MPEG ISO/IEC 13818-3 International standard; MPEG-2 Audio.*
7. *DVB subtitling system; working draft 2.0; TM 1398 rev 2.*

APPLICATION INFORMATION

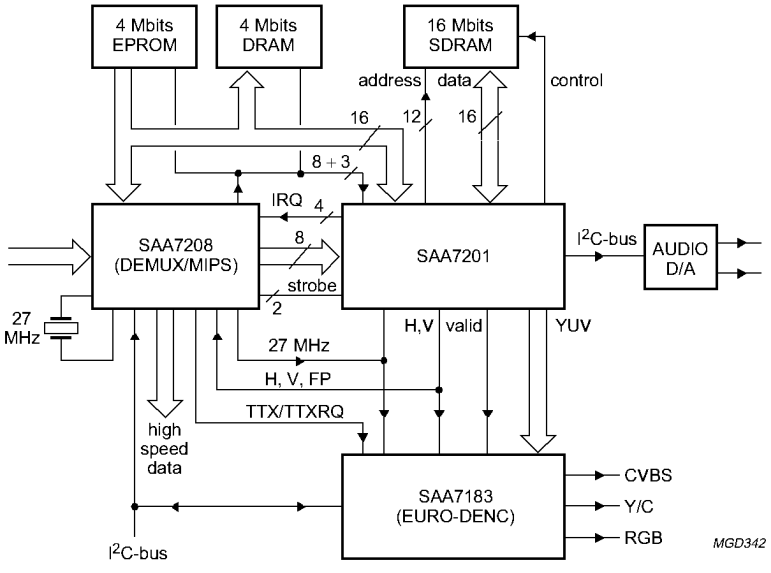


Fig.24 Application diagram.

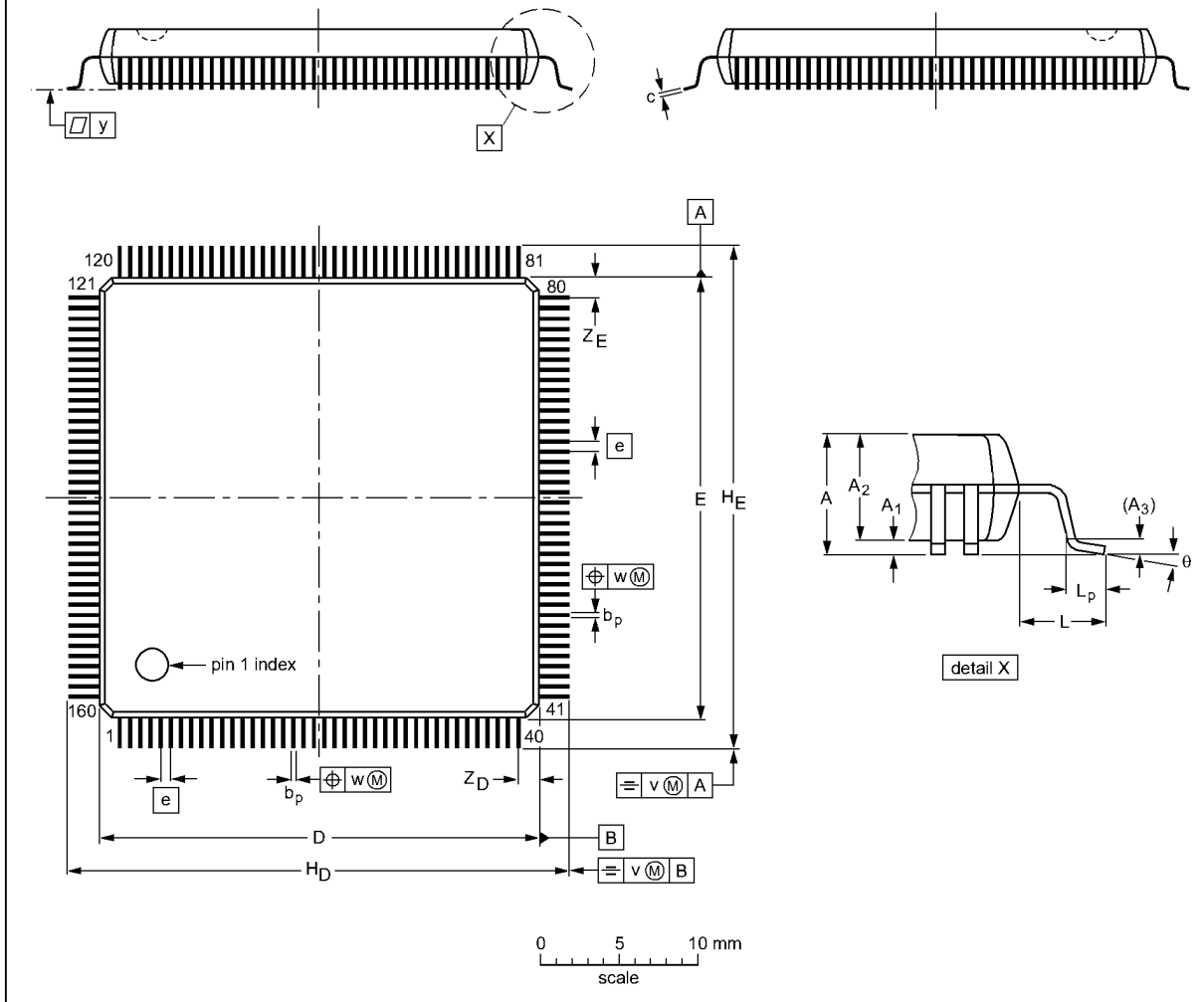
APPENDIX

Syntax of pixel-data-sub-block

data type	pixel-code-string of N-coded words		end of string
2 bit/pixel	01	1 pixel in colour 1	
	10	1 pixel in colour 2	
0001 0000	11	1 pixel in colour 3	00 00 00
	00 01	1 pixel in colour 0	
	00 00 01	2 pixels in colour 0	
	00 1L LL CC	L pixels (3 to 10) in colour C	
	00 00 10 LL LL CC	L pixels (12 to 27) in colour C	
	00 00 11 LL LL LL LL CC	L pixels (29 to 284) in colour C	
4 bit/pixel	0001	1 pixel in colour 1	
↓	↓	↓	↓
0001 0001	1111	1 pixel in colour 15	0000 0000
	0000 1100	1 pixel in colour 0	
	0000 1101	2 pixels in colour 0	
	0000 0LLL (L>0)	L pixels (3 to 9) in colour 0	
	0000 10LL CCCC	L pixels (4 to 7) in colour C	
	0000 1110 LLLL CCCC	L pixels (9 to 24) in colour C	
	0000 1111 LLLL LL11 CCCC	L pixels (25 to 280) in colour C	
8 bit/pixel	00000001	1 pixel in colour 1	00000000-- --00000000
↓	↓	↓	↓
0001 0010	11111111	1 pixel in colour 255	
	00000000 0LLLLLLL	L pixels (1 to 127) in colour 0	
	00000000 1LLLLLLL CCCCCCCC	L pixels (3 to 127) in colour C	

PACKAGE OUTLINE

QFP160: plastic quad flat package;
160 leads (lead length 1.95 mm); body 28 x 28 x 3.4 mm; high stand-off height **SOT322-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_D	H_E	L	L_p	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	3.95	0.40 0.25	3.70 3.15	0.25	0.40 0.25	0.23 0.13	28.1 27.9	28.1 27.9	0.65	32.2 31.6	32.2 31.6	1.95	1.1 0.7	0.3	0.15	0.1	1.5 1.1	1.5 1.1	8^o 0^o

Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT322-1		MO-112				~~97-08-04~~ 99-12-27

SOLDERING

Introduction to soldering surface mount packages

This text gives a very brief insight to a complex technology. A more in-depth account of soldering ICs can be found in our *"Data Handbook IC26; Integrated Circuit Packages"* (document order number 9398 652 90011).

There is no soldering method that is ideal for all surface mount IC packages. Wave soldering can still be used for certain surface mount ICs, but it is not suitable for fine pitch SMDs. In these situations reflow soldering is recommended.

Reflow soldering

Reflow soldering requires solder paste (a suspension of fine solder particles, flux and binding agent) to be applied to the printed-circuit board by screen printing, stencilling or pressure-syringe dispensing before package placement.

Several methods exist for reflowing; for example, convection or convection/infrared heating in a conveyor type oven. Throughput times (preheating, soldering and cooling) vary between 100 and 200 seconds depending on heating method.

Typical reflow peak temperatures range from 215 to 250 °C. The top-surface temperature of the packages should preferable be kept below 220 °C for thick/large packages, and below 235 °C for small/thin packages.

Wave soldering

Conventional single wave soldering is not recommended for surface mount devices (SMDs) or printed-circuit boards with a high component density, as solder bridging and non-wetting can present major problems.

To overcome these problems the double-wave soldering method was specifically developed.

If wave soldering is used the following conditions must be observed for optimal results:

- Use a double-wave soldering method comprising a turbulent wave with high upward pressure followed by a smooth laminar wave.
- For packages with leads on two sides and a pitch (e):
  - larger than or equal to 1.27 mm, the footprint longitudinal axis is **preferred** to be parallel to the transport direction of the printed-circuit board;
  - smaller than 1.27 mm, the footprint longitudinal axis **must** be parallel to the transport direction of the printed-circuit board.

  The footprint must incorporate solder thieves at the downstream end.
- For packages with leads on four sides, the footprint must be placed at a 45° angle to the transport direction of the printed-circuit board. The footprint must incorporate solder thieves downstream and at the side corners.

During placement and before soldering, the package must be fixed with a droplet of adhesive. The adhesive can be applied by screen printing, pin transfer or syringe dispensing. The package can be soldered after the adhesive is cured.

Typical dwell time is 4 seconds at 250 °C.
A mildly-activated flux will eliminate the need for removal of corrosive residues in most applications.

Manual soldering

Fix the component by first soldering two diagonally-opposite end leads. Use a low voltage (24 V or less) soldering iron applied to the flat part of the lead. Contact time must be limited to 10 seconds at up to 300 °C.

When using a dedicated tool, all other leads can be soldered in one operation within 2 to 5 seconds between 270 and 320 °C.

Suitability of surface mount IC packages for wave and reflow soldering methods

PACKAGE	SOLDERING METHOD	
	WAVE	REFLOW[1]
BGA, HBGA, LFBGA, SQFP, TFBGA	not suitable	suitable
HBCC, HLQFP, HSQFP, HSOP, HTQFP, HTSSOP, HVQFN, SMS	not suitable[2]	suitable
PLCC[3], SO, SOJ	suitable	suitable
LQFP, QFP, TQFP	not recommended[3][4]	suitable
SSOP, TSSOP, VSO	not recommended[5]	suitable

Notes

1. All surface mount (SMD) packages are moisture sensitive. Depending upon the moisture content, the maximum temperature (with respect to time) and body size of the package, there is a risk that internal or external package cracks may occur due to vaporization of the moisture in them (the so called popcorn effect). For details, refer to the Drypack information in the *"Data Handbook IC26; Integrated Circuit Packages; Section: Packing Methods"*.
2. These packages are not suitable for wave soldering as a solder joint between the printed-circuit board and heatsink (at bottom version) can not be achieved, and as solder may stick to the heatsink (on top version).
3. If wave soldering is considered, then the package must be placed at a 45° angle to the solder wave direction. The package footprint must incorporate solder thieves downstream and at the side corners.
4. Wave soldering is only suitable for LQFP, TQFP and QFP packages with a pitch (e) equal to or larger than 0.8 mm; it is definitely not suitable for packages with a pitch (e) equal to or smaller than 0.65 mm.
5. Wave soldering is only suitable for SSOP and TSSOP packages with a pitch (e) equal to or larger than 0.65 mm; it is definitely not suitable for packages with a pitch (e) equal to or smaller than 0.5 mm.

DATA SHEET STATUS

DATA SHEET STATUS[1]	PRODUCT STATUS[2]	DEFINITIONS
Objective data	Development	This data sheet contains data from the objective specification for product development. Philips Semiconductors reserves the right to change the specification in any manner without notice.
Preliminary data	Qualification	This data sheet contains data from the preliminary specification. Supplementary data will be published at a later date. Philips Semiconductors reserves the right to change the specification without notice, in order to improve the design and supply the best possible product.
Product data	Production	This data sheet contains data from the product specification. Philips Semiconductors reserves the right to make changes at any time in order to improve the design, manufacturing and supply. Changes will be communicated according to the Customer Product/Process Change Notification (CPCN) procedure SNW-SQ-650A.

Notes

1. Please consult the most recently issued data sheet before initiating or completing a design.
2. The product status of the device(s) described in this data sheet may have changed since this data sheet was published. The latest information is available on the Internet at URL http://www.semiconductors.philips.com.

DEFINITIONS

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 60134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

DISCLAIMERS

Life support applications — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

ICs with MPEG-2 functionality — Use of this product in any manner that complies with the MPEG-2 Standard is expressly prohibited without a license under applicable patents in the MPEG-2 patent portfolio, which license is available from MPEG LA, L.L.C., 250 Steele Street, Suite 300, Denver, Colorado 80206.

Philips Semiconductors – a worldwide company

Argentina: see South America

Australia: 3 Figtree Drive, HOMEBUSH, NSW 2140,
Tel. +61 2 9704 8141, Fax. +61 2 9704 8139

Austria: Computerstr. 6, A-1101 WIEN, P.O. Box 213,
Tel. +43 1 60 101 1248, Fax. +43 1 60 101 1210

Belarus: Hotel Minsk Business Center, Bld. 3, r. 1211, Volodarski Str. 6,
220050 MINSK, Tel. +375 172 20 0733, Fax. +375 172 20 0773

Belgium: see The Netherlands

Brazil: see South America

Bulgaria: Philips Bulgaria Ltd., Energoproject, 15th floor,
51 James Bourchier Blvd., 1407 SOFIA,
Tel. +359 2 68 9211, Fax. +359 2 68 9102

Canada: PHILIPS SEMICONDUCTORS/COMPONENTS,
Tel. +1 800 234 7381, Fax. +1 800 943 0087

China/Hong Kong: 501 Hong Kong Industrial Technology Centre,
72 Tat Chee Avenue, Kowloon Tong, HONG KONG,
Tel. +852 2319 7888, Fax. +852 2319 7700

Colombia: see South America

Czech Republic: see Austria

Denmark: Sydhavnsgade 23, 1780 COPENHAGEN V,
Tel. +45 33 29 3333, Fax. +45 33 29 3905

Finland: Sinikalliontie 3, FIN-02630 ESPOO,
Tel. +358 9 615 800, Fax. +358 9 6158 0920

France: 7 - 9 Rue du Mont Valérien, BP317, 92156 SURESNES Cedex,
Tel. +33 1 4728 6600, Fax. +33 1 4728 6638

Germany: Hammerbrookstraße 69, D-20097 HAMBURG,
Tel. +49 40 2353 60, Fax. +49 40 2353 6300

Hungary: Philips Hungary Ltd., H-1119 Budapest, Fehervari ut 84/A,
Tel: +36 1 382 1700, Fax: +36 1 382 1800

India: Philips INDIA Ltd, Band Box Building, 2nd floor,
254-D, Dr. Annie Besant Road, Worli, MUMBAI 400 025,
Tel. +91 22 493 8541, Fax. +91 22 493 0966

Indonesia: PT Philips Development Corporation, Semiconductors Division,
Gedung Philips, Jl. Buncit Raya Kav.99-100, JAKARTA 12510,
Tel. +62 21 794 0040 ext. 2501, Fax. +62 21 794 0080

Ireland: Newstead, Clonskeagh, DUBLIN 14,
Tel. +353 1 7640 000, Fax. +353 1 7640 200

Israel: RAPAC Electronics, 7 Kehilat Saloniki St, PO Box 18053,
TEL AVIV 61180, Tel. +972 3 645 0444, Fax. +972 3 649 1007

Italy: PHILIPS SEMICONDUCTORS, Via Casati, 23 - 20052 MONZA (MI),
Tel. +39 039 203 6838, Fax +39 039 203 6800

Japan: Philips Bldg 13-37, Kohnan 2-chome, Minato-ku,
TOKYO 108-8507, Tel. +81 3 3740 5130, Fax. +81 3 3740 5057

Korea: Philips House, 260-199 Itaewon-dong, Yongsan-ku, SEOUL,
Tel. +82 2 709 1412, Fax. +82 2 709 1415

Malaysia: No. 76 Jalan Universiti, 46200 PETALING JAYA, SELANGOR,
Tel. +60 3 750 5214, Fax. +60 3 757 4880

Mexico: 5900 Gateway East, Suite 200, EL PASO, TEXAS 79905,
Tel. +9-5 800 234 7381, Fax +9-5 800 943 0087

Middle East: see Italy

Netherlands: Postbus 90050, 5600 PB EINDHOVEN, Bldg. VB,
Tel. +31 40 27 82785, Fax. +31 40 27 88399

New Zealand: 2 Wagener Place, C.P.O. Box 1041, AUCKLAND,
Tel. +64 9 849 4160, Fax. +64 9 849 7811

Norway: Box 1, Manglerud 0612, OSLO,
Tel. +47 22 74 8000, Fax. +47 22 74 8341

Pakistan: see Singapore

Philippines: Philips Semiconductors Philippines Inc.,
106 Valero St. Salcedo Village, P.O. Box 2108 MCC, MAKATI,
Metro MANILA, Tel. +63 2 816 6380, Fax. +63 2 817 3474

Poland: Al.Jerozolimskie 195 B, 02-222 WARSAW,
Tel. +48 22 5710 000, Fax. +48 22 5710 001

Portugal: see Spain

Romania: see Italy

Russia: Philips Russia, Ul. Usatcheva 35A, 119048 MOSCOW,
Tel. +7 095 755 6918, Fax. +7 095 755 6919

Singapore: Lorong 1, Toa Payoh, SINGAPORE 319762,
Tel. +65 350 2538, Fax. +65 251 6500

Slovakia: see Austria

Slovenia: see Italy

South Africa: S.A. PHILIPS Pty Ltd., 195-215 Main Road Martindale,
2092 JOHANNESBURG, P.O. Box 58088 Newville 2114,
Tel. +27 11 471 5401, Fax. +27 11 471 5398

South America: Al. Vicente Pinzon, 173, 6th floor,
04547-130 SÃO PAULO, SP, Brazil,
Tel. +55 11 821 2333, Fax. +55 11 821 2382

Spain: Balmes 22, 08007 BARCELONA,
Tel. +34 93 301 6312, Fax. +34 93 301 4107

Sweden: Kottbygatan 7, Akalla, S-16485 STOCKHOLM,
Tel. +46 8 5985 2000, Fax. +46 8 5985 2745

Switzerland: Allmendstrasse 140, CH-8027 ZÜRICH,
Tel. +41 1 488 2741 Fax. +41 1 488 3263

Taiwan: Philips Semiconductors, 5F, No. 96, Chien Kuo N. Rd., Sec. 1,
TAIPEI, Taiwan Tel. +886 2 2134 2451, Fax. +886 2 2134 2874

Thailand: PHILIPS ELECTRONICS (THAILAND) Ltd.,
60/14 MOO 11, Bangna Trad Road KM. 3, Bagna, BANGKOK 10260,
Tel. +66 2 361 7910, Fax. +66 2 398 3447

Turkey: Yukari Dudullu, Org. San. Blg., 2.Cad. Nr. 28 81260 Umraniye,
ISTANBUL, Tel. +90 216 522 1500, Fax. +90 216 522 1813

Ukraine: PHILIPS UKRAINE, 4 Patrice Lumumba str., Building B, Floor 7,
252042 KIEV, Tel. +380 44 264 2776, Fax. +380 44 268 0461

United Kingdom: Philips Semiconductors Ltd., 276 Bath Road, Hayes,
MIDDLESEX UB3 5BX, Tel. +44 208 730 5000, Fax. +44 208 754 8421

United States: 811 East Arques Avenue, SUNNYVALE, CA 94088-3409,
Tel. +1 800 234 7381, Fax. +1 800 943 0087

Uruguay: see South America

Vietnam: see Singapore

Yugoslavia: PHILIPS, Trg N. Pasica 5/v, 11000 BEOGRAD,
Tel. +381 11 3341 299, Fax.+381 11 3342 553

For all other countries apply to: Philips Semiconductors,
Marketing Communications, Building BE-p, P.O. Box 218, 5600 MD EINDHOVEN,
The Netherlands, Fax. +31 40 27 24825

Internet: http://www.semiconductors.philips.com

Printed in The Netherlands 753504/03/pp36 Date of release: 2001 Mar 28 Document order number: 9397 750 08176

Let's make things better.

PHILIPS

EXHIBIT G

SGS-THOMSON MICROELECTRONICS

STi3520

MPEG AUDIO / MPEG-2 VIDEO INTEGRATED DECODER

BRIEF DATA

- SINGLE CHIP COMBINING THE DECODING FUNCTIONS OF THE STi3500A VIDEO DECODER AND THE STi4500 AUDIO DECODER
- ON-CHIP PLL ALLOWING FULL CHIP OPERATION WITH TWO EXTERNAL CLOCKS
- VIDEO DECODER FULLY SUPPORTS MPEG-2 MAIN PROFILE/MAIN LEVEL (MP@ML)
- AUTOMATIC VIDEO ERROR CONCEALMENT
- ENHANCED ON-SCREEN DISPLAY GENERATOR : 16 COLORS/REGION, LINKED LIST MEMORY MANAGEMENT
- AUDIO DECODER SUPPORTS LAYERS I & II OF MPEG
- ALL POPULAR PCM AUDIO OUTPUT FORMATS SUPPORTED
- STANDARD 8-BIT INTERFACE FOR MICROCONTROLLER AND COMPRESSED DATA INPUT
- SUPPORT FOR SYNCHRONOUS DRAM
- 3.3V POWER SUPPLY, I/Os 5V TTL COMPATIBLE
- 0.5µm CMOS TECHNOLOGY

APPLICATIONS

- DBS RECEIVER
- DIGITAL TV RECEIVER
- DIGITAL CABLE TV RECEIVER

DESCRIPTION

The video decoder is a real-time video decompression processor supporting the MPEG-1 and MPEG-2 standards at video rates up to 720 x 480 x 60Hz or 720 x 576 x 50Hz. Picture format conversion for display is performed by a vertical and a horizontal filter (sample rate converter). External DRAM, typically of size 16 Mbits is required.

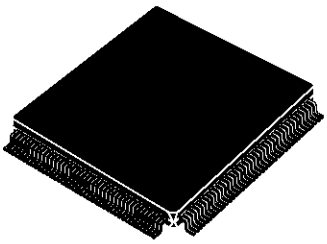
The audio decoder is compliant with layers I and II of the MPEG standard. Sampling rates of 32, 44.1 and 48kHz can be used.

This specification refers to versions 3.1 or later (marking CB or higher).

The STi3520 requires minimal support from an external microcontroller, which is mainly required to initialise the video decoder at the start of every picture. Separate audio and video data streams are input through the 8-bit data port. Time stamps are detected and made available to the microcontroller for the management of audio/video synchronization.

User-defined bitmaps may be superimposed on the displayed picture through use of the on-screen display function. These bitmaps are written directly into the DRAM memory by the microcontroller.

Undetected bitstream errors which would cause decoder errors activate the error concealment functions.



PQFP160
(Plastic Quad Flat Pack)

ORDER CODE : STi3520CV

EXHIBIT H

CS6651

MPEG-2 Video Decoder for FPGA

AMPHION™
Virtual Components for the Converging World

The CS6651 MPEG2 decoder is designed to provide high performance solutions for a broad range of motion image applications. This highly integrated application specific virtual component (ASVC) is for standard definition video, compliant with ISO/IEC 13818-2 (MPEG2) and capable of decoding video streams at the Main Profile at Main Level (MP@ML). The CS6651 is at home in mainstream consumer applications and can also decode MPEG1 (ISO/IEC 11172-2) bitstreams. The CS6651 been handcrafted by Amphion for optimal performance while minimizing power consumption and silicon area.

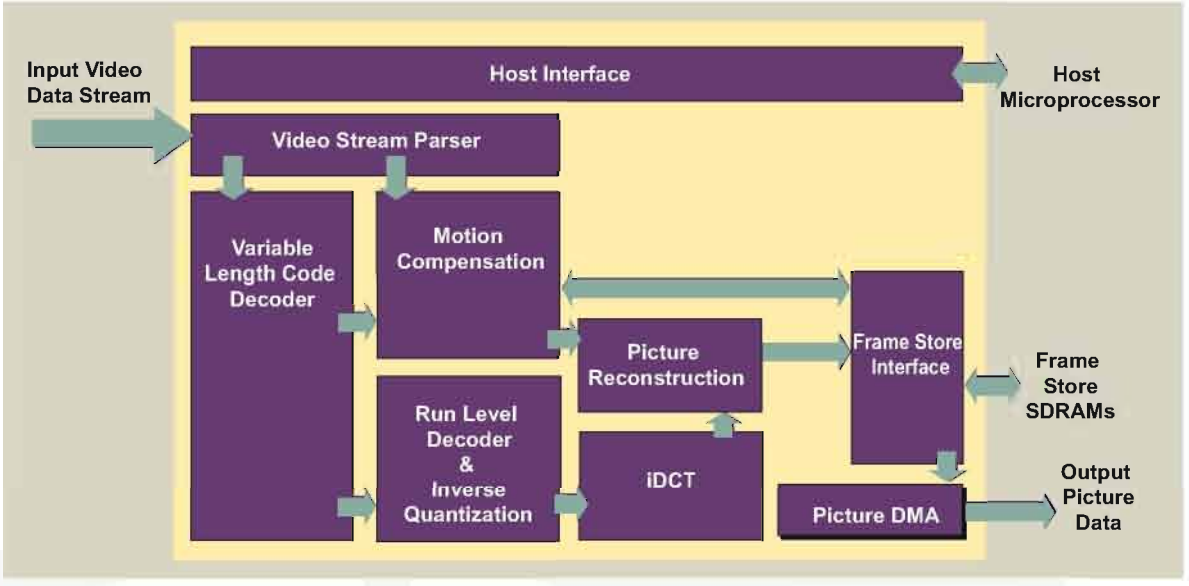


Figure 1: CS6651 Overview Diagram

DECODER FEATURES

- **Supports progressive scan and interlaced streams**
- **ISO/IEC 13818-2 (H.262) Compliant**
  - MP@ML
  - Decodes ISO/IEC11172-2 (MPEG1) Constrained Parameter bitstreams
- **High performance solution for MPEG2 decoding**
  - Supports input bit rates up to 30Mbit/sec
  - Real time decode and display of MP@ML
- **Supports PAL and NTSC SDTV resolutions and frame rates**
- **Bitstream error detection and recovery**
- **Glueless interface to external SDRAM**
- **Capable of standalone stream decoding or host CPU controlled operation**
- **Fully synchronous design with host shut-down and restart control**

APPLICATIONS

- **Digital cable and satellite set-top decoder box for SDTV**
- **DVD Players**
- **PC video hardware accelerator**

CS6651 FUNCTIONAL DESCRIPTION

The CS6651 ASVC is a highly integrated MPEG2 video decoder suitable for a wide range of video applications. The CS6651 accepts the input video elementary stream as aligned bytes from conditional access decryption, transport stream demulti-plexer, or similar source. The maximum average input bit rate is 30Mbits/sec. The core can operate in a default mode on an input stream without the intervention of a host CPU. In this mode pictures will be decoded from the video stream and output in correct display order. A host CPU has access to a full range of information and control to manipulate the behavior of the decoder to permit audio/video synchronization, pan and scan and letterbox conversion, and various trick modes.

The output from the core is provided by a highly configurable pixel stream DMA (Direct Memory Access) engine. This engine allows adjustable output video component sequencing and provides external logic with control over the display of the picture.

To meet the bandwidth requirements of MP@ML decoding, a bank of two dedicated SDRAM chips is used. These SDRAM chips are commodity 64Mbit SDRAMs in 2Mx32 configuration.

FUNCTIONAL BLOCK OVERVIEW

VIDEO STREAM PARSER

The Video Stream Parser unit extracts various encoding parameters from the input video stream and any requested user specific data contained within the stream, such as closed-caption or teletext data. This information is contained in headers at each layer of the stream and may be used throughout the rest of the decoding and reconstruction process. Selected user data is stored to buffer space and made available to the host CPU. Having removed header information from the stream, the Video Stream Parser unit passes the variable length encoded picture data to the Variable Length Code (VLC) Decoder unit. A range of parameters describing the overall stream and the picture currently being decoded is made available to the rest of the decoder.

VARIABLE LENGTH CODE DECODER

The Variable Length Code Decoder unit decodes the Huffman- style variable length encoded picture data. The outputs of this unit include the Discrete Cosine Transform (DCT) block run-level information for the Inverse DCT (iDCT) unit and decoded macroblock motion vectors for the motion compensation unit as well as a number of information fields describing the section of the picture currently being decoded. These decoded fields are made available to the rest of the decoder.

RUN-LEVEL DECODER & INVERSE QUANTIZATION

The output run-level information from the VLC decoder is converted into complete blocks of 64 quantized DCT coefficients by the Run-Level decoder. These coefficients are passed to the Inverse Quantizer for conversion back to actual DCT coefficients. To perform this, the Inverse Quantizer keeps track of a number of tables and scale factors, all extracted from the input video stream.

