means for evaluating the headers to determine the formats of the corresponding packets, and

means for causing the packets to arrive at the module corresponding to their formats.

31. The system according to claim 29 further including

a time-division-multiplexed bus connecting the voice data processing modules and the router.

32. The system of claim 29 wherein the first means includes

means for receiving the first data stream,

means for processing the first data stream to extract information according to the format corresponding to that module, and

means for formatting the extracted information in the intermediate format.

33. The system of claim 32 wherein the first means also includes

buffering means for rate adaption.

34. The system of claim 32 wherein the first means also includes

means for performing cell delay variation compensation.

35. The system according to claim 32 further including

a time-division-multiplexed bus connecting the voice data processing modules and the router.

36. The system of claim 35 wherein the means for formatting the extracted information into the intermediate format includes

means for placing the extracted information into appropriate slots in the time-division-multiplexed bus.

37. The system of claim 32 wherein the second means includes

means for assembling needed information from the intermediate format,

means for organizing the assembled information according to the format corresponding to this module, and

means for placing the organized information into the second data stream.

38. The system according to claim 37 further including

a time-division-multiplexed bus connecting the voice data processing modules and the router.

39. The system of claim 38 wherein the means for assembling includes

means for selecting the information from appropriate slots of the time-division-multiplexed bus.

40. The system of claim 29 further comprising

a time slot interchanger coupled to the voice data processing modules.

41. The system of claim 29 wherein the switch includes

an egress control circuit for arbitrating among the voice data processing modules.

42. The system of claim 29 wherein the router includes

ATM ingress circuitry coupled to the voice data processing modules.

43. The system of claim 42 wherein the ATM ingress circuitry includes

header extraction circuitry for extracting a VPI/VCI header from the first data stream.

44. The system of claim 43 wherein the ATM ingress circuitry includes

means for forming module identification signals from the extracted header.

45. The system of claim 43 wherein the ATM ingress circuitry includes

means for forming data type signals from the extracted header.

46. An interworking device receiving a first data stream in a plurality of formats, the device comprising:

a plurality of data processing modules capable of operating in parallel, each of the modules including

first means for converting the first data steam from one of the plurality of formats to a data stream in an intermediate format, and

second means for converting data streams in the intermediate format to a second data stream;

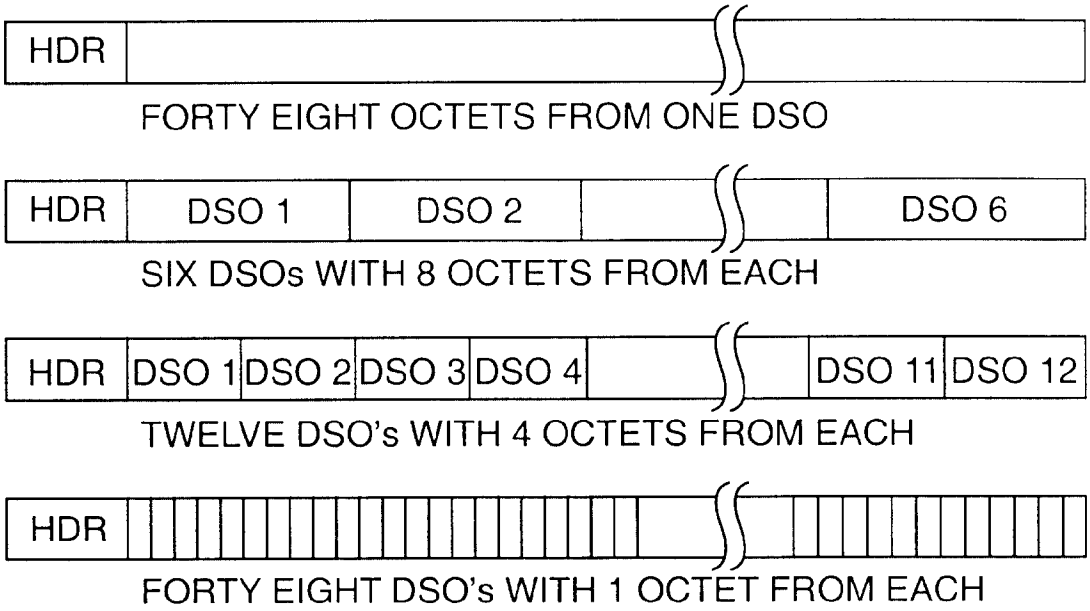
a router for sending to the appropriate one of the modules portions of the first data stream with the corresponding formats; and

a switch for switching the data stream in the intermediate format between different ones of the modules.

SMART & BIGGAR

OTTAWA, CANAGA

PATENT AGENTS



DELAY AND BANDWIDTH REQUIRED FOR DIFFERENT SIZE "TRUNKS" IN THE VOICE OVER ATM MULTIPLEX FORMATS

NUMBER OF DSO's PER CELL	NUMBER OF BYTES FROM EACH DSO	ATM BANDWITH PER VC	DELAY IN mSec
1	48	70,667	6
2	24	141,333	3
3	16	212,000	2
4	12	282,667	1.5
6	8	424,000	1
8	6	565,333	0.75
12	4	848,000	0.5
16	3	1,130,667	0.375
24	2	1,696,000	0.25
48	1	3,392,000	0.125

FIG. 1

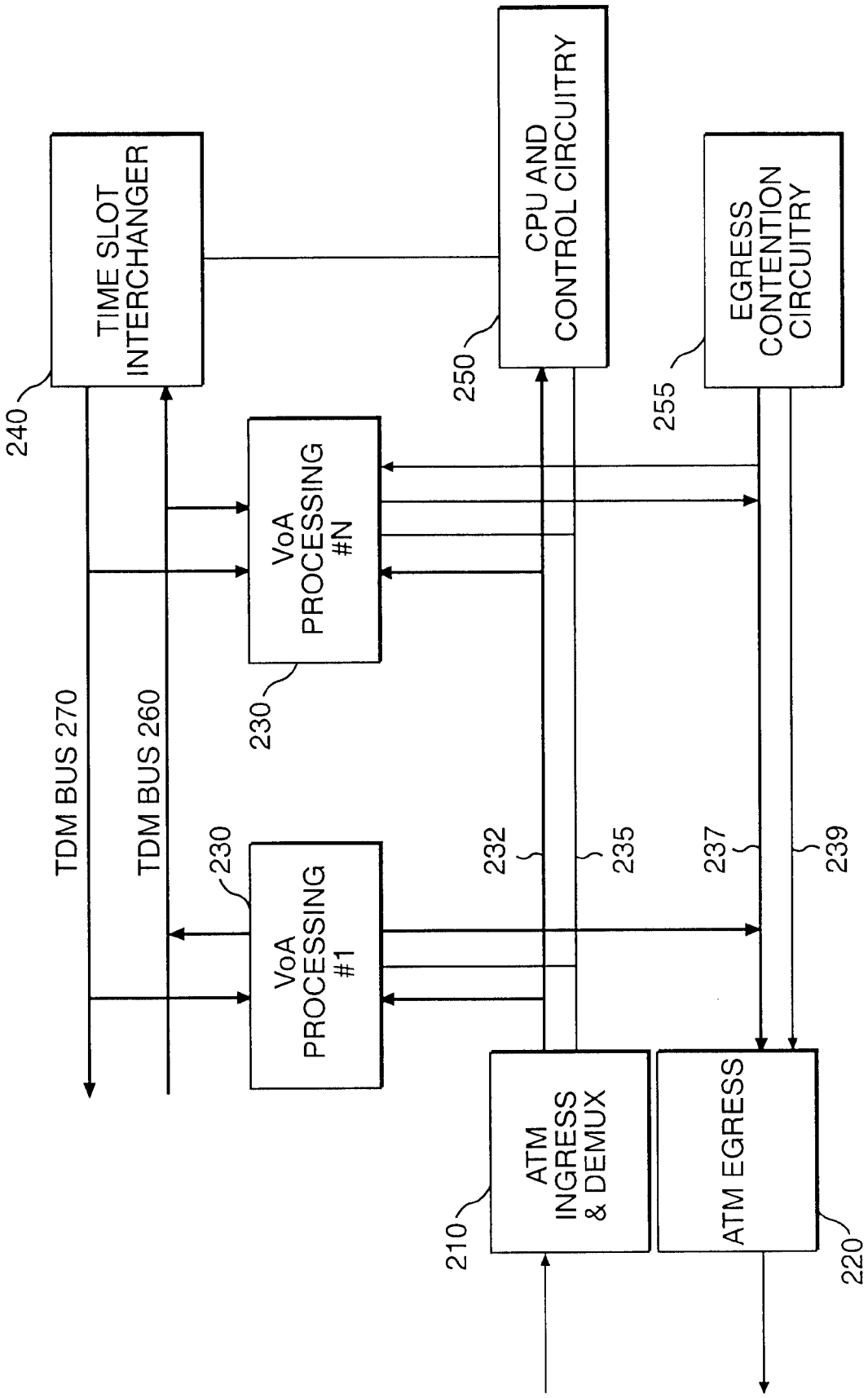

TIME SLOT INTERCHANGER
240
CPU AND CONTROL CIRCUITRY
250
EGRESS CONTENTION CIRCUITRY
255
VoA PROCESSING #N
230
VoA PROCESSING #1
230
TDM BUS 270
TDM BUS 260
232
235
237
239
ATM INGRESS & DEMUX
210
ATM EGRESS
220

