10-12-00

A/PRov

PROVISIONAL APPLICATION COVER SHEET [37 CFR 1.53(c)]

This is a request for filing a PROVISIONAL APPLICATION under 35 U.S.C. §111(b) and 37 CFR 1.51(a)(2)

Date : October 10, 2000

Docket No. : 40715/SAH/B600

EXPRESS MAIL NO. EL496225445US

Mail to: BOX PROVISIONAL PATENT APPLICATION

INVENTOR(S)/APPLICANT(S) (LAST NAME, FIRST NAME, MIDDLE INITIAL, RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY)

Alexander G. Macinnis, Los Altos, California

____ Additional inventors are being named on separately numbered sheets attached hereto.

TITLE OF THE INVENTION (280 characters max)

SYSTEM AND METHOD FOR PERSONAL VIDEO RECORDING

ENCLOSED APPLICATION PARTS

_27_ Specification (number of pages)
_1_ Drawings (number of sheets)
____ Small Entity Statement
____ Assignment
____ Other (specify):

FEE AND METHOD OF PAYMENT

_X_ A check for the filing fee of $150.00 is enclosed.
The Commissioner is hereby authorized to charge any fees under 37 CFR 1.16 and 1.17 which may be required by this filing to Deposit Account No. 03-1728. Please show our docket number with any charge or credit to our Deposit Account. **A copy of this letter is enclosed.**

____ No filing fee enclosed.

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

_X_ No ___ Yes, the name of the U.S. Government agency and the Government contract number are:

Please address all correspondence to **CHRISTIE, PARKER & HALE, LLP, P.O. Box 7068, Pasadena, CA 91109-7068, U.S.A.**

Respectfully submitted,

CHRISTIE, PARKER & HALE, LLP

By _____

Art Hasan
Reg. No. 41,057
626/795-9900

PROVISIONAL APPLICATION FILING ONLY

SAH/plg
SAH PAS278953.1-*-10/10/00 8 03 PM

40715/SAH/B600

SYSTEM AND METHOD FOR PERSONAL VIDEO RECORDING

FIELD OF THE INVENTION

The present invention relates to the decoding and presentation of digital audio-video streams, and more particularly to a system and method for implementing features of personal video recording ("PVR") such as fast forward play and reverse play at various speeds, while maintaining adequate representation of the video.

BACKGROUND OF THE INVENTION

PVR is gaining popularity with television viewers. Examples of PVR systems are the TiVO system and the Replay system, available from Phillips Electronics and Sony. PVR allows a viewer to store the contents of a television broadcast and playback the broadcast contents at a later time. PVR systems typically receive and decode analog video from broadcast sources, compress the decoded video and audio into a digital format such as MPEG, and record the compressed data onto a hard drive. For playback, the systems read and decode the compressed digital video and audio from the hard drive and output the decoded video and audio in an analog format for display.PVR systems preferably include fast forward and reverse display functions, also known as "trick modes," whereby the viewer can reach a desired point in the broadcast by viewing the video pictures faster than the normal display rate or viewing them in reverse. It is particularly desirable that the PVR systems present the pictures during fast forward and reverse displays with a quality adequate to allow the viewer to readily locate the desired point in the broadcast. The fast forward display and reverse display are sometimes referred to as "cue" and "review".

Present PVR systems typically perform the function of encoding the video into a digital format, that allows for cueing and reviewing during playback of the stored contents. In digital cable systems, for example, the video is encoded in a digital format before it ever reaches the PVR system. Such a format may not be conducive to enabling cue and review with adequate picture quality. For example, performing trick modes with no-I-picture digitally encoded streams, such as General Instrument/Motorola's progressive refresh encoded streams or streams provided by AT&T's Headend in the Sky ("HITS") service, is generally more difficult than with MPEG video using a conventional IBP GOP structure, since in progressive refresh encoded streams or HITS streams there are no I pictures to which the decoder can perform random access.

SUMMARY OF THE INVENTION

The present invention provides a system and method for PVR to record digitally encoded data streams, and to playback and decode the recorded streams. In a preferred embodiment, the system implements PVR features in conjunction with a digital cable box or other source of digitally encoded video, such as a network server. The invention applies to all types of digital audio-video streams, whether they have I pictures or not. It may be used with streams that have no I pictures and that have a "progressive refresh" pattern of using intra-slices (I-slices). The system preferably enables "trick modes", i.e., fast forward and reverse displays at various speeds, and preferably produces an adequate representation of the video during the trick modes of operation.