INVERSE DCT

This high performance unit performs the inverse quantization of 8x8 DCT-encoded Y, Cr and Cb pixel blocks. This key unit is capable of streaming data through continuously; transforming every 64 clock cycles an entire block of 8x8 DCT coefficients into an 8x8 block of pixel samples or estimated sample corrections.

MOTION COMPENSATION

Where the video data is encoded as an estimate using previous pictures and a set of corrections, the Motion Compensation unit forms the estimated pixel values. The Motion Compensation unit takes decoded motion vectors from the Variable Length Code Decoder unit and translates them into row and column coordinates within the pictures from which the estimations are being made. The reference samples for these coordinates are requested from the Frame Store Interface and the resulting pixels combined where necessary to form the estimated values for the block being decoded.

PICTURE RECONSTRUCTION

The Picture Reconstruction unit combines decoded pixels or corrections from the iDCT unit with the estimated pixels from the Motion Compensation Unit and writes the resulting pixels to the Frame Store, ready for subsequent display or reference.

FRAME STORE INTERFACE

The Frame Store is required for the storage of the two reference pictures used in the MPEG2 algorithm to form the estimated pixels. It also stores the frame currently being decoded and another frame currently being displayed. This allows the decoding and the display operations to be decoupled making audio/video synchronization simpler to maintain.

The Frame Store is implemented using two SDRAM chips which are commodity PC133 64Mbit parts, each with 2Mx32 organization. The memory interface runs at the core speed and can be directly connected to the SDRAM chip.

The Frame Store SDRAM Interface handles the mapping of pixel read and write requests from the Motion Compensation, Picture Reconstruction and Picture DMA units into linear memory addresses. Additionally, the host interface can access the memory banks. Arbitration between the various accessing units and memory transaction queues are all maintained by the SDRAM Interface.

PICTURE DISPLAY DMA

The Picture Display DMA has a double-byte output interface which can carry Y, Cr or Cb pixel data. Y and Cr or Cb data can be output simultaneously as 16-bit wide values or sequentially as four separate bytes. The Picture Display DMA unit will upsample the chrominance vertically to provide a 4:2:2 output. The Display DMA engine has the capability to be programmed by the host CPU to display only a certain portion of the picture or, in stand-alone mode, will display the entire coded picture.

A number of handshake signals are provided on the Picture Display DMA interface to allow the external logic to control the timing of the pixel output stream and to control the end of the current scan row or picture display. Outputs indicate to the external logic the nature of the pixel being currently driven; and end of row and end of picture flags are available to allow, for example, sync pulse generation.

HOST INTERFACE

When the CS6651 is operating with the assistance of a host CPU, a number of additional features can be accessed. All interfacing between the host and the CS6651 is performed through the Host Interface unit. This unit allows read/write access to all the internal control, status and video stream parameter registers contained within the decoder.

The Host Interface also provides a simple 32-bit read/write access to the Frame Store SDRAM. Normally, the areas of the SDRAM used for storage of picture data cannot be accessed by the Host Interface; however, a bypass mode allowing direct access is provided for system diagnostic tests, etc.

A number of conditions arising from the decoding of the video stream may require the software on the CPU to be alerted. An interrupt controller within the Host Interface unit provides a simple Interrupt Request signal and an interrupt status and mask register.

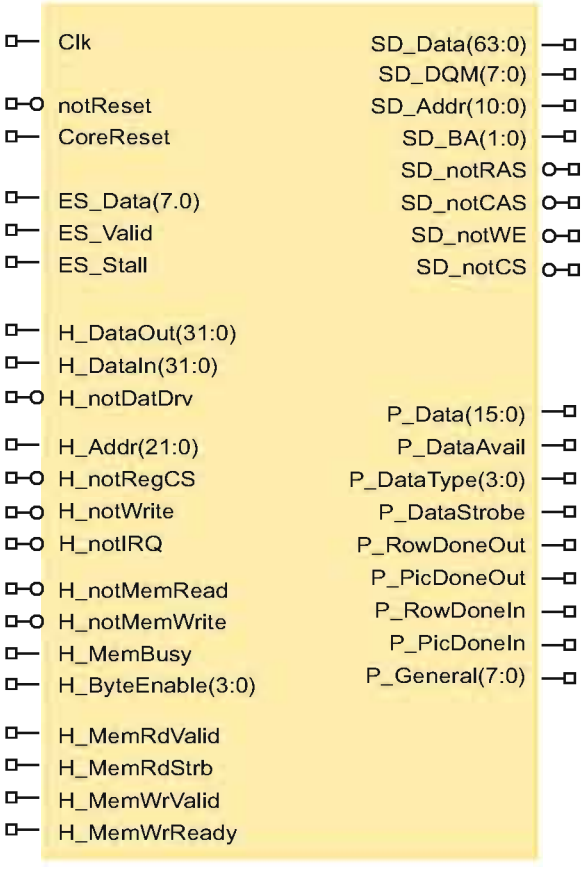


Figure 2: CS6651 Symbol and Pin Description

Table 1: Global Signals

Signal	I/O	Description
Clk	I	Core Clock. Master clock used for all logic and the external SDRAM interface. This clock should also be routed to the external SDRAM chips. This clock should be 27 MHz.
notReset	I	Core reset. Asynchronous, active low global core reset
CoreReset	I	Core reset. Synchronous, active high core reset

Table 2: Input Interface

Signal	I/O	Description
ES_Data[7:0]	I	Elementary Stream Data, byte aligned video elementary stream data from the Conditional Access decryption unit or transport stream demux. Maximum average input bit rate is 30Mbits/s
ES_Valid	I	Data Valid Strobe. ES_Data is latched on the positive edge of Clk when ES_Valid is asserted, and ES_Stall is de-asserted.
ES_Stall	O	Data Stall. Input data may be bursted into the core at a rate higher than the specified maximum 30Mbit/sec. In this case the core will indicate that it temporarily cannot receive any more data by assertion of ES_Stall. ES_Data will not be latched while ES_Stall is asserted.

Table 3: Picture Output Interface

Signal	I/O	Description
P_Data[15:0]	O	Picture Output Data. Output from the decoded picture display DMA engine. Contains either Y, Cr or Cb, as indicated by P_DataType. In 16 bit mode, the upper 8 bits carry Y and the lower 8 bits carry either Cr or Cb as indicated by P_DataType.
P_DataStrobe	I	Data Valid Strobe. Indicates that the external logic will consume the current P_Data on the next rising edge of clock. This signal is also used to qualify the P_RowDoneIn and P_PicDoneIn signals.
P_DataAvail	O	Picture Data Available. Indicates that the DMA engine has been configured and is running and that P_Data carries a valid picture sample.
P_DataType[3:0]	O	Picture Data Type, indicates the type of sample on P_Data. the bottom two bits carry the component identification as follows: 00 = Y1, 01 = Y2, 10 = Cb, 11 = Cr. The top two bits carry display frame/field information as follows: 00 = progressive, 01 = undefined, 10 = top field, 11 = bottom field.
P_RowDoneIn	I	Last Pixel In Row. This input can be used to terminate a row scan and move on to the next. This may be used with pan and scale external logic. This input is ignored in certain DMA engine configurations. Should be asserted for the last byte of the pixel sample group – the engine will move to the next row after the last component for the group is taken.
P_PicDoneIn	I	Last Pixel In Picture. Indicates to the DMA engine that the display of the picture is complete at the end of the current pixel. The engine will revert to idle mode. This input is ignored in certain DMA engine configurations. Should be asserted for the last byte of the pixel sample group – the engine will stop after the last component for the group is taken.
P_RowDoneOut	O	Last Pixel In Row. This output can be programmed to indicate the last component of the last pixel of the row. This requires correct configuration of the DMA engine row length register.
P_PicDoneOut	O	Last Pixel In Picture. This output can be programmed to indicate the last component of the last pixel of the picture. This requires correct configuration of the DMA engine vertical size register.
P_General[7:0]	O	General Outputs. These outputs directly reflect the programmed value in the DMA General Output register. They can be used by the host CPU to inform the display logic of specific display parameters such as PAL/NTSC encoding information etc.

Table 4: Frame Store Interface

Signal	I/O	Description
SD_Data[63:0]	I/O	SDRAM Data Bus. Bidirectional read/write databus to the external SDRAM
SD_Addr[10:0]	O	SDRAM Address Bus. Carries row or column addresses or commands to the external SDRAM.
SD_BA[1:0]	O	SDRAM Bank Address. Indicates selected bank for the current SDRAM command.
SD_DQM[7:0]	O	SDRAM DQ Mode. Used to control burst transfers of data to/from the SDRAM.
SD_notRAS	O	SDRAM Row Address Strobe. Strobes a row address or command into the SDRAM.
SD_notCAS	O	SDRAM Column Address Strobe. Strobes a column address or command into the SDRAM.
SD_notWE	O	SDRAM Write Enable. Indicates to the SDRAM that a write command is required.
SD_notCS	O	SDRAM chip select. Initiates a command to the SDRAM.

Table 5: Host Interface

Signal	I/O	Description
H_DataIn[31:0]	I	Host Data Input. Host Write data into the core.
H_DataOut[31:0]	O	Host Data Output. Host Read data from the core.
H_notDatDrv	O	Host Data Drive. Indicates that a read is active. This can be used to control external tristate drivers if required. Active low.
H_Addr[21:0]	I	Host Address. Used to select a register for read/write, or a Frame Store SDRAM word to be accessed.
H_notRegCS	I	Host Chip Select. Active low enable signal controls all host register accesses.
H_notWrite	I	Host Write Select. If asserted when H_notRegCS is asserted, the register addressed by H_Addr will have the value on H_DataIn assigned to it on the rising edge of the Clk signal, if the appropriate byte write enable signal is also asserted. If de-asserted when H_notRegCS is asserted then a register read is initiated and H_DataOut will show the selected registers data on the next clock cycle.
H_notIRQ	O	Host interrupt request. Active low output
H_ByteEnable[3:0]	I	Host Byte Write Enables. Used on write accesses to control which bytes in a register or SDRAM word actually get written.
H_notMemWrite	O	Host Memory Write Access. Initiates an SDRAM host write transaction.
H_MemBusy	O	Host Memory Interface Busy. Indicates that a memory access transaction is in progress. This can be used to insert read wait states and to stall for posted writes to complete.
H_MemRdValid	O	Host Memory Read Data Valid. Indicates that the read data is available on the H_DataOut port.
H_MemRdStrb	I	Host Memory Read Data Strobe. Indicates that the host will consume the data from the H_DataOut port on the next rising edge of Clk.
H_MemWrValid	I	Host Memory Write Data Valid. Indicates that the host has placed valid write data on the H_DataIn port. Note that H_ByteEnable should be valid at the same time as the data.
H_MemWrReady	O	Host Memory Write Data Ready, indicates that the core is ready to consume the data on H_DataIn on the positive edge of Clk when it is signalled as valid with H_MemWrValid

TIMING DIAGRAMS

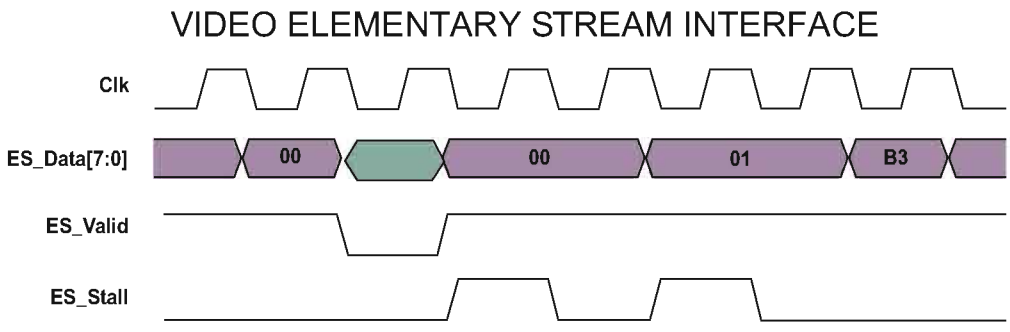


Figure 3: Using ES_Valid and ES_Stall

In the above figure, the MPEG2 Video Sequence Start Code is being loaded into the decoder core. The value on ES_Data is loaded by the core when ES_Valid is asserted and ES_Stall is not being asserted. The external stream data source logic can use ES_Valid to indicate the presence of real video data. The CS6651 core will assert ES_Stall when it is temporarily unable to accept any more bytes. The average data rate entering the core can be up to 30 Mbits/second. The data rate is further constrained by the maximum frame rate defined in MPEG2 MP@ML for the resolution of image coded into the stream. If the core is unable to process data due to the input frame rate exceeding the display frame rate, then it will assert ES_Stall.

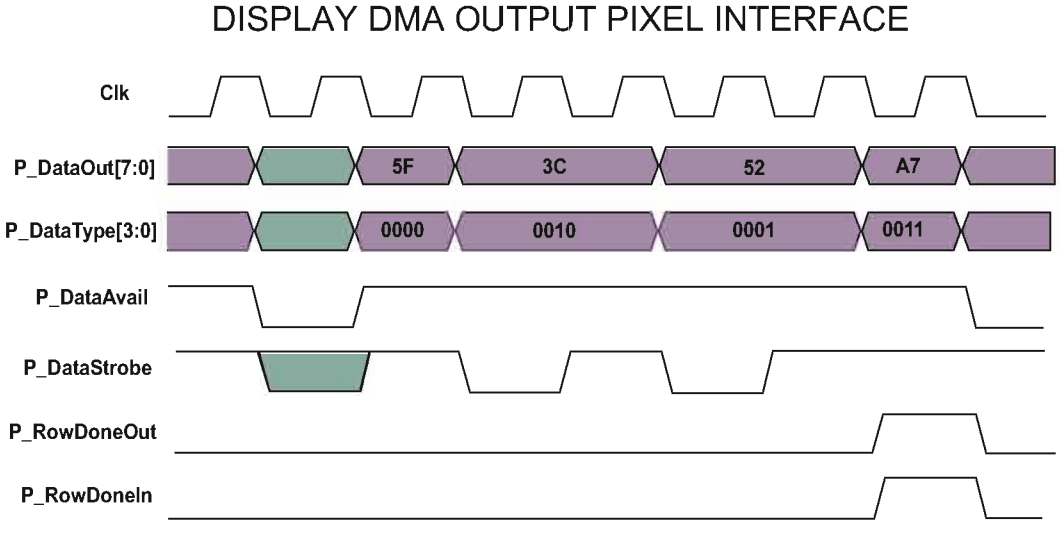


Figure 4: Picture DMA Outputs

In this example, the 4:2:2 sampled pixel set consists of the luminance (Y) values 5F and 52, the blue chrominance difference value 3C, and the red chrominance difference value A7. The bottom two bits of P_DataType are indicating the sample type currently being output on P_DataOut. The top two bits indicate that a progressive frame is being output.

This waveform shows P_DataAvail is not asserted for the clock cycle before the pixel group commences. During this time the P_DataOut and P_DataType values are undefined and P_DataStrobe is ignored. Also, the initial clock cycles of the 3C and 52 values are not accepted by the external logic, P_DataStrobe is not asserted, so the CS6651 continues to drive those old values for another clock cycle.

In this example the sample set is the last in the current row, so P_RowDoneOut is asserted. The diagram also shows how the external logic can indicate P_RowDoneIn to the core. In this case the signal has no effect since P_RowDoneOut was asserted already. Use of P_PicDoneOut and P_PicDoneIn is similar.

P_General is not shown here and simply reflects the value currently programmed into the Display DMA Controller's GeneralDataValue register.

THE SDRAM INTERFACE

The SDRAM interface timing is completely specified by JEDEC SDRAM standards.

HOST INTERFACE

The host interface is effectively in two parts, a Configuration and Status register read and write section and a Host to memory read and write access section.

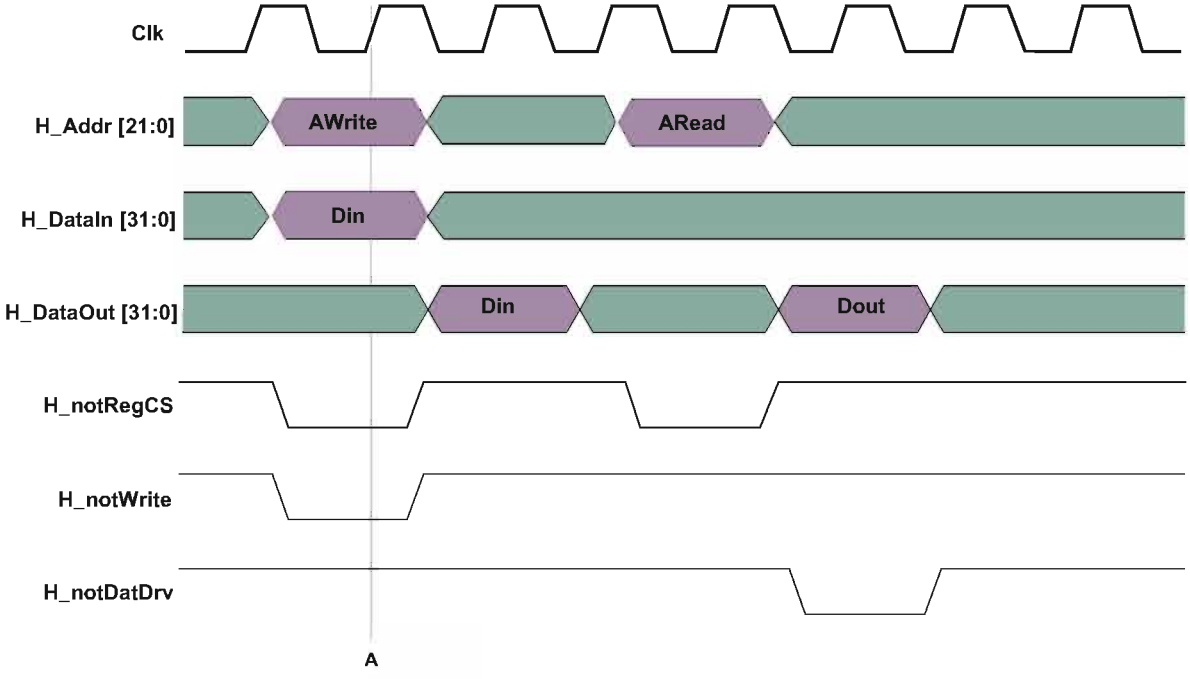


Figure 5: Host Write to Configuration Register and Host Read of Configuration or Status Register

Figure 5 illustrates a host write to a configuration register and a host read of a configuration or status register. The write happens on the rising edge of the Clk signal indicated by the line A. The H_DataOut bus reflects the new value in the register on the next clock cycle. The read cycle is synchronous. Whenever H_notRegCS is asserted and the H_notWrite signal is not, then H_notDatDrv will be asserted in the following cycle, when H_DataOut is valid. This can be used to enable tristate drivers on a bi-directional host data bus, if required.

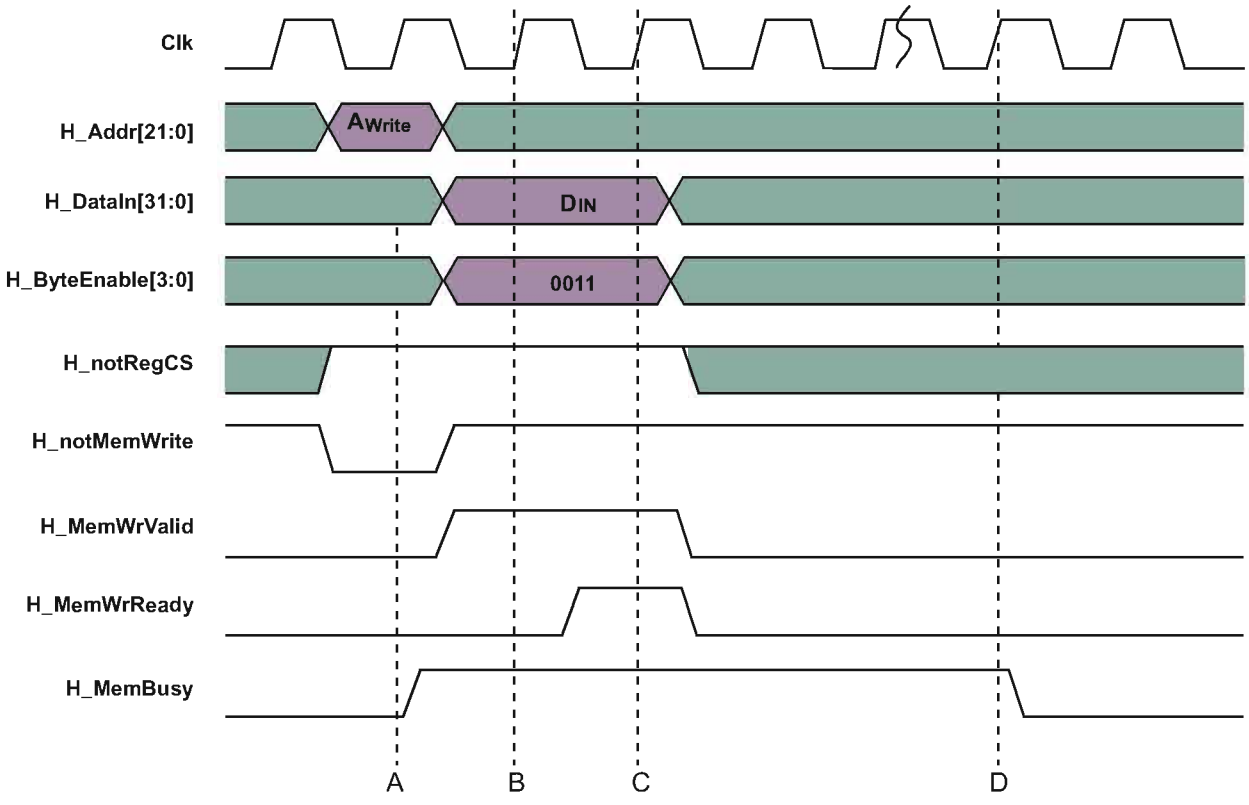


Figure 6: Host Writing Location in Memory

In Figure 6, the host is writing a location in memory. On clock cycle A the write address is latched since H_notMemWrite was asserted by the host. A single clock cycle of data is made available to the decoder core on clock cycle B using H_MemWrValid. On clock cycle C the core loads the data and performs the write. The data load is indicated by the assertion of H_MemWrReady.

In this example the bottom two bytes of H_DataIn, i.e. 15:0, were enabled. Only these two bytes of the memory will be written. Bits 31:16 of the memory location will remain unchanged.

The above diagram shows the write spread out over three clock cycles. Normally, when the memory interface is not busy and the host has the address and data available together, clock cycles A, B and C will all be on the same rising edge. H_MemBusy remains asserted until clock cycle D, when the memory write actually reaches the memory controller. This signal can be used to insert wait states into a host interface controller if necessary.

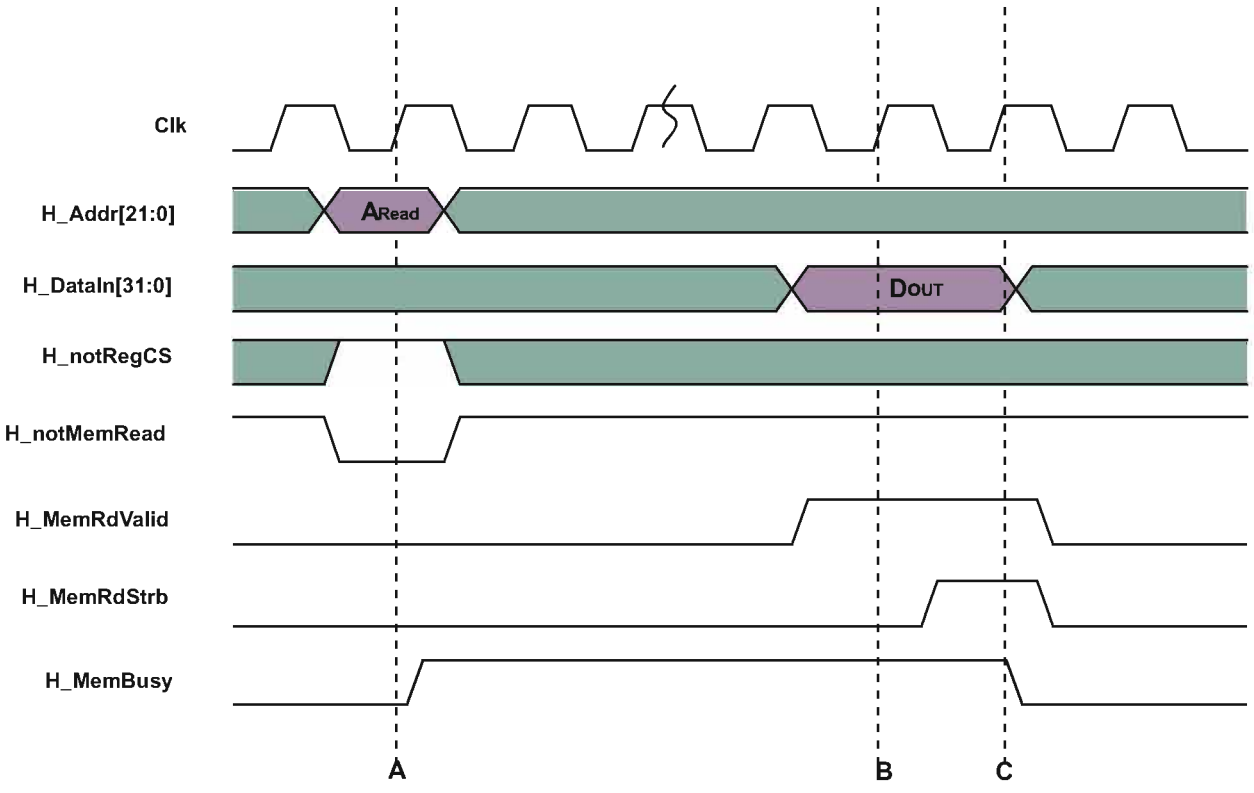


Figure 7: Host Reading Location in Memory

In Figure 7, the host is reading a location in memory. On clock cycle A the read address is latched due to H_notMemRead being asserted by the host. On this edge H_MemBusy will be asserted until the read is completed. A single clock cycle of data is made available by the decoder core on clock cycle B using H_MemRdValid. On clock cycle C the external logic loads the data and completes the read, indicated by the assertion of H_MemRdStrb. On this clock cycle the core clears H_MemBusy. The entire word is read from memory and made available on H_DataOut, regardless of the state of the H_ByteEnable port.

CS6651 DESIGN METHODOLOGY

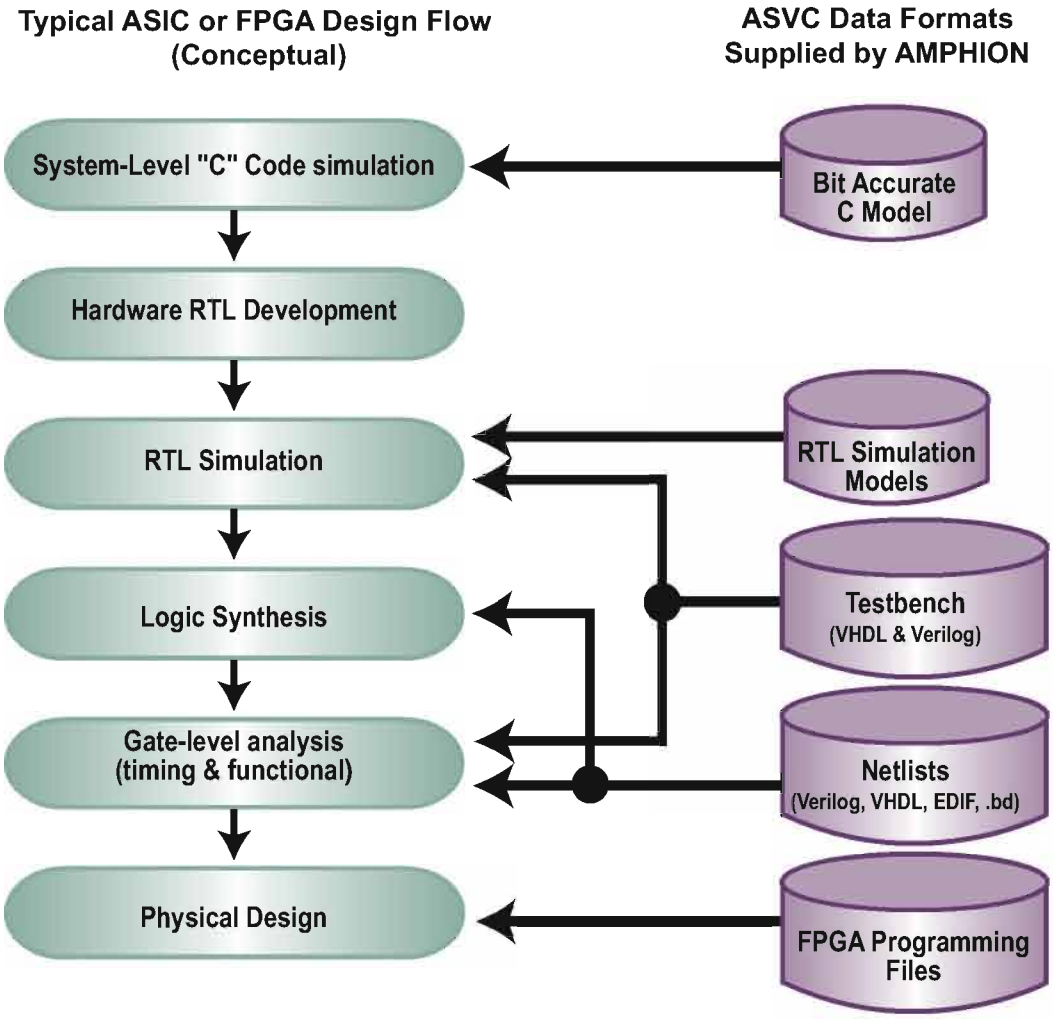


Figure 8: Design Data Formats Supplied by Amphion

AMPHION™

AVAILABILITY AND IMPLEMENTATION INFORMATION

PROGRAMMABLE LOGIC CORES FOR ACTEL SILICON DEVICES

For ASIC prototyping or for projects requiring the fast time-to-market of a programmable logic solution, Amphion's programmable logic core solutions offer the silicon-aware performance tuning found in all Amphion products, combined with the rapid design times offered by today's leading programmable logic solutions.