FIG. 2

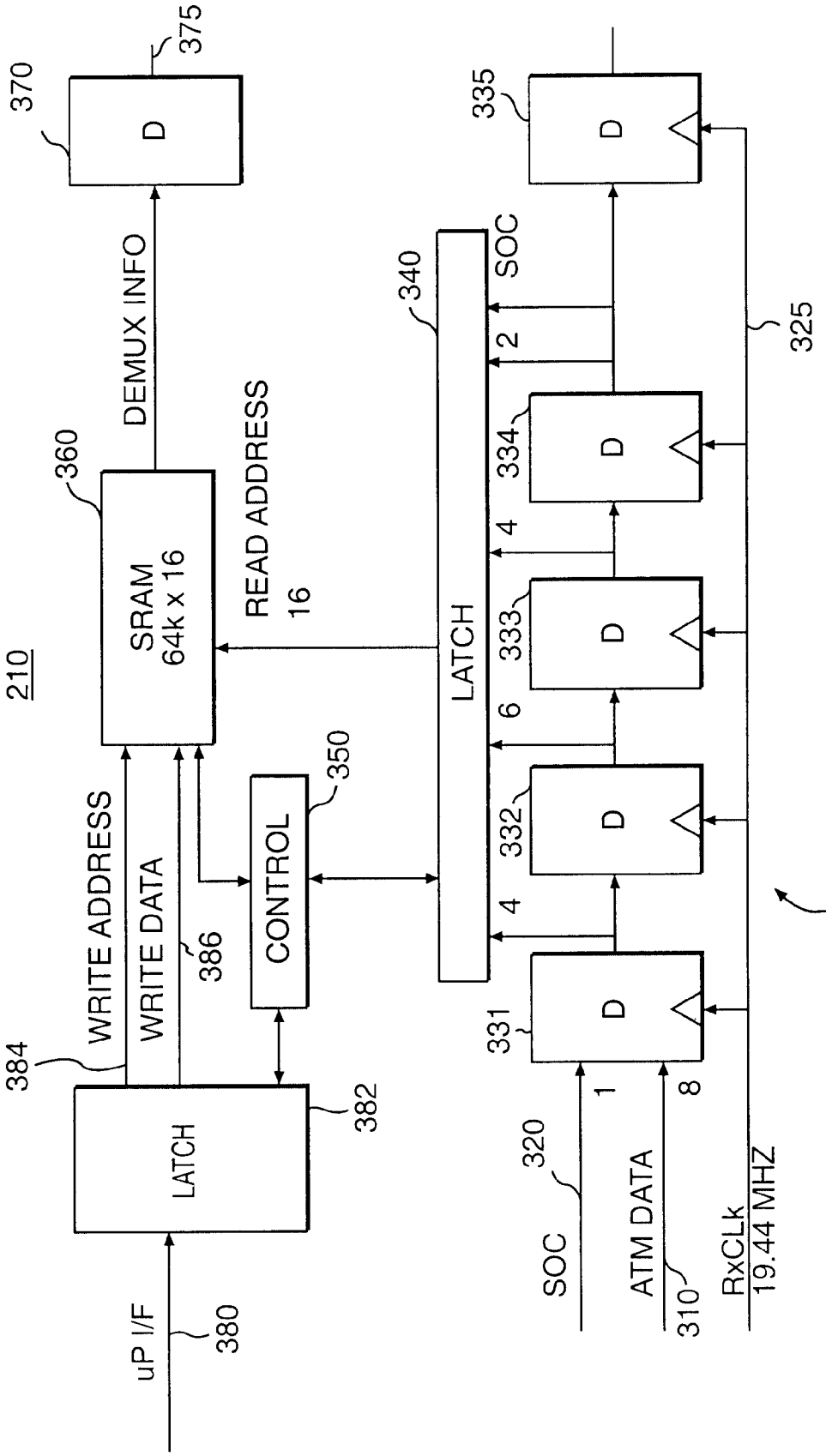

210
uP I/F
380
LATCH
382
384
WRITE ADDRESS
WRITE DATA
386
CONTROL
350
SRAM
64k x 16
360
READ ADDRESS
16
DEMUX INFO
370
D
375
LATCH
340
SOC
2
4
6
4
1
8
SOC
320
ATM DATA
310
RxCLk
19.44 MHZ
325
330
331
332
333
334
335
D

FIG. 3

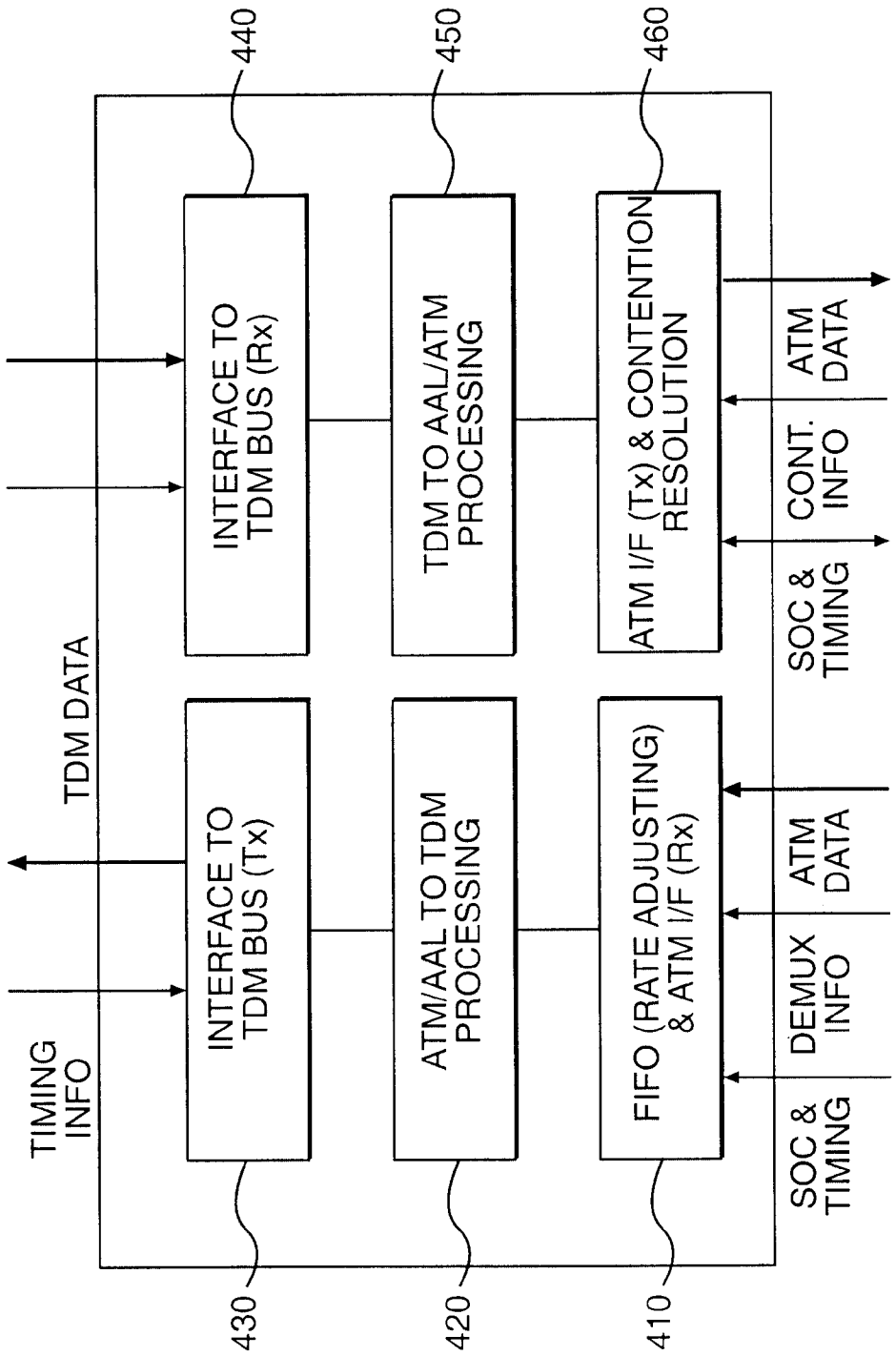

TIMING INFO
TDM DATA
INTERFACE TO TDM BUS (Tx)
ATM/AAL TO TDM PROCESSING
FIFO (RATE ADJUSTING) & ATM I/F (Rx)
INTERFACE TO TDM BUS (Rx)
TDM TO AAL/ATM PROCESSING
ATM I/F (Tx) & CONTENTION RESOLUTION
SOC & TIMING
DEMUX INFO
ATM DATA
SOC & TIMING
CONT. INFO
ATM DATA
410
420
430
440
450
460

FIG. 4

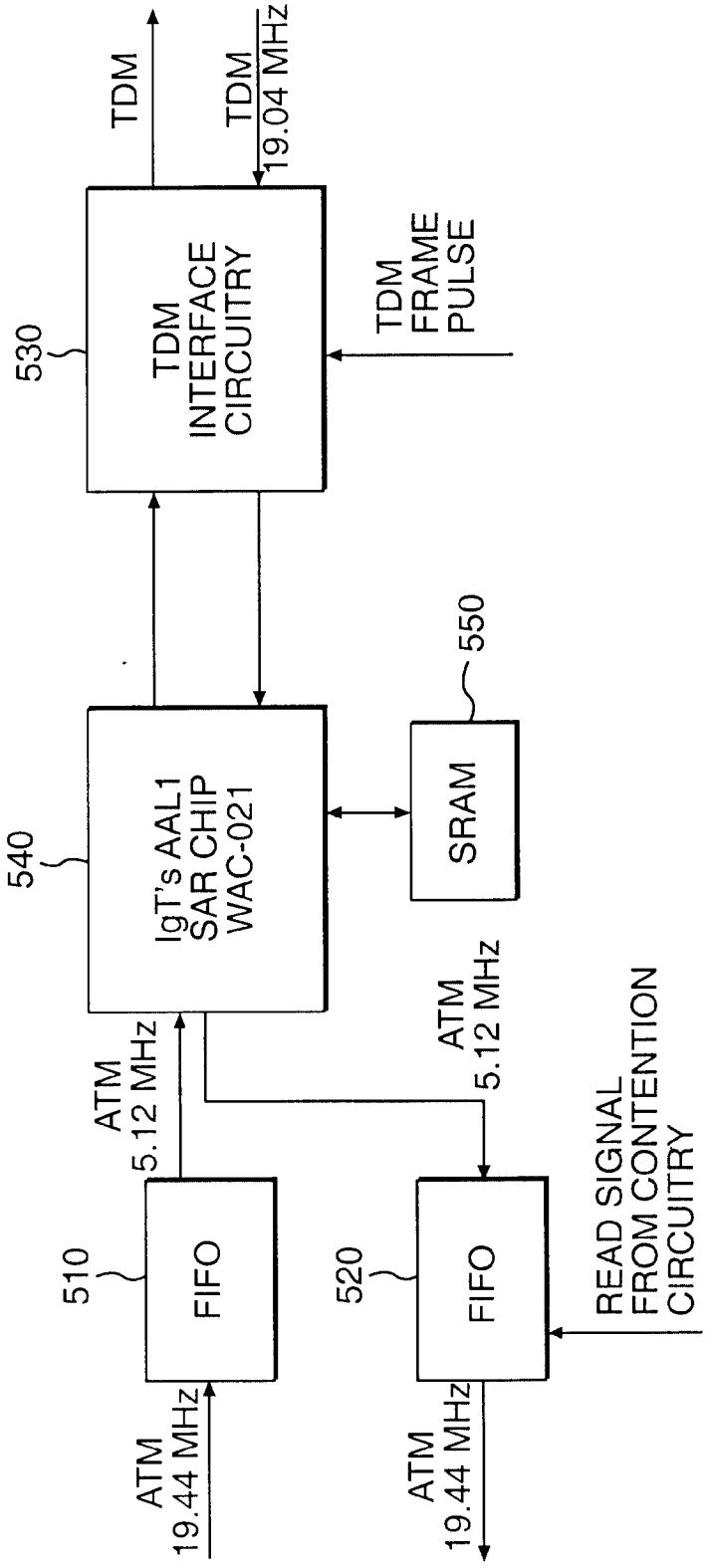

TDM
TDM
19.04 MHz
530
TDM
INTERFACE
CIRCUITRY
TDM
FRAME
PULSE
550
540
IgT's AAL1
SAR CHIP
WAC-021
SRAM
ATM
5.12 MHz
ATM
5.12 MHz
READ SIGNAL
FROM CONTENTION
CIRCUITRY
510
FIFO
520
FIFO
ATM
19.44 MHz
ATM
19.44 MHz