In accordance with one embodiment of the present invention, the system enables recording and playback of no-I-picture digital encoded streams by decoding video pictures faster than the display rate, thereby enabling trick modes. Using this fast

decode method, streams may be played at faster than normal speed by decoding the entire stream at a faster rate and displaying only those pictures that are aligned appropriately with the available display times. Reverse decode and display may be implemented by starting at a suitable entry point, decoding quickly up to the point of the desired picture, and displaying the result, and repeating the process. Index tables of entry points may be required for proper operation in some modes. These tables point to what may be referred to in the context of the present invention as E Pictures or "Entry Pictures", i.e., those whose first row is an I-slice starting the pattern of progressive I-slices.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a timing diagram of decoding pictures having a pattern of progressive I-slices in one embodiment of the present invention; and

FIG. 2 is a timing diagram of reverse decoding and displaying pictures having a pattern of progressive I-slices in one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides a system and method for PVR to record digitally encoded data streams, and to playback and decode the recorded streams. In a preferred embodiment, the system implements PVR features in conjunction with a digital cable box or other source of digitally encoded video, such as a network server. The invention applies to all types of digital audio-video streams, whether they have I pictures or not. It may be used with streams that have no I pictures ("no-I-picture streams") and that have, for example, a "progressive refresh" pattern of using intra-slices (I-slices). The system preferably enables "trick modes", i.e., fast forward and reverse displays at various speeds, and preferably produces an adequate representation of the video during the trick modes of operation.

Exemplary No-I-Picture Progressive Refresh Coding Structure

The system may receive MPEG video coded using the "progressive refresh" coding structure. The term "progressive" as used herein is not to be confused with progressive picture coding, i.e. non-interlace frames. The "progressive" method is also known as "all-P" (actually, it is "no I pictures", with some additional constraints.) The HITS (Headend In The Sky) system often uses the progressive refresh coding structure, so sometimes progressive may be referred to as "HITS" for brevity, although HITS does not necessarily mean progressive refresh, nor are all progressive refresh streams from the HITS system. The progressive coding is specified as one mode that General Instruments/Motorola encoders operate. Progressive coded streams are proper MPEG-2 video streams, with some special constraints.

Progressive coding is described as follows:

- There are no I pictures. This means there are no pictures that can be decoded without depending on other decoded pictures.

- There are of course P pictures (predicted pictures), and optionally there may be B pictures (bidirectional pictures).
- Within P pictures, there are some rows that are coded as intra-slices (I-slices). There is a specified pattern for coding I-slices, with options in this specification.
- Anywhere from 1 to 9 rows are encoded as I-slices in each P-picture; this number is called the "refresh depth". A typical default setting for the encoder, for example, is M=3 (PBBPBBPBB pattern) and refresh depth of 6.
- I-slices follow a consistent pattern; this applies only to P pictures, not to B pictures. Call the value of refresh_depth "D". In one picture, the first D rows are I-slices. The remaining rows are predictively coded, i.e. P-slices. In the next picture, the first D rows are not I-slices, and the second D rows are. In the 3rd picture, the first 2*D rows are not I-slices, and the 3rd D rows are. This continues until all rows in the consecutive sequence of pictures have been represented in I-slices. In addition, there is a constraint on motion vectors. For each picture where the I-slices are not at the top of the picture, the motion vectors are constrained to point only at rows that are at the same vertical height or higher, that is, they can only point at rows that have been refreshed via I-slices since the most recent picture with the first row being an I-slice.

This means that in order to perform a random access into a progressive stream, if the decoding starts with a picture whose first row is an I-slice, then successive P pictures can be decoded until a picture is decoded where the bottom row is an I-slice, and from that picture on, the resulting pictures are fully decoded and capable of being presented.

Within the context of this invention, a picture whose first row is an I-slice may be referred to as an "entry picture" or an

"E picture". A set of pictures starting with an entry picture and ending with the last picture before the next E picture may be referred to as a "block of pictures" or "BOP". The last picture in a BOP is the first "fully decoded" picture in the BOP.

If a random access starts anywhere other than at an E picture, there is no value in decoding the pictures before the next E picture, since the picture before the first E picture cannot be fully decoded, since some or many of the rows are predicted from, or the prediction predecessors are predicted from, macroblocks that were never decoded. Therefore, in a preferred embodiment, the most efficient method for performing random accesses into a progressive stream is to always start with an E picture.

Fast Forward

To decode no-I-picture video streams in fast forward, the compressed video is read from disk, moved to the decoder, and decoded at rates that are preferably faster than the normal picture rate by a ratio equal to the desired fast forward ratio. One suitable decoder is contained in the BCM7030 graphics display system, marketed by Broadcom Corporation, Irvine, California, aspects of which are described in U.S. patent appplication (Atty docket 36971/SAH/B600), filed August 18, 2000, the contents of which are hereby incorporated by reference. The BCM7030 graphics display system can decode I and P pictures at up to nine times the normal video display rate. The invention may be used with any other system that provides fast decode, or may be used with conventional decoders, with a commensurate degradation of picture quality.