Table 7: CS6651 Programmable Logic Cores

PRODUCT ID#	SILICON VENDOR	PROGRAMMABLE LOGIC PRODUCT	DEVICE RESOURCES USED (LOGIC)	DEVICE RESOURCES USED (MEMORY)	AVAILABILITY
CS6651	Actel	Contact Amphion or Actel	TBD	TBD	Contact us

ABOUT AMPHION

Amphion (formerly Integrated Silicon Systems) is the leading supplier of speech coding, video/ image processing and channel coding application specific silicon cores for system-on-a-chip (SoC) solutions in the broadband, wireless, and mulitmedia markets.

Web: www.amphion.com

Email: info@amphion.com

CORPORATE HEADQUARTERS

Amphion Semiconductor Ltd
50 Malone Road
Belfast BT9 5BS
Northern Ireland, UK

Tel: +44.28.9050.4000
Fax: +44.28.9050.4001

EUROPEAN SALES

Amphion Semiconductor Ltd
CBXII, West Wing
382-390 Midsummer Boulevard
Central Milton Keynes
MK9 2RG England, UK

Tel: +44 1908 847109
Fax: +44 1908 847580

WORLDWIDE SALES & MARKETING

Amphion Semiconductor, Inc
2001 Gateway Place, Suite 130W
San Jose, CA 95110

Tel: (408) 441 1248
Fax: (408) 441 1239

CANADA & EAST COAST US SALES

Amphion Semiconductor, Inc
Montreal
Quebec
Canada

Tel: (450) 455 5544
Fax: (450) 455 5543

SALES AGENTS

Voyageur Technical Sales Inc
1 Rue Holiday
Tour Est, Suite 501
Point Claire, Quebec
Canada H9R 5N3

Tel: (905) 672 0361
Fax: (905) 677 4986

Phoenix Technologies Ltd
3 Gavish Street
Kfar-Saba, 44424
Israel

Tel: +972 9 7644 800
Fax: +972 9 7644 801

SPINNAKER SYSTEMS INC
Hatchobori SF Bldg. 5F 3-12-8
Hatchobori, Chuo-ku
Tokyo 104-0033 Japan

Tel: +81 3 3551 2275
Fax: +81 3 3351 2614

JASONTECH, INC
Hansang Building, Suite 300
Bangyidong 181-3, Songpaku
Seoul Korea 138-050

Tel: +82 2 420 6700
Fax: +82 2 420 8600

SPS-DA PTE LTD
21 Science Park Rd
#03-19 The Aquarius
Singapore Science P ark II
Singapore 117628

Tel: +65 774 9070
Fax: +65 774 9071

EXHIBIT I

Texas Instruments

Application Report
SPRA649 - March 2000

MPEG-2 Video Decoder: TMS320C62x Implementation

Ngai-Man Cheung *Texas Instruments Incorporated*

ABSTRACT

This application report describes the implementation of the MPEG-2 video decoder on the TMS320C62x DSP. The MPEG-2 video standard specifies the decompression and coded representation for entertainment-quality digital video, and is widely used in different digital video systems including DVB, DTV, DVD, DSS, etc. The decoder software implements all the MPEG-2 main-profile-at-main-level functionality, and conforms to the eXpressDSP™ Algorithm Standard (xDAIS) to enhance reusability. This report describes different aspects of the decoder software, including algorithm overview, coding guidelines, decoder APIs, memory requirement, and performance.

Contents

List of Figures

List of Tables

1 Introduction

This application report describes the implementation of the MPEG-2 video decoder on the TMS320C62x DSP. The decoder software implements all the MPEG-2 main-profile-at-main-level functionality, and conforms to the eXpressDSP Algorithm Standard (xDAIS) to enhance reusability. In the following sections we will describe different aspects of the decoder software, including algorithm overview, coding guidelines, decoder APIs, memory requirement, and performance.

1.1 Algorithm Overview

The *MPEG-2 video* [1,2] standard specifies the decompression and coded representation for entertainment-quality digital video. It is widely used in different digital video systems, including DTV (digital television), DVB (Digital Video Broadcast), DSS (direct satellite system), and DVD (digital versatile disc). The MPEG-2 video decoder plays an important role in consumer electronics like DVD players, set-top boxes, and DSS units. Compared with the hardware implementation, software implementation of the decoder is more flexible, easier to be customize for different applications, and easier to upgrade with new features. Also, the programmability of the device offers the advantage of putting multiple functions (e.g., video decoding, modem function, and speech control interface) in the same hardware platform.

We have implemented the MPEG-2 *main-profile-at-main-level* video decoder, which has the maximum input bit rate of 15 Mbps (megabit per second) and *chrominance* format of 4:2:0. This is the most common format and is being used in many applications.

2 Algorithm Description

Figure 1 shows the MPEG-2 video-decoding algorithm. The MPEG-2 standard employs a number of techniques to achieve high compression ratio while preserving good video quality.

2.1 Interframe Coding Using Motion Compensation

Motion compensation (MC) achieves compression by using the fact that within a short sequence of pictures, the scenes are similar and many objects move only a short distance. By using these temporal redundancies, many parts of the current picture could be predicted by the previously decoded pictures. In MC, the picture is divided into blocks. Two-dimensional motion vectors are computed to tell where to retrieve blocks of pixel values from the previously decoded pictures to predict the block of pixels of the current picture. Compression is achieved by encoding the motion vectors and prediction error instead of the block of pixels. The prediction error has less spatial redundancy and can be compressed effectively by transform coding.

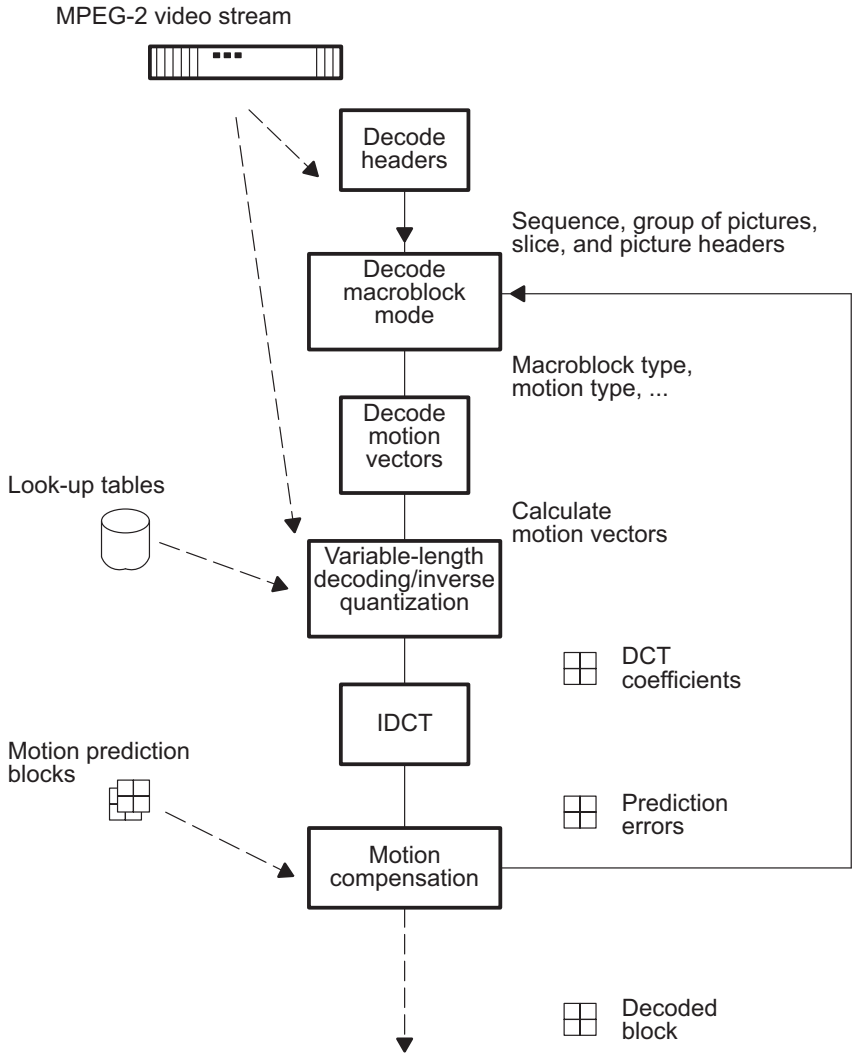


Figure 1. MPEG-2 Video Decoding Algorithm

2.2 Transform Coding Using Discrete Cosine Transform

During encoding, the discrete cosine transform (DCT) is applied to the prediction error in interframe coded macroblock and the pixel values in intraframe coded macroblock. The picture is divided into blocks of 8-by-8 pixels. The DCT transforms the pixel values into another block of the same size, consisting of the horizontal and vertical spatial frequency coefficients representing the detail of the block. While the energy of the image signal and prediction error can be distributed randomly across a block, the energy of the DCT block is concentrated on the low frequency. Compression is achieved by using a quantizer with quantization steps varied by the frequency, according to psycho-visual characteristics such that quantization noise is unlikely to be perceived. Also, many high-frequency coefficients are very small and have a value of zero after quantization. Compression can be achieved by using a zig-zag order to gather the coefficients of value zero, and encoding the block into a series of zero-run and level pairs with run-length encoding.

2.3 Variable-Length Coding

The variable-length coding (VLC) assigns each run-level pair a code word based on the frequency of occurrence of the pair. Pairs that occur more frequently are assigned short code words while those that occur less frequently are assigned long code words. Compression is achieved by the fact that overall, the more frequent shorter code words dominate.

3 Decoder Implementation

In this section, we describe different aspects of our implementation of the video decoder.

3.1 Features

These are the features of the video decoder software:

- The whole video decoding is software-based working on the programmable DSP. There is no hardware assistance of decoding. This ensures maximum flexibility.
- The decoder is completely MPEG-2 main-profile-at-main-level compliant. We have tested the decoder thoroughly with the official MPEG-2 compliance test-streams and have verified that the decoder is completely compliant with the specification. This ensures both the correctness of the algorithm implementation and the quality of the output picture.
- The decoder could also handle MPEG-1 constrained parameters bit-streams (CPB).
- The decoder is xDAIS [3] compliant. We implemented all the xDAIS rules and most of the guidelines. This ensures the decoder algorithm can be easily integrated into different framework systems and environments. Please note that as xDAIS itself may undergo some changes, the software will be updated to reflect these changes.
- The decoder is multichannel enabled. The decoder is reentrant and can handle several different decoding channels simultaneously (this is subject to further testing). The decoder can be interrupted at any place other than the software pipeline code. The interrupt latency shall be less than 10 μs at 250 MHz.

3.2 Decoder Structure

The decoder is divided into the following modules (see Figure 2):

- VLD, which includes functions to perform variable-length decoding, run-length expansion and dequantization
- IDCT, which includes functions to perform inverse discrete cosine transform
- Motion compensation address calculation, which includes functions to calculate the reference blocks location and to fetch the blocks into internal memory
- Motion compensation kernel, which includes functions to calculate the prediction pixels
- Miscellaneous functions to decode header information, motion vectors, etc.
- Implementation of IALG and IRTC interfaces as required in xDAIS

The modules are glued together with the decoder control code. The control code invokes the functions in different modules as well as passes and receives the data.

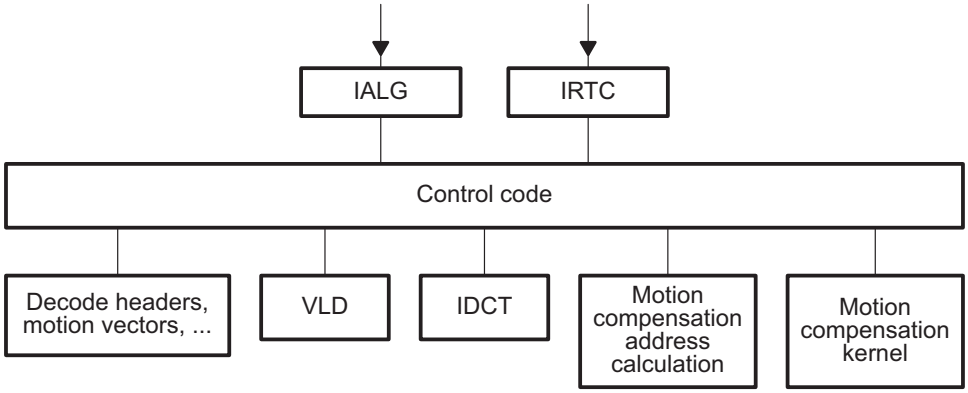


Figure 2. MPEG-2 Video Decoder Structure

3.3 Coding Guidelines

The decoder program is a mixed C and TMS320C62x assembly language implementation. The coding follows all the xDAIS rules. Some of them are:

- The decoder is reentrant.
- All the data references are fully relocatable. Also, all the decoder code is fully relocatable.
- All external definitions are prefixed with MPEG2VDEC or MPEG2VDEC_ti.

Please refer to the xDAIS document for a complete listing of coding rules.

3.4 Interrupt Issues

The decoder can be interrupted at any place other than the software pipelined loops. The maximum interrupt latency is less than 10 μs at 250 MHz, as recommended in the xDAIS guideline.

3.5 Multichannel Implementation

The decoder is reentrant and can be used in multichannel environment. The decoded information of each channel is retained in the algorithm instance object, MPEG2VDEC_Handle. Client or framework uses the decoder's API (Application Programming Interfaces) MPEG2VDEC_create to create the algorithm instance object for each decoding channel. After that, the framework passes the instance object to the API MPEG2VDEC_apply to decode a picture.

4 Interfacing With the Decoder

The decoder can be configured to run on its own or link to some framework systems. In the latter case, framework can use the decoder's APIs to interface with the decoder. In this section we will describe the decoder APIs. Also, we will give an example framework code to illustrate the interfacing.

4.1 Decoder APIs

The decoder APIs include:

- MPEG2VDEC_init

```
Void MPEG2VDEC_init(Void);
```

Parameters
NULL.

Return Value
NULL.

Description
Decoder initialization. Should be the first call to the decoder.

- MPEG2VDEC_create

```
MPEG2VDEC_Handle MPEG2VDEC_create(
                const IMPEG2VDEC_Fxns *fxns,
                const MPEG2VDEC_Params *prms);
```

Parameters

Parameter	Meaning
const IMPEG2VDEC_Fxns *fxns	Functions table
const MPEG2VDEC_Params *prms	Creation parameter

Return Value
MPEG2VDEC algorithm instance handle.

Description
Create an algorithm instance object. Call this for every decoding channel.

- MPEG2VDEC_delete

```
Void MPEG2VDEC_delete(MPEG2VDEC_Handle handle);
```

Parameters

Parameter	Meaning
MPEG2VDEC_Handle handle	MPEG2VDEC algorithm instance handle

Return Value
NULL.

Description
Delete an algorithm instance object. Call this after completion of a decoding channel.

- MPEG2VDEC_exit

```
Void MPEG2VDEC_exit(Void);
```

Parameters
NULL.

Return Value
NULL.

Description
Decoder finalization.

- MPEG2VDEC_apply

```
Void MPEG2VDEC_apply(MPEG2VDEC_Handle handle,
                     Int *input[],Int *output[]);
```

Parameters

Parameter	Meaning
MPEG2VDEC_Handle handle	MPEG2VDEC algorithm instance handle.
Int *input[1]	Address of the function code, functionCode. The functionCode could be FUNC_DECODE_FRAME or FUNC_START_PARA.
Int *input[2]	Starting of external input bit-stream buffer.
Int *input[3]	Address of the size of the external input bit-stream buffer.
Int *output[1]	Address of the output parameter buffer.
Int *output[2]	Starting of the external output frame buffer.
Others	Reserve for framework specific extension.

Return Value
NULL.

Description
This applies the decoder to the input bit stream and outputs the result in the output buffer. The function code, input[1], should be FUNC_START_PARA at the beginning of a video sequence and FUNC_DECODE_FRAME afterward. The pass-in algorithm instance object identifies the decoding channel.

Framework should call the API MPEG2VDEC_init to initialize the decoder. After that, framework should call the MPEG2VDEC_create to create an algorithm instance object for each channel. The algorithm instance object contains all the status information for each decoding channel. Then, framework can call the MPEG2VDEC_apply to apply the decoder to the input bit stream.

4.1.1 Input Raw Data

Framework passes the MPEG-2 raw input data to the decoder through an input buffer starting at input[2] when calling MPEG2VDEC_apply. The size of the input buffer is pointed by input[3]. The input buffer is organized in a circular fashion. Framework is responsible for filling the buffer and ensuring enough input data to feed the decoding of one picture. Framework can learn how much input data the decoder has consumed by the variable ((DECODE_OUT *) (output[1]) ->next_wptr), which points to the head of the circular input buffer. The input buffer must be a multiple of 4 bytes and aligned on a 4-byte boundary. We recommend the input buffer size to be 512 KB or more. The input buffer should reside in external memory.

4.1.2 Output Decoded Picture

The MPEG2VDEC algorithm returns the decoded picture in the output frame buffer pointed by output[2] at the return of MPEG2VDEC_apply. The algorithm requires keeping four output frames, so the output frame buffer should be of size 4 x Picture_Size at the minimum, where Picture_Size = Picture_Height x Picture_Width x 1.5. Moreover the output frame buffer has to be aligned on a 4-byte boundary. We recommend the output buffer size to be 2440 KB, which can handle 720x576 4:2:0 video properly.

The output picture is stored in the 4:2:0 YU12 format. When algorithm returns, the client should check the variable ((DECODE_OUT *) (output[1]) ->outputting) to see if a decoded picture is ready, and if so the output picture could be found at the memory location ((DECODE_OUT *) (output[1]) ->outframe). The output picture is always in frame format. This is to avoid the confusion in decoding interlaced video when the output picture can be in frame or field format within the same sequence.

4.1.3 Output Parameters

The decoder returns the output parameter in output[1] at the return of MPEG2VDEC_apply. The parameter is either the structure START_OUT at the beginning of a sequence or DECODE_OUT afterward, as shown in Figure 3.

```
/*********************************************************/
/* Output parameter at the beginning of sequence.        */
/*********************************************************/
typedef struct _START_OUT {
  Int fault;                          /* Any problem occur?    */
  Int ld_mpeg2;                       /* MPEG-2 stream?        */
  Int bit_rate;                       /* Input bit rate        */
  Int picture_rate;                   /* Output picture rate   */
  Int vertical_size;                  /* Ori. pic. dimension   */
  Int horizontal_size;
  Int coded_picture_width;            /* Coded pic. dimension  */
  Int coded_picture_height;
  Int chroma_format;
  Int chrom_width;
  Int prog_seq;                       /* Progressive seq.?     */
} START_OUT;
/*********************************************************/
/* Output parameter afterward.                           */
/*********************************************************/
typedef struct _DECODE_OUT {
  Int fault;                          /* Any problem occur?    */
  Int pict_type;                      /* I, P or B-pic.        */
  Int pict_struct;
  Int next_wptr;                      /* Head of cir. input    */
  Int topfirst;
  Int end_of_seq;                     /* End of sequence?      */
  Int outputting;                     /* Output any frame?     */
  SmUns outframe;                     /* Starting of out pic.  */
} DECODE_OUT;
```

Figure 3. Video Decoder Output Parameters

4.2 Example Framework Code

Figure 4 shows an example framework code to illustrate the usage of the decoder's APIs.

```
#define SHARE_MPEG2_RDBUF_SIZE    (128 * 1024)
unsigned int share_bsbuf_storage[SHARE_MPEG2_RDBUF_SIZE];
/* 512KB input buffer. 4-bytes alignment. */
#define MAX_PICT_SIZE             0x098800
/* 610KB. Enough for 720x576 4:2:0. */
#define NO_OF_FRAME_BUF           4
unsigned char frame_all_storage[NO_OF_FRAME_BUF * MAX_PICT_SIZE];
/* Output buffer. 4-bytes alignment. */
#define MAXPARAM                  5
int *in[MAXPARAM];
int *out[MAXPARAM];
int h_share_mpeg2_rdbuf_size = SHARE_MPEG2_RDBUF_SIZE;
int functionCode;
...
MPEG2VDEC_Handle mpeg2vdec;
/*********************************************************/
/* To do -- fill up the whole input buffer               */
/*********************************************************/
old_ptr = 0; /* head of circular buffer */
MPEG2VDEC_init();
mpeg2vdec = MPEG2VDEC_create(&MPEG2VDEC_TI_IMPEG2VDEC, NULL);
in[1] = &functionCode;                  out[1] = (int *) &out_para[0];
in[2] = (int *) &share_bsbuf_storage[0]; out[2] = (int *) &frame_all_storage[0];
in[3] = &h_share_mpeg2_rdbuf_size;
functionCode = FUNC_START_PARA;
MPEG2VDEC_apply(mpeg2vdec, in, out);          /* decode sequence header */
while (! (decode_out-> end_of_seq) ){         /* not end of sequence    */
  functionCode = FUNC_DECODE_FRAME;
  MPEG2VDEC_apply(mpeg2vdec, in, out);        /* decode one picture     */
  decode_out = (DECODE_OUT *)(out[1]);
  if (decode_out-> outputting) {
    /*********************************************/
    /* To do -- output the frame                 */
    /* starting at location decode_out-> outframe */
    /*********************************************/
  }
  /*************************************************/
  /* To do -- fill the input buffer between        */
  /* old_ptr and decode_out->next_wptr from source */
  /*************************************************/
  old_ptr = decode_out->next_wptr;
} /* while */
MPEG2VDEC_delete(mpeg2vdec);
MPEG2VDEC_exit();
/* End of program */
```

Figure 4. An Example Framework to Interface the Video Decoder

5 Running the Program

This section describes the procedure to build and run the video decoder.

5.1 Build Procedure

To build the code:

1. Compile and assemble individual files with

   cl6x –@cl.cmd

   or use the Code Composer Studio™ (CCS) project file Mpg2vdec.mak.

2. Create the decoder library Mpeg2vdec_ti.lib with

   ar6x @ar.cmd

3. Link the decoder library with the target system.

The code directories contain all the files to compile, assemble, and link the decoder. The CCS project file Mpg2vdec.mak compiles and assembles all the source files, and links them with the linker file Mpg2vdec.cmd to produce an example stand alone executable Mpg2vdec.out. The executable Mpg2vdec.out can be loaded into the DSP and tested with the Windows™ GUI interface program. The object files can also be packaged into the decoder library Mpeg2vdec_ti.lib using the ar6x command and linked to the target system by following the above steps.

5.2 Run the Program

To run the example executable:

1. Load and start the DSP executable Mpg2vdec.out using CCS or evm6xldr.
2. Start the Windows GUI interface program Mplay.exe.
3. In the Mplay.exe program, select File –> Open, and choose an MPEG-2 video file.
4. In the Mplay.exe program, select Control –> Play to start decoding.

If you have Microsoft™ DirectDraw™ software package on your computer, then you would be able to see the decoded video.

5.3 Test and Validation

We have tested the decoder thoroughly with the official MPEG-2 compliance test streams and have verified that the decoder is completely compliant with the requirements. This ensures both the correctness of the algorithm implementation and the quality of the output picture.

6 Memory Requirements and Performance

The section reports the memory requirements and performance of the decoder.

Table 1 lists all data and program memory requirements.

Table 1. Memory Requirements of the Decoder

	Internal Memory	External Memory
Data Memory		
Heap data memory	7552 (7.4 KB)	3022848 (2952 KB)
Stack space data memory	16384 (16 KB)	
Static data memory	10616 (10.4 KB)	
Total	34552 (33.8 KB)	3022848 (2952 KB)
Program Memory		
Total	65504 (64 KB)	26432 (26 KB)

The external memory includes the input bit-stream buffer and the output frame buffers. They are allocated by framework and passed to the decoder as arguments of API MPEG2VDEC_apply.

We have benchmarked the video decoder with several MPEG-2 video streams. The results are shown in Table 2. The decoder software was benchmarked on a C6201 DSP with internal memory configured as mapped mode (cache disabled). The performance was interpreted by the cycle count measured in CCS. Please contact Texas Instruments for the latest performance information.

Table 2. Performance of the Decoder

Test Stream	Information	Format	Bit-rate (Mbps)	Numbers of picture	Performance (MHz)	
Public domain MPEG-2 stream	Mobl_080.m2v. Available from http://www.mpeg.org.	704x576 x25fps	7.629	375	Av. Max.	214 239
DVD test stream #1	Vts_05_1.vob in Panasonic DVD Demonstration Disc.	720x480 x30fps	9.346	10000	Av. Max.	204 255
DVD test stream #2	Vts_40_1.vob in Philips DVD Demonstration Disc.	720x576 x25fps	9.346	1000	Av. Max.	190 220
MPEG-2 compliance test stream	Bitstream gi_9.m2v in MPEG-2 test suite.	720x480 x30fps	14.305	16	Av. Max.	261 267

7 References

1. ISO/IEC 11172-2, *Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbits/s, Part 2: Video (MPEG-1 video standard)*.
2. ISO/IEC 13818-2, *Generic coding of moving pictures and associated audio information, Part 2: Video (MPEG-2 video standard)*.
3. Texas Instruments, *The eXpressDSP Algorithm Standard (xDAIS): Rules and Guidelines*, September 1999 (SPRU352).

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

EXHIBIT J

Equator Software Reference

Datasheet: MAP-CA DSP MPEG-4 Video Decoder

Equator Technologies, Inc.

May 11, 2001

Document Number: SWR.DS.M4VD.2001.05.11

Equator Software Reference
Datasheet: MAP-CA DSP MPEG-4 Video Decoder

May 11, 2001

The MAP-CA digital signal processor was jointly developed by Equator Technologies, Inc., and Hitachi, Ltd.

MAP-CA DSP MPEG-4 video decoder at a glance

The Equator Technologies, Inc. MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package provides an example implementation of MPEG-4 video image decoding on the MAP-CA digital signal processor. This software can be used by developers as a reference for creating their own MPEG-4 decoders for the MAP-CA DSP.

Features

The MAP-CA DSP MPEG-4 video decoder provides a number of important features for software developers:

- implementation as a well-documented set of C source code modules that utilize MAP-CA DSP Media Intrinsics C extensions
- inclusion of highly optimized, re-usable software modules, designed to allow users of the MAP-CA DSP to quickly adopt these modules to their own code base
- examples of efficient and parallel use of resources on the MAP-CA DSP, including
    - DataStreamer DMA controller
    - VLx coprocessor
    - Media Intrinsics C extensions
    - data cache and instruction cache

Benefits

The MAP-CA DSP MPEG-4 video decoder provides easily adaptable software modules for software developers. The benefits include

- substantial savings in development costs
- greatly improved time to market

Performance

The MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code is intended to provide a basis for customer development of MPEG-4 decoder software, and has not yet been fully optimized for performance, nor does it contain comprehensive error handling or other production-level code routines. As such, it does not demonstrate the full potential performance of optimized code running on the MAP-CA DSP.

The MAP-CA DSP is capable of decoding two Simple Profile at Simple Level (SP@SL) MPEG-4 video streams in real time, with sufficient cycles left to perform other tasks, such as the audio and system decoding.

Processor support

The MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package fully supports the MAP-CA digital signal processor.

Development platform support

The MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package fully supports the Shark series of evaluation and development platforms from Equator Technologies.

Functionality

The MAP-CA DSP MPEG-4 video decoder passes all MPEG-4 standard conformance streams. This decoder supports 4:2:0 coded video format and Simple Profile at Simple Level (SP@SL) video streams. The decoder performs DC prediction and AC prediction. The current version of this decoder does not support error checking.

Ordering information

To order the MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package, use part number 098-0208-10.