FIG. 5

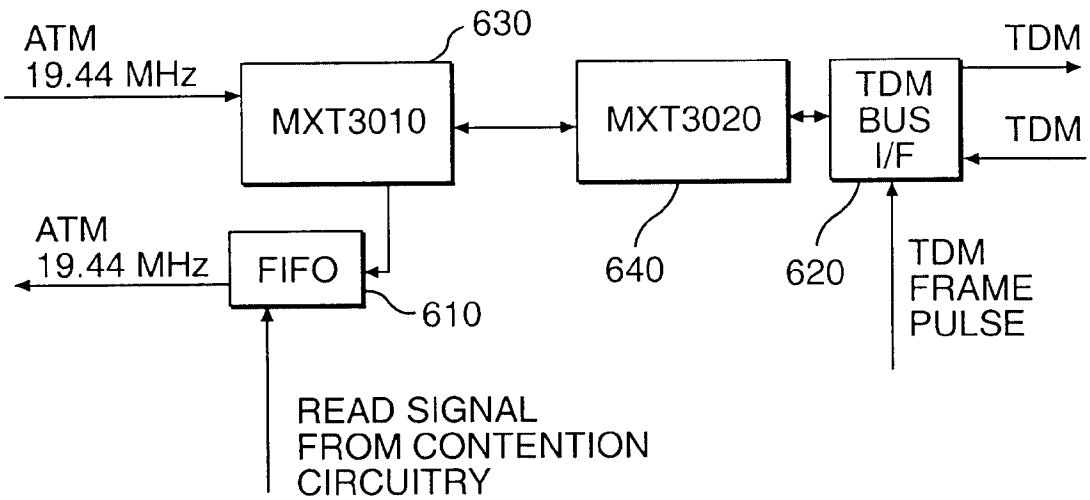

ATM
19.44 MHz
630
MXT3010
MXT3020
TDM
BUS
I/F
TDM
TDM
ATM
19.44 MHz
FIFO
610
640
620
TDM
FRAME
PULSE
READ SIGNAL
FROM CONTENTION
CIRCUITRY

FIG. 6

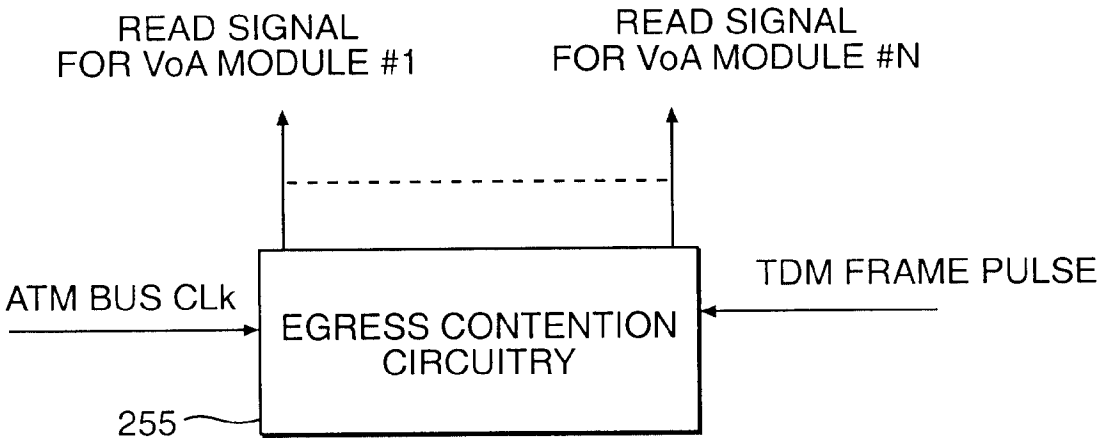
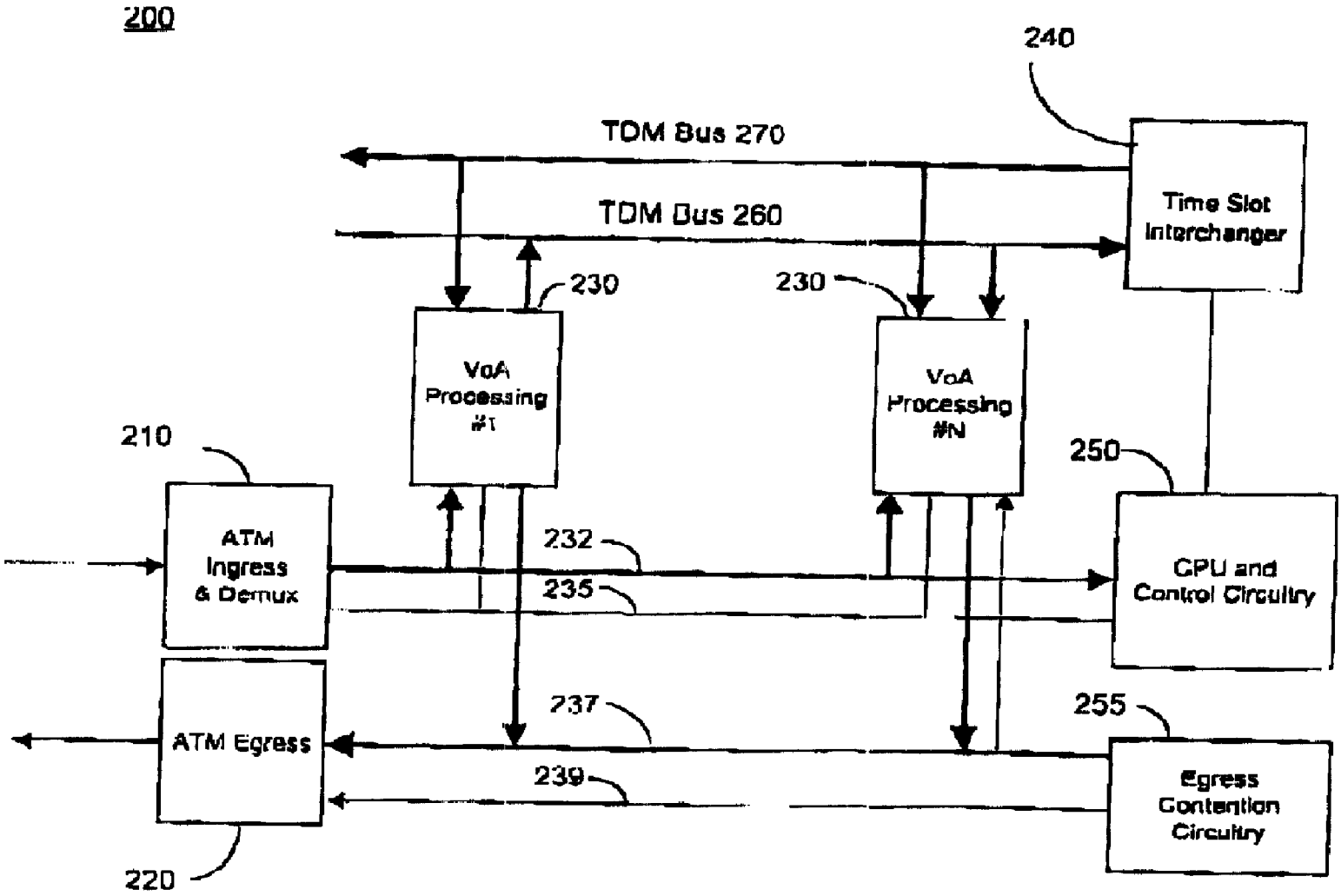

READ SIGNAL
FOR VoA MODULE #1
READ SIGNAL
FOR VoA MODULE #N
ATM BUS CLk
EGRESS CONTENTION
CIRCUITRY
TDM FRAME PULSE
255

FIG. 7


200
240
TDM Bus 270
TDM Bus 260
Time Slot Interchanger
230
230
VoA Processing #1
VoA Processing #N
210
250
ATM Ingress & Demux
232
235
CPU and Control Circuitry
ATM Egress
237
255
239
Egress Contention Circuitry
220

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 614 317 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94301252.6**

(51) Int. Cl.[5] : **H04N 7/13**

(22) Date of filing : **22.02.94**

(30) Priority : **05.03.93 JP 45112/93**

(43) Date of publication of application :
07.09.94 Bulletin 94/36

(84) Designated Contracting States :
DE FR GB IT NL

(71) Applicant : **SONY CORPORATION**
7-35 Kitashinagawa 6-chome
Shinagawa-ku
Tokyo 141 (JP)

(72) Inventor : **Koyanagi, Hideki, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)
Inventor : **Sumihiro, Hiroshi, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)
Inventor : **Emoto, Seiichi, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)
Inventor : **Wada, Tohru, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)

(74) Representative : **Robinson, Nigel Alexander Julian et al**
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Video signal decoding.

(57) A digital video signal that has been encoded using motion- compensated prediction, transform encoding, and variable-length coding, is decoded using parallel processing. Frames of the video signal are divided into slices (1, 2, 3, 4) made up of a sequence of macroblocks (MB). The signal to be decoded is slice-wise divided for parallel variable-length decoding. Each variable-length-decoded macroblock is divided into its constituent blocks for parallel inverse transform processing. Resulting blocks of difference data are added in parallel to corresponding blocks of reference data. The blocks of reference data corresponding to each macroblock are read out in parallel from reference data memories (44, 45, 46, 47) on the basis of a motion vector (83) associated with the macroblock. Reference data corresponding to each macroblock is distributed for storage among a number of reference data memories.