If the no-I-picture video stream has B pictures, those pictures can be skipped entirely unless they are to be displayed, and in general B pictures need not be displayed. Therefore the

rate at which the decode and display process progresses through the stream may be:

- 9x for streams with no B pictures, and without skipping compressed data
- 27x for streams with B pictures (M=3), and without skipping compressed data
- Arbitrarily fast rates with skipping compressed data

Here we assume that desired fast forward rates are 3x and 10x normal speed, and that streams with B pictures are normally available. Slightly slower rates for the higher speed, i.e. 8x or 9x, are acceptable.

In fast forward without skipping data, the process is prerably as follows:

- As in all PVR, the rate of decoding and display is driven by the decoder, not by the data source, which in this case is a hard disk drive (HDD)
- The video decoder decodes MPEG video as fast as it can for the desired number of coded pictures per displayed picture. For example, for 10x fast forward, for every displayed picture, the decoder preferably decodes 10 coded pictures and prepares the last one for display. In cases where the stream has B pictures, the B pictures may be skipped rather than being decoded. If the display picture falls on a B picture, the nearest I or P picture may be decoded and displayed instead.
- As the compressed data is evacuated from the compressed data buffer and from the PVR playback buffer, stream data is demanded and read from the HDD. The rate of reading data from the HDD is nominally increased from the normal data rate by the fast forward ratio, although this rate is not necessarily a constant due to several factors.

Details of decode buffer management and data transfer are discussed in a later section.

In fast forward at faster rates requiring data skipping, the preferred process is as follows:

- An index table is preferably used to indicate where the entry pictures are in the compressed stream. Depending on the specifics of the stream, entry pictures may typically be spaced between 30 and 15 pictures apart.
- For each picture to be displayed, the data read from the HDD and placed in the decode buffer starts with the beginning of an entry picture. The entry picture and successive pictures are decoded until a completely reconstructed picture has been produced and is ready for display. The number of pictures that need to be decoded ranges, for example, from 5 (refresh depth = 6) to 30 (refresh depth = 1). The number of coded pictures that are passed (decoded or skipped) ranges, for example, from 10 (no B pictures, refresh depth = 9) to 30 (either no B pictures, refresh depth = 1 or with B pictures, M = 3, and refresh depth = 3).
- Note that it is not necessary to decode all slices of all pictures. For the entry picture, only the I-slices (rows) at the top of the picture - the number of rows equal to the refresh depth – need to be decoded. For the second picture, the (2 x refresh depth) rows at the top need to be decoded, and so on until I-slices have been decoded at the bottom of the picture. On average, this allows the decoder to progress through the pictures at twice the rate than it would otherwise, assuming the decoder can find the start of each picture without parsing all of each picture.
- Once a set of pictures, starting from an entry picture and ending with a displayable picture, has been decoded, a new

set of pictures is decoded, starting with a new entry picture.

Reverse

The system preferably provides for trick mode PVR with no-I-picture streams in reverse, from slow speeds such as 1x to higher speeds. Decoding of MPEG video generally always progress forwards, even when the desired direction of pictures to be displayed progresses backwards. The method of the present invention involves starting at entry frames, decoding forwards until the desired fully decoded picture is reached, and displaying the resulting picture while again decoding a portion of the stream, stopping the decode process at an earlier picture, and possibly starting the decode process from an earlier coded picture if necessary to achieve a fully decoded picture. It may or may not be possible to achieve 30 unique displayed pictures per second with a 1x reverse play rate; this depends on the structure of the video stream, including the refresh depth, and the decode performance. The process is illustrated in FIG. 1 as follows:

- Begin decoding at an entry picture, as indicated by the index table (entry picture A in FIG. 1). This entry picture precedes the first picture that can be displayed by, for example, a difference of 10 to 30 pictures, depending on the refresh depth and the value of M (one more than the number of consecutive B pictures every P or I pictures). The entry picture precedes the picture that is desired for display by a figure that may be greater than the above - typically up to 53 pictures - depending on the relationship between the desired picture and the location of the entry pictures. That is, there may be another entry picture between the first picture decoded and the picture desired for display, but if decoding starts with the second entry

picture, the result would not be a completely decoded picture.