For more information, contact your nearest Equator sale office:

Corporate Headquarters
Equator Technologies, Inc.
1300 White Oaks Road
Campbell, CA 95008
Phone: (408) 369-5200
FAX: (408) 371-9106
Email: info@equator.com
URL: http://www.equator.com

Western North America
Equator Technologies, Inc.
19782 MacArthur Blvd, Suite 210
Irvine, California 92612
Phone: (949) 260-0974
FAX: (949) 263-0926
Email: info@equator.com

Eastern North America
Equator Technologies, Inc.
571 West Lake Avenue, Suite 12
Bay Head, New Jersey 08742
Phone: (732) 892-5221
FAX: (732) 892-5234
Email: info@equator.com

Japan
Equator Japan KK
Harmony Wing 5F
1-32-4 Hon-cho, Nakano-ku
Tokyo, Japan 164-0012
Phone + 81-3-5354-7530
FAX: +81-3-5354-7540
Email: info@equator.com

Europe
Equator Europe
Les Algorithmes
Batiment Aristote A 06410 Biot 2000 Route Des Lucioles
Sophia Antipolis, France
Phone: +33 (0) 492944716
FAX: +33 (0) 492944717
Email: info@equator.com

To locate your local Equator sales office, or for further information, go to http://www.equator.com

Type style conventions

With the exception of section and subsection headings, the formatting of text in this document follows the following conventions:

Normal descriptive text is presented in Times New Roman font.

Italicized Times New Roman text is used for document titles.

Underlined Times New Roman text is used for emphasis in normal descriptive text.

Any input or output text for any computer program is presented in `Courier New` font. This includes source code, command-line text, and program output.

`Italicized Courier New` text is used for any portion of a path, including individual file names.

`Bold Courier New` text is used for any placeholder for a set of text input or output items for a program.

4

Table of contents

Chapter 1 Overview of the MAP-CA DSP

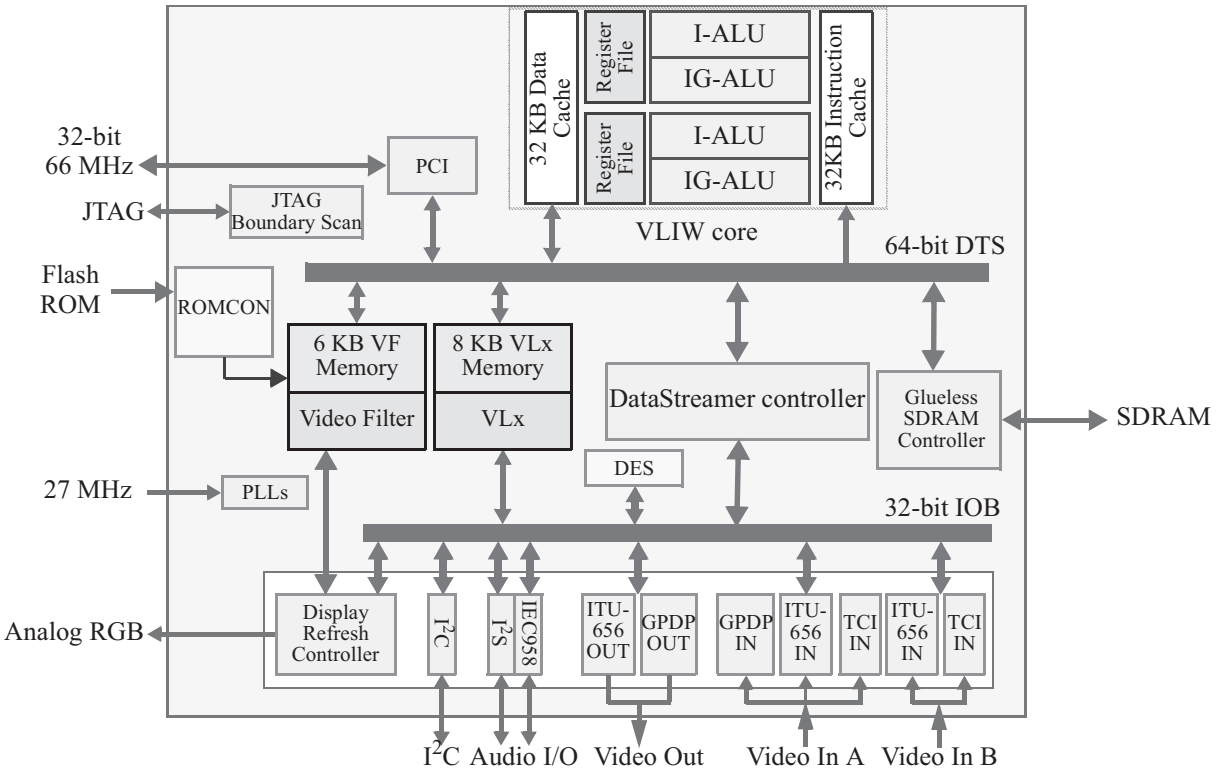


Figure 1-1: MAP-CA DSP block diagram

The MAP-CA digital signal processor is a high-performance, fully programmable processor developed for various multimedia applications that require real-time processing of video, audio, and image data. The MAP-CA DSP is capable of executing up to 30 billion operations per second on 8-bit data. The MAP-CA DSP also integrates several peripheral devices on the chip, such as input and output ports for real-time video and audio data.

1.1 VLIW core CPU

Much of the computational power of the MAP-CA DSP comes from its VLIW (very long instruction word) core central processing unit. In addition to offering high performance on intensive numeric and multi-dimensional matrix operations found in video and signal processing applications, the VLIW core CPU offers a high level of programmability and supports serial and irregular code found in control and data-driven functions. This joint ability to handle serial processing as well as parallel processing avoids the need for a separate host processor for control functions, thus making the MAP-CA DSP a cost-effective solution for consumer and embedded systems.

The VLIW core architecture is a fusion of general-purpose microprocessor and DSP paradigms. Like most modern RISC microprocessors, the VLIW core uses a traditional load/store architecture so that all operations are performed on the data available in the registers. Also integrated on the chip are a 32-kilobyte, 4-way set-associative data cache with true least-recently-used (LRU) cache replacement policy; a 32-kilobyte, 2-way set-associative instruction cache; and full virtual memory support with multiple translation look-aside buffers (TLB). On the other hand, like most modern DSPs, the VLIW core supports a wide array of partitioned and application-specific operations, which can be very efficient when used to operate on multimedia data such as image bitmaps and audio samples.

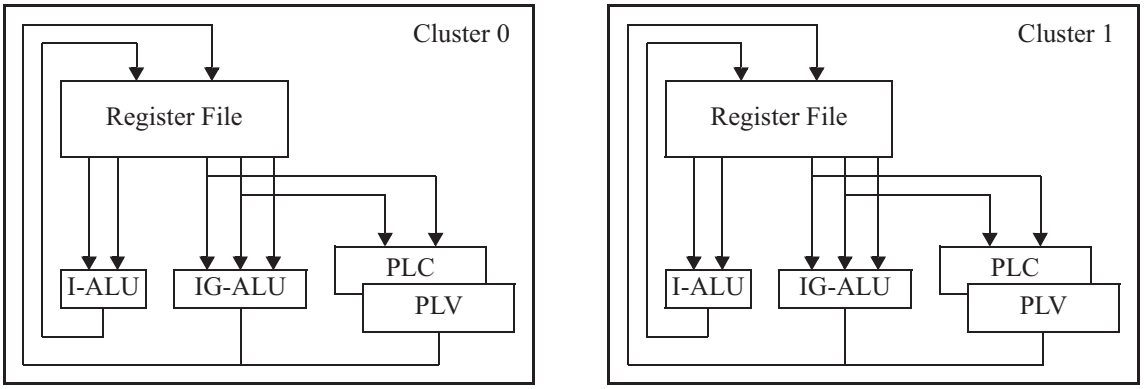


Figure 1-2: Simplified block diagram of two execution clusters in the MAP-CA DSP

The MAP-CA DSP's VLIW core consists of two execution clusters, as shown above. Each cluster contains an integer arithmetic and logic unit (I-ALU), an integer and graphics arithmetic and logic unit (IG-ALU), a register file consisting of thirty-two general purpose 64-bit registers, and a set of predicate and special registers. The I-ALU and IG-ALU on a cluster operate concurrently and independently of one other. They are controlled by separate opcode fields in the instruction word but share a register file and program counter.

The I-ALU executes 32-bit integer arithmetic operations, 32- and 64-bit load and store operations, and branch operations. This unit is primarily used for memory accesses, program flow control, and address calculations. The IG-ALU has a 64-bit adder and shifter, which can be partitioned into 8, 16, 32, or 64 bits for SIMD (single instruction, multiple data) operations. For example, a byte-wise partitioned subtract operation performs eight subtractions, each on an 8-bit partition. In other words, when a partitioned operation is executed, the same base operation (e.g., addition or subtraction) is applied to each partition independently. We thus get eight 8-bit results packed into a 64-bit value for byte-wise partitioned operations.

The IG-ALU is also capable of performing 128-bit integer MAD (mean absolute difference) operations as well as 128-bit vector sum operations. This allows the sum of absolute differences and inner product operations to each be computed with a single instruction, greatly improving the performance of video compression and decompression applications. One IG-ALU can execute eight 16-bit fixed-point MAD operations each cycle.

1.1.1 Pipelining of VLIW core instructions

Because of the way the execution stages are pipelined in the hardware, each execution unit is capable of independently issuing one instruction per cycle. Most integer instructions take only one cycle to complete, but partitioned instructions take multiple cycles for the result to become available and be written to the destination register. However, since a new instruction can be issued on every cycle, even

though the one previously issued has not been completed, the effective number of execution cycles per pipelined instruction can always be one.

1.1.2 Register files

The register file on each cluster consists of thirty-two 64-bit general-purpose registers. Since the I-ALU and IG-ALU on each cluster are connected to the same register file, each register can be accessed as a single 64-bit quantity or a pair of 32-bit quantities. This unique feature is very useful in transposing a 2-dimensional matrix, as discussed in later sections.

1.2 Video and graphics coprocessors

The MAP-CA DSP integrates on-chip coprocessors to remove from the VLIW core tasks that do not make efficient use of the VLIW core's wide data path. One such coprocessor on the MAP-CA DSP is the Variable Length Encoder/Decoder (VLx), a 16-bit microprocessor specifically designed for variable-length decoding and encoding. The Video Filter (VF) is a coprocessor integrated on the MAP-CA DSP to handle the scaling of images at the pixel clock rate.

1.2.1 Variable Length Encoder/Decoder (VLx)

The Variable Length Encoder/Decoder (VLx) is a 16-bit RISC microprocessor, with an instruction set optimized for bit-serial processing. The VLx's dedicated RAM (VLmem) consists of two 2-kilobyte banks for data and a 4-kilobyte bank for instructions.

The primary use of the VLx is the decoding and encoding of variable-length codes (VLCs). The bit-serial nature of the coded data and data dependencies in the control flow of video coding algorithms do not make efficient use of the highly parallel VLIW core. Such algorithms can be implemented more efficiently on serial processors like the VLx, which runs concurrently with the VLIW core. By using the VLx for sequential bit-parsing algorithms, the VLIW core is free to process more computationally intensive parallel code.

1.2.2 Video Filter (VF)

The Video Filter (VF) can scale an image to an arbitrary size using a separable convolution filter with five horizontal and three vertical taps. The VF can take images in YUV color format with 4:2:0 or 4:2:2 chroma sampling and produce output images with 4:4:4 chroma sampling to the Display Refresh Controller.

1.3 DataStreamer DMA controller

In several signal and image processing algorithms, the same sequence of operations is repeatedly performed on subsets of a large set of data. The access pattern of the data is typically sequential and known beforehand. Although most processors with cache memory allow a small subset of the data to be brought into the cache, this scheme does not fit well in signal and image processing algorithms. In the cache memory architecture, copying of the data from external memory to cache memory is initiated only after the data are found to not exist in the cache memory (cache miss). It takes several cycles to copy the

data from memory to cache, and during this time the processor is usually idle and cannot execute other instructions. In other words, on cache-based architectures, the data transfer from external memory to the cache does not overlap with data processing.

Since data access patterns are often regular and well known beforehand, many DSPs have employed a DMA (direct memory access) engine that can transfer a data set between external memory and the cache in parallel with the processing of a previous data set. For example, the Texas Instruments TMS320C80 contains the Transfer Controller, which can transfer data without the processor's intervention. Only the transfer parameters — such as the starting address and the byte to transfer — are set up by the processor at the beginning of the processing of a data set; once the Transfer Controller starts transferring the data, the processor is not involved in individual data transfers.

One of the disadvantages of this sort of DMA engine is that the on-chip memory must have a separate address space from the external memory, because the source and destination addresses specified by the transfer parameters must be unique. This necessitates that the on-chip memory be used only as a scratchpad memory rather than as cache memory.

The MAP-CA DSP provides the best characteristics of both approaches with the DataStreamer DMA controller, a sophisticated 64-channel DMA controller with 8 kilobytes of dedicated on-chip buffer memory. All 64 channels can be allocated at the same time; the DataStreamer DMA controller uses a priority-based scheme to schedule which channel will be active at any given time. In contrast with previous solutions using DMA engines, cache memory does not need to have a separate address space.

A DataStreamer DMA controller transfer is specified as a memory-to-memory copy, with the cache coherency mode specifying different types of transfer. If the source address is specified to be coherent and the destination address is specified to be non-coherent, then data will be read from the cache (assuming the data is already in the cache) and data will be written directly to external memory. Similarly, if the source address is non-coherent and the destination address is coherent, then data will be read directly from external memory and written to the cache. See the figure below for an illustration of the latter example.

Each memory-to-memory transfer requires two channels: one source channel and one destination channel. A buffer allocated from the 8-kilobyte buffer memory is used to hold the data temporarily between the source and destination channels. The source and destination channels, along with the buffer, constitute a data flow called a 'path'. For transfers to or from an I/O device, only one channel is specified, since the other channel is connected directly to the I/O device.

The transfer parameters — such as the starting address and the transfer size — are specified in a data structure called a 'descriptor'. The descriptors are allocated in memory; the DataStreamer DMA controller reads one descriptor at a time as it transfers data. Multiple descriptors can be chained, one pointing to another, so that transfers of data from disjoint memory locations or with different geometry can be performed without interrupting the VLIW core.

Unmodified C code will run through the normal cache mechanism.

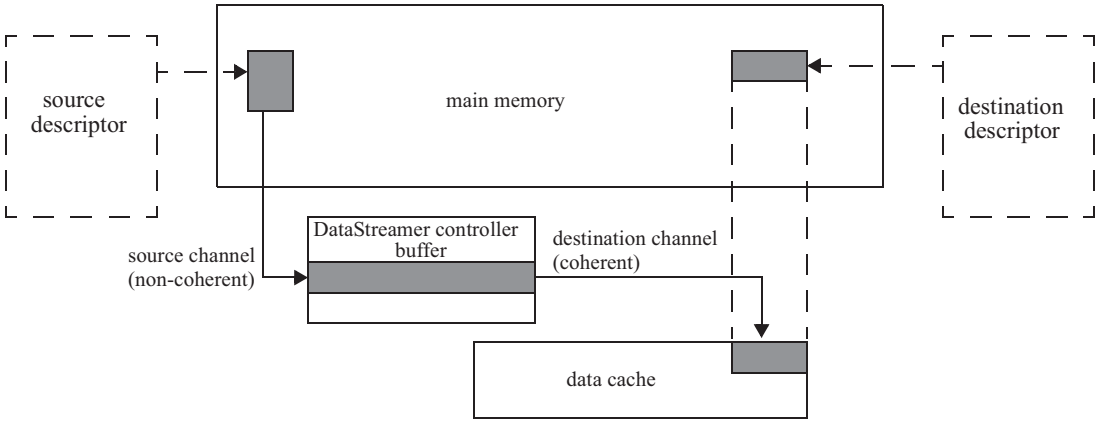


Figure 1-3: Transferring data from memory to the data cache using the DataStreamer DMA controller

1.4 I/O devices

The MAP-CA DSP integrates several I/O devices, including the Transport Channel Interface (TCI) to receive MPEG transport or program streams, a PCI bus interface for connection to another MAP-CA DSP or host system, and an I^2C bus interface. For the display of images on a monitor, the MAP-CA DSP also has a Display Refresh Controller (DRC) that can accept image data in various formats and can multiplex between multiple image streams. The display timing is completely programmable in the DRC, supporting a range from interlaced NTSC scans to high-resolution progressive scans with up to 1280×1024 pixels. The output from the DRC can be sent to the internal DAC (digital-to-analog converter) for display on a high-resolution PC monitor or to an external NTSC encoder chip for display on an NTSC monitor.

1.5 Pipelined nature of decoding on the MAP-CA DSP

The VLIW core, coprocessors, DataStreamer DMA controller, and Display Refresh Controller operate in a pipelined fashion. For example, while the VLx coprocessor is decoding a macroblock, the VLIW core can be processing the previous macroblock. Also, while the Display Refresh Controller is scanning out a frame, the VLx and VLIW core can be decoding the next frame.

Chapter 2 MPEG-4 video decoder

2.1 Overview of the MAP-CA DSP MPEG-4 video decoder

MPEG-4 decoding on the MAP-CA DSP utilizes several functional blocks in parallel. First, the variable-length codes (VLCs) in the input bitstream are decoded by the VLx coprocessor. The decoded information is then passed to the VLIW core, which performs the pixel-processing operations — such as the inverse quantisation, inverse discrete cosine transform (IDCT), and half-pel interpolation — to reconstruct the coded picture. The reconstructed picture can be displayed on an NTSC monitor by the Display Refresh Controller (DRC) and an external NTSC encoder chip. Data transfers between the VLx, the VLIW core, the DRC, and memory are handled by the DataStreamer DMA controller so that the processing units do not wait for data to arrive from memory.

2.1.1 Variable length decoding

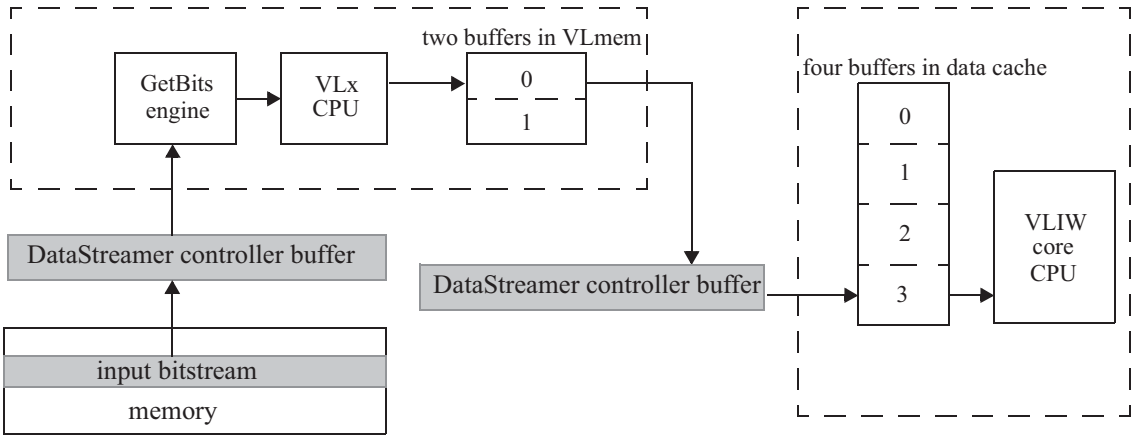


Figure 2-1: Transferring DCT coefficient blocks from the VLx to the VLIW core

The input bitstream arrives at the VLx via a DataStreamer DMA controller buffer. For a standalone configuration, the entire input bitstream is usually loaded into the memory by the host PC. In this case, the starting address is set to the beginning location of the bitstream in the memory, and the transfer size is set to the size of the bitstream. The DataStreamer DMA controller can loop the descriptor, so the bitstream can be repeated indefinitely if desired. For a streaming configuration where a small portion at a time of the input bitstream is passed from the host PC or TCI, the transfer descriptors specify a set of circular buffers in memory.

The VLx receives data from the DataStreamer DMA controller through the GetBits engine (GB). The VLx can request an arbitrary number of bits (up to 16) from the GetBits engine, which supplies the data from its 112-bit input buffer. When the GetBits engine runs out of bits, it reads the next 32 bits of data from the DataStreamer DMA controller buffer.

The VLx has three stages of processing and uses a dedicated region in its coprocessor memory to store the data in each stage. The first stage decodes all of the header symbols. The decoded information is stored in the header information buffer. The VLx then decodes the macroblock symbols and stores them into a macroblock information buffer. The third stage produces DCT coefficients in 8×8 matrix form.

Each macroblock in the MPEG-4 bitstream can contain up to six 8×8 blocks of DCT coefficients. The VLIW core allocates four circular buffers in the data cache to receive the DCT coefficients from the VLx, as shown in the figure above. The DataStreamer DMA controller source channel reads one macroblock of data from one of two buffers in the coprocessor memory and the destination channel stores them to one of four buffers in the data cache. After the transfer of a macroblock is complete, the source and destination channels advance to their next buffer, waiting for a command from VLx to transfer the next macroblock.

The transfer of macroblocks from the coprocessor memory to the data cache is done while the VLIW core is processing the previous macroblock. During the transfer, the VLIW core does not need to check for the arrival of each and every block. When the VLIW core is ready to process the next macroblock, it checks to see if all of the blocks from a macroblock have been transferred.

2.1.2 VLIW core processing

The VLIW core performs all of the pixel-processing operations. Using the header and macroblock information and the DCT coefficients from VLx, the VLIW core performs the inverse quantisation, inverse DCT, motion compensation, half-pixel interpolation, and pixel additions. The VLIW core also uses the DataStreamer DMA controller to load the pixel data from the reference pictures and to store the reconstructed picture.

2.1.2.1 Data flow

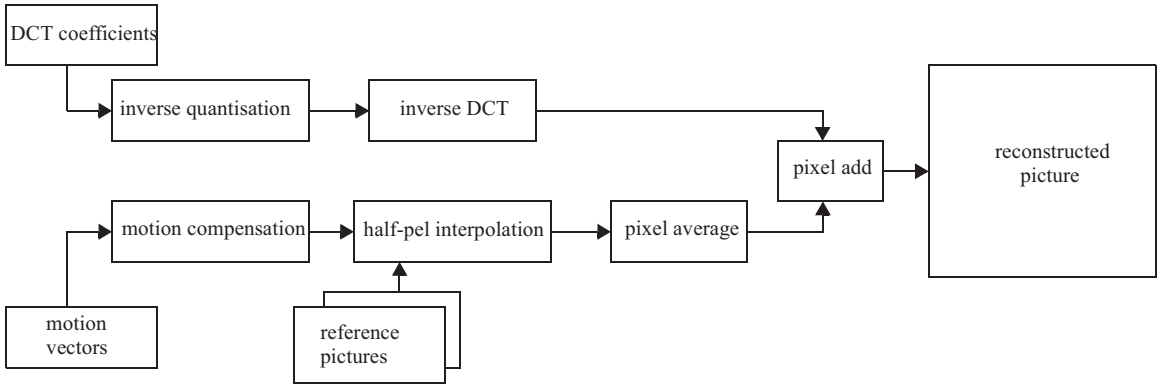


Figure 2-2: Data flow paths in VLIW core processing.

The VLIW core operates on a macroblock unit; this means that a complete set of operations is performed on a macroblock before the next macroblock is processed. There are two paths of data flow in the VLIW core, as shown above. The first path involves the inverse quantisation followed by the inverse DCT. This path produces six 8×8 arrays of 16-bit values, representing the reconstructed macroblock of the motion-compensated residual values. The second path performs motion compensation on the macroblock of reference pictures using the half-pel motion vectors decoded by VLx. This path also involves half-pel interpolation and bidirectional interpolation on previous and future reference macroblocks. The results from the two paths are then combined in the pixel addition stage, producing the reconstructed pixels of a macroblock.

2.1.2.2 Instruction cache management

The MAP-CA DSP MPEG-4 video decoder software has been designed to maximize performance by minimizing instruction cache misses during execution. The MAP-CA DSP MPEG-4 video decoder has been designed to break up the decoding algorithm along frame processing lines, using separate software routines for decoding each type of frame picture: I, P, and B. The routines for these frame types are designed so that each routine can fit into the 32K instruction cache. Alignment of functions is forced to 16K to ensure that the instruction cache is fully utilized.

Instruction cache misses are further reduced through the use of inline expansion of functions. Inlining can increase program performance by removing function call overhead and revealing additional opportunities for parallel scheduling of operations and other optimizations. All of the function calls inside the frame processing routines are inlined.

2.1.2.3 Data cache management

The MAP-CA DSP MPEG-4 video decoder software has been designed to maximize performance by minimizing data cache misses during execution. This is accomplished through the use of a specially designed C language structure, `DCACHE_MAP`, that contains a linear map of all locally used memory. This structure allows the MAP-CA DSP MPEG-4 video decoder to use no global variables; all of the shared variables used in the program are contained in the `DCACHE_MAP` structure.

2.1.3 Video display

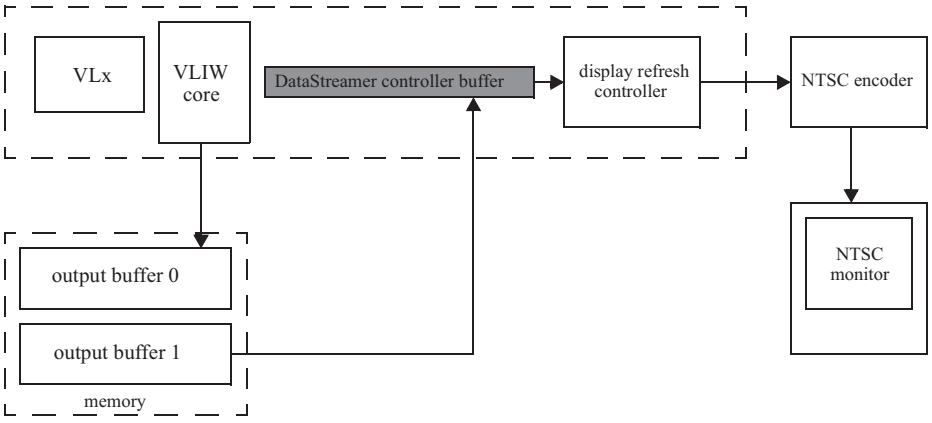


Figure 2-3: Data flow in displaying an image frame to an NTSC monitor through the DRC

The MAP-CA DSP MPEG-4 video decoder supports 4:2:0 coded video output.

An image frame to be displayed on a screen is transferred from the memory to the Display Refresh Controller (DRC) via the DataStreamer DMA controller, as shown above. Since the destination of the DataStreamer DMA controller transfer is an I/O device, only the source channel needs to be set up for this path. When the output device is an interlaced NTSC monitor, the source channel descriptor must be set up to read the scan lines in an interlaced order: all of the active scan lines in the first field are transferred first, followed by all of the active scan lines in the second field. The DRC generates the horizontal and vertical timing information and sends the formatted digital video data conforming to the ITU-R BT.656 standard to an external NTSC encoder chip.

It is also possible for the DRC to route the pixel data to an integrated digital-to-analog converter (DAC) on the MAP-CA DSP for display on a PC monitor. Only the progressive-scan timings are supported in

this mode. The Video Filter can be used to do the necessary conversion from the interlaced source image to a progressive format.

2.1.4 Synchronization

For the correct display of images, a new frame should be displayed only after the all of the scan lines in the current frame have been scanned out. A frame is read from the output buffer to the DRC via a DataStreamer DMA controller buffer. Associated with this buffer is a source channel descriptor that gives the starting address and size of the frame to be transferred. The starting address of a frame is read only at the beginning of the transfer of the frame. Therefore, as long as the transfer of a frame has begun, the starting address of the associated source channel descriptor can be changed without affecting the transfer of that frame. Before queuing a new frame, the core checks to see that the transfer of the previous frame has begun. If the transfer of the previous frame has already begun, the starting address of the source channel descriptor is modified to point to the next frame to be scanned out. This descriptor chains to itself (loops) at the end of the transfer of a frame to begin the transfer of the next frame — the one now pointed to by the new starting address. This scheme ensures that the transfer of a frame to the DRC will be completed before the transfer of the next frame begins, preventing a frame from being prematurely overwritten on the display by a new frame.

It is also necessary to synchronize decoding of frames in the VLIW core with the display. As the DataStreamer DMA controller is passing data from one output buffer to the DRC, the VLIW core is writing data to the other output buffer. Since the MAP-CA DSP MPEG-4 video decoder can decode a frame faster than real-time, it is necessary to synchronize decoding of frames in the VLIW core with the display to prevent the output buffer for the frame currently being scanned out from being overwritten by a new frame. The code running on the VLIW core incorporates a mechanism for waiting for the previous frame to be completely scanned out before the VLIW overwrites the output buffer from which that frame was read. A wait function checks for the current transfer address in the DataStreamer DMA controller; before starting to decode a new frame, the VLIW core calls this function to make sure that the buffer it will decode into is not currently being displayed. If the buffer is currently being displayed, the function will not return until the all of the lines in the frame have been scanned out.

2.2 Current implementation limitations

The current version of the MAP-CA DSP MPEG-4 video decoder passes all MPEG-4 standard conformance streams. This decoder supports 4:2:0 coded video format and Simple Profile at Simple Level. This includes AC prediction and short and long headers.

The current version of this decoder does not support error checking and does not function under the beta version of the VxWorks operating system.

2.3 Example bitstreams and sample application

The MPEG-4 video decoder library is shipped with example bitstreams and a sample application ready to run — once built from the source code — on any MAP-CA DSP platform.

2.4 Accuracy

In order to achieve consistent picture quality across different decoder implementations and technologies, the MPEG-4 standard uses the IEEE standard specification 1180-1990 to limit the inaccuracy of the IDCT outputs.