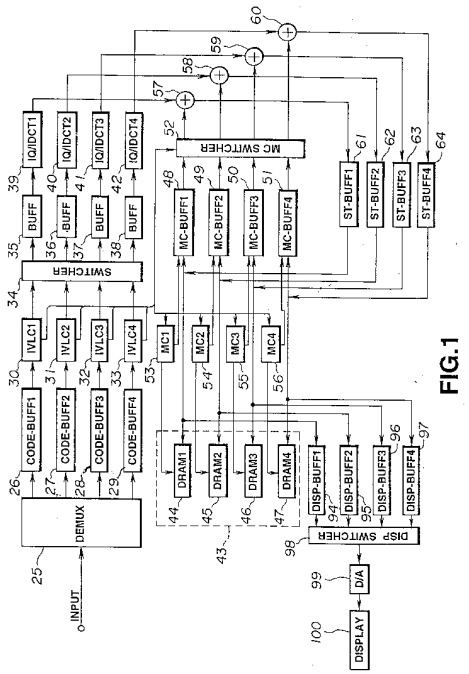


FIG.1

EP 0 614 317 A2

This invention relates to decoding of prediction-coded video signals, and more particularly is directed to the application of parallel processing to such decoding.

It is known to perform compression coding on video data which represents a moving picture in order to reduce the quantity of data to be recorded and/or transmitted. Such data compression may be useful, for example, in recording/reproducing systems using recording media such as magnetic tape or optical disks, and is also useful in transmission systems such as those used for video teleconferencing, video telephones, television broadcasting (including direct satellite broadcast), and the like. For example, it has been proposed by the Moving Picture Experts Group (MPEG) to compression-code moving picture video data utilizing motion-compensated prediction, transform processing using an orthogonal transformation such as the discrete cosine transform (DCT), and variable-length coding. A system for decoding and reproducing such compression-coded video data is illustrated in block diagram form in Figure 14 of the accompanying drawings.

As shown in Figure 14, a sequence of compression-coded video data is provided at an input terminal 101 for processing, in turn, by an inverse VLC (variable-length coding) circuit 102, an inverse quantization circuit 103, and an inverse DCT circuit 104. An adding circuit 105 forms a reconstructed frame of video data on the basis of a difference signal provided from the inverse DCT circuit 104 and predictive picture data (reference data) provided from a motion compensation circuit 106. The resulting reconstructed video data is stored in a frame memory 107.

The motion compensation circuit 106 forms the predictive picture data from reconstructed data previously stored in frame 107 on the basis of motion compensation information (including, for example, motion vectors) extracted from the input signal and supplied to the motion compensation circuit 106 by the inverse VLC circuit 102. Alternatively, with respect to frames for which predictive coding was not performed, such as "intra-frame" coded data, the motion compensation circuit 106 simply provides the value "0" to the adder 105. Reconstructed frames of video data are output from the frame memory 107 via a digital-to-analog converter 108 for display by a display device 109.

As the number of pixels in each frame of the video signal has increased from, for example, the 352 x 240 frame used for video telephones to the 720 x 480 frame used in the NTSC format or the 1920 x 1024 frame in a HDTV (high definition television) system, it was found to be difficult to perform the necessary processing using only one processor and one program execution sequence. For this reason, it has been proposed to divide each frame of the video data into a plurality of subframes, as illustrated in Figure 16 of the accompanying drawings, and then to provide a respective processor for each of the plurality of subframes, so that coding and decoding are performed with parallel processing by the plurality of processors. For example, Figure 15 of the accompanying drawings is a block diagram of a decoding system provided in accordance with this proposal.

In the system of Figure 15, input sequences of encoded video data, each representing a respective subframe, are respectively provided via input terminals 110-113 to processors (decoder blocks) 114-117. The processors 114-117 decode the respective data sequences based upon data supplied from frame memories 119-122, which store respective subframes and are assigned to respective ones of the processors 114-117. For example, processor 114 stores a subframe of decoded data in the memory 119. In order to provide motion compensation, a switching logic circuit 118 provided between the processors 114-177 and the frame memories 119-122, permits the processor 114 to read out data from an adjacent portion of the frame memory 120 as well as from all of frame memory 119. The switching logic circuit 118 also provides frames of output video data from the memories 119-120, via a digital-to-analog converter 123 for display on a display device 124.

The four data sequences respectively provided to the processors 114-117 can, for practical purposes, be combined into a single data sequence by providing headers for controlling multiplexing of the data sequence. For this purpose, a separation block (not shown) is provided upstream from the decoder for separating the combined data sequence into the four sequences to be provided to the respective processors. Examples of parallel processing techniques which use division of a video frame into subframes are disclosed in U.S. Patent No. 5,138,447 and Japanese Patent Application Laid Open No. 139986/1992 (Tokkaihei 4-139986).

As just described, according to the conventional approach, the video frame was generally divided into subframes which were processed in parallel by respective processors. However, when a frame is divided in this manner, there are restrictions on the extent to which the processors can access data that is outside of the processor's respective subframe. Although, as indicated above, a processor can access a region that adjoins its respective subframe, the extent of such access is limited in order to keep the scale of the switching logic circuit 118 from becoming unduly large. As a result, the degree of compression efficiency is reduced, and there are variations in the quality of the reproduced picture at the boundary between the subframes, which may result in visible artifacts at the subframe boundary.

In addition, the processing for compression-coding is carried out completely separately for each of the subframes, which makes it impossible to provide

compression-coding on the basis of data blocks in other subframes, a limitation that is not present when the frame is not divided into subframes. Accordingly, the compression coding method must be changed to accommodate the division into subframes, resulting in a lack of compatibility and a loss in compression efficiency.

Furthermore, if header data is added to the data sequence to be recorded or transmitted in order to provide for multiplexing the data sequence into the respective sequences provided to the parallel processors, the additional header data increases the overhead in the recorded data with a corresponding loss of efficiency, and it may also be necessary to change the coding procedure, and so forth.

In accordance with a first aspect of the present invention, there is provided an apparatus for decoding a coded video signal that represents an image frame, said coded video signal having been divided into a plurality of slices each of said slices being a sequence of macroblocks, each of said macroblocks being a two-dimensional array of picture elements of said image frame, said coded video signal being a bit stream that represents a sequence of said slices which together represent said image frame, said bit stream including a plurality of synchronizing code signals, each of which is associated with a respective one of said slices for indicating a beginning of the respective slice, the apparatus comprising:

a plurality of decoding means each for decoding a respective portion of said coded video signal that represents said image frame; and

distributing means responsive to said synchronizing code signals for distributing said slices among said plurality of decoding means.

According to a second aspect of the invention, there is provided an apparatus for decoding input signal blocks that were formed by transform encoding and then variable-length encoding blocks of video data, the apparatus comprising:

decoding means for variable-length decoding a series of said input signal blocks;

parallel data means for forming plural parallel data streams, each of which includes respective ones of said series of input signal blocks which were variable-length decoded by said decoding means; and

a plurality of inverse transform means each for receiving a respective one of said parallel data streams and for performing inverse transform processing on the variable-length decoded signal blocks in the respective data stream.

In preferred embodiments of the apparatus just described, the decoding circuit is one of a plurality of decoding circuits for variable-length decoding respective series of input signal blocks, and the apparatus further includes a distributing circuit for forming the respective series of input signal blocks to be decoded by the plural decoding circuits from a bit stream representing an image frame, and the respective series of input signal blocks are formed in response to synchronizing signals provided at predetermined intervals in the bit stream representing the image frame.

According to a third aspect of the invention, there is provided an an apparatus for decoding an input digital video signal which includes groups of blocks of prediction-coded difference data, each of said groups consisting of a predetermined plurality of said blocks and having a respective motion vector associated therewith, each of said blocks of prediction-coded difference data having been formed on the basis of the respective motion vector associated with the respective group which includes said block, the apparatus comprising:

output means for supplying in parallel blocks of prediction-coded difference data contained in one of said groups of blocks;

reference data means for supplying in parallel plural blocks of reference data, each of said blocks of reference data being formed on the basis of the motion vector associated with said one of said groups of blocks and corresponding to one of said blocks of prediction-coded difference data supplied by said output means; and

a plurality of adding means each connected to said output means and said reference data means for adding a respective one of said blocks of prediction-coded difference data and the corresponding block of reference data.

In preferred embodiments of the invention, the reference data circuit includes a plurality of reference data memories from which reference data is read out in parallel on the basis of the motion vector associated with that group of blocks, a plurality of buffer memories for temporarily storing reference data read out from the plurality of reference data memories and a distribution circuit. According to one alternative embodiment of this aspect of the invention, each of the buffer memories is associated with a respective one of the reference data memories and is controlled on the basis of the motion vector for reading out the reference data temporarily stored therein, and the distributing circuit is connected between the buffer memories and the adding circuits and distributes the reference data stored in the buffer memories among the adding circuits on the basis of the motion vector. According to another alternative embodiment of this aspect of the invention, each of the buffer memories is associated with one of the adding circuits and the distributing circuit is connected between the reference data memories and the buffer memories for distributing among the buffer memories, on the basis of the motion vector associated with that group of blocks, the reference data read out from the reference data memories.

According to a fourth aspect of the invention,

there is provided a method of decoding a coded video signal that represents an image frame, said coded video signal having been divided into a plurality of slices each of said slices being a sequence of macroblocks, each of said macroblocks being a two-dimensional array of picture elements of said image frame, said coded video signal being a bit stream that represents a sequence of said slices which together represent said image frame, said bit stream including a plurality of synchronizing code signals, each of which is associated with a respective one of said slices for indicating a beginning of the respective slice, the method comprising the steps of:

providing a plurality of decoding means each for decoding a respective portion of said coded signal that represents said video frame; and

distributing said slices among said plurality of decoding means in response to said synchronizing code signals.

The data representing each macroblock may be distributed block-by-block among the plurality of memories or line-by-line in a cyclical fashion among the plurality of memories.

A video signal decoding apparatus may be provided in which the input coded signal is distributed for parallel processing among several decoding circuits on the basis of synchronizing code signals that are provided in the signal in accordance with a conventional coding standard. In this way, parallel decoding can be precisely carried out on the basis of synchronizing signals provided in accordance with a conventional coding method and during time periods available between the synchronizing signals. In this way, restrictions on the conventional coding method can be reduced.