- Decode forwards from the entry picture until the desired picture has been decoded, and display the result. B pictures need not be decoded, unless the picture to be displayed is a B picture. Note that not all slices need to be decoded, just as in the case of fast forward at high rates while skipping data. Decoding only the necessary slices can approximately double the average decode rate.
- The compressed data is preferably retained in a DRAM buffer after it has been decoded, as it will be decoded again.
- Optionally, and for best performance, one intermediate decoded picture can be stored in memory to facilitate the decoding of future pictures. The picture to be stored is the last picture in the first set of pictures, starting with the first entry picture, before the 2nd entry picture. This is the same as the first fully decoded picture as illustrated in FIG. 1.
- Once the first picture has been displayed, again begin decoding, either from the same entry picture as before, or from the entry picture that precedes the previous entry picture. The same entry picture as before is used again if the final decoded picture can be completely decoded, as determined by the structure of the bit stream, but if this is not possible, then decoding starts at the next earlier entry picture. For example, it is generally not possible to completely decode and display a picture between the entry picture A and the first fully decoded picture if decoding starts at the entry picture A in FIG. 1. To completely decode the any picture between the entry picture A and the first fully decoded picture, the decoding preferably should start at an entry picture earlier than the entry picture A.

- The process repeats for each picture to be displayed. Since decoding of the requisite number of pictures may take more than the display time of a picture, the currently displayed picture may be displayed for 2, 3 or more frame times. For each decode sequence, the entry picture where the decoding process starts may be the same as, or an earlier entry picture than the previous one, as required to ensure a correctly decoded picture results.

- If the optional method is implemented whereby the first fully decoded picture is stored, the successive pictures in reverse order can be decoded by starting from the entry picture following the reference picture (start from Entry Picture B in FIG, 1) and proceeding to the picture to be displayed. This can make a substantial reduction in the amount of decoding that is required to produce some pictures for display. However, once the reverse play progresses past the decoded reference picture, decoding generally must start again from a new, earlier entry picture as indicated above.

Scenario for Reverse Play

FIG. 2 illustrates exemplary decoding for reverse play by the system. The first picture to be displayed is picture B in FIG. 2. In order to decode B, decoding must start from picture A, an entry (E) picture 2. In order to display picture C, normally decoding must again start from picture A and continue through picture C. This process repeats until picture D has been decoded and displayed. In order to decode picture E, decoding must start at picture F, an E picture 1, which is earlier than the picture A, and proceed forwards up to picture E.

In order to display one new picture in reverse order, the complete previous block of pictures (BOP) starting from an entry

picture (E picture) generally must be decoded (the B pictures, if any, do not need to be decoded), in addition to all of the P pictures leading up the picture to be displayed.

An optional technique that can save a substantial amount of decoding in some cases, involves the expense of one additional frame buffer. In FIG. 2, if picture D is stored during the first decoding pass starting at picture A, then decoding of picture B can start using picture D as a reference and decoding the successive pictures until picture C is decoded. For each picture to the left of picture C and to the right of picture D, decoding can start with picture D once picture D has been fully decoded. After picture D has been displayed, decoding generally must start at picture F in order to decode and display picture E. If the decoded version of picture G is stored in a frame buffer, the process can repeat for pictures to the left of picture E, and so on.

Index Tables

Benefits of Index Tables

In order to perform reverse play with predictable order and rate of pictures, and reasonable efficiency, decoding preferably starts from carefully chosen entry points, namely entry pictures. In order to do this, the required coded data should be in a buffer ready to be decoded, and the decoder must know where the appropriate E picture is in the buffer. This typically requires an index table indicating where these E pictures are, on the disk and in the buffer.

A signficant performance improvement can be had if the index table indicates the starting location of every coded picture. Only some of the rows need to be decoded in some pictures - the top D (refresh depth) rows of an entry picture, the top 2*D rows from the next picture, and so on. In order for the decoder to

skip the rows that do not need to be decoded, and go directly to the next picture, it is preferable that the decoder knows where each picture starts. This information can be obtained from an index table or from a start code (row) table created by the transport engine in a preferred embodiment.

System with No Index Table

Alternatively, the system can be implemented without the use of index tables created before playback. In this alternative method, in the decode process the host obtains data from the hard disk and passes it to the decoder for analysis. Typically the decoder can scan through the compressed data stream and search for picture start codes, slice start codes and other data that help to identify entry points in the stream, much faster than the decoder can perform a full decode of all the compressed video data. As before, we can consider the various specific cases that make up all the possible decode scenarios.

Normal or Slow Forward Playback

In normal or slow speed playback, after starting decode there is no need to identify entry points. Therefore it does not matter whether there is an index table or not once decoding of the stream has started, and this method is essentially the same as methods that use index tables. Start of decode can begin at any point, which may be the start of a file representing a short clip, or an arbitrarily selected point in the video stream, or a point indicated by any means, including a directory. The directory, if there is one, need not indicate all or any entry points, it can point to any point in the stream where a user may be interested in starting decode, such as the start of a major scene.

Fast Forward without Skipping

In fast forward playback where skipping of data is not required, this method is again similar to methods that use index tables, since the decoder can process all contiguous compressed data. It is not necessary for the decoder to know the locations of entry points in the stream.