The MAP-CA DSP MPEG-4 video decoder IDCT implementation meets the IEEE specification in all tests. The overall mean error and mean square error are -0.00007 and 0.00816, respectively. The allowable limits are 0.0015 and 0.02, respectively. The largest mean error and mean square error in any of the 8×8 locations are 0.0029 and 0.0098, respectively, well within the allowed values of 0.015 and 0.06, respectively. The peak error at any of the 64 locations is one in magnitude, also meeting the specification.

2.5 Performance

The MAP-CA DSP is capable of decoding two Simple Profile at Simple Level (SP@SL) MPEG-4 video streams in real time, with sufficient cycles left to perform other tasks, such as the audio and system decoding.

2.5.1 I-frame performance

The first I-frame of the *coastguard.m4v* bitstream (included with the MAP-CA DSP MPEG-4 video decoder) was processed for this performance evaluation. The following table presents cycles needed for each sub-algorithm to process the frame:

IntraDCpred	203,515 cycles
IntraACpred	231,332 cycles
UpdateIntraACpred	154,662 cycles
Iquant	394,667 cycles
idct	255,886 cycles
PixelAddI	158,351 cycles
OutputData	73,017 cycles
PadBorder	18,342 cycles
Miscellaneous	1,631,111 cycles
Total	3,120,883 cycles

The “Miscellaneous” row of this table encompasses all tasks not specifically mentioned in the sub-algorithm performance breakdown (e.g., waiting for data from the VLx).

2.5.2 P-frame performance

Of the first 40 frames of the *coastguard.m4v* bitstream, the largest P-frame (the 27th frame overall, the 26th P-frame) was processed for this performance evaluation. The following table presents cycles needed for each sub-algorithm to process the frame:

ProcSkippedP	not needed for this frame
PadBorder	not needed for this frame
ComputeMV	87,760
LoadRefP	94,602
UpdateIntraACpred	304,503
IntraDCpred	not needed for this frame
IntraACpred	not needed for this frame
Iquant	334,587
idct	234,128
InterpP	166,146
PixelAddP	219,146
PixelAddI	not needed for this frame
OutputData	64,373
PadBorder	15,958
Miscellaneous	1,382,952
Total	2,904,155

The “Miscellaneous” row of this table encompasses all tasks not specifically mentioned in the sub-algorithm performance breakdown (e.g., waiting for data from the VLx).

The MAP-CA DSP is capable of decoding two Simple Profile at Simple Level (SP@SL) MPEG-4 video streams in real time, with sufficient cycles left to perform other tasks, such as the audio and system decoding.

2.6 MPEG-4 video decoder API

The MAP-CA DSP MPEG-4 video decoder is written with an application program interface (API) that aids in the management of caches and buffers, the setting up of the DataStreamer DMA controller and VLx, initialization of data items, and various other low-level tasks.

The MAP-CA DSP MPEG-4 video decoder API uses the `Mpeg4VdecStatus` structure to communicate between API elements. The pointers `frameJustDecoded`, `frameToQueueForDisplay`, `frameToDecode`, `frameToDecodeIntoNext` point to the internal frame buffers of the decoder and are rotated with the API calls. The `dcachePtr` points to the structure containing all local memory references. The `decodeValid` field is set if the decoder determines a new picture exists in the stream. The `frame_no` field is the frame number of the frame to be decoded next. The `result` field is reserved for error flags.

```
typedef struct
{
unsigned char *frameJustDecoded;
unsigned char *frameToQueueForDisplay;
unsigned char *frameToDecode;
unsigned char *frameToDecodeIntoNext;
//    unsigned char *BframeToDecodeIntoNext;    // For PB-mode
//    unsigned char *BframeToQueueForDisplay;   // For PB-mode display
void          *dcachePtr;
int           decodeValid;
int           pb_mode;
int           frame_no;
SCODE         result;
} MPEG4VdecStatus;
```

```
void
MPEG4VideoDecodeOpen(
    void * bitstreamAddr,
    unsigned int bitstreamNumFrames,
    unsigned int bitstreamLength,
    MPEG4VdecStatus *mpeg4VdecStatus);
```

This call does the following.

1. Initializes the data cache pointer to allow for data cache management.
2. Initializes the frame buffer pointers for the double buffering of the input and output frame buffers.
3. Initializes various arrays, tables, and data values.
4. Sets up the DataStreamer DMA controller.
5. Sets up the GetBits data for the VLx.
6. Loads and starts the VLx program.
7. Initializes the `Mpeg4VdecStatus` structure.

```
void
MPEG4VideoDecodeFrame(MPEG4VdecStatus *mpeg4VdecStatus);
```

This call does the following.

1. Decodes an I, P, or B frame based on the next picture header in the bitstream.
2. Outputs the decoded frame into a frame buffer in SDRAM.
3. Rotates the output frame buffer's pointer.
4. Modifies `Mpeg4VdecStatus` appropriately.

Ordering information

To order the MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package, use part number 098-0208-10.

For more information, contact your nearest Equator sale office:

Corporate Headquarters
Equator Technologies, Inc.
1300 White Oaks Road
Campbell, CA 95008
Phone: (408) 369-5200
FAX: (408) 371-9106
Email: info@equator.com
URL: http://www.equator.com

Western North America
Equator Technologies, Inc.
19782 MacArthur Blvd, Suite 210
Irvine, California 92612
Phone: (949) 260-0974
FAX: (949) 263-0926
Email: info@equator.com

Eastern North America
Equator Technologies, Inc.
571 West Lake Avenue, Suite 12
Bay Head, New Jersey 08742
Phone: (732) 892-5221
FAX: (732) 892-5234
Email: info@equator.com

Japan
Equator Japan KK
Harmony Wing 5F
1-32-4 Hon-cho, Nakano-ku
Tokyo, Japan 164-0012
Phone + 81-3-5354-7530
FAX: +81-3-5354-7540
Email: info@equator.com

Europe
Equator Europe
Les Algorithmes
Batiment Aristote A 06410 Biot 2000 Route Des Lucioles
Sophia Antipolis, France
Phone: +33 (0) 492944716
FAX: +33 (0) 492944717
Email: info@equator.com

To locate your local Equator sales office, or for further information, go to http://www.equator.com

EXHIBIT K

SOFTWARE MPEG-2 VIDEO DECODER ON A 200-MHz, LOW-POWER MULTIMEDIA MICROPROCESSOR

Kouhei Nadehara, Hanno Lieske† and Ichiro Kuroda

C&C Media Res. Labs., NEC Corp.
4-1-1, Miyazaki, Miyamae-ku,
Kawasaki 216, Japan
{nade,kuroda}@ccm.CL.NEC.co.jp

†University of Hannover
Schneiderberg 32,
D-30167 Hannover, Germany
lieske@mst.uni-hannover.de

ABSTRACT

This paper presents a low-power, 32-bit RISC microprocessor with a 64-bit "single-instruction multiple-data" multimedia coprocessor, V830R/AV, and its MPEG-2 video decoding performance. This coprocessor basically performs multimedia-oriented four 16-bit operations every clock, such as multiply-accumulate with symmetric rounding and saturation, and accelerates computationally intensive procedures of the video decoding; an 8×8 IDCT is performed in 201 clocks. The processor employs the Concurrent Rambus DRAM interface, and facilities for controlling cache behaviors explicitly by software to speed up enormous memory accesses necessary to motion compensation. The 200-MHz V830R/AV processor with the 600-Mbyte/sec. Concurrent Rambus DRAMs decodes MPEG-2 MP@ML video in real-time (30 frames/sec.).

1. INTRODUCTION

Multimedia signal processing, such as compression and decompression of voice, audio and video, is indispensable in consumer electronic products such as video games, digital video disc players and set-top boxes, as well as personal computers and workstations. Multimedia signal processing is so demanding that it is necessary to incorporate application specific ICs or digital signal processors (DSPs) in addition to a main general-purpose processor.

In progress of processor technology, there is a strong requirement to implement multimedia equipment using software on a general-purpose processor. These systems could be easily compliant to multiple standards by software upgrades. Moreover, it is quite advantageous in system size, power and cost to implement systems' multimedia capability in software without additional signal processing hardware.

Some high-end processors for personal computers and workstations have introduced multimedia-oriented execution units and associated instructions [1, 2]. These processors have already attained sufficient signal processing performance to decode MPEG-2 main profile at main level (MP@ML) bitstreams in real-time [3, 4]. It was not the case, however, in inexpensive consumer electronic products, because low-cost, embedded processors have not introduced multimedia-oriented facilities so aggressively.

For software signal processing in low-cost, low-power systems, a low-power, embedded RISC processor with a 64-bit multimedia coprocessor, named V830R/AV, has been developed. This processor provides high signal processing performance up to 1.6 GOPS by "single-instruction multiple-data (SIMD)-type" parallel operations and a fast 600-Mbyte/sec. Concurrent Rambus DRAM interface.

In this paper, the V830R/AV processor architecture is described first. Next, the processor's signal processing performance is evaluated taking an MPEG-2 video decoding as an example.

2. PROCESSOR ARCHITECTURE

The V830R/AV processor is a new member in the NEC's V800 embedded RISC family [5]. This processor is designed to support real-time signal processing of broadcast-quality video. It integrates a 64-bit multimedia coprocessor, the Concurrent Rambus interface, and 16-Kbyte, 4-way, set-associative instruction and data caches, with the V830-compatible integer execution pipeline (Figure 1). The integer execution pipeline integrates a 1-clock throughput 32-bit integer/fixed-point multiply-accumulator for high-precision signal processing such as audio encoding and decoding [6].

The processor can issue two instructions simultaneously when the current and the next instructions can be issued to multimedia and integer execution units, respectively. This asymmetric two-way superscalar capability makes maximum use of both execution units, while simplifying an instruction decoder.

The V830R/AV processor is fabricated in a 0.25μm, 4-level metal layer CMOS technology, resulting in 3.9 million

0-7803-4428-6/98 $10.00 © 1998 IEEE

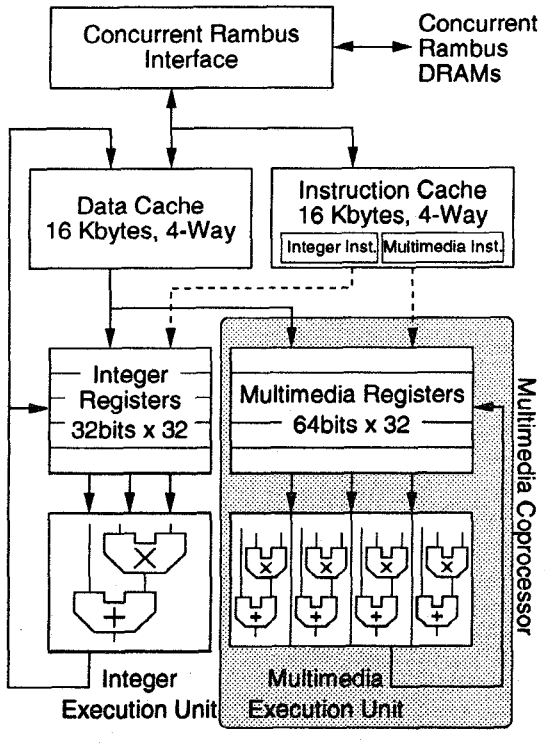


Figure 1: Processor Core Block Diagram.

transistors. The processor core clock frequency is 200 MHz at 2.5V power supply. It dissipates less than 2.0 W.

2.1. Multimedia Coprocessor

The multimedia coprocessor performs SIMD parallel operations on eight 8-bit, four 16-bit and two 32-bit packed data in thirty-two 64-bit multimedia registers. This large multimedia register file eliminates instructions for saving or restoring intermediate results. The coprocessor mainly supports four 16-bit data type, which is sufficient precision for video applications. Table 1 shows the multimedia instruction set supported by the coprocessor, called MIX2 (Multimedia Instruction eXtension 2). The multimedia execution unit is fully pipelined, and has 1-clock throughput and fixed 4-clock latency for simplicity in both the chip design and software programming.

2.2. Cache Management

In embedded processors, which cannot afford large multi-level caches, cache misses impose heavy penalties on their performance. To make matters worse, caching strategy is not very effective on multimedia data, which usually have very large working set and less temporal locality. Therefore, this processor incorporates mechanisms to control cache behaviors explicitly by software according to memory access

Table 1: MIX2 Instruction Set.

Instruction	8-bit ×8	16-bit ×4	32-bit ×2	64-bit ×1
Arithmetic				
Vector Addition	○	○	○	
Vector Subtract	○	○	○	
Vector Multiply		○		
Vector Multiply Accumulate		○		
Scalar Addition		○		
Scalar Subtract		○		
Scalar Multiply		○		
Scalar Multiply Accumulate		○		
Compare				
Vector Maximum		○		
Vector Minimum		○		
Scalar Maximum		○		
Scalar Minimum		○		
Format Conversion				
Pack	○	○		
Interleave	○	○	○	
Logical				
And				○
Or				○
Xor				○
Not				○
Shift				
Logical Left		○	○	○
Logical Right		○	○	○
Arithmetic Right		○	○	○
Motion Estimation				
Absolute Difference		○		
Shift and Add			○	

characteristics of multimedia applications, in addition to general automatic mechanisms to reduce miss penalties.

For efficient software execution, it is necessary to reduce both cache miss counts and miss penalties per cache miss. First, to reduce cache miss counts, the instruction and data caches have a "freeze" attribute bit in each 64-byte cache line to suppress cache line replacements. When this bit is set, the current data reside in the cache line; the data stored in a "frozen" line is accessible without a cache miss.

Second, to reduce apparent penalties per cache miss, this processor employs a non-blocking data cache which allows one pending miss. The processor deals with a data cache miss in parallel to instruction execution, when there is a enough time between a load instruction which caused the miss and an instruction which refers to the load result. Third, the processor has a "preload" instruction to fill the specified data to the data cache prior to load instructions. This instruction further increases the possibility to handle cache misses in background and prevents the processor pipeline from stalling.

2.3. External Memory

The processor employs a fast memory interface called Concurrent Rambus [7, 8]. The Concurrent Rambus interface

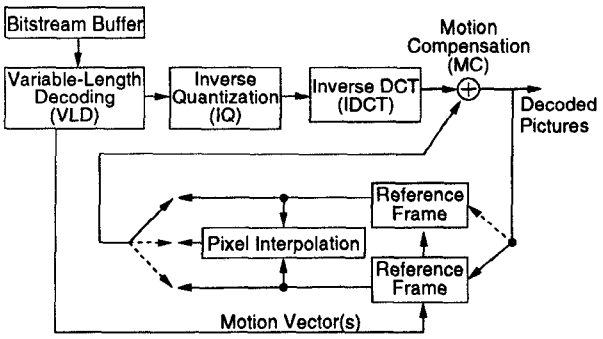


Figure 2: MPEG-2 Video Decoding Process.

has data bandwidth up to 600 Mbyte/sec., because it transfers access commands and data through the 8-bit bus on both edges of fast 300-MHz clock. This interface can provide twice the bandwidth than the synchronous DRAM interface with a 32-bit, 66-MHz bus, with much less pin count. This fast memory interface also reduces cache miss penalties per miss.

3. SOFTWARE MPEG-2 VIDEO DECODING

An MPEG-2 video software decoder for the V830R/AV processor has been developed, based on the sample decoder implementation, *mpeg2decode*, from MPEG Software Simulation Group [9]. The original decoder compiled by a C compiler without MIX2 instructions, takes 891 M instructions/ 1.5 G clocks to decode 30 frames of a 4 Mbps MPEG-2 MP@ML bitstream; it is 7.5 times the processor performance. Therefore the decoder has been optimized by rewriting the macroblock layer and subsidiary functions in the MIX2 assembler.

The MPEG-2 video decoding process mainly comprises of four procedures as shown in Figure 2; variable length decoding (VLD), inverse quantization (IQ), inverse discrete cosine transform (IDCT), and motion compensation (MC).

3.1. VLD and IQ

A variable-length codeword is extracted from the current bit position of an MPEG-2 video bitstream in VLD. Theoretically, variable-length codes are decoded one by one, because it is impossible to determine each code length in advance.

A *doubleword shift* instruction in the integer execution unit can efficiently extract a 32-bit word containing a current variable-length cordword even when the codeword is stored across two bitstream buffer entries (Figure 3). IQ is performed every time the codeword is decoded.

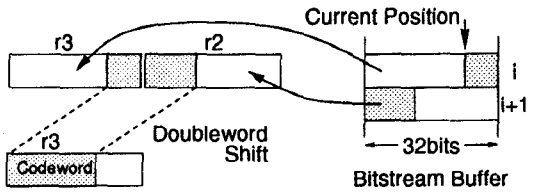


Figure 3: VLD Using Doubleword Shift Instruction.

The software decoder "freezes" data cache lines correspond to variable-length code tables to avoid cache misses during code table lookups. For the effective use of the 16-Kbyte data cache, small multi-level code tables are employed instead of a large single-level code table. Each element is packed into bit fields for further table compaction.

3.2. IDCT

Since IDCT is a multiplication intensive procedure with high data parallelism, this procedure is very appropriate to the SIMD coprocessor. SIMD multiply-accumulate instructions are very suitable for fast IDCT. An 8×8 2D IDCT is first decomposed into two consecutive 1D 8-point IDCT, and four 1D 8-point IDCTs are performed in parallel by SIMD multiply-accumulate instructions with symmetric rounding. This instruction contributes to simple and fast implementation of an IDCT compliant to the IEEE1180 standard [10].

Thirty-two 64-bit registers in the coprocessor are large enough to hold all the 8×8 elements during an IDCT. They eliminate instructions for saving and restoring intermidiate results. As a result, an 8×8 2D IDCT is performed in 272 instructions/201 clocks at the V830R/AV processor, which is 6.8 times faster than the original.

3.3. MC

A macroblock is reconstructed by adding difference signals, which are the IDCT outputs, and reference block(s) pointed by motion vector(s). There are heavy cache miss penalties in MC, because of extensive read and write accesses to external frame buffers in main memory.

Figure 4 shows how the preload instruction is used to mitigate cache miss penalties in reading reference macroblocks. There is not sufficient time to preload a current reference macroblock shown in solid lines between a macroblock's motion vector decode and the reference macroblock access, since motion vectors and difference signals are successive in a bitstream.

Therefore, preload instructions are issued for the next MC. Assuming that motion vectors are similar in adjacent macroblocks, the center of the next reference macroblock

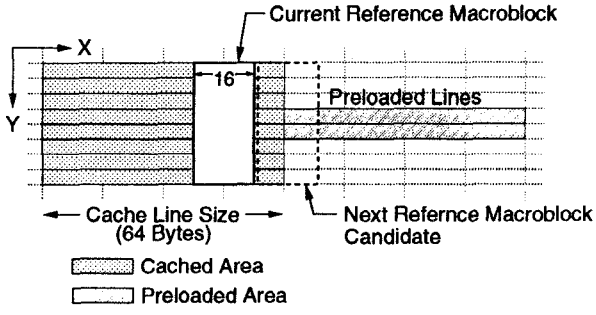


Figure 4: Preloading Reference Macroblocks.

area shown in dashed lines is considered to be a prospective candidate for preloading. Therefore, preload instrcutions are issued to the hatched area.

The preload instruction is also effective to allocate cache lines corresponding to a current block in a reconstructed frame to reduce write cache miss penalties. These explicit cache control greatly reduces a clock count necessary in MC to 14% compared to the original version.

3.4. Implementation Results

Figure 5 shows clock and instruction counts necessary to decode a 30 frame, 4 Mbps, MPEG-2 MP@ML video sequence for the original and optimized decoders. After optimizing software decoder using MIX2 instruction set and explicit cache control. clock and instruction counts are reduced to 196.7 M and 168.2 M, respectively. This result shows the 200-MHz V830R/AV has enough performance to decode MPEG-2 MP@ML video (720x480 pixels, 30 frames/sec.) in real-time.

4. CONCLUSION

The software MPEG-2 video decoder implemented on the low-power, low-cost, RISC microprocessor with a 64-bit multimedia coprocessor, V830R/AV, is presented. This processor's multimedia instructions, which perform parallel operations on multimedia registers, reduce the dynamic instruction count for decoding 30 frames of MPEG-2 video from 891 million to 168.2 million. In addition, cache line freezing and cache preloading mechanisms reduce the clock count to 196.7 million. As a result, the 200-MHz V830R/AV processor with the Concurrent Rambus DRAMs achieves software MPEG-2 MP@ML video decoding in real-time. It shows that this processor can provide sufficient signal processing performance to deal with broadcast-quality video in low-power, low-cost, consumer products.

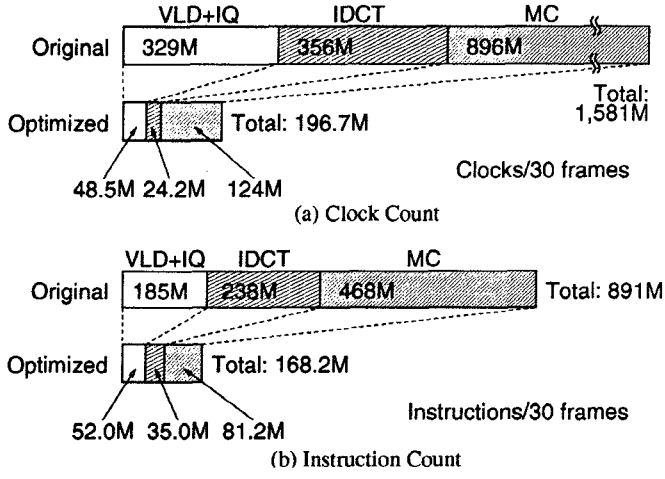


Figure 5: Software Decoder Performance.

5. ACKNOWLEDGMENTS

The authors would like to thank Dr. Kazunori Ozawa, Dr. Takao Nishitani, and Mr. Takashi Miyazaki of the NEC R&D group for their continous support.

6. REFERENCES

[1] Marc Tremblay et al., "VIS Speeds New Media Processing," IEEE Micro, pp. 10–20, Aug. 1996.

[2] Linley Gwennap, "Intel's MMX Speeds Multimedia," Microprocessor Report, Vol. 10, No. 3, pp. 1, 6–10, Micro Design Resources, Mar. 5, 1996.

[3] ISO/IEC 13818-2, "Information Technology — Generic Coding of Moving Pictures and Associated Audio," 1995.

[4] Masao Ikekawa et al., "A Real-Time Software MPEG-2 Decoder for Multimedia PCs," Proc. of Int'l Conf. on Consumer Electronics, pp. 2–3, Jun. 1997.

[5] Tomohisa Arai et al., "Embedded Multimedia Superscalar RISC Processor with Rambus Interface," IEEE Hot Chips Symposium IX, Aug. 1997.

[6] Kouhei Nadehara et al., "Low-Power Multimedia RISC," IEEE Micro, Vol. 15, No. 6, pp.20–29, Dec. 1995.

[7] Masaki Kumanoya et al., "Advances in DRAM interfaces," IEEE Micro, Vol. 15, No. 6, pp.30–36, Dec. 1995.

[8] Allen Roberts, "Direct Rambus: The New Memory Standard," Conference Materials of Microprocessor Forum, Oct. 1997, also seen at Rambus WWW page: ⟨URL:http://www.rambus.com/ html/mp_forum.html⟩.

[9] MPEG Software Simulation Group WWW page: ⟨URL:http://www.mpeg.org/MSSG/⟩.

[10] "IEEE Standerd Specifications for the Implementations of 8×8 Inverse Discreate Cosine Transform," Std 1180-1190, Dec. 1990.

EXHIBIT L

Design of an MPEG-2 Codec

Konstantinos Konstantinides, Cheng-Tie Chen, Ting-Chung Chen, Hown Cheng, and Fure-Ching Jeng

In 1988, the International Organization for Standardization (ISO) established the Moving Pictures Experts Group (MPEG) with the mission to develop standards for the coded representation of moving pictures and associated audio for digital storage media. The work of the MPEG group led to the development of several ISO standards, including ISO 11172, known as MPEG-1 [1]; ISO 13818, known as MPEG-2 [2]; and ISO 14496, known as MPEG-4 [3].

In recent years, digital video has become a key ingredient in entertainment, digital broadcasting, education, and business. The adoption of the MPEG standard offered consumers a new generation of products, such as DVD players, digital TV, personal video recorders, and DVD recorders. Among the key components for all these new products are integrated circuits (ICs) for MPEG-2 encoding and decoding. While the first generation of MPEG ICs [4] offered playback-only capability, the second generation of MPEG-encoding solutions [5], [6] allowed for a new class of affordable digital video recording products. These ICs integrated complete MPEG-2 video encoders and decoders (codecs); however, a complete audio/video (a/v) system would still require additional hardware for audio processing (encoding and decoding) and for multiplexing or demultiplexing the audio and video streams [7].

The third generation of MPEG ICs [8], [9] pushed system integration even further by integrating both audio and video encoding and decoding into a single chip. In addition to real-time audio and video coding, these ICs provide flexible host interfaces, programmable support for a/v multiplexing and demultiplexing, and other functions, such as video pre- and post-processing of video data, and on-screen display (OSD). These features increase chip complexity and go far beyond the mandatory specifications of the ISO standards, but they are essential for the development of cost-effective, MPEG-based consumer appliances.

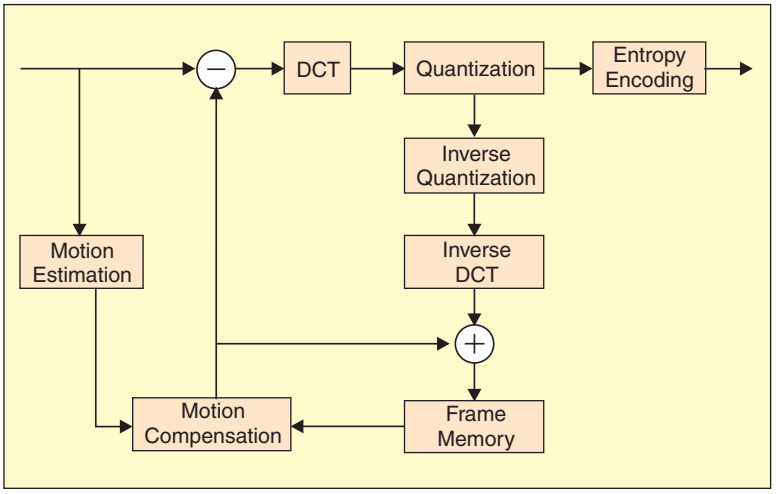
In this article, we review the design of a single-chip MPEG-2 a/v codec, covering the design process from the MPEG specifications and system requirements to the final design. After a brief overview of MPEG-1 and MPEG-2 standards, we examine the system requirements using as an example a universal serial bus (USB)-based MPEG-2 real-time digital video recorder. Finally, we present in more detail the hardware and software architecture of a specific MPEG a/v codec.

MPEG Overview

The MPEG standards are published in several parts, covering such areas as video, audio, systems, and compliance

1053-5888/02/$17.00©2002IEEE

testing. The video and audio parts of the MPEG standard specify the coded representation of the video and audio data (which are referred to as elementary streams) and a *decoding* process for the reconstruction of MPEG-coded bit streams. The systems part specifies how audio, video, and ancillary data can be multiplexed and provides guidelines for buffer management. Even though it is beyond the scope of this article to provide a detailed explanation of MPEG, in this section, we do provide a brief review of the basic principles so that a reader with little or no prior exposure to the standard can have a better understanding of the core requirements for the design of an MPEG-2 processor.



▲ *1. Block diagram of an MPEG encoder.*

MPEG Video

As mentioned before, MPEG does not define the encoding process. Based on the specifications of the coded bit stream and the requirements of the decoder, Fig. 1 shows the functions that need to be executed by a typical MPEG-1 or MPEG-2 video encoder [4].

MPEG uses a variety of tools for video coding, including:

▲ block-based discrete cosine transform (DCT) and quantization;
▲ block-based motion estimation and compensation;
▲ entropy coding for the lossless compression of motion vectors (MVs) and the quantized DCT coefficients.

MPEG defines three main types of coded pictures:

▲ Intra-pictures (I-pictures) are compressed without reference to any other pictures in the coded bit stream.
▲ Predicted-pictures (P-pictures) are coded using motion-compensated prediction from past I-pictures or P-pictures.
▲ Bidirectionally predicted pictures (B-pictures) are coded using motion-compensated prediction from either past and/or future I-pictures or P-Pictures.

I-pictures provide fast random access, but B-pictures yield the best compression. An MPEG stream can consist of only I-pictures or a combination of I,P, and B pictures. The output bitstream is divided into a group of pictures (GOPs). Each GOP has at least one I-picture and must end with either an I- or P-picture. A typical GOP has 15 pictures.

While the core of MPEG-2 video coding is the same as in MPEG-1, MPEG-2 differs in two main areas: a) it extends the nonscalable syntax of MPEG-1 with additional support for the coding of interlaced signals, and b) it provides a scalable syntax, which allows for a layered coding of the video stream.

The MPEG committee understood from the beginning the importance of cost-effective integration of the MPEG standards into silicon. Hence, to assist product development, it decided to use a tool-kit approach. The MPEG-2 syntax is divided into profiles and levels, which can be considered a collection of tools that satisfy the specific requirements of major applications. For example, the Main Profile at Main Level (MP@ML) addresses the needs of consumer applications. Table 1 shows the most common nonscalable profiles and levels in MPEG-2 (see "Digital Video Resolutions" for more details on video resolutions).