In addition, the data may be sequenced on the basis of "slices" which are a standard subdivision of a video frame constituting a plurality of macroblocks and the slices of data are distributed among decoding circuits so that high speed parallel decoding may be carried out.

Further, each of the blocks making up a macroblock may be distributed to a respective inverse transformation circuit so that inverse transform processing can be carried out simultaneously in parallel for all of the blocks of a macroblock, and the inverse transform blocks are then combined, in parallel, with reference data to recover the video signal which had been predictive-coded. The reference data, in turn, may be provided from parallel memories at the same time on the basis of the motion compensation vector for the particular macroblock, and in such a way that there is no need to place restrictions on the motion-compensation carried out during the predictive coding. For example, there is no need to limit the range of the motion vector.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of an embodiment of an apparatus for decoding a moving picture video data signal;

Figure 2 is a schematic illustration of a manner in which video data corresponding to an image frame is distributed for decoding;

Figure 3 is a timing diagram which illustrates operation of a buffer memory provided in the apparatus of Figure 1;

Figure 4 is a block diagram which illustrates a code buffering arrangement provided upstream from variable-length decoder circuits provided in the apparatus of Figure 1;

Figure 5 is a timing diagram which illustrates operation of the code buffering arrangement shown in Figure 4;

Figure 6 is a block diagram which shows an alternative code buffering arrangement provided upstream from variable-length decoder circuits provided in the apparatus of Figure 1;

Figure 7 is a timing diagram which illustrates operation of the code buffering arrangement shown in Figure 6;

Figures 8(A), 8(B) and 8(C) together schematically illustrate a manner in which reference data is provided on the basis of a motion vector to adders that are part of the apparatus of Figure 1;

Figure 9 is a timing diagram which illustrates an operation for providing reference data to the adders which are part of the apparatus of Figure 1;

Figure 10 is a block diagram of another embodiment of an apparatus for decoding a moving picture video data signal;

Figure 11(A), 11(B) and 11(C) together schematically illustrate a manner in which reference data is provided on the basis of a motion vector to adders that are part of the apparatus of Figure 10;

Figures 12(A) and 12(B) together schematically illustrate an alternative manner in which reference data is provided on the basis of a motion vector to the adders which are part of the apparatus of Figure 10;

Figure 13 is timing diagram which illustrates an operation for providing reference data according to the example shown in Figure 12;

Figure 14 is a block diagram of a conventional apparatus for decoding and reproducing a moving picture video data signal;

Figure 15 is a block diagram of a portion of a conventional apparatus for decoding and reproducing a moving picture video data signal by means of parallel processing; and

Figure 16 schematically illustrates operation of the conventional decoding apparatus of Figure 15.

A preferred embodiment of the invention will now be described, initially with reference to Figure 1.

Figure 1 illustrates in block diagram form an apparatus for decoding a moving picture video data signal that has been coded according to a proposed MPEG standard system.

An input bit stream representing the coded video data signal is provided to a demultiplexer 25, by means of which the input signal is distributed, slice-by-slice, to code buffers 26-29.

Figure 2 illustrates the slice-by-slice distribution of the input data. As is well known to those who are skilled in the art, each slice is a sequence of macroblocks transmitted in raster scanning order. The starting point of each slice is indicated by a synchronizing code signal, and the slices are provided so that transmission errors and the like can be confined to a single slice, because after an error occurs, proper coding can resume at the synchronizing code signal provided at the beginning of the subsequent slice. Accordingly, the demultiplexer 25 is provided with a circuit which detects the synchronizing code signals, and distribution of the input signal among the code buffers 26-29 is carried out in response to the detected synchronizing code signals.

As is also well known, the motion vectors provided with respect to each macroblock, and the DC coefficients for each block, are differentially encoded. In other words, only the difference between respective motion vectors for the current macroblock and the preceding macroblock is encoded and transmitted, and also, only the difference between the respective DC coefficient for the present block and that of the preceding block are coded and transmitted.

As indicated in Figure 2, the first, fifth, ninth, etc. slices of each image frame are stored in the first code buffer 26, and these slices are provided for variable-length decoding by a variable-length decoder circuit 30. Similarly, the second, sixth, tenth, etc. slices of the image frame are stored in the second code buffer 27 for variable-length decoding by variable-length decoder circuit 31; the third, seventh, eleventh, etc. slices are stored in the third code buffer 28 for variable-length decoding by the variable-length decoder circuit 32; and the fourth, eighth, twelfth, etc. slices are stored in the fourth code buffer 29 for variable-length decoding by the variable-length decoder circuit 33.

According to the example shown in Figure 2, the number of macroblocks in each slice is fixed, so that it will not be necessary for any of the variable-length decoders to wait. As result, decoding carried on by the variable-length decoders is synchronized and is carried out efficiently.

It will be understood that, although the number of macroblocks per slice is fixed, the number of bits per slice in the input signal will vary because of variable-length encoding. Nevertheless, the number of macroblocks per slice output by each variable-length decoding circuit is the same according to this example.

In the example shown in Figure 2, each slice is shown as being one macroblock high and extending horizontally entirely across the image frame, so that each slice consists of one row of macroblocks. However, it is also within the contemplation of this invention to provide for slices having a fixed length in terms of macroblocks that is longer or shorter than one row of macroblocks. It is further contemplated that the number of macroblocks per slice may be variable within each frame and/or from frame to frame and that the positions of slices within a frame may vary. In case variable-length slices are provided within a frame, it will be appreciated that the number of macroblocks distributed to each of the variable-length decoders may be unbalanced, in which case some of the variable-length decoders may be required to output filler macroblocks (all zeros for example) until other decoders have "caught up". Furthermore, it is provided that variable-length decoding of slices from the next image frame will not proceed until all of the slices of the current frame have been variable-length decoded.

It will be recognized that any loss of decoding efficiency that results from the occasional need to interrupt the processing by some of the variable length decoders is compensated for by the fact that the coding can be performed with slices that have a variable length in terms of macroblocks.

Details of the variable-length decoding processing will now be described.

Data which has been decoded by the respective variable length decoders are transferred to buffer memories 35-38 by way of switcher 34. Figure 3 illustrates the manner in which data is distributed to, and output from, the buffer memories 35-38. It will be noted that, upstream from the buffers 35-38, processing had been performed in a slice-wise parallel manner, but downstream from the buffers 35-38 processing is performed in a block-wise parallel manner. In particular, the four blocks of luminance data making up a macroblock are output in parallel from respective ones of the buffer memories 35-38. (It will be understood that a macroblock also includes chrominance blocks. For example, in the 4:2:2 format, each macroblock includes four blocks of chrominance data in addition to the four blocks of luminance data. The discussion from this point forward will deal only with the luminance data blocks, it being understood that the corresponding four chrominance data blocks can be processed in a similar manner.)

Referring again to Figure 3, it will be seen that the variable length decoders 30-33 respectively output simultaneously the respective first block of the first through fourth slices. The respective first blocks are distributed among the buffer memories 35-38 so that the first block of the first slice (i.e., the first block of the first macroblock of the first slice) is stored in the first buffer memory 35, the second block of the first slice is stored in the second buffer memory 36. the

third block of the first slice is distributed to the third buffer memory 37, and the fourth block of the first slice is distributed to the fourth buffer memory 38. As a result, all four blocks of a single macroblock can be read out in parallel by the respective buffer memories 35-38, so that block-wise parallel processing can be accomplished downstream. Such processing includes conventional inverse transform processing in accordance with zig-zag scanning.

In the example just discussed, each buffer memory preferably has two banks which each have the capacity of storing four data blocks.

The block-wise parallel data provided from the buffer memories 35-38 is subjected to inverse quantization and inverse discrete cosine transform processing in parallel at processing blocks 39-42. Thereafter, motion compensation processing for the four blocks of the macroblock is also carried out in parallel. Reference picture data for each macroblock is extracted from previously reproduced (i.e., previously reconstructed) image data stored in a frame memory 43. The reference picture data is formed on the basis of the motion vector which corresponds to the macroblock being processed and is used to form decoded data in combination with difference data output from the processing blocks 39-42. In this example, since motion compensation processing is carried out in parallel for each macroblock (four blocks) of luminance data, the motion vectors provided to motion compensation processing blocks 53-56 from the variable length decoders 30-33 always correspond to each other at any given time. For this reason, an MC (motion compensation) switcher 52 is used to switch a data bus, so that it is possible to provide motion compensation processing of the reference data transferred to MC buffer memories 48-51 in such a manner that memory accessing by the motion compensation processing blocks 53-56 does not overlap. As a result, the motion compensation search range, and accordingly the permissible range of the motion vector, is not limited. Details of motion compensation processing will be provided below.

Reproduced decoded image data formed in parallel at adders 57-60 is stored via four parallel processing paths in the frame memory 43 by way of storage buffers 61-64. Moreover, sequences of images for which the reproduced (reconstructed) data is stored in memory 43 are output to a digital-to-analog converter 99 through display buffer memories 94-97, and a display switcher 98 which is switched according to appropriate display timing. The D/A converted signal is then displayed on a display device 100.

There will now be described, with reference to Figure 4, details of a buffering arrangement provided upstream from the variable length coders of the apparatus of Figure 1.

As shown in Figure 4, an input signal bit stream is received at an input terminal 65 and provided therefrom to a demultiplexer 66 which divides the bit stream at the beginning of each slice and distributes the slices among code buffer memories 67-70. The slices of data are output respectively from the code buffer memories 67-70 to variable-length decoders 71-74, and variable-length decoded data is respectively output from each of the variable-length decoders 71-74 via output terminals 75-78.

The buffering and decoding operations carried out by the circuitry shown in Figure 4 will now be described with reference to the timing diagram shown in Figure 5.

In particular, the input bit stream received at the terminal 65 is divided at the beginning of each slice by the demultiplexer 66. Because synchronizing code signals indicative of the beginning of each slice are included at intervals corresponding to a plural number of macroblocks (such intervals each being referred to as a slice), the synchronizing code signals are detected at the demultiplexer 65 for the purpose of performing the division of the bit stream into slices.

As shown in Figure 5, a sequence of the resulting slices are written in a cyclical fashion into the code buffer memories 67-70. In particular, slice 1, slice 5, slice 9, etc. are written into the code buffer memory 67; slice 2, slice 6, slice 10, etc. are written into the code buffer memory 68; slice 3, slice 7, slice 11, etc. are written into the code buffer memory 69; and slice 4, slice 8, slice 12, etc. are written into the code buffer memory 70.