Fast Forward with Skipping

In this case, the decoder does not decode all of the contiguous compressed data (in addition to potentially skipping B pictures if they exist); rather, it skips some compressed data between instances of decoding pictures. The decoder can benefit from knowing where to re-start decoding after skipping data. This case can be considered to be two separate scenarios: One in which the decoder scans all the compressed data and finds entry points, while decoding only some of the data, and another case in which the decoder does not scan all the data - typically some of the data is not even read from the HDD.

In the first scenario, a contiguous compressed data stream is fed to the decoder by the host. The decoder first scans all the data and finds significant entry points, such an I pictures or E pictures, possibly with the assistance of the host, in much the same way that an index table would be built during record time. The locations of the entry points found are recorded by the decoder, and optionally reported to the host. The decoder decodes a picture starting at one entry point, and then skips ahead by one or more entry points and decodes starting at another entry point, using a location just previously found. The process repeats as long as indicated by the host. The selection of which entry point to start decoding each time may be made by the host, using the entry points just found and indicated to the decoder via an API, or it may be made by the decoder, responding to a

more general command from the host such as "play fast forward at 20x normal speed". In the latter case the host does not need to know the locations of the entry points.

In the second scenario, i.e. when the decoder does not receive nor scan all of the contiguous data, typically the host retrieves compressed data from the hard drive according to some pattern which may be arbitrarily unrelated to the location of entry points in the stream. For example, the host may retrieve one complete cylinder of data from the hard drive, then skip a cylinder, then read the next cylinder, etc. The data that is fetched from the HDD is passed to the decoder. The decoder then scans the data as above, searching for entry points. The decoder decodes starting at entry points just found. Alternatively, the decoder need not scan all the data it receives, rather it can decode from the beginning of each block of data received and decode until it has decoded a fully decodable picture; however this method is less efficient in terms of decoder performance. The main drawback to perform fast forward while skipping data on the HDD is that the temporal spacing of the pictures that are retrieved, decoded and displayed may not be uniform, resulting in uneven-ness of motion in the displayed result. On the other hand, skipping entire cylinders, for example, is a very efficient way of retrieving the compressed data stream from the HDD very quickly.

Reverse Play without Skipping

As in the case of fast forward without skipping, in this case the host feeds all of the compressed data to the decoder, and the decoder scans the data searching for entry points. Of course, since this is reverse play, the data is typically fed to the decoder in blocks, with each block being fed in forward order, and succeeding blocks starting earlier in time, i.e. the order of the blocks themselves is reversed from normal forward

play. As the decoder finds entry points, it keeps a record of them, and optionally passes this information to the host. As above, the decoder performs decoding in a way that produces reverse play at the desired speed. In the case of progressive-I-slice streams (no-I-picture streams), decoding succeeding pictures for display may require decoding the same data multiple times, starting at the same E picture. After some number of decoding passes starting at the same E picture, the decoder begins decoding at the next earlier E picture. The identification of which E picture (or other entry point e.g. I picture) at which to start decoding may be made by the host, using the locations of the entry points just found and communicated by the host, or it may be made by the decoder, optionally via an API indicating abstractions such as "start decoding at the next earlier entry point" or more abstract commands such as "play in reverse order at 1x speed", leaving more decisions up to the decoder. In the preferred implementation, when the decoder scans the compressed data and finds entry points, it indicates the addresses of these entry points to the host, and for decoding and display the host indicates the starting point for each decoding pass.

Reverse Play with Skipping

This case is similar to that of fast forward with skipping, with a few differences. The main difference is the operation when the compressed video stream is a no-I-picture stream (e.g, "HITS" stream). The method is similar to fast forward with skipping. As in the case of fast forward, in one embodiment, all the data is fed to the decoder (albeit with the blocks in reverse play order), and in another embodiment in which not all the data is fed to the decoder, e.g. if not all the data is read from the hard disk.

Data is preferably fed to the decoder in blocks. The decoder scans through the data and searches for entry points. Searching

is performed in the forward order of the data stream. The entry points found are recorded, and optionally reported to the host. The decoder may decode one or more pictures starting at each entry point, particularly in the case of no-I-picture streams. Succeeding entry points are in the reverse order from normal play. The decoder may start decoding from non-consecutive entry points in order to implement faster reverse play speeds. The entry points from which to start decoding may be specified by the host, using the locations of entry points just found, or they may be decided by the decoder. In the preferred implementation, the host indicates the location for each decode pass, using the location of entry points searched by the decoder and communicated to the host

The data blocks fed to the decoder may make up a contiguous data stream (even though the blocks are in reverse order, so the data is not received contiguously), or it may make up a non-contiguous stream, i.e with sections of data skipped between blocks. This would typically occur if the host does not read all the data from the HDD, e.g. in order to implement faster speeds of reverse play. The operation is essentially the same whether or not data is skipped as it is fed to the decoder, except that there may be entry points from which no pictures can be fully decoded, especially in the case of no-I-picture streams if the last picture in a BOP is not fully contained within the blocks of data sent to the decoder. The decoder or (preferably) the decoder in conjunction with the host find and examine entry points within the compressed bit stream, and decide which entry points can productively be used for decoding. Preferably the host directs the decoder at which entry points to start decoding via an API. Alternatively the decoder decides at which entry points to start decoding.