In addition to the profiles and levels shown in Table 1, MPEG-2 defines the MPEG-2 4:2:2 Profile@ML. Here the tools address the needs of professional video production environments, where a video stream may be com-

Table 1. Nonscalable Profiles and Levels in MPEG-2 Video Coding.

Levels		Profiles (Nonscalable)	
		Simple 4:2:0 (I,P)	**Main 4:2:0 (I,B,P)**
High	Max resolution/rate (Hz)	N/A	1920 × 1152/60
	Bit rate (Mbit/s)	N/A	80
High-1440	Max resolution/rate (Hz)	N/A	1440 x 1152/60
	Bit rate (Mbit/s)	N/A	60
Main	Max resolution/rate (Hz)	720 × 576/30	720 × 576/30
	Bit rate (Mbit/s)	15	15
Low	Max resolution/rate (Hz)	N/A	352 × 288/30
	Bit rate (Mbit/s)	N/A	4

The adoption of the MPEG standard offered consumers a new generation of products, such as DVD players, digital TV, personal video recorders, and DVD recorders.

pressed and decompressed across several generations. The 4:2:2 Profile allows for higher bit rates (up to 50 Mbit/s), 4:2:2 chrominance subsampling, and higher precision in DCT coding. Most consumer-grade MPEG encoders support the main and simple profiles at the main and low levels.

From a chip implementation point of view, MPEG leaves chip architects with a number of areas where they can make design tradeoffs that affect not only implementation cost but also video quality. Such areas include the following.

▲ *Video preprocessing and postprocessing.* Since video encoding can be done in multiple resolutions, there is a need for a preprocessing unit for scaling, filtering, and color subsampling (from 4:2:2 to 4:2:0) of the input video. Similarly, during video playback, there is a need for chroma up-sampling (from 4:2:0 to 4:2:2), picture up-scaling, and other post-processing functions, such as the removal of blocking artifacts.

None of the algorithms for these functions is defined in the MPEG standard. For example, for scaling, designers have a variety of algorithms to choose from: from simple pixel removal or duplication to spline-based interpolation.

▲ *Motion estimation algorithm.* Motion estimation is the most compute-intensive operation in MPEG, and there is no limit on the choices of algorithms and tradeoffs a designer can make [4]. For example, full-search, block-based motion estimation yields better quality, but requires more hardware and higher memory bandwidth than hierarchical methods.

As another example of design tradeoffs in this area, many consumer MPEG-2 encoders do not support field prediction. They support only frame prediction. In frame prediction, during motion estimation, the two fields from an interlaced TV input are treated as a single frame (as done in MPEG-1). In field prediction, each field is treated separately, and MVs can point to a field in another reference frame or to a field in a current frame. Field prediction yields better quality for interlaced input, but requires additional compute power and memory bandwidth.

▲ *Bit allocation and rate control.* The rate control (or bit allocation) algorithm used during encoding can be a major quality differentiator among MPEG encoders, specially when coding at low bit rates (below 3-4 Mbit/s) See "Rate Control Considerations" for more details on rate control algorithms.

Digital Video Resolutions

The specifications for a digital video signal are defined by the International Telecommunications Union (ITU) Recommendation BT.601 [17]. An ITU-601 image (also referred to by video professionals as full D1 image) is defined as 720 pixels x 480 lines at 60-Hz interlaced (NTSC) or 720 pixels x 576 lines at 50 Hz interlaced (PAL). (Many consumer devices support only 704 pixels of horizontal resolution; this is also referred to as full D1.)

A color ITU-601 image has one luminance (Y) and two chrominance color components (Cb and Cr). Each of the chroma frames has 360 pixels x 480 lines in NTSC or 360 pixels x 576 lines in PAL. This is referred to as 4:2:2 subsampling. The MPEG-1 and MPEG-2 Main Profile at Main Level use a 4:2:0 subsampling format, where the resolution of the chroma components is half of the luminance resolution in both the horizontal and vertical dimensions. MPEG-1 also defines the source input format (SIF), with a luminance resolution of 352 x 240 pixels at 30 Hz or 352 x 288 pixels at 25 Hz. Other popular formats are half-D1 (352 x 480 at 30 Hz), 2/3-D1 (480 x 480 at 30 Hz), the common interchange format (CIF) (352 x 288), and quarter-CIF (QCIF) (176 x 144). Half-D1 and 2/3-D1 resolutions are used mostly for coding video at rates below 3 Mbit/s. CIF and QCIF resolutions are used mostly for video transmissions at rates below 1.5 Mbit/s.

ITU Recommendation BT.656 [18] defines another popular interface for transmitting digital video. Instead of the conventional horizontal and vertical synchronization video signals (HSYNC and VSYNC) used by BT.601, BT.656 uses unique timing codes that are embedded within the video stream. This reduces the number of wires required for a BT.656 video interface.

Audio Coding

Digital audio is an integral part of any audio-visual system. There are two main compression schemes for digital audio: MPEG audio and Dolby Digital, also known as AC-3 [10]. Both schemes combine transform-based coding and psychoacoustic models to remove audio information that is perceptually irrelevant.

MPEG audio is an integral part of the MPEG standards [1], [2], [4] and defines three layers of compression. Each successive layer offers better compression performance, but at a higher complexity. In recent years, Layer III (MP3) has been the coder of choice for portable audio players and for distributing audio files over the Internet; however, most a/v applications require only Layer II encoding support.

Dolby Digital was developed and is licensed by Dolby Laboratories. This was the first compression scheme to support the coding of up to six channels. Dolby Digital is

mandatory for HDTV transmission and DVD authoring in North America.

System Specifications

While the MPEG systems standards specify generic guidelines for the multiplexing of audio and video elementary streams into program streams, most of the implementation details are application specific. For example, MPEG specifies the syntax of a program stream packet, which combines a packet header with other audio, video, or data packets, but puts no constraints on its size. However, the DVD format restricts the size of a program packet to 2048 bytes. Most MPEG-based consumer appliances provide support for the following application-specific specifications: video CD (VCD), super VCD (SVCD), and DVD.

VCD [11] was developed by JVC, Matsushita, Philips, and Sony. It uses MPEG-1 audio and video to compress a/v material into CDs. SVCD [12] (or Chaoji VCD) is based on VCD and allows for MPEG-2-based recordings on low-cost CDs [7]. SVCD uses MPEG-2 and supports higher picture resolutions (up to 2/3-D1) and higher bit rates (up to 2.6 Mbit/s) than VCD. SVCD was developed by the Chinese Ministry of Information, and it is used mostly in Asia. Both VCD and SVCD support only two channels of audio.

The video DVD format [13], developed by members of the DVD Forum, provides the best possible video and audio quality. Video is recorded in full D1 resolution and a DVD-video disc may have up to eight audio tracks. Audio can be in either linear PCM, Dolby Digital, or MPEG (Layer II). Audio is also sampled at 48 kHz, compared to the 44.1 kHz used for audio CDs. Most DVD players can play VCD discs, and many can play SVCD discs as well. Table 2 shows the main characteristics of the VCD, SVCD, and DVD formats.

Design Example

In every video recording or playback system, MPEG ICs need to be connected to a variety of other devices, such as memory, NTSC/PAL video encoders and decoders, audio converters, system controllers, and recording media. In this section we will review the most common system-interface requirements for an MPEG codec IC. As an example, we look into the system design of a USB-based digital video recorder and player (DVR), targeted as a peripheral for computers with a USB port and their own recording media.

Fig. 2 shows a block diagram of the design. The recording device can be either the computer's hard disk or a recordable CD or DVD drive. The DVR consists of an MPEG a/v codec, memory, a USB controller, an optional TV tuner (for TV recording), and analog to digital (A/D) and digital to analog (D/A) audio and video converters.

Encoding

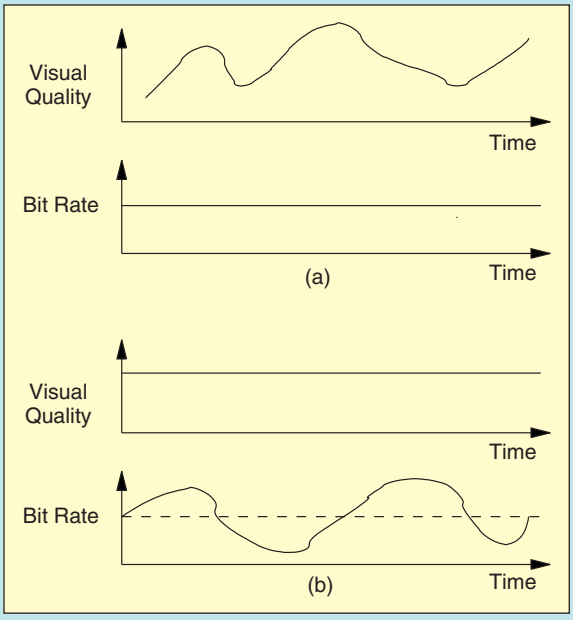
Analog video from either the TV tuner or another video source, such as a videocassette recorder (VCR) or camcorder, is demodulated using the NTSC/PAL decoder and is passed to the MPEG codec. The setup and control of the NTSC/PAL decoder is handled by either the MPEG codec or the USB controller using an inter-IC bus (I^2C). This is a two-wire serial bus (one wire for clock

Rate Control Considerations

Rate control or bit allocation is one of the most important differentiating factors among MPEG video encoders. A good rate control algorithm maintains the target bit rate while it preserves the best possible video quality.

There are two types of rate control: constant bit rate (CBR) control and variable bit rate (VBR) control. In CBR, the main goal is to maintain the average bit rate within each GOP. Thus, the average bit rate remains constant, but the output quality varies depending on the complexity of the input source [19], [20] (see part (a) in the figure below). CBR works quite well when the variations in the input source are gradual, but, it will yield poor perceptual quality when there are sudden changes, such as increased motion or scene cuts.

In VBR, the goal is to maintain constant picture quality. Hence, a VBR algorithm will try to allocate more bits in the segments with high activity (or motion) than in the segments with low activity. This results in better picture quality, but video coded using VBR has higher fluctuations in the average bit rate than video coded using CBR (see (b) in the figure below). Examples of two-pass and one-pass-VBR algorithms can be found in [20] and [21].



▲ *CBR versus VBR rate control in MPEG. (a) CBR, (b) VBR.*

and the other for data) developed by Philips [14]. Input video can be overlayed by on-screen graphics and passed back to the NTSC/PAL encoder for video loop-back. Digital video to the MPEG IC can be either in ITU-R 601 or ITU-R 656 video formats. (See "Digital Video Resolutions" for details on video formats.)

Analog audio is digitized by an audio A/D converter and linear PCM data is transferred to the codec using an Inter-IC Sound (I^2S) interface [15]. This is a three-wire, serial, bus interface (clock, data, and word select) designed by Philips, and it is considered the defacto standard for the inter-connection of audio devices.

The audio and video are compressed by the MPEG codec, and the compressed bit stream is transferred to the recording device using the USB controller and the USB bus. The USB controller serves also as the system controller of the DVR. It downloads microcode and recording or playback parameters and commands from the host computer to the MPEG codec. It communicates with the MPEG codec via the codec's host interface bus. Most microcontrollers have interfaces that resemble either an Intel-like or Motorola-like microprocessor. These interfaces include address and data buses and special control signals for data read and writes, direct memory transfers, and interrupts.

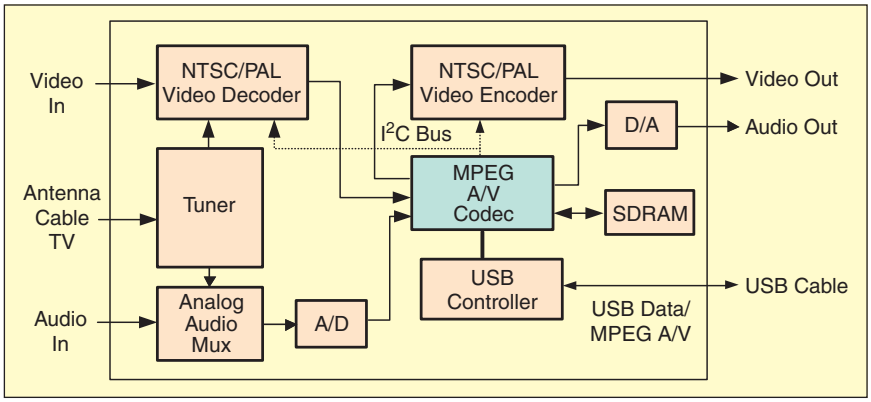
Decoding

During decoding, using the USB controller, compressed a/v data are transferred from the playback drive to the MPEG codec, where they are demultiplexed and decoded into audio and video streams. Digital video is transferred via the output digital video bus to an NTSC/PAL encoder for playback on a monitor or TV. Digital LPCM audio is transferred via an I^2S bus to a digital to analog audio converter for audio playback. As before, the settings of the NTSC/PAL encoder are controlled using the I^2C interface.

In summary, the architects of an MPEG codec have to support a variety of system interfaces so that their devices can be connected without expensive glue-logic to as many peripheral devices as possible. Since each additional interface adds both pins and control logic, the designers have to trade off between market requirements and design cost. The design of a specific codec is described in the next section.

Table 2. Key characteristics of VCD, SVCD, and DVD Formats.

Disc Format	Video Stream	Audio Stream	Picture Size	Max. Rate (Mbit/s)
VCD	MPEG-1	MPEG-1	SIF	1.5
SVCD	MPEG-2	MPEG-1	2/3 D1	2.6
DVD	MPEG-2	MPEG-1, Dolby, LPCM	D1	10.08



▲ *2. Block diagram of USB-based digital video recorder and player.*

Design of an MPEG-2 Codec

In this section we describe the design and architecture of the CS92288 MPEG-2 a/v codec from Cirrus Logic. This is a half-duplex (that is, it either encodes or decodes), MPEG-2, MP@ML, audio and video codec with programmable system multiplexor and demultiplexor and OSD.

Overall Architecture

Fig. 3 shows the major functional units of the codec. These units include: the RISC microcontroller, the video interface unit (VIU), the audio interface unit (AIU), the video engine unit (VEU), the audio engine (DSP), the host interface unit (HIU), and the SDRAM control unit (DCU)

All blocks intercommunicate using two major buses: a 64-bit wide data bus (D-Bus) and a 16-bit wide register bus (R-Bus). In addition to the above seven major blocks, the I^2C CTRL block provides control for external NTSC/PAL video encoders and decoders. The PLL block provides clocking for all internal blocks and also external memory. Given an input 27-MHz clock, all internal components operate at 108 MHz. A separate audio PLL can be used to provide an output master clock to external audio A/D and D/A converters.

RISC Microcontroller

This is an embedded, programmable, 32-bit ARC RISC processor [16]. It performs multiplexing of audio and video elementary streams into program or transport streams and demultiplexing of MPEG program or transport streams. The programmability of the RISC core allows the codec to support a variety of recording formats.

A key operation of the ARC microcontroller is to control all subunits of the codec. All these units operate independently; however, from time to time, they may request special service requests from the central controller. Then,

depending on the priority of the service requests (also known as interrupts), the ARC processor has to switch between tasks, a process known as context switching.

In traditional context switching, the processor performs the following tasks: a) saves the current state of all CPU resources (such as register values), b) switches to the new task, and c) upon completion, restores the state of the CPU so that it can continue executing the original task.

One solution for context switching is to use a single RISC processor running a real-time operating system (RTOS). In this case, context switching time is very important, and unless the RISC processor is very powerful it is very difficult to guarantee a predictable behavior for time-critical tasks.

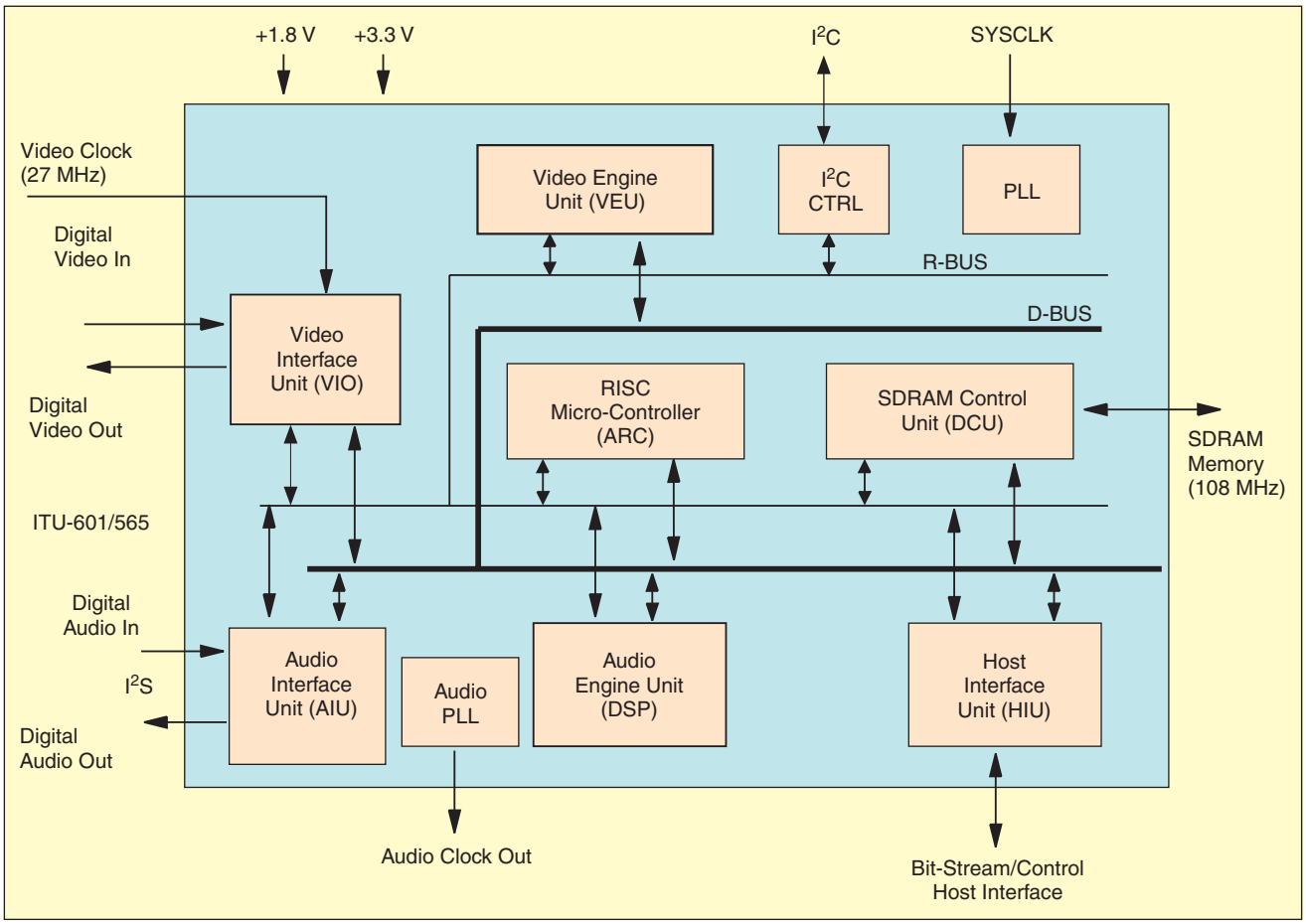
In our MPEG codec, not all tasks have the same priority. In general, we distinguish two distinct types of tasks: time-critical tasks, such as video compression, and non-time-critical tasks, such as multiplexing of audio and video and user communications. Hence, another solution for context switching could be to use two different processors: one processor for time-critical tasks and one for noncritical tasks. However, this solution requires to double the hardware requirements for the chip's controller.

Our solution is different. We decided to use a single ARC core, but enhance its architecture so that time-critical tasks can be handled with guaranteed performance. The modified core features additional memory mapping and interrupt controlling schemes that allows it to handle both critical timing requirements and traditional software applications. A real-time operating system is still used, but only for non-time-critical applications.

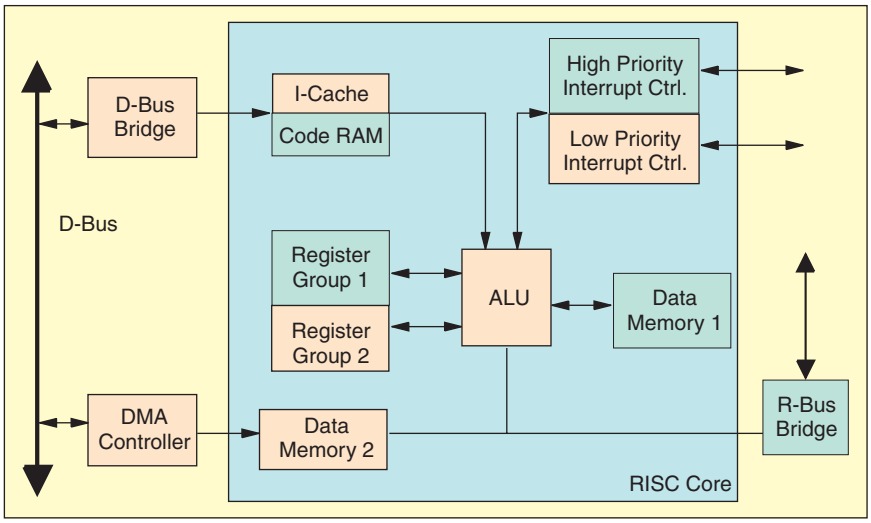
Fig. 4 shows a simplified block diagram of the modified ARC core and its interface circuitry to the rest of the architecture. In Fig. 4, shaded blocks show our enhancements to the original ARC core for time-critical tasks. Specifically, we have added a dedicated interrupt controller, dedicated instruction and data memories, and a separate set of ALU registers. Since all time-critical tasks are interrupt driven and have their own memory space, context switching for time-critical tasks is performed much faster and with predictable performance.

Host Interface Unit

The host interface is used to communicate with the host controller and external EPROM or flash memory. It supports a variety of communication protocols, including 16-bit Motorola- or Intel-like interfaces and a generic 8-bit interface. The host interface has a glueless interface to USB controllers, and it may also be used in PC-based host systems using a PCI bridge interface. The HIU is also used for the I/O of the compressed bit streams between the codec and an external controller.



▲ *3. Block diagram of the MPEG-2 a/v codec.*



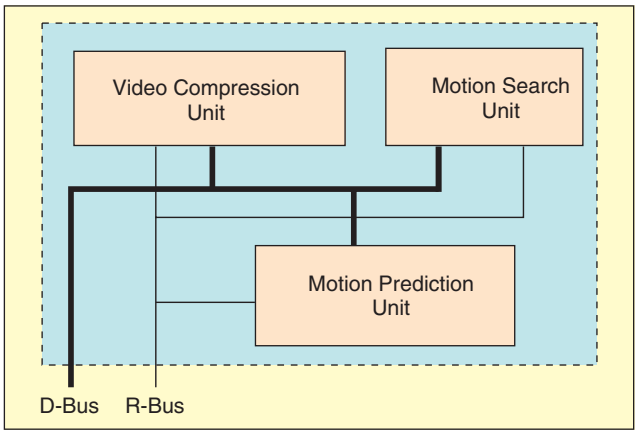
▲ *4. Block diagram of the modified ARC core.*

Audio Interface Unit

The AIU provides the interface between the codec and external audio devices. Audio samples are transferred in and out of the codec using I^2S signaling. During audio encoding, the CS92288 can be either a slave or a master to the external A/D. As a master, it can use a separate audio PLL to provide a user- configurable output clock for external audio A/D and D/As. Thus, if there is no other master audio clock, this feature eliminates the need for a separate, external, clock crystal.

Video Engine Unit

The VEU consists of dedicated hardware units that compute all MPEG video compression and decompression functions, as described by the block diagram in Fig. 1. Fig. 5 shows a block diagram of the VEU. It includes a video compression unit (VCU), a motion search unit (MSU), and the motion prediction unit (MPU). During encoding, it operates on the video data and generates an MPEG-compliant video elementary stream. It performs, among its many functions, motion estimation and compensation, DCT, quantization, rate control, and variable length coding.



▲ *5. Block diagram of VEU.*

During decoding, it operates on a video elementary stream and generates decompressed video frames. It performs variable length decoding, inverse quantization, IDCT, and motion compensation. The IDCT output is fully compliant with the IEEE-1180 accuracy requirements.

Motion Search and Prediction Units

The MSU computes all necessary MVs at a half-pel accuracy. The estimation algorithm involves two steps. In the first step, the motion estimation unit computes MVs with a coarser accuracy. Next, the motion refinement unit refines the MVs into half-pel accuracies. MSU supports frame estimation and field estimation, including 16×16, 16×8, top-to-top, top-to-bottom, bottom-to-top, and bottom-to-bottom estimation. The motion-search range is up to ± 63.5 pixels horizontally and ± 31.5 pixels vertically. For each macroblock, up to six candidate MVs may be generated for B-pictures and up to three candidate MVs may be generated for P-pictures.

The computation of the MVs is based on minimizing the sum of absolute errors [4]; however, to reduce the computational load of a full search, in the first phase of motion estimation, the computations are simplified by adopting a method similar to hierarchical motion search [4], where the macroblock pixels are down-sampled into a lower resolution. Fig. 6 shows a simplified block diagram of the MEU block [5]. It includes local RAMs for storing the search area and the current macroblock, multiplexers, an array of sum-of-absolute-differences (SAD) blocks and comparators. At each cycle, the array of SAD units computes the SAD between pixels in a row of the current macroblock and pixels in a row of the search window. After the SAD operation, an array of comparators computes the MVs.

In addition to motion prediction and compensation, the MPU computes a measure of activity for each input frame, which is being used by the rate control algorithm.

Audio Engine

The audio engine provides the core processing power for all audio-related functions. Since an MPEG codec needs to support a variety of audio algorithms, such as MPEG audio and Dolby Digital, a programmable digital signal processor (DSP) is more cost effective and flexible than hardwired logic.

In this codec, the audio engine consists of a proprietary, 24-bit, general purpose, programmable DSP. The DSP features a 24×24-bit multiplier and can perform a multiply-accumulate operation in a single cycle, with no overhead pipeline delay. It features dual data memory banks and a separate program memory.

A separate six-channel DMA engine provides a seamless interface between the external and internal memories and automatic translation from 64 to 24 bits. The audio engine can support all popular audio formats, such as Dolby Digital and MPEG audio. In addition, it can be programmed for audio post-processing, such as enhanced stereo or virtual surround.

SDRAM Control Unit

The DCU provides an interface between all functional units and the off-chip memory (SDRAM) storage. It sustains real-time audio and video encoding and decoding at 30 frames per second. The DCU arbitrates the requests from all function units and generates all necessary handshake and control signals for both the requesting unit and the external SDRAM. The data storage boundaries are user-programmable for custom applications.

Because of the large data bandwidth needed to do motion estimation and data I/O, the chip requires 8 Mbytes of synchronous DRAM (SDRAM) running at 108 MHz. Not all of this memory is being used, but this size allows for coding at least two B-pictures between P-pictures, say, IBBPBBP. Note that, in this case, the codec needs to store at least four frames of input video before it starts coding the first B-picture.

In addition to storing uncompressed video and audio data, the SDRAM is used for storing: the video and audio compressed elementary streams, the multiplexed program stream, MVs, DSP program and data, and the ARC program and data.

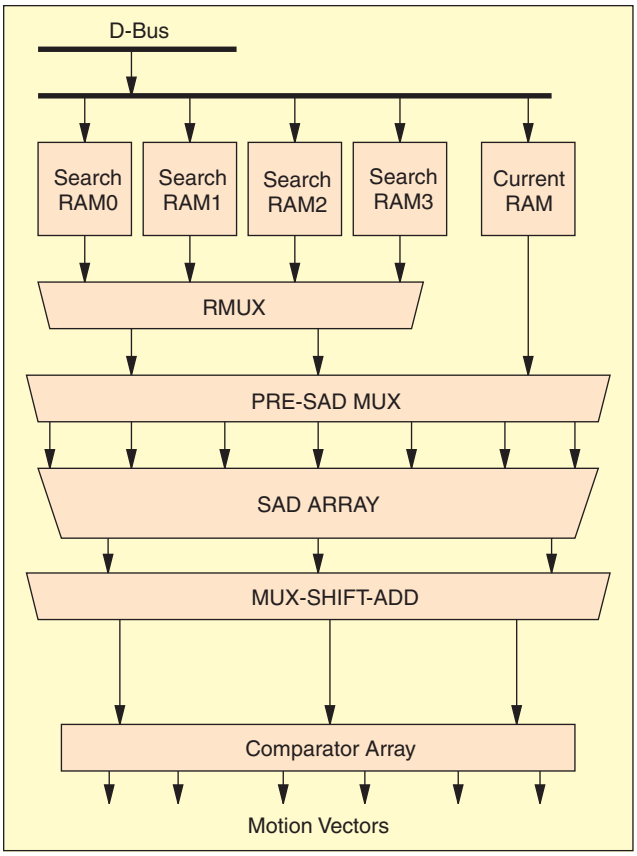
Video Interface Unit

As mentioned before, MPEG leaves quite open the design and implementation of all pre- and post-processing functions, such as filtering, and color downsampling and upsampling. Because of the variety of application in digital video encoding, there is a need for a programmable and flexible VIU.