At a point when slice 4 has been written into the code buffer memory 70, the slices 1-4 are respectively read out in parallel from the code buffer memories 67-70 to the four variable-length decoders 71-74 and variable-length decoding begins.

The variable-length decoders 71-74 each complete decoding processing of a macroblock from a respective slice within the same time. Decoded data produced by variable-length decoder 71 is output via terminal 75; decoded data produced by variable-length decoder 72 is output via terminal 76; decoded data produced by variable-length decoder 73 is output via terminal 77; and decoded data produced by variable-length decoder 74 is output via terminal 78. All of the decoded data is supplied to the switcher 34 (Figure 1). In addition, decoded motion vector data is provided from the variable-length decoders to the MC switcher 52 and motion compensation processing blocks 53-56.

It should be understood that, in Figure 5, the symbol "1-1" shown in the output of IVLC1 (variable-length decoder 71) is indicative of the first block of slice 1. Similarly, for example, "4-1" shown in the output of IVLC4 (variable-length decoder 74) is indicative of the first block of slice 4.

An alternative code buffering arrangement provided upstream from the variable-length decoders is shown in Figure 6.

In Figure 6, the input bit stream is again received at an input terminal 65 and provided therefrom to a demultiplexer 79, at which the bit stream is divided at the beginning of each slice. Immediately downstream from the demultiplexer 79 is a code buffer memory 80 which has respective regions in each of which a slice of data can be stored. Additional buffer memories 90-93 are provided downstream from the buffer memory 80. In a similar manner to the arrangement of Figure 4, the buffered data output from each of the buffer memories 90-93 is provided to a respective one of the variable-length decoders 71-74, and the decoded data output from the variable-length decoders 71-74 is provided at respective output terminals 75-78.

Operation of the code buffering arrangement shown in Figure 6 will now be described with reference to the timing diagram of Figure 7.

As before, the input bit stream provided from the terminal 65 is divided at the beginning of each slice by the demultiplexer 79 on the basis of synchronizing code signals provided at intervals corresponding to a number of macroblocks.

As shown in Fig. 7, respective slices are written in a cyclical fashion into the regions 1-4 of the buffer memory 80. In particular, slice 1, slice 5, slice 9, etc. are written into region 1; slice 2, slice 6, slice 10, etc. are written into region 2; slice 3, slice 7, slice 11, etc. are written into region 3; and slice 4, slice 8, slice 12, etc. are written into the region 4.

At a point when slice 4 has been written into region 4, the data stored in the four regions are sequentially read out from the code buffer memory 80. As a result, slices 1, 5, 9, etc. are read out from region 1 and written into buffer memory 90; slices 2, 6, 10, etc., are read out from region 2 and written into buffer memory 91; slices 3, 7, 11, etc. are read out from region 3 and written into buffer memory 92, and slices 4, 8, 12, etc. are read out from region 4 and written into buffer memory 93.

At a time when the contents of region 4 have been written into the buffer memory 93, the data respectively stored in the buffer memories 90-93 is read out in parallel to the variable-length decoders 71-74, and decoding processing starts at that time.

The variable-length decoders 71-74 each complete the decoding processing of a respective macroblock within the same time. Decoded data produced by variable length decoder 71 is output via terminal 75; decoded data produced by variable-length decoder 72 is output via terminal 76; decoded data produced by variable-length decoder 73 is output via terminal 77; and decoded data produced by variable-length decoder 74 is output via terminal 78. This decoded data is supplied to the switcher 34, and in addition, decoded motion vector data is supplied from the variable-length decoders to MC switcher 52 and to motion compensation processing blocks 53-56.

As was the case with Figure 5, in Figure 7 the symbol "1-1" is indicative of the first block in slice 1, which is decoded by variable-length decoder 71, while "4-1" is indicative of the first block of slice 4, which is decoded by the variable-length decoder 74.

With respect to the buffering arrangement shown in Figure 4, it is possible to use certain distribution methods with respect to input data streams which have a processing unit which is shorter than a slice and are included in a layer (known as an "upper layer") which has processing units which are longer than a slice. With respect to an input data stream which has such a format, it is possible to simultaneously write the upper layer into the code buffer memories 67-70 in order to provide parallel data to the variable-length decoders 71-74. Alternatively, the bit stream for the upper layer can be written into one of the four code buffer memories so that the upper layer is decoded by only one of the four variable-length decoders, with parameters being set at the other variable-length decoders. According to another possible method, an additional processor is provided to decode the upper layer bit stream so as to set parameters at the four variable-length decoders.

On the other hand, using the arrangement shown in Figure 6, the upper layer bit stream can be written into one of the four regions of the buffer memory 80 and the contents of that region can be simultaneously written into the buffer memories 90-93 for parallel processing by the variable-length decoders 71-74. According to an alternative method, the upper layer bit stream is written into one of the four regions of the buffer memory 80 so that the data is written into one of the four buffer memories 90-93 and is then decoded by one of the four variable-length decoders in order to set parameters at the other variable-length decoders.

According to another alternative method, a separate processor is provided to decode the upper layer bit stream in order to set parameters at the four variable-length decoders. As a further method, the demultiplexer 79 repeatedly writes the upper layer bit stream into the four regions of the buffer memory 80 so that the data is simultaneously written from each region into the buffer memories 90-93 for parallel processing in the variable-length decoders 71-74.

In these ways, distribution of the data stream, and parallel processing thereof, can be carried out on the basis of parameters included in the data stream.

Details of decoding processing with respect to motion-compensated predictive-coded data will now be described.

Figure 8(A) illustrates a manner in which reference image data is distributed among and stored in DRAMs 44-47 making up the frame memory 43. Each image frame is, as indicated above, divided into macroblocks, and each macroblock is formed of four blocks. Each of the four blocks is, in this particular example, an 8 x 8 array of pixel elements, and each of

the blocks constitutes one of four quadrants of its respective macroblock. The data with respect to each macroblock is divided among the four DRAMs 44-47. In particular, all of the first blocks (upper left blocks) of all of the macroblocks are stored in DRAM 44, all of the second blocks (upper right blocks) of all of the macroblocks are stored in DRAM 45, all of the third blocks (lower left blocks) of all of the macroblocks are stored in DRAM 46, and all of the fourth blocks (lower right blocks) of all of the macroblocks are stored in DRAM 47. Accordingly, it will be seen that the reference data is distributed among DRAMs 44-47 in a checkered pattern.

Continuing to refer to Figure 8(A), the square labelled 81 represents the geometric area of the image frame which corresponds to the macroblock which is currently being decoded (reconstructed), and reference numeral 82 represents the motion vector associated with that macroblock, according to the example shown in Figure 8(A). In addition, the reference numeral 83 represents the reference data stored in the DRAMs 44-47 and indicated by the motion vector 82 as corresponding to the current macroblock 81. The data represented by the shaded square 83 is read out from the DRAMs 44-47 under control of motion compensation processing blocks 53-56 on the basis of the motion vector 82. In particular, the data corresponding to the "DRAM1" portion of the square 83 (i.e., a central portion of the square 83) is read out from DRAM 44 to motion compensation buffer 48 under the control of motion compensation processing block 53. Similarly, the portions of the shaded square 83 which overlap with squares labelled "DRAM2" (i.e., central portions of the left and right sides of the square 83) are read out from DRAM 45 to motion compensation buffer 49 under control of motion compensation processing block 54. Also, the portions of the shaded square 83 which overlap the squares labelled "DRAM3" (i.e., the central portions of the upper and lower edges of the square 83) are read out from DRAM 46 to motion compensation buffer 50 under control of motion compensation processing block 55. Finally, the portion of the shaded square 83 which overlaps with squares labelled "DRAM4" (i.e., corner regions of the square 83) are read out from the DRAM 47 to motion compensation buffer 51 under control of motion compensation processing block 56.

Figure 8(B) is a schematic illustration of the reference data read out from the respective DRAMs 44-47 and stored in respective motion compensation buffers 48-51. This data stored in the four motion compensation buffers 48-51 represents the reference data for the macroblock which is currently to be reconstructed. However, the data as stored in the individual motion compensation buffers does not correspond to the data required for each of the adders 57-60. Therefore, the MC switcher 52 is provided between the motion compensation buffers 48-51 and the adders 57-60 so that the correct reference data is distributed from the motion compensation buffers to the adders. The reference data which is supplied to each of the adders 57-60 is schematically illustrated in Figure 8(C).

Figure 9 illustrates the timing, according to the example shown in Figure. 8(A), at which data read out from the motion compensation buffers 48-51 is routed among the adders 57-60.

The processing of the four blocks making up the macroblock proceeds, as indicated before, in parallel, with the respective first lines of each of the blocks being processed simultaneously, then the second lines, and so forth. With respect to the first lines of the blocks, initially, at a starting time tl (Figure 9), data from motion compensation buffer 51 is routed to adder 57, data from motion compensation buffer 50 is routed to adder 58, data from motion compensation buffer 49 is routed to adder 59, and data from motion compensation buffer 48 is routed to adder 60. At a changeover point in the processing of the first lines, indicated by time t2 in Figure 9, the routing is changed so that data from motion compensation buffer 50 is routed to adder 57, data from motion compensation buffer 51 is routed to adder 58, data from motion compensation buffer 48 is routed to adder 59, and data from motion compensation buffer 49 is routed to adder 60. This routing state continues until the end of the first line (indicated by time t3) and then the procedure that was followed for the first lines is carried out again with respect to the second lines. The same procedure is then continued through the nth lines, but upon completion of the nth lines of the block, as indicated at time t4, a different routing pattern is established for the beginning of the (n + 1)th lines. According to this pattern, data from motion compensation buffer 49 is provided to adder 57, data from motion compensation buffer 48 is provided to adder 58, data from motion compensation buffer 51 is provided to adder 59, and data from motion compensation buffer 50 is provided to adder 60. This routing arrangement continues until a changeover point in the (n + 1)th lines, indicated by time t5, at which the routing arrangement is changed so that data from motion compensation buffer 48 is routed to adder 57, data from motion compensation buffer 49 is routed to adder 58, data from motion compensation buffer 50 is routed to adder 59, and data from motion compensation buffer 51 is routed to adder 60. On the completion of the process for the (n + 1)th line (indicated by time t6), the procedure carried out for the (n + 1)th lines is repeated with respect to each of the remaining lines of the blocks until the last (eighth) lines have been processed, at which point (indicated by time t7) processing for the macroblock is complete. Processing for the next macroblock then begins, on the basis of the motion vector associated with the next macroblock.