Compressed Data Buffer Management

For PVR playback, whether the stream is a GOP structured stream with I pictures, or a no-I-picture stream (e.g., progressive I-slice (HITS) stream without I pictures), management of compressed data buffers is important, especially for fast forward and reverse play.

The following principles and guidelines may assist in the design of the system:

- It is OK to interrupt the host once per display field time and to expect the host to answer each interrupt within one field time, but it is not reasonable to expect very fast interrupt response from the host.
- Compressed data is laid out on the disk (HDD) in ways that are suitable for the HDD deisgn and for high performance of the disk. Therefore we cannot assume any alignment of compressed pictures and disk sectors. Any compressed picture may span two or more sectors.
- Memory buffers in the host, which hold data read from the HDD, are designed for system efficiency and performance. The size of these buffers is generally a power of 2, and they are designed to align well with HDD sectors. Therefore any compressed picture may space two or more buffers. Any compressed picture may start anywhere within a buffer.
- For some trick modes, such as fast forward at rates beyond the fast decoding rate, and for reverse play, it may be necessary for the host to skip sectors on the HDD, and it may be necessary for the decoder to skip some data in some buffers or to skip some buffers entirely.
- Video decoders normally have compressed data buffers (CDB); there may be a significant difference between the time compressed data enters the CDB and the time that same data

is decoded and possibly displayed. Data generally must be moved from a source, e.g. the host's data buffers, into the decoder's CDB, before the data can be decoded. Moving data into the CDB can be done by the decoder alone, by the host or other device, or by a combination. The preferred solution is for the decoder to move the data from the host buffers into the CDB. It's possible for a decoder's CDB to be the same memory that makes up the host's buffers (i.e. for HDD access), but this is not preferred.

- The compressed data stored on the HDD may be Elementary Streams (ES), sequences of PES packets, Transport Streams (TS), or possibly modified versions of any of these or other formats.

In view of the above factors, the preferred design for managing compressed data buffers is to use a linked list of buffers. Each buffer is of some power of two size, such as 32kB or 64kB. Larger logical buffers are constructed by linking buffers together. Smaller logical buffers are created by using portions of the fixed size buffers and pointing to the required data. Linked lists are constructed through the use of a descriptor list, where each descriptor points to a buffer and to the next descriptor. In one embodiment, video Elementary Streams (ES) are stored on the HDD, put into host buffers, retrieved by the decoder, and decoded. Elementary streams may be modified from the standard MPEG definition, e.g. to include PTS and DTS time stamp information, and possibly PCR or SCR or equivalent time stamp information. In another embodiment, transport streams (TS) are stored on disk and played from disk. In some cases the data transport hardware (outside of the decoder proper) records data to memory and plays data from memory using the linked list buffer structure. For record, this hardware can perform searching of key data elements to create index tables, and it can also scramble or encrypt the data to be recorded, after optionally descrambling

(or decrypting) the scrambling applied to the data before it is received, typically the network conditional access scrambling. For playback, this hardware can descramble the data and pass data to the decoder, including optional flow control to avoid buffer overflow in the decoder.

There are at least two implementations that may be desired. In one embodiment, the preferred design is to have decoder pull data from the buffers, using the linked list to direct the addressing of the buffers. In another embodiment, the preferred design is to have another block, the "data transport", retrieve data from the linked list buffers and feed it to the video decoder. The same linked list structure with the same descriptor list may be used in both embodiments.

Linked List Definition

Linked lists are controlled using a list of buffer descriptors. The preferred buffer descriptor format is indicated here:

Word	Element	Bits	Description
0	reserved	31:26	Reserved
0	Start Address	25:0	This is the buffer start address. It is a byte address. Byte resolution is to be supported.
1	reserved	31:26	Reserved
1	Buffer Length	25:0	This is the buffer length in bytes. Byte resolution is to be supported.
2	Interrupt Enable	31	If this bit is high, an interrupt is generated when the record/playback circuit has finished processing this descriptor

2	reserved	30:0	Reserved - This could be used as an interrupt count value if that feature is needed. In the BCM7030, playback circuit only interrupts at the end of a buffer playback.
3	reserved	31:26	Reserved
3	Next Descriptor Address	25:4	This is the DRAM address of the next descriptor. It is a 16-byte aligned address. Descriptors can only begin at 16-byte aligned addresses (i.e. bits 3:0 of the address are 4'h0, and are not part of the descriptor field).
3	reserved	3:1	Reserved
3	Last Descriptor Indicator	0	This field is programmed with the value 1'b0 when the Next Descriptor address is valid. This field is programmed with the value 1'b1 when this is the last descriptor in the link list.