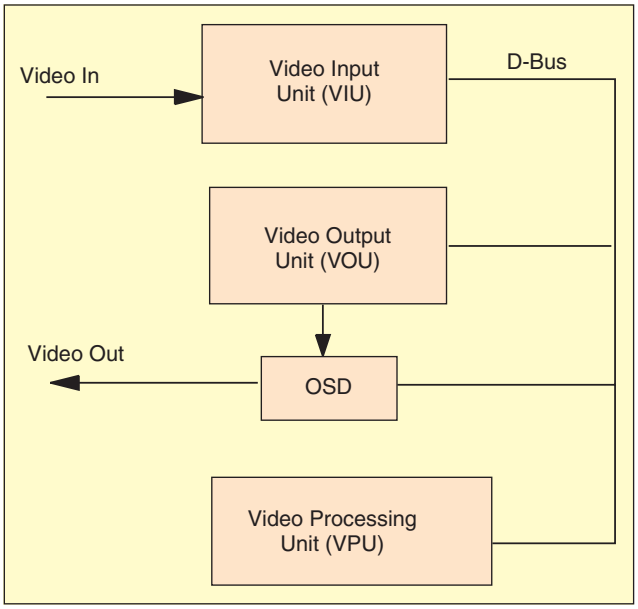
Fig. 7 shows a block diagram of the VIO. It includes the video input unit (VIU), the video output unit (VOU), the video processing unit (VPU), and the OSD unit. The VIU selects the input video active area and performs chroma conversion, detection of repeated fields (to reduce the compression effort), spatial and/or temporal noise prefiltering, and other data operations that facilitate the subsequent encoding processes.

The VOU can perform a variety of postprocessing operations, including horizontal and vertical scaling, and video format and frame-rate conversion. The OSD block mixes text and/or graphics from the OSD buffer (in SDRAM) with the output of the VOU and generates a correctly sequenced ITU-R BT.601 or 656 4:2:2 output video stream. The VPU is a separate processing unit that operates in parallel with the VIU and VOU. A short description of its features is given later in this section. The flexible architecture of the VIO unit allows it to operate in a number of different configurations.

For example, Fig. 8 shows the flow of operations when the VIO is used during encoding. The input video is captured by the VIU and is transferred to SDRAM. The buffered input is passed first to the VOU and then to the OSD unit, where it is mixed with text or graphics from the OSD buffers. The output of the OSD unit provides digital loop-back of the input video, overlaid with on-screen text or graphics.

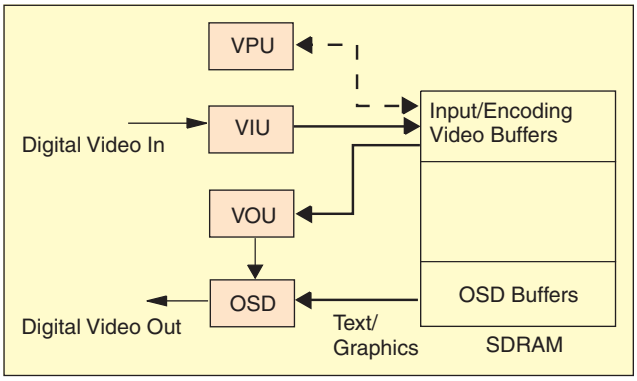


▲ *6. Block diagram of the motion estimation unit (MEU).*



▲ *7. Block diagram of VIU.*

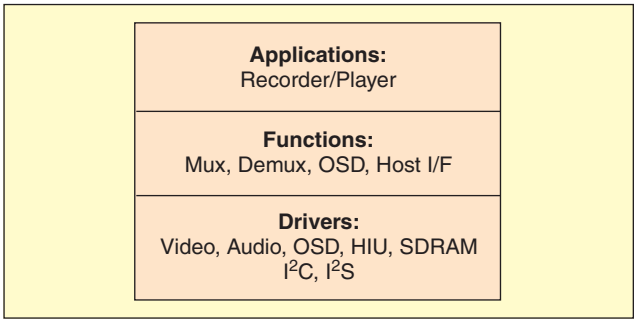
If necessary, input video can also be preprocessed by the VPU. Among its functions, the VPU can initialize the video frame buffer with specific YCbCr values (e.g., blue screen generation), copy data from one video buffer to another, or scale data from one frame-buffer region to another frame-buffer region.



▲ *8. VIO operation during encoding.*

Table 3. Key Features of the A/V Codec.

Video coding	MPEG-1, MPEG-2 MP@ML, SP@ML, MP@LL, I/P/B frames
Audio coding Resolutions	Dolby Digital, MPEG (all layers) Up to 24-bit/sample
Resolutions Rates	D1, 2/3 D1, 1/2 D1 CIF, SIF, QCIF 24, 25 , 29.97, 30 Hz
Prediction Rate Control	Adaptive field/frame/16×8 Adaptive frame/field DCT Adaptive inter/intra CBR, VBR (one-pass) I-only to 30 Mb/s
Interfaces	Video: ITU-601/656 I^2C, I^2S,SDRAM,EPROM, Flash, 16-b/8-b parallel
Other	Pre-, post-filtering 4:2:2 to 4:2:0 4:2:0 to 4:2:2 Temporal/spatial filtering Frame-rate conversion 8-bit OSD



▲ *9. Software architecture layers in the MPEG codec.*

Bringing Everything Together

Consider now the time between two input video frames. The following operations occur in parallel during encoding:

▲ Raw input and audio data are captured and stored into SDRAM. Input video data may be prefiltered and scaled by the VIU.

▲ The video engine generates a video elementary stream.

▲ The DSP generates an audio elementary stream.

▲ The ARC controller monitors the HIU for host commands, multiplexes audio and video elementary streams into a program stream, monitors a/v synchronization, and provides rate control.

▲ The HIU interacts with an external controller so that the compressed a/v data are transferred from the SDRAM to an external recording device.

Table 3 summarizes the key features of the codec.

Software Architecture

Fig. 9 shows the overall software architecture for our codec. At the top layer, we have the top-level applications, such as video recording or playback. These applications are written purely in C. Below, we have a collection of high-level functions (such as OSD, multiplexing, demultiplexing, and host interface). These functions are written mostly in C, but critical tasks may be written in assembly as well. RTOS services, such as threads and memory management, are provided for all functions in these two top layers.

The bottom layer includes miscellaneous application and hardware drivers. These include such functions as: I^2C and I^2S control, the control for the encoding and decoding of audio and video elementary streams, a/v synchronization, and OSD control. These are all considered time-critical functions and are written mostly in assembly. For the time-critical functions, all necessary data and instructions are preallocated in special memory and register banks, as it was shown by the shaded blocks in Fig. 4.

During encoding, the top-level recording applications can generate bit streams that comply with all popular formats (VCD, SVCD, and DVD), but they still allow users a variety of programming options for prefiltering, coding and multiplexing audio and video elementary streams, and for OSD. During decoding, the playback application will automatically parse and decode the input bit stream, but users can still program specific OSD and data transfer options.

Conclusions

In this article we presented a brief overview of the critical engineering concerns for the design and implementation of an MPEG-2 a/v codec IC. By taking into consideration the overall system requirements in consumer-based digital video recording, we designed the codec with unique

211

and flexible RISC controller and VIUs, a programmable audio DSP, and hardwired video processing units. The codec has been implemented using a standard-cell library in 0.18 µm CMOS technology. At 108 MHz, it consumes about 1 W of power.

Acknowledgments

We would like to thank all the hardware and software engineers that have contributed to the design and implementation of the CS92288 codec. We also thank the anonymous reviewers for their valuable comments and suggestions.

Konstantinos Konstantinides is with the video recording group in Cirrus Logic where he divides his time between marketing and advanced R&D. Prior to Cirrus, he was Director of Advanced Products at Stream Machine and Member of Technical Staff at HP Laboratories, where he was involved in the areas of digital signal and video processing. He is coauthor with V. Bhaskaran of two books on *Image and Video Compression Standards—Algorithms and Architectures* (Norwell, MA: Kluwer, 1995, 1997). He is member of the IEEE Signal Processing Society Technical Committee on the Design and Implementation of Signal Processing Systems. He holds a Ph.D. from the University of California at Los Angeles.

Cheng-Tie Chen is a Vice President of Engineering in the video recording group in Cirrus Logic, where he is a specialist in video compression systems and signal processing. Prior to Cirrus, he was Vice President of Engineering in Stream Machine and was with Eastman Kodak Research Laboratories and Bellcore (now Telcordia). He was a guest special issue co-editor for Optical Engineering on Visual Communications and was an Associate Editor for *IEEE Transaction on Circuits and Systems for Video Technology* and the *Journal of Visual Communication and Image Representation*. He has a Ph.D. (EE) from the University of Pennsylvania.

Ting-Chung Chen is currently with Cirrus Logic. He has 21 years of experience in video signal processing, digital signal processing, embedded system implementation and VLSI design. Prior to joining Cirrus Logic he was chief scientist at Stream Machine. He designed video compression products at Compression Labs and worked at Bell Labs and Bell Communications Research. He has over 45 journal and conference publications in the area of MPEG compression algorithms, HDTV coding, motion compensation, high-order entropy coding, and wireless visual communications. He holds a Ph.D. (EE) from Rice University.

Hown Cheng is currently with Cirrus Logic where he is Senior Manager in charge of ASIC development. Prior to joining Cirrus, he has held technical positions in Stream Machine, OPTi, WD, and C&T. He has 17 years of experience in designing high speed data path, CPU/embeded systems and video processors. He holds an M.S.E.E. degree from North Carolina State University.

Fure-Ching Jeng is currently in charge of systems development in the video recording group of Cirrus Logic. Prior to Cirrus, he was Vice President of R&D in Stream Machine and worked on video coding at Bell Communications Research. He has published more than 15 articles about digital image and video compression. He holds a Ph.D. in computer and systems engineering from Renesselaer Polytechnic Institute.

References

[1] *Coding of Moving Pictures and Associated Audio for Digital Storage Media Up to 1.5 Mbits/s,* ISO/IEC JTC1 CD 11172, 1992.

[2] *Generic Coding of Moving Pictures and Associated Audio,* ISO/IEC JTC1 CD 13818, 1994.

[3] *Information Technology—Coding of Audio-Visual Objects,* ISO/IEC 14496, 1999.

[4] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures,* 2nd ed. Boston, MA: Kluwer, 1997.

[5] C.T. Chen, T.C. Chen, C. Feng, C-C. Huang, F-C. Jeng, K. Konstantinides, F-H. Lin, M. Smolenski, and E. Haly, "A single-chip MPEG-2 video encoder/decoder for consumer applications," in *Proc. ICIP-99*, 1999.

[6] DVX C-Cube Microsystems. DVxpress-MX25 and DVxpress-MX50, product brief, 1999.

[7] M. Smolenski, T. Fink, K. Konstantinides, D. Frankenberger, and C. Peplinski, "Design of a personal digital video recorder/player" in *Proc. 2000 IEEE Workshop on Signal Processing Systems: Design and Implementation (SiPS 2000)*, 2000, pp. 3-12.

[8] C.T. Chen, T.C. Chen, F-C. Jeng, H. Cheng, and K. Konstantinides, "A single-chip MPEG-2 MP@ML audio/video encoder/decoder with a programmable video interface unit," in *Proc. ICASSP-01*, 2001.

[9] GlobeSpan. iTVC12 MPEG-2 encoder, product brief, May 2001.

[10] *Digital Audio Compression Standard (AC-3),* United States Advanced Television Systems Committee, ATSC, Dec. 1995.

[11] JVC, Matsushita, Philips, and Sony. Video CD specification, version 2.0. Philips Consumer Electronics, 1995.

[12] *1998 People's Republic of China Electronic Industry Standard. Chaoji VCD (Super VCD) system specification,* The Ministry of Information Industry of the People's Rebuplic of China, SVCD SJ/T 11196, 1998.

[13] J. Taylor, *DVD Demystified*, 2nd ed. McGraw-Hill, 2000.

[14] Philips Semiconductors. T*he I^2C-Bus Specification, version 2.1*. Philips Semiconductors, Jan. 2000.

[15] Philips Semiconductors. *I^2S Bus Specification*, June 1996.

[16] Argonaut Technologies, Ltd. *ARC Interface Manual*, 1999.

[17] *Encoding Parameters of Digital Television for Studios,* ITU-R BT.601-5, 1994.

[18] *ITU-R BT.656-4 interfaces for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of recommendation,* ITU-R BT.601 (Part A), 1994.

[19] *Test Model 5: Version 2.* ISO, ISO/IEC JTC1/SC29/WG11/N0400, 1993.

[20] P.H. Westerink, R. Rajagopalan, and C.A. Gonzales, "Two-pass MPEG-2 variable-bit-rate encoding," *IBM J. Res. Devel.*, vol. 43, no. 4, pp. 471-488, July 1999.

[21] N. Mohsenian, R. Rajagopalan, and C.A. Gonzales, "Single-pass constant- and variable-bit-rate MPEG-2 video compression," *IBM J. Res. Devel.*, vol. 43, no. 4, pp. 489-509, July 1999.

EXHIBIT M

THE D30V/MPEG MULTIMEDIA PROCESSOR

THIS PROCESSOR INTEGRATES A DUAL-ISSUE RISC WITH MINIMAL HARDWARE SUPPORT TO IMPLEMENT A REAL-TIME MPEG-2 DECODER. ITS SMALL CHIP AREA AND EASY PROGRAMMING ARE ADVANTAGEOUS FOR MULTIMEDIA APPLICATIONS.

Hidehiro Takata,
Tetsuya Watanabe,
Tetsuo Nakajima,
Takashi Takagaki,
Hisakazu Sato,
Atsushi Mohri,
Akira Yamada,
Toshiki Kanamoto,
Yoshio Matsuda,
Shuhei Iwade, and
Yasutaka Horiba
Mitsubishi Electric Corporation

MPEG-2 decoding and encoding are important applications for multimedia systems. Real-time capability and low-cost implementation are the main design considerations for these systems. Due to the high computational requirements of real-time applications, multimedia systems typically use special-purpose processors to handle data.[1] However, due to the inherent inflexibility of their designs, these dedicated processors are of little use in various application environments—digital videocassette recorders, for example. This article introduces Mitsubishi's D30V/MPEG multimedia processor, which integrates a dual-issue RISC[2,3] with minimal hardware support for a real-time MPEG-2 decoder. This approach is advantageous because of the small chip area it requires and the flexibility of the easy-to-program RISC processor for multimedia applications.

The 2.3-V, 300-MHz multimedia processor uses instruction decode and clock skew control to achieve the MPEG-2 decoding. The processor contains a CPU core, dedicated hardware, a 32-Kbyte data RAM, and a 64-Kbyte instruction RAM. We met the 3.3-ns clock's timing requirements for fetch and decode from the large instruction RAM by using advanced circuit techniques including dynamic logic[4] together with place-and-route tools. The careful design of global and local clock networks achieves a clock skew of 0.238 ns. The chip's 6.7 million transistors are integrated in an area of 8.77×8.28 mm^2 in a 0.25-micron CMOS process.

Decoding chip

High-performance hardware provides a flexible system solution to the problem of MPEG-2 decoding. Processors must decode 243,000 8×8-pixel blocks each second for video decoding in NTSC format of the MPEG-2 main profile at main level (MP@ML).[5] Video decoding accounts for more than 80% of the MPEG-2 decoding computations. The D30V/MPEG, running at 243 MHz, uses 1,000 cycles to decode one block.

Figure 1 illustrates the MPEG-2 decoding process. MPEG-2 video decoding starts with a single bitstream, Huffman-decoded[5] into 12-bit data elements. The D30V core, a two-way VLIW and two-way SIMD processor, separates the MPEG-2 system bitstream into video and audio bitstreams. The D30V decodes the data via a well-defined algorithm and sends the final information to an output device. The D30V decodes a bidirectionally predicted non-intra[5] video block in less than 800 cycles of a

0272-1732/99/$10.00 © 1999 IEEE

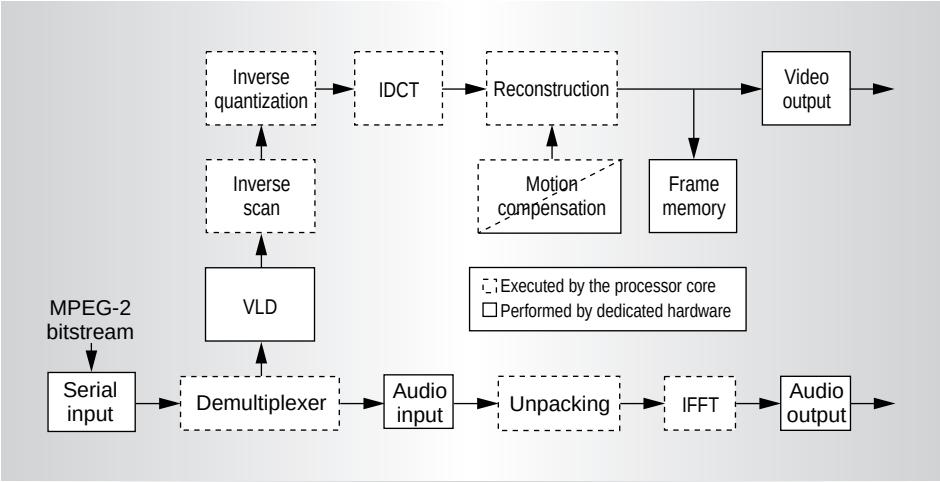


Figure 1. MPEG-2 video and audio decoding.

243-MHz clock.[6] Video decoding requires two special, dedicated hardware blocks in addition to the core: the variable-length decode unit (VLD) and the block loader (see Figure 3 on next page). The block loader transfers macro blocks between the external memory and the on-chip memory and computes the half-pixel-element operation for the motion compensation process. No special hardware is necessary for audio decoding. The D30V core performs all the remaining operations.

Running at 243 MHz, the processor has an available throughput of 243 million execution cycles per second. Since decoding an 8×8-pixel video block requires 800 cycles in the worst-case scenario—bidirectionally predicted non-intra blocks,[7]—the processor uses 200 million cycles per second. Of the remaining available cycles, 5 million per second are needed for system decoding and 40 million per second for AC-3 (Digital Audio Compression Standard) audio decoding. The D30V takes 4,076 cycles to compute the 128-point, complex IFFT used in AC-3 decoding. Figure 2 illustrates the allocation of available throughput to the D30V's tasks running at 243 MHz.

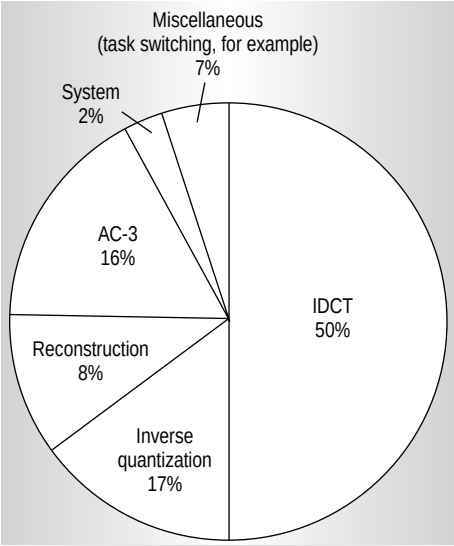


Figure 2. D30V core decoding tasks.

Acronyms

DVD	digital video disk
DVC	digital videocassette
HDTV	high-definition television
IDCT	inverse discrete cosine transform
IFFT	inverse fast Fourier transform
MPEG	Motion Pictures Expert Group
NTSC	National Television System Committee
RISC	reduced instruction set computer
SDTV	standard-definition television
SIMD	single instruction, multiple data
VLC/VLD	variable-length coder/decoder
VLIW	very long instruction word

Chip description

Figure 3 shows a block diagram of the D30V/MPEG. The chip consists of the multimedia processor core, the 64-Kbyte instruction RAM, the 32-Kbyte data RAM, peripherals, and a DRAM interface block. The peripherals include the system bus interface, a cyclic redundancy check (CRC) block, a vari-

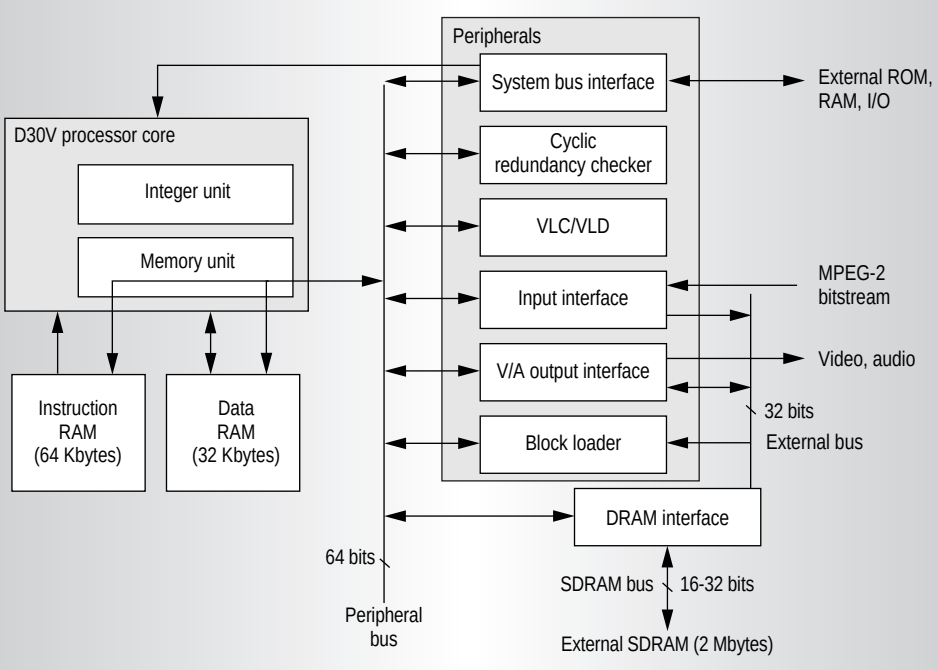


Figure 3. D30V/MPEG block diagram.

Table 1. D30V/MPEG chip features.

Feature	Description
Instructions	102 subinstructions
Parallelism	Two-way VLIW, two-way SIMD
Register file	Sixty-four 32-bit general-purpose registers, two 64-bit accumulators
On-chip RAMs	64 Kbytes (instruction), 32 Kbytes (data), 4 Kbytes (VLC/VLD tables)
Clock frequency	243 MHz (max. 300 MHz)
Peak performance	1.2 GOPS
Supply voltage	2.5 V (internal), 3.3 V (external)
Power (typical)	2.0 W (2.5 V at 243 MHz)
Chip size	8.77 mm × 8.28 mm (6,760,000 transistors)
Processor core size	7.4 mm^2 (360,000 transistors)
Peripheral size	9.4 mm^2 (694,000 transistors)
Process technology	0.25-micron, 1-poly, 4-metal CMOS
L (effect)	0.21-micron (n-channel transistor), 0.235-micron (p-channel transistor)
Package	257-pin grid array

able-length coder/decoder, an input interface block, a video/audio output interface block, and the block loader. The memory unit, one of two execution units in the D30V core (the other is the integer unit), controls data transfer between the core and the peripherals via the high-bandwidth peripheral bus. The D30V core reads and writes commands into control registers in the peripherals with load/store instructions. The peripherals transfer block data to and from the data RAM with direct memory access. The DRAM interface block controls data transfer between the peripherals and the synchronous DRAMs (SDRAMs), which are the external memory.

Table 1 summarizes the D30V/MPEG's characteristics. The processor exploits two modes of parallelism through the memory unit and the integer unit. It executes the dual-issue instructions through two-way VLIW and two-way subword operations, for a maximum of four operations per cycle. It achieves a peak sustained throughput of 1.2 billion operations per second when running at 300 MHz. The power dissipation at 243 MHz with a 2.5-V power supply is 2.0 W in typical conditions.

Figure 4 is a chip micrograph of the D30V/MPEG. We laid out the processor core's data path by hand and used the core as a hard macro. We used Verilog HDL, synthesis, and place-and-route tools to design the seven peripheral blocks, which are located at the left and bottom of the chip.

Multimedia instructions

The D30V/MPEG incorporates several enhancements to speed up multimedia processing. The functional units in the integer and memory units can operate on either 32-bit data elements or two 16-bit data elements, as specified by each subinstruction. Tables 2 and 3 list subinstructions that operate on a single pair of halfwords (16-bit data) and on subwords (two pairs of halfwords). The single halfword operations of Table 2 are flexible, allowing the operands to be stored in either the upper or lower halfword of a register. A single halfword

operation operates on two 16-bit data elements in two independent registers to generate a 16- or 32-bit result. For example, ADDH can operate on the high halfword of one register with the low halfword of another register to generate a 16-bit result. JOIN concatenates any two 16-bit data elements in the high and low portions of registers to generate a 32-bit result. This feature effectively doubles the number of available registers to 128 for many multimedia applications.

The processor can perform the same subword operation on two sets of 16-bit data elements in the upper and lower 16-bit portions of registers to generate two 16-bit results or two 32-bit results, respectively. To maximize the deliverable throughput, a subword multiplication operating on two pairs of 16-bit data elements generates either two 16-bit results (MUL2H) or two 32-bit results (MULX2H), depending on the desired operand precision.

For digital signal-processing applications, we included instructions such as load-store with modulo addressing, block-repeat, and multiply-accumulate. Load-store instructions can access memory operands with an autoincrement or autodecrement addressing mode. For these load-store instructions, the D30V hardware, using the MOD_S and MOD_E registers, executes modulo addressing. Also, a block-repeat instruction executes branches at the end of a loop with no delay penalties. The RPT_S, RPT_E, and RPT_C registers keep the start address, the end address, and the repeat count of a loop specified by a block-repeat instruction. Therefore, the hardware executes the branches for the block-repeat instructions without using any clock cycles. The multiply-accumulate instruction performs a 32×32-bit multiply followed by a 64-bit addition; the multiplication result can shift one bit to the left before accumulation to support fixed-point arithmetic. The processor also supports multiply-subtract subinstructions.

Saturate and add sign subinstructions increase the processor's performance for video applications. The saturate subinstructions (SAT, SATZ, SAT2H, SATZ2H, SATH) saturate 32-bit or two 16-bit data elements with any precision specified by operands. The add sign subinstructions (ADDS, ADDS2H) add 16-bit or 32-bit data with the sign of the third operand to generate the second operand.

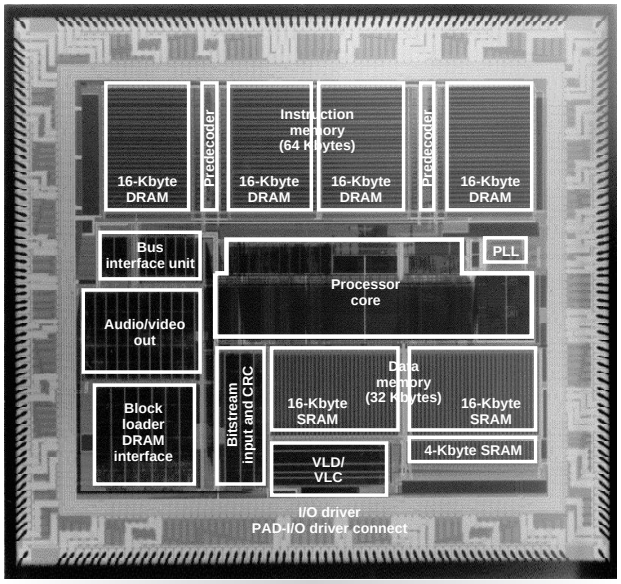


Figure 4. D30V/MPEG chip micrograph.

Table 2. Halfword operations.

Subinstruction	Operation
ADDH	Add a halfword to another halfword
SUBH	Subtract a halfword from another halfword
JOIN	Join two halfwords to generate a word
SATH	Saturate a word into a halfword
MULH	Multiply a halfword with another halfword to generate a word
LDH	Load a halfword
STH	Store a halfword

Table 3. Subword operations.

Subinstruction	Operation
ADD2H	Add two pairs of halfwords
ADDS2H	Add the sign bits of two halfwords to two halfwords
SUB2H	Subtract two pairs of halfwords
AVG2H	Average two pairs of halfwords
MUL2H	Multiply two pairs of halfwords
MULX2H	Multiply two pairs of halfwords with extended precision
SAT2H	Saturate two halfwords
SATZ2H	Saturate two halfwords to positive
SRA2H	Shift arithmetic right two halfwords
SRL2H	Shift logical right two halfwords
ROT2H	Rotate two halfwords
LD4BH	Load 4 bytes into four halfwords with unpacking
ST4HB	Store four halfwords into 4 bytes with packing

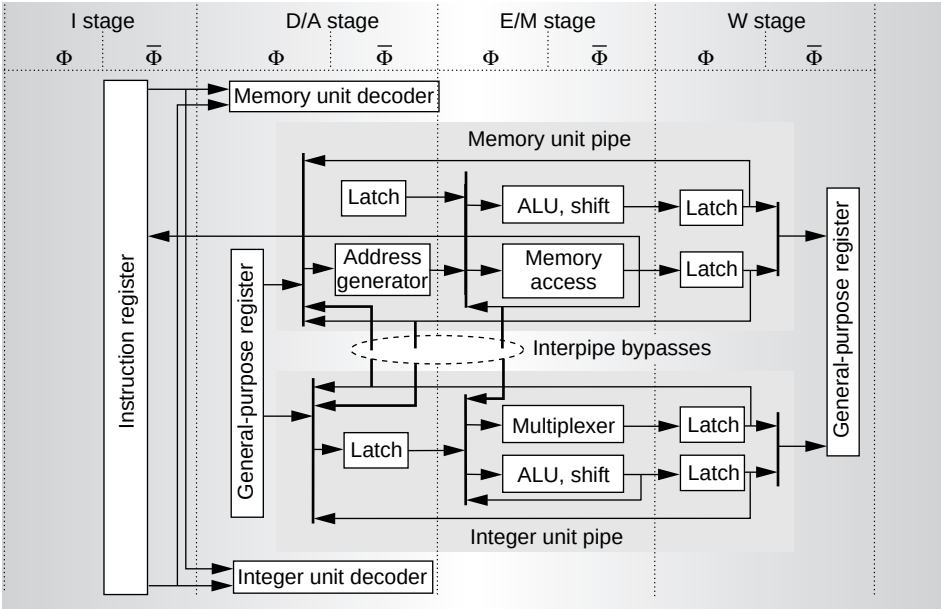


Figure 5. The D30V/MPEG's multiple-bypass structure (ϕ: high clock; $\bar{\phi}$: low clock).