It will be appreciated that the reference data sup-

plied to the adders 50-60 is added by the adders to the current difference data supplied thereto from the processing circuits 39-42 so that macroblocks of reconstructed image data are produced. It will also be recognized that the storage of the reference data according to the above-described checkered pattern in the frame memory 43, and the above-described method of reading out, buffering, and switching the reference data makes it possible to provide motion-compensation decoding processing without any restriction on the range of the motion vector, and in such a manner that memory accesses do not overlap.

In the embodiment illustrated in Figure 1, the MC switcher 52 is provided between the motion compensation buffers 48-51 and the adders 57-60. However, according to an alternative embodiment, shown in Figure 10, the MC switcher 52 can be provided between the DRAMs 44-47 and the motion compensation buffers 48-51, with each of the buffers 48-51 connected directly to, and providing data exclusively to, a respective one of the adders 57-60.

A method of operating the embodiment illustrated in Figure 10 will be described with reference to Figures 11(A)-(C).

Figure 11(A) is similar to Figure 8(A), and shows a square 84 which represents the geometric area corresponding to the macroblock currently being processed, motion vector 85 associated with the current macroblock, and a shaded square 86 which represents the appropriate reference data for the current macroblock as indicated by the motion vector 85. It will also be noted that the reference data is distributed for storage among the DRAMs 44-47 in a block-wise manner according to the same checkered pattern shown in Figure 8(A).

Under control of the motion compensation processing blocks 53-56, and on the basis of the motion vector for the current macroblock, data is read out from the DRAMs 44-47 and routed to the motion compensation buffers 48-51 by the MC switcher 52 so that all of the reference data to be provided to the adder 57 is stored in the motion compensation buffer 48, all of the reference data to be provided to the adder 58 is stored in the motion compensation buffer 49, all of the reference data to be provided to the adder 59 is stored in the motion compensation buffer 50, and all of reference data to be provided to the adder 60 is stored in the motion compensation buffer 51. Referring to Figures 11(A) and (B), it will be noted that the data represented by the upper left quadrant of the shaded square 86 is stored in the motion compensation buffer 48, the data represented by the upper right quadrant of the shaded square 86 is stored in the motion compensation buffer 49, the data represented by the lower left quadrant of the shaded square 86 is stored in the motion compensation buffer 50, and the data represented by the lower right quadrant of the shaded square 86 is stored in the motion compensation buffer 51. More specifically, during an initial read out period, data is simultaneously read out from all four of the DRAMs 44-47 and routed such that data from a portion of the DRAM 47 is stored in motion compensation buffer 48, while data from a portion of DRAM 46 is stored in motion compensation buffer 49, data from a portion of DRAM 45 is stored in motion compensation buffer 50, and data from a portion of DRAM 44 is stored in motion compensation buffer 51. During a second read out period there is again simultaneous reading out of data from the four DRAMs, but now the routing is such that data from a portion of DRAM 46 is stored in motion compensation buffer 48, data from a portion of DRAM 47 is stored in motion compensation buffer 49, data from a portion of DRAM 44 is stored in motion compensation buffer 50, and data from a portion of DRAM 45 is stored in motion compensation buffer 51. Moreover, during a third read out period, again there is simultaneous read out from all of the DRAMs, but routing is performed so that data from a portion of DRAM 45 is stored in motion compensation buffer 48, data from a portion of DRAM 44 is stored in motion compensation buffer 49, data from a portion of DRAM 47 is stored in motion compensation buffer 50, and data from a portion of DRAM 46 is stored in motion compensation buffer 51. Then, during a final read out period, data is simultaneously read out from four DRAMs and routed such that data from a portion of DRAM 44 is stored in motion compensation buffer 48, data from a portion of DRAM 45 is stored in motion compensation buffer 49, data from a portion of DRAM 46 is stored in motion compensation buffer 50, and data from a portion of DRAM 47 is stored in motion compensation buffer 51.

It will be observed that data from every one of the four DRAMS is thus stored in each of the motion compensation buffers. Moreover, with reading of the data from the DRAMs and control of the MC switcher 52 on the basis of the motion vector for the current macroblock, memory access can be performed without overlap.

Also, because each of the motion compensation buffers are associated exclusively with a respective adder, and the reference data has been stored appropriately therein, as shown in Figure 11(C), there is also no difficulty in accessing the motion compensation buffers.

There will now be described, with reference to Figures 12 and 13, in addition to Figure 10, an alternative method of operating the embodiment of Figure 10 so that the appropriate reference data is stored in each of the motion compensation buffers 48-51.

As indicated in Figure 12(A), according to this alternative method of operation, the reference data is distributed line-by-line among the DRAMS 44-47, rather than block-by-block, as in the technique shown in Figure 11(A). For example, referring again to Figure 12(A), the data for the first line of each macroblock

(i.e., the first line of the first and second blocks of the macroblock), is stored in DRAM 44, the second line of data of each macroblock is stored in DRAM 45, the third line of data for each macroblock is stored in DRAM 46, the fourth line of each macroblock is stored in DRAM 47, the fifth line of each macroblock is stored in DRAM 44, and so forth, continuing in a cyclical fashion, line-by-line. It should be understood that the data for the ninth line of each macroblock (i.e., the first line of data in the third and fourth blocks of each macroblock) is stored in DRAM 44, whereas the data for the last line of each macroblock (i.e., the last line of the last two blocks of the macroblock) is stored in DRAM 47. Accordingly, the reference data is distributed among the DRAM 44-47 according to a striped pattern, rather than the checkered pattern of Figure 11(A).

In Figure 12(A), the square labelled 87 represents the geometric area which corresponds to the macroblock which is currently to be decoded, the motion vector 88 is the motion vector associated with the current macroblock, and the square 89 represents the appropriate reference data for the current macroblock, as indicated by the motion vector 88.

Figures 12(B) and Figure 13 indicate the sources of data and the timing according to which the appropriate reference data is stored in the motion compensation buffers 48-51. As before, data is read out from the DRAMS 44-47 and routed by MC switcher 52 under the control of the motion compensation processing blocks 43-56 and on the basis of the motion vector for the current macroblock.

In particular, during a first time slot, the reference data corresponding to the first line of the first block is read out from DRAM 47 and stored in motion compensation buffer 48. During the same time slot, reference data corresponding to the eighth line of the second block is read out from DRAM 46 and stored in motion compensation buffer 49. reference data for the seventh line of the third block is read out from DRAM 45 and stored in motion compensation buffer 50. and reference data for the sixth line of the fourth block is read out from DRAM 44 and stored in motion compensation buffer 51.

In the next (second) time slot, a one line shift in routing occurs, so that reference data for the second line of the first block is read out from DRAM 44 and stored in motion compensation buffer 48, reference data for the first line of the second block is read out from DRAM 47 and stored in motion compensation buffer 49, reference data for the eighth line of the third block is read out from DRAM 46 and stored in motion compensation buffer 50, and reference data for the seventh line of the fourth block is read out from DRAM 45 and stored in motion compensation buffer 51.

The one-line shifts are continued in each of the succeeding six time slots so that the data is read out, routed and stored in the motion compensation buffers according to the pattern shown in Figures 12(D) and 13. It will be observed that memory access occurs, as before, without overlapping.

As a result, the reference data which is to be supplied to adder 57 is stored in motion compensation buffer 48, reference data which is to be supplied to adder 58 is stored in motion compensation buffer 49, reference data which is to be supplied to adder 59 is stored in motion compensation buffer 50, and reference data which is to be supplied to adder 60 is stored in motion compensation buffer 51. Again, there is no problem with overlapping memory accesses with respect to the motion compensation buffers.

Although the above embodiments of the present invention have been described with respect to a decoding apparatus, it should be understood that the same could also be applied to a local decoder provided in a data encoding apparatus.

The moving picture video data decoding apparatus provided in accordance with this invention distributes an input data stream for parallel decoding processing on the basis of synchronizing code signals present in the data stream, and the decoding processing is continuously carried out within a time period between synchronizing codes. Accordingly, there is no limitation placed on the coding method with respect to time periods between synchronizing codes. Thus, parallel decoding processing can be carried out with respect to data that has been encoded by a conventional method, which difference-codes motion vectors, DC coefficients and the like on the basis of differences between a current block and a previous block.

In addition, in the decoding apparatus provided in accordance with this invention, the blocks making up a macroblock are simultaneously processed in parallel so that video data that has been encoded by a conventional encoding method, without modification, can be reproduced at high speed.

Furthermore, decoding of motion-compensation coded video data can be carried out with parallel readout of reference data from a plurality of memory banks based on the same motion vector, so that a plurality of reference data memory banks and motion compensation circuits can be operated in parallel to carry out high speed processing on the basis of a conventional encoding method that is not modified by limiting the range of motion vectors, or by placing other limitations on motion prediction.

As used in the specification and the following claims, the term "image frame" should be understood to mean a signal representing a picture upon which motion-compensated predictive coding is performed. As will be understood by those skilled in the art, such a picture may be formed, for example, of a progressive-scanned video frame, one field of an interlace-scanned video frame, or two fields which together make up an interlace-scanned video frame.

In at least preferred embodiments there is provided a method and apparatus for decoding a video signal in which a plurality of memory units and motion compensation devices are operated in parallel to process video data encoded according to a known standard, and without limiting the range of motion vectors used for predictive coding or requiring similar restrictions on motion predictive compression-coding.

Having described specific preferred embodiments of the present invention with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various changes and modifications may be effected by one skilled in the art without departing from the scope of the invention as defined in the appended claims.

Claims

1. An apparatus for decoding a coded video signal that represents an image frame, said coded video signal having been divided into a plurality of slices (1, 2, 3, 4), each of said slices being a sequence of macroblocks (MB), each of said macroblocks being a two-dimensional array of picture elements of said image frame, said coded video signal being a bit stream that represents a sequence of said slices which together represent said image frame, said bit stream including a plurality of synchronizing code signals, each of which is associated with a respective one of said slices for indicating a beginning of the respective slice, the apparatus comprising:

   a plurality of decoding means (30, 31, 32, 33), each for decoding a respective portion of said coded video signal that represents said image frame; and

   distributing means (25) responsive to said synchronizing code signals for distributing said slices among said plurality of decoding means.