Filling Buffers from Hard Disk

Playing from Buffers

On playback from linked list buffers, there are several distinct cases to consider:

- Forward direction playback of contiguous data. This includes normal speed play, slow forward play, pause, and fast forward play using the method of decoding at faster than normal rates without skipping data (except B pictures may be skipped).

- Fast forward playback while skipping data. In this case the decoder skips some data that is on the hard disk, and in general it skips data that is in the buffers. For example, I frames may be decoded while ignoring the pictures that intervene between I frames; blocks of pictures (BOPs) may be decoded while skipping some BOPs.

Contiguous Forward Decode

For the first case, decoding contiguous data in the forward direction, buffer management may be accomplished as follows. The linked list of buffer acts like a FIFO, and in fact this mode could be implemented as easily using a FIFO structure. The decoder starts decoding from some initial entry point, and thereafter it follows all of the data in the buffers, in order. The linked list is followed from the first buffer, one buffer after the other, in the order specified in the list. The preferred means to control the decoder is for the host to tell the decoder, via an API call, to start decoding from a specified point in the linked list's descriptor list, which may include an offset into the first buffer. The decoder follows the linked list. There may be additional API functions for the host to indicate to the decoder the intended decode rate, and functions for the decoder to tell the host its current progress through the stream, where it is currently in the linked list, and other information. The host and the decoder may interact dynamically to adjust such parameters as the desired decode rate, either to adapt to an inability of the decoder to maintain a desired rate, or to change the intended rate of play.

Fast Forward with Skipping

In the second case, the decoder skips some data in the buffers, and it decodes only targeted pictures or sets of pictures (or BOPs). In the preferred method the host tells the decoder, via

an API call, where in the buffer set to start decoding, for every instance in which data is to be skipped. For example, the host may wish the decoder to decode all I pictures, and only I pictures. Assuming the host has an index table from which it can determine the location within the buffers of the compressed I pictures, the host tells the decoder to decode a first I picture which starts at or following a specified location in a specified buffer in the linked list; and then to decode a second I picture at another location, and so on. The host may tell the decoder to decode one picture, or more than one picture for every specified entry point, and in case the decoder is to decode more than one picture the host may optionally instruct the decoder to skip some pictures. The host typically tells the decoder to start decoding at a specified point in the linked list of buffers, while the decoder typically stops decoding at a picture that is implied parametrically, rather than a point in the data stream that is spelled out by the host. For example, the host may tell the decoder to stop at the first fully decoded picture (the end of a BOP in the case of a no-I-picture (e.g., HITS) stream, or it may tell the decoder to stop at the Nth picture after the entry point. For no-I-picture (HITS) streams, the decoder typically decodes a BOP after each skip. It is most efficient to start decoding at an E picture, and to complete decoding with the first fully decoded picture. It is of course possible to start decoding before an E picture, but this is not preferred as there is no benefit. It is possible to continue decoding beyond the first fully decoded picture, in case the desired picture to display is further into the stream. The host uses its knowledge of the decoder's progress through the linked list as well as its progress in decoding the stream in order to know when to indicate to the decoder what to do next. For each instance in which the host instructs the decoder to decode one or more pictures starting from a specified location in a linked list, the decoder

may have to follow the linked list through two or more buffers in the list, and it may stop before retrieving all the data from the last buffer in the list.

Reverse Play

In the reverse play case, more considerations and complications come into play. For reverse play using only I pictures, the process is very similar to fast forward with skipping. This typically results in very fast, or jerky, reverse play. For smooth normal speed reverse play, preferably the decoder starts at an entry point - this may be an I picture, or an E picture, depending on the stream construction - and decodes forward to the desired picture to display, and then decodes again from the same entry point and decodes forward again but stopping at an earlier picture, as described in previous sections of this document. The same compressed data is re-used. This requires the behavior of the data buffers to be different from that of a FIFO, in which the data would be discarded after it is decoded once. Continuing with normal speed reverse play, after decoding some of the same data multiple times, the decoder begins decoding at an earlier point in the stream, and in some cases it continues to decode from the earlier data up through some of the later data that has been previously stored and decoded. This process involves pre-pending coded data to the sequence of data to be decoded. In the preferred design, both the re-use and the pre-pending of data are implemented through the use of the linked list of buffers. The host modifies the descriptor list as required, and for each time the decoder starts decoding from a specified entry point in the buffers, the host directs the decoder to start decoding from a specified point in the linked list. As in the case with fast forward (above), the host tells the decoder not only to start decoding from a specified point in the list, but also to stop at a picture that may be indicated

indirectly, such as at the first fully decoded picture, or at the Nth picture, or at the Nth picture after the first fully decoded picture.