These operations, executed by simple hardware in one clock cycle, increase MPEG video-processing performance.

Implementation issues

To achieve the frequency of 243 MHz, we designed the D30V/MPEG for a 3.3-ns, 300-MHz cycle time, operating at 2.3 V and 27°C. We provided 19% of the operation power (57 MHz) for worst-case process and temperature conditions. Because of the instruction memory's large capacity (64 Kbytes), the implementation of the high-speed instruction fetch is critical. Another important issue is controlling the clock skew.

Floor planning

Figure 5 shows the processor's pipeline and operand bypass mechanism. The memory unit and the integer unit pipes execute two subinstructions in parallel. Register operation instructions execute in the instruction fetch (I), instruction decode (D), execution (E), and write-back (W) stages. Memory access instructions execute in the instruction fetch (I), instruction decode and address generation (D/A), memory access (M), and write-back (W) stages. Operand bypasses can take place not only in each pipe but also between the pipes. The operand fetched from memory in the M stage of the memory unit pipe is bypassed to the following instructions both in the memory unit and integer unit pipes. The result of a multiply operation in the I stage of the integer unit pipe is bypassed to the following instructions in both pipes.

Multimedia applications requiring many parallel and multiply operations frequently use these two interpipe bypasses. We have obtained a 12% speedup via these bypasses in the MPEG block-decoding program. We have also eliminated unimportant bypasses and used interlock controls to insert wait cycles for the eliminated bypasses.

Figure 6 shows the instruction memory floor plan and timing chart of the instruction fetch and decode stages. To achieve 300-MHz access, we allow 3.3 ns for each stage's cycle time. The I stage performs address decoding and memory reading. The D stage executes instruction decoding and generates control signals for the memory and integer units. The amount of instruction decoding in the D stage is large because of the complicated interlock controls. Therefore, we separated the interlock controls into two functions: generating interlock conditions and generating interlock signals. Each function is divided into a predecode block and a decode block, both implemented with dynamic circuits to shorten access time.

To make matters worse, because of the large instruction memory, the access path—6 mm from a memory cell to the decoder—is inherently long. Therefore, we placed the predecode blocks on the access paths. In the high-clock (ϕ) I stage, the address decoder decodes the instruction address and drives it to the instruction memory blocks. In the low-clock ($\bar{\phi}$) I stage, the word lines are activated, and the sense amplifiers detect the difference in the bit lines and forward the memory data to the predecode units. The predecode results are repeated at the center of the data path and decoded in the D stage (ϕ). In the D stage ($\bar{\phi}$), the MU and IU controls generate the control signals for the memory and integer units. The

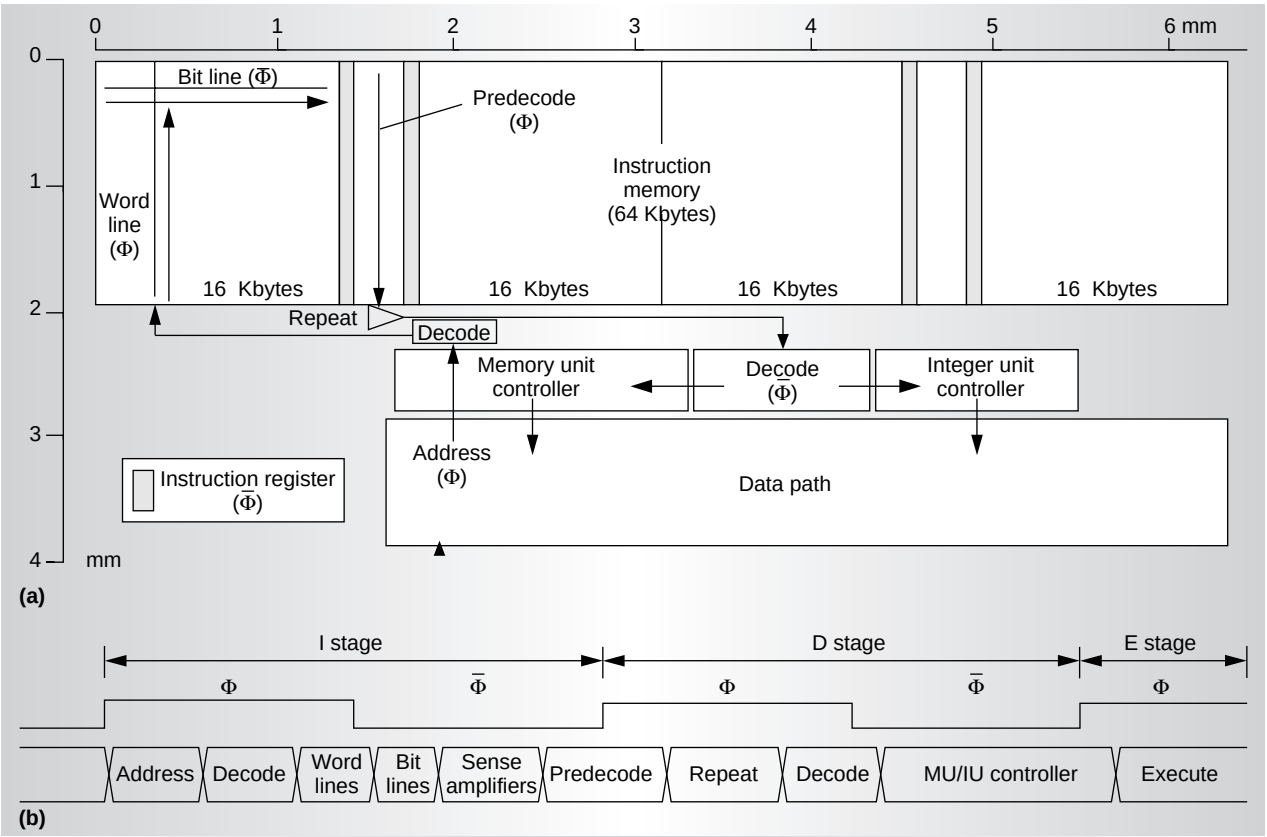


Figure 6. Instruction memory floor plan (a) and timing chart (b).

data path begins instruction execution immediately at the rising edge of the clock in the E stage.

To shorten instruction memory read-out time, we used a selective sensing circuit, as shown in Figure 7. The least-significant bit (LSB) address activates one of the coupled sense amplifiers, and the selection occurs in the latch. In this way, we avoided a slow 32-to-1 selector and maximized the output's speed. As a result, we achieved an access speed for the 64-Kbyte memory of a memory half that capacity.

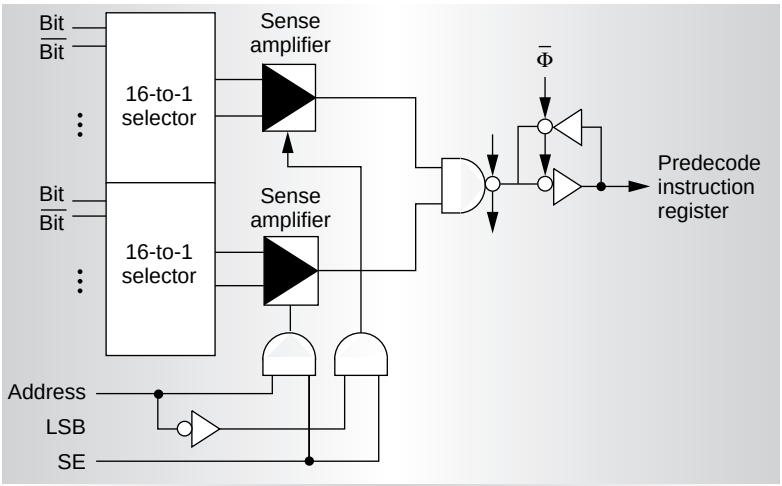


Figure 7. Double sense amplifiers and selective latch.

Predecode/decode with dynamic circuits

Both the predecode and decode blocks use dynamic circuits, which guarantee synchronization of these operations at the clock edge. The predecode dynamic gates are enabled at the rising edge of the D stage, and the decode dynamic gates at the falling edge of the D stage. Figures 8a and 8b (next page) show typical dynamic circuits for the predecode block. The domino circuit in Figure 8a realizes a 7-input AND, 8-input OR function. The dynamic circuit in Figure 8b realizes a compare followed by an OR of the results. Figure 8c shows typical dynamic circuits for the decode block. They use a combination of static and dynamic logic to shorten the total operation time. The 3-input AND, 4-input

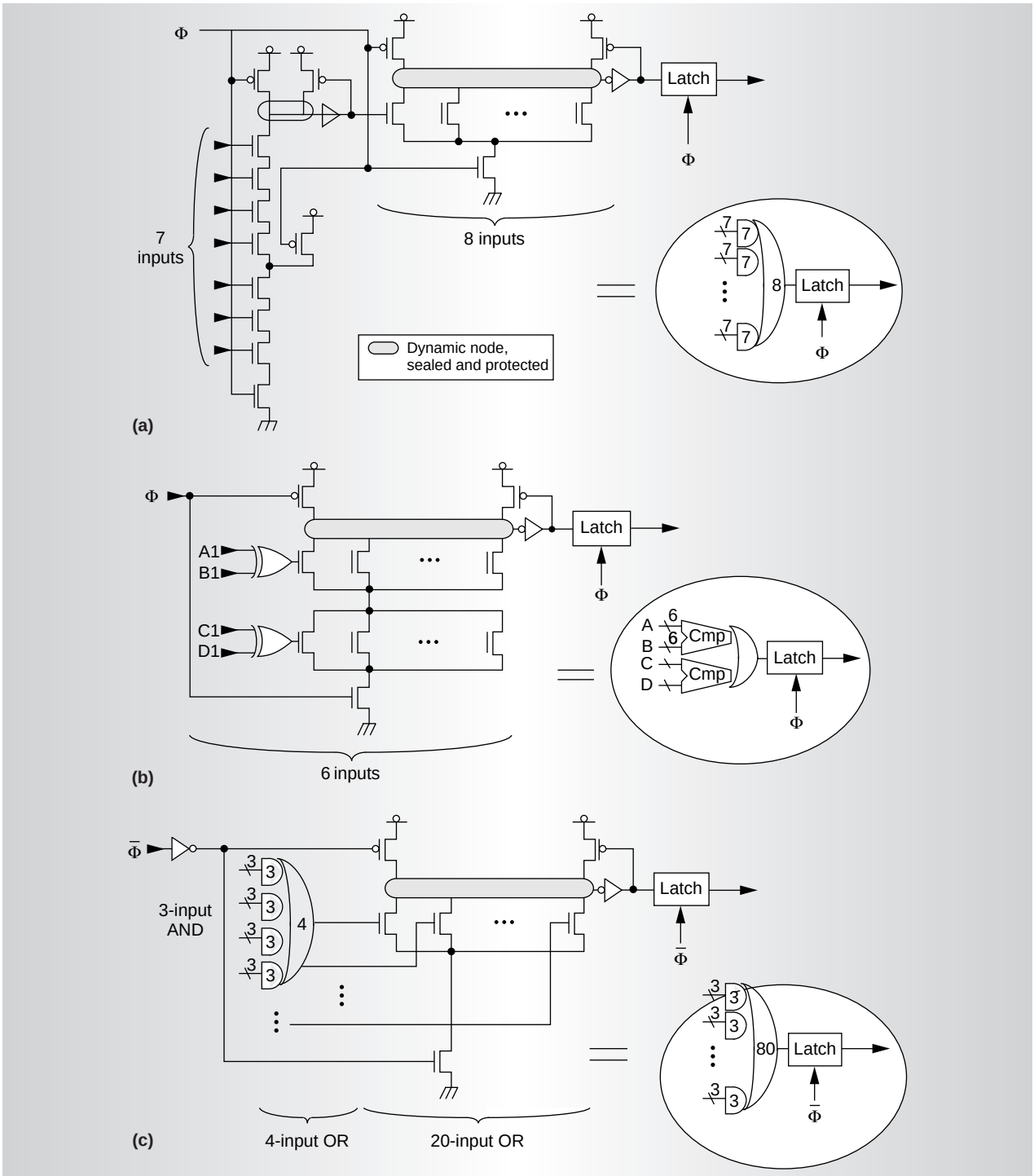


Figure 8. A 7-input AND, 8-input OR domino circuit (a); a two-comparison, 2-input OR dynamic circuit (b); a 3-input AND, 80-input OR combination circuit (c).

OR static CMOS logic completes before the falling edge of the clock, followed by the 20-input OR dynamic circuit. The combined 3-input AND, 80-input OR static and dynamic logic fully optimizes the complete clock frequency for fast throughput to the data path.

We implemented the physical layout of the predecode and decode blocks using a place-

and-route tool. The tool registers the dynamic circuits as standard cells. To protect the dynamic nodes from noise, they are sealed and protected against overlap routing, as shown in Figures 8a–c.

Clock skew control

Because of the large die, large memory, and high-speed instruction fetch, controlling clock skew is especially important. In the D30V/MPEG, we minimized preskew by centrally routing and driving the clock to the eight main clock drivers, as shown in Figure 9. We chose the size and position of the repeaters carefully to minimize delay from the phase-locked loop and to ensure sharp rise and fall times. The eight main clock drivers drive the clock mesh in the blocks implemented by the place-and-route tool and the data path blocks. In the tool-implemented blocks, a place-and-route option for the shortest and straightest connection to the clock helps reduce the clock skew.

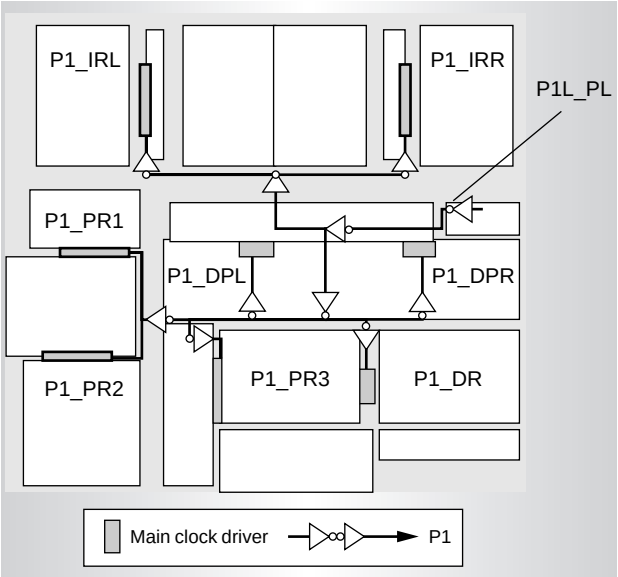


Figure 9. Positions of clock drivers and repeaters.

We used mesh routing networks for all processor core blocks, including the instruction RAM, the data RAM, and the peripherals. The outputs of the eight main drivers connect to the mesh routing networks.

Results of SPICE simulation

Figure 10 shows results of SPICE simulation of the D30V/MPEG's instruction fetch and decode. The simulation conditions were the Mitsubishi 0.25-micron CMOS process parameters at 2.3 V and 27°C. In this case, the results are for the path shown in Figure 6, including the predecode and decode circuits shown in Figures 8b and 8c. In the I stage, the processor completed instruction memory access and input to the predecode block. In the first half of the D stage, it finished the output of the predecode block and the input of the decode block. It executed instruction fetch and instruction decode in one and a half cycles, meeting the 3.3-ns, 300-MHz cycle time requirement.

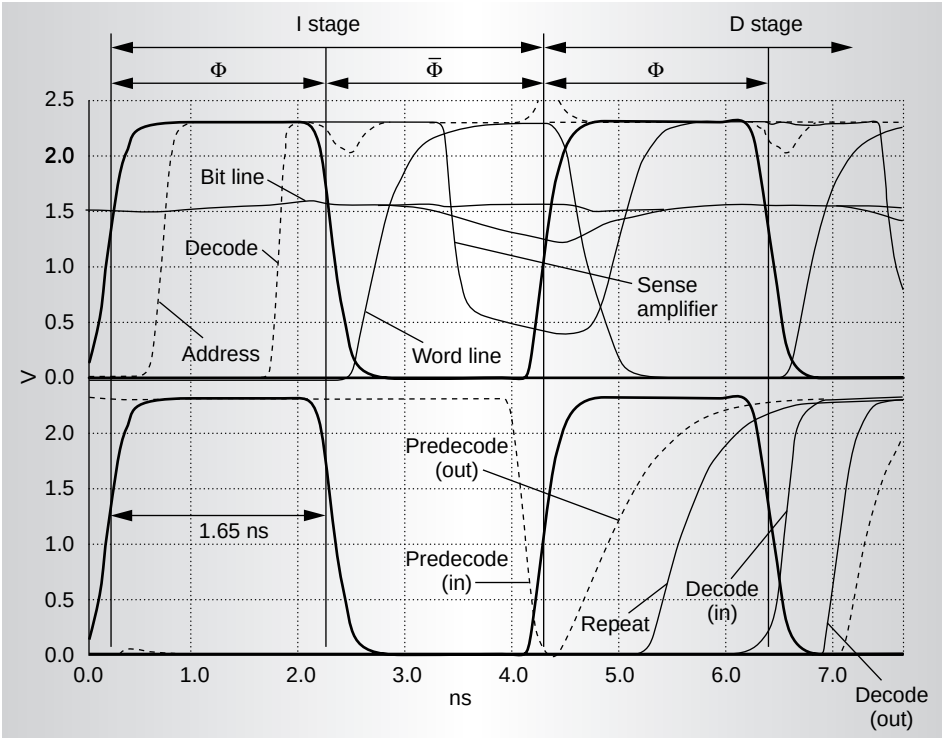


Figure 10. SPICE results of instruction fetch and decode.

Figure 11 (next page) shows the results of the clock skew simulation. We extracted and simulated the RC nets of the clock lines at data points over the entire chip. The worst skew, 0.238 ns, occurs in the left side of a block

implemented by the place-and-route tool. The clock skew, except for peripheral blocks, is 0.150 ns in the worst case. This is 4.5% of a 3.3-ns (300-MHz) clock cycle. The simulation conditions are 2.5 V and 27°C.

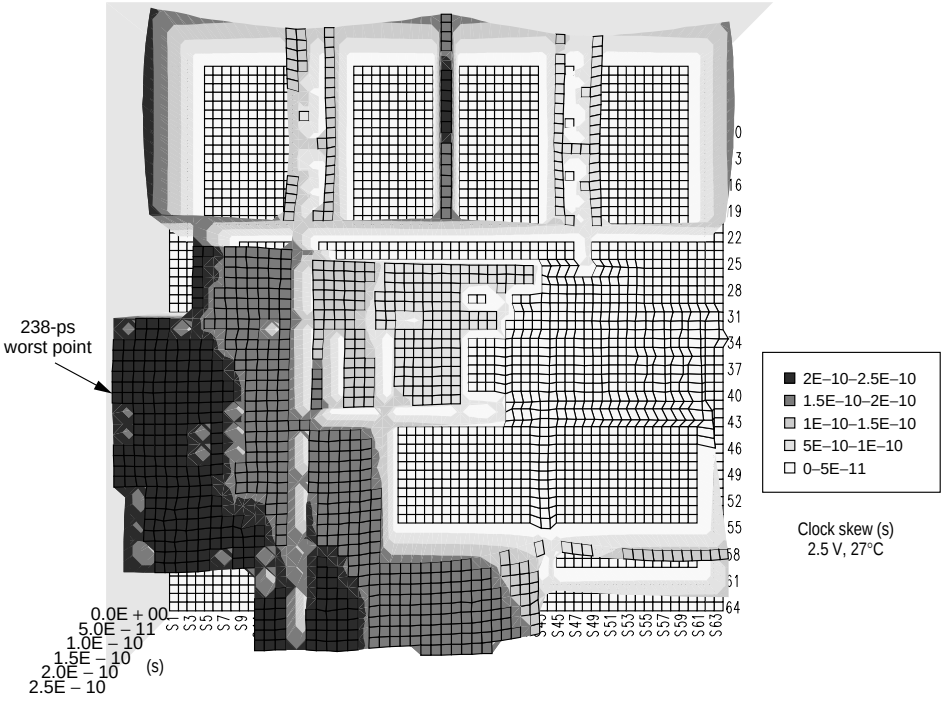


Figure 11. Clock skew results.

Chip evaluation results

Figure 12 shows the schmoo plot of the chip running at 300 MHz with a 2.3-V power supply. In this case, we executed an ALU bypass operation. We executed the interlock controls during instruction fetch and decode.

Figure 13 shows the chip's power dissipation, depending on the supply voltage and the running frequency. The total I_{CC} current when executing IDCT operations at 243 MHz with a 2.5-V power supply is 807 mA. The power dissipation for this case is 2.0 W.

Applications of the D30V/MPEG include DVD and DVC encoder/decoders and SDTV/HDTV endoding/decoding systems. In our

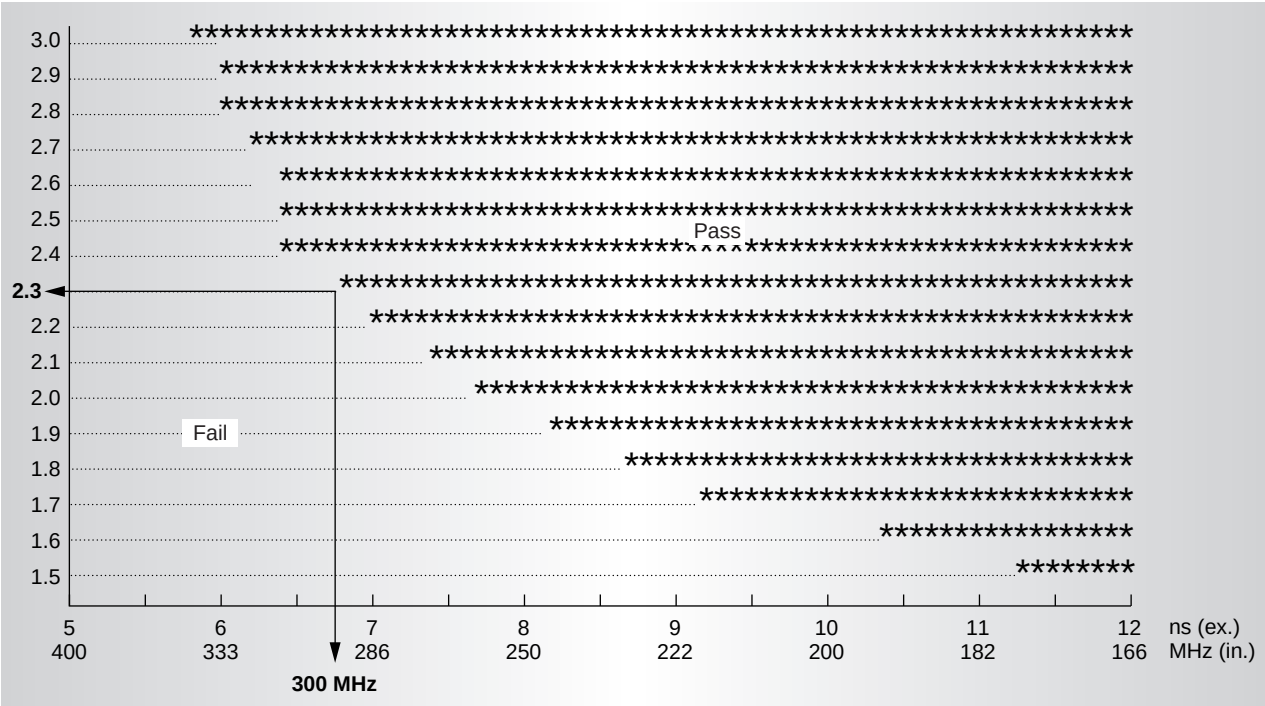


Figure 12. Schmoo plot (ALU bypass operation).

future work on the multimedia processor, we plan process migrations, speed enhancement, and processor core reuse. MICRO

Acknowledgments

We thank T. Yoshida, K.M. Clark, and all the engineers involved in this project at Mitsubishi Electric Corporation for their help.

References

1. M. Mizuno et al., "A 1.5W Single-Chip MPEG-2 MP@ML Encoder with Low-Power Motion Estimation and Clocking," *ISSCC Digest of Technical Papers*, IEEE Press, Piscataway, N.J., 1997, pp. 256-257.
2. T. Yoshida et al., "A 2V 250MHz Multimedia Processor," *ISSCC Digest of Technical Papers*, 1997, pp. 266-267.
3. A. Yamada et al., "Real-Time MPEG-2 Encoding and Decoding with a Dual-Issue RISC Processor," *Proc. IEEE Custom Integrated Circuits Conf.*, IEEE Press, 1997, pp. 225-228.
4. H. Takata et al., "2V 250MHz Multimedia Processor Design," *Proc. Int'l Workshop Advanced LSIs*, IEEE Electron Devices Society, Tokyo Chapter, 1997, pp. 124-131.
5. ISO-IEC/JTC1 SC29, DIS 13818, Part 2, Int'l Organization for Standardization, Geneva, 1994.
6. A. Mohri et al., "A Real-Time Digital VCR Encode/Decode and MPEG-2 Decode LSI Implemented on Dual-Issue RISC Processor," *Proc. 24th European Solid-State Circuits Conf.*, IEEE Press, 1998. pp. 180-183.
7. E. Holmann et al., "Real-Time MPEG-2 Software Decoding with a Dual-Issue RISC Processor," *Proc. VLSI Signal Processing IX*, 1996, pp. 105-114.

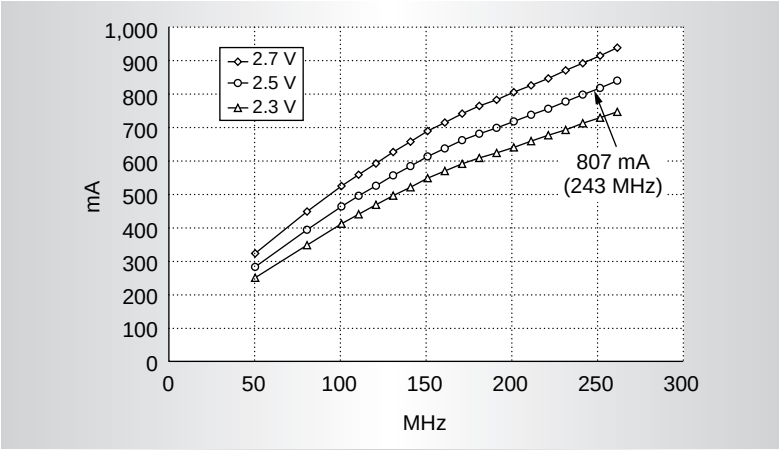


Figure 13. Power dissipation (circuit total).

Takashi Takagaki, a research engineer in the LSI Design Center, develops LSI design technology and designs ASICs. **Hisakazu Sato**, deputy manager of the Multimedia Processor Architecture Group, works on DSP architecture design.

Atsushi Mohri, a senior engineer in the Processor Architecture Department of the Information Technology R&D Center, worked on designing the processor architecture. **Akira Yamada**, manager of the Multimedia Processor Architecture Group's processor section, designs DSP architectures. **Toshiki Kanamoto** is a research engineer in the Semiconductor EDA Department. He is responsible for interconnect parasitic extraction and circuit analysis. **Yoshio Matsuda**, manager of the Advanced Circuit Design Group, supervises the development of advanced logic circuit design. **Shuhei Iwade** is the manager of the System LSI Design R&D department. **Yasutaka Horiba**, deputy general manager of the Semiconductor Group, manages development of DSP and system-level LSIs.

The authors contributed in various ways to the development of the D30V/MPEG multimedia processor. **Hidehiro Takata** manages the Implementation Section of the Advanced Circuit Design Group. He helped implement the system LSIs, focusing on the clock skew analysis and the dynamic circuit design. **Tetsuya Watanabe**, a senior research engineer, contributed to the design of the digital PLL and the system LSI timing verification. **Tetsuo Nakajima** is a research engineer in Mitsubishi's LSI Development Center, where he worked on implementing the system LSIs.

Send questions and comments to Hidehiro Takata, System LSI Design R&D Dept., System LSI Development Center, Mitsubishi Electric Corp., 4-1 Mizuhara, Itami, Hyogo, 664-8641 Japan; takata@lsi.melco.co.jp.