The decoder can optionally make use of a fully decoded picture as a starting point for additional decoding, particularly in reverse play of no-I-picture streams. In the preferred design, when this feature is implemented, the host directs the decoder to begin decoding at the start of a BOP immediately following the compressed data associated with a previously decoded and stored fully decoded picture, while indicating also the re-use of the stored decoded picture. The host commands the decoder to do this via an API call.

WHAT IS CLAIMED IS:

1. A method for enabling trick modes in a personal video recording system, the method comprising the steps of:

receiving a digitally encoded stream including video pictures for display at a display rate

decoding the video pictures at a rate faster than the display rate;

displaying video pictures that are aligned with available display times.

2. The method of claim 1, wherein the personal video recording system is implemented in a digital cable box.

3. A method for enabling reverse play in a personal video recording system, the method comprising the steps of:

(a) receiving a digitally encoded stream including video pictures and

(b) decoding the video pictures starting at an entry point;

(c) decoding up to a desired picture;

(d) displaying the decoded pictures;

(e) repeating at least steps (b), (c) and (d).

4. The method of claim 3 further comprising the step of accessing an index table of entry points, wherein the entry points point to entry pictures containing rows, and wherein a first one or more rows of the entry pictures is an I-slice which starts a pattern of progressive I-slices.

5. A personal video recording system for storing and playback a digitally encoded data stream, the personal video recording system comprising:

means for storing the digitally encoded data stream, including video pictures for display at a display rate;

means for playing back and decoding the video pictures; and

means for selecting an entry point at which decoding of video pictures can start.

6. The personal video recording system of claim 5 wherein the means for selecting an entry point includes an index table that associates the entry point with a particular video picture.

7. The personal video recording system of claim 5 wherein the means for playing back and decoding the video pictures is capable of playing back and decoding the video pictures at a rate faster than the display rate.

8. The personal video recording system of claim 7 wherein the particular video picture associated with the entry point is an entry picture, and wherein one or more first rows of the entry picture is an I-slice which starts a pattern of progressive I-slices.

9. The personal video recording system of claim 8 wherein fast forward display of the video pictures is implemented through random access of the entry picture when the digitally encoded data stream does not contain an I-picture.

10. The personal video recording system for claim 8 wherein reverse display of the video pictures is implemented through random access of the entry picture when the digitally encoded data stream does not contain an I-picture.

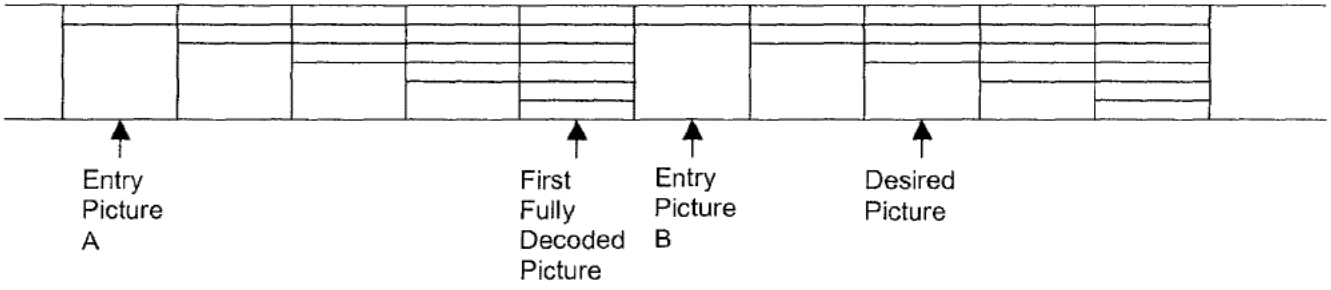

FIG. 1

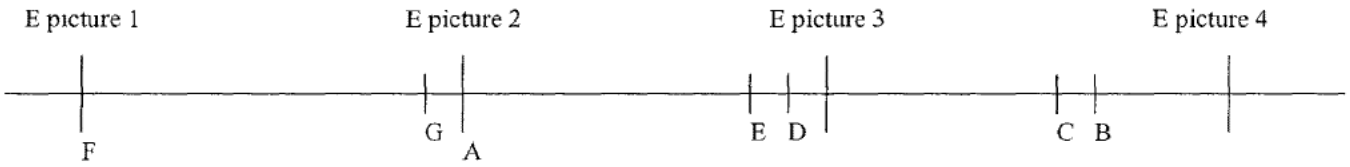

FIG. 2