2. An apparatus according to claim 1, wherein said plurality of decoding means is fewer in number than said plurality of slices into which said coded video signal which represents said image frame was divided, and said distributing means distributes said slices in cyclical fashion among said decoding means.

3. An apparatus according to any one of claims 1 and 2, wherein each of said slices represents a portion of said image frame which is one macroblock high and extends horizontally entirely across said image frame.

4. An apparatus according to claim 3, wherein each of said macroblocks is a 16 x 16 array of said picture elements.

5. An apparatus for decoding input signal blocks that were formed by transform encoding and then variable-length encoding blocks of video data, the apparatus comprising:

   decoding means (30, 31, 32. 33) for variable-length decoding a series of said input signal blocks;

   parallel data means (34) for forming plural parallel data streams, each of which includes respective ones of said series of input signal blocks which were variable-length decoded by said decoding means; and

   a plurality of inverse transform means (39, 40, 41, 42) each for receiving a respective one of said parallel data streams and for performing inverse transform processing on the variable-length decoded signal blocks in the respective data stream.

6. An apparatus according to claim 5, wherein said decoding means is one of a plurality of decoding means for variable-length decoding respective series of input signal blocks; and further comprising distributing means (25) for forming said respective series of input signal blocks to be decoded by said plural decoding means from a bit stream representing an image frame and in response to synchronizing signals provided at predetermined intervals in said bit stream representing said image frame.

7. An apparatus for decoding an input digital video signal which includes groups of blocks (83) of prediction-coded difference data, each of said groups consisting of a predetermined plurality of said blocks (MB) and having a respective motion vector (82) associated therewith, each of said blocks of prediction-coded difference data having been formed on the basis of the respective motion vector associated with the respective group which includes said block, the apparatus comprising:

   output means (39, 40, 41, 42) for supplying in parallel blocks of prediction-coded difference data contained in one of said groups of blocks;

   reference data means (43, 53, 54, 55, 56, 48, 49, 50, 51) for supplying in parallel plural blocks of reference data, each of said blocks of reference data being formed on the basis of the motion vector associated with said one of said groups of blocks and corresponding to one of said blocks of prediction-coded difference data supplied by said output means; and

   a plurality of adding means (57, 58, 59, 60) each connected to said output means and said

reference data means for adding a respective one of said blocks of prediction-coded difference data and the corresponding block of reference data.

8. An apparatus according to claim 7, wherein each of said groups of blocks is a macroblock which includes four blocks of prediction-coded data and said plurality of adding means consists of four adders (57, 58, 59, 60) operating in parallel.

9. An apparatus according to any one of claims 7 and 8, wherein said reference data means comprises:

a plurality of reference data memories (44, 45, 46, 47) from which reference data is read out in parallel on the basis of said motion vector associated with said one of said groups of blocks;

a plurality of buffer memories (48, 49, 50, 51), each for temporarily storing the reference data read out from a respective one of said plurality of reference data memories and for reading out the temporarily stored data on the basis of said motion vector associated with said one of said group of blocks; and

distributing means (52) connected between said buffer memories and said adding means for distributing among said plurality of adding means, on the basis of said motion vector associated with said one of said groups of blocks, the reference data read out from said plurality of buffer memories.

10. An apparatus according to any one of claims 7 and 8, wherein said reference data means comprises:

a plurality of reference data memories (44, 45, 46, 47) from which reference data is read out in parallel on the basis of said motion vector associated with said one of said groups of blocks;

a plurality of buffer memories (48, 40, 50, 51), each connected to a respective one of said adding means, for temporarily storing reference data read out from said plurality of reference data memories and for supplying the temporarily stored reference data to its respective adding means; and

distributing means (52) connected between said reference data memories and said buffer memories for distributing among the plurality of buffer memories, on the basis of said motion vector associated with said one of said groups of blocks, the reference data read out from the plurality of reference data memories.

11. An apparatus according to claim 10, wherein each of said buffer memories temporarily stores reference data read out from every one of said reference data memories.

12. An apparatus according to any one of claims 7 to 11, wherein said input digital video signal includes input signal blocks that were formed by transform encoding and then variable-length encoding blocks of prediction-coded difference data, and said output means comprises:

decoding means (30, 31, 32, 33) for variable-length decoding a series of said input signal blocks;

parallel data means (34) for forming plural parallel data streams, each of which includes respective ones of said series of input signal blocks which were variable-length decoded by said decoding means; and

a plurality of inverse transform means (39, 40, 41, 42) each for receiving a respective one of said parallel data streams and for performing inverse transform processing on the variable-length decoded signal blocks in the respective data stream to form blocks of prediction-coded difference data that are supplied to said adding means.

13. An apparatus according to claim 12, wherein said decoding means is one of a plurality of decoding means (30, 31, 32, 33) for variable-length decoding respective series of input signal blocks; and further comprising distributing means (25) for forming said respective series of input signal blocks to be decoded by said plural decoding means from a bit stream representing an image frame and in response to synchronizing signals provided at predetermined intervals in said bit stream representing said image frame.

14. A method of decoding a coded video signal that represents an image frame, said coded video signal having been divided into a plurality of slices (1, 2, 3, 4), each of said slices being a sequence of macroblocks (MB), each of said macroblocks being a two-dimensional array of picture elements of said image frame, said coded video signal being a bit stream that represents a sequence of said slices which together represent said image frame, said bit stream including a plurality of synchronizing code signals, each of which is associated with a respective one of said slices for indicating a beginning of the respective slice, the method comprising the steps of:

providing a plurality of decoding means (30, 31, 32, 33), each for decoding a respective portion of said coded signal that represents said video frame; and

distributing said slices among said plurality of decoding means in response to said synchronizing code signals.

15. A method according to claim 14, wherein said

plurality of decoding means is fewer in number than said plurality of slices into which said coded video signal which represents said image frame was divided, and said distributing step includes distributing said slices in cyclical fashion among said decoding means.

16. A method according to claim 14, wherein each of said slices represents a portion of said image frame which is one macroblock high and extends entirely across said image frame.

17. A method according to claim 16, wherein each of said macroblocks is a 16 x 16 array of said picture elements.

18. A method of decoding input signal blocks that were formed by transform encoding and then variable-length encoding blocks of video data, the method comprising the steps of:

variable-length decoding a series of said input signal blocks;

forming plural parallel data streams, each of which includes respective ones of said variable-length decoded series of input signal blocks; and

performing, in parallel, inverse transform processing on the variable-length decoded signal blocks in the respective data streams.

19. A method according to claim 18, further comprising the steps of:

forming in parallel plural series of input signal blocks from a bit stream representing an image frame of input video signals and in response to synchronizing signals provided at predetermined intervals in said bit stream representing said frame of input signals; and

variable-length decoding, in parallel, the plural series of input signal blocks.

20. A method according to claim 19, further comprising the step of distributing variable-length decoded input signal blocks from every one of said plural series of input signal blocks to each of said plural parallel data streams.

21. A method of decoding an input digital video signal which includes groups of blocks of prediction-coded difference data, each of said groups consisting of a predetermined plurality of said blocks and having a respective motion vector associated therewith, each of said blocks of prediction-coded difference data having been formed on the basis of the respective motion vector associated with the respective group which includes said block, the method comprising the steps of:

outputting in parallel blocks of prediction-coded difference data contained in one of said groups of blocks;

reading out in parallel from memory, on the basis of the motion vector associated with said one of said groups of blocks, plural blocks of reference data, each of said blocks of reference data corresponding to one of said blocks of prediction-coded difference data; and

respectively adding, in parallel, the blocks of prediction-coded difference data contained in said one of said groups of blocks and the corresponding blocks of reference data.

22. A method according to claim 21, wherein said reading out step comprises the sub-steps of:

reading out the reference data from a plurality of memories on the basis of the motion vector associated with said one of said groups of blocks;

distributing, on the basis of the motion vector associated with said one of said groups of blocks, the reference data read out from the plurality of memories;

temporarily storing the distributed reference data; and

reading out the temporarily stored data.

23. A method according to any one of claims 21 and 22, wherein said input digital video signal includes input signal blocks that were formed by transform-encoding and then variable-length encoding blocks of prediction-coded difference data, said outputting step comprising the sub-steps of:

variable length decoding a series of said input signal blocks;

forming plural parallel data streams, each of which includes respective ones of said variable-length decoded series of input signal blocks; and

performing, in parallel, inverse transform processing on the variable-length decoded signal blocks in the respective data streams.

24. A method according to claim 23, further comprising the steps of:

forming in parallel plural series of input signal blocks from a bit stream representing an image frame of input video signals and in response to synchronizing signals provided at predetermined intervals in said bit stream representing said frame of input signals; and

variable-length decoding, in parallel, the plural series of input signal blocks.

25. A method of decoding a prediction-coded video signal that represents an image frame, said prediction-coded video signal having been divided

into a plurality of macroblocks, each of said macroblocks being a two-dimensional array of picture elements of said image frame, the method comprising the steps of:

providing a plurality of memories each for storing reference data which corresponds to a respective portion of said image frame, said plurality of memories together storing reference data which represents a complete image frame; and

distributing data representing a reconstructed image frame for storage in said plurality of memories such that a portion of each macroblock of the reconstructed image frame is stored in each of said plurality of memories.

26. A method according to claim 25, wherein said macroblocks are each composed of a predetermined number of two-dimensional blocks and each of said plurality of memories stores corresponding blocks from all of the macroblocks of an image frame.

27. A method according to claim 26, wherein said plurality of memories consists of first, second, third and fourth memories, said macroblocks are each composed of four blocks which respectively represent upper left, upper right, lower left and lower right quadrants of the respective macroblock, and said distributing step comprises:

storing in the first memory the blocks representing the upper left quadrants of all of the macroblocks;

storing in the second memory the blocks representing the upper right quadrants of all of the macroblocks;

storing in the third memory the blocks representing the lower left quadrants of all of the macroblocks; and

storing in the fourth memory the blocks representing the lower right quadrants of all of the macroblocks.

28. A method according to any one of claims 25, 26 and 27, wherein said distributing step comprises storing a first line of each of said macroblocks in a first one of said plurality of memories and storing a second line of each of said macroblocks in a second one of said plurality of memories.

29. A method according to claim 28, wherein each of said macroblocks is composed of a number of lines that is an integral multiple of a number of memories that forms said plurality of memories, and said distributing step comprises distributing said lines of each macroblock in cyclical fashion among said memories.

30. A method according to claim 29, wherein each of said macroblocks is composed of sixteen lines, and said number of memories is four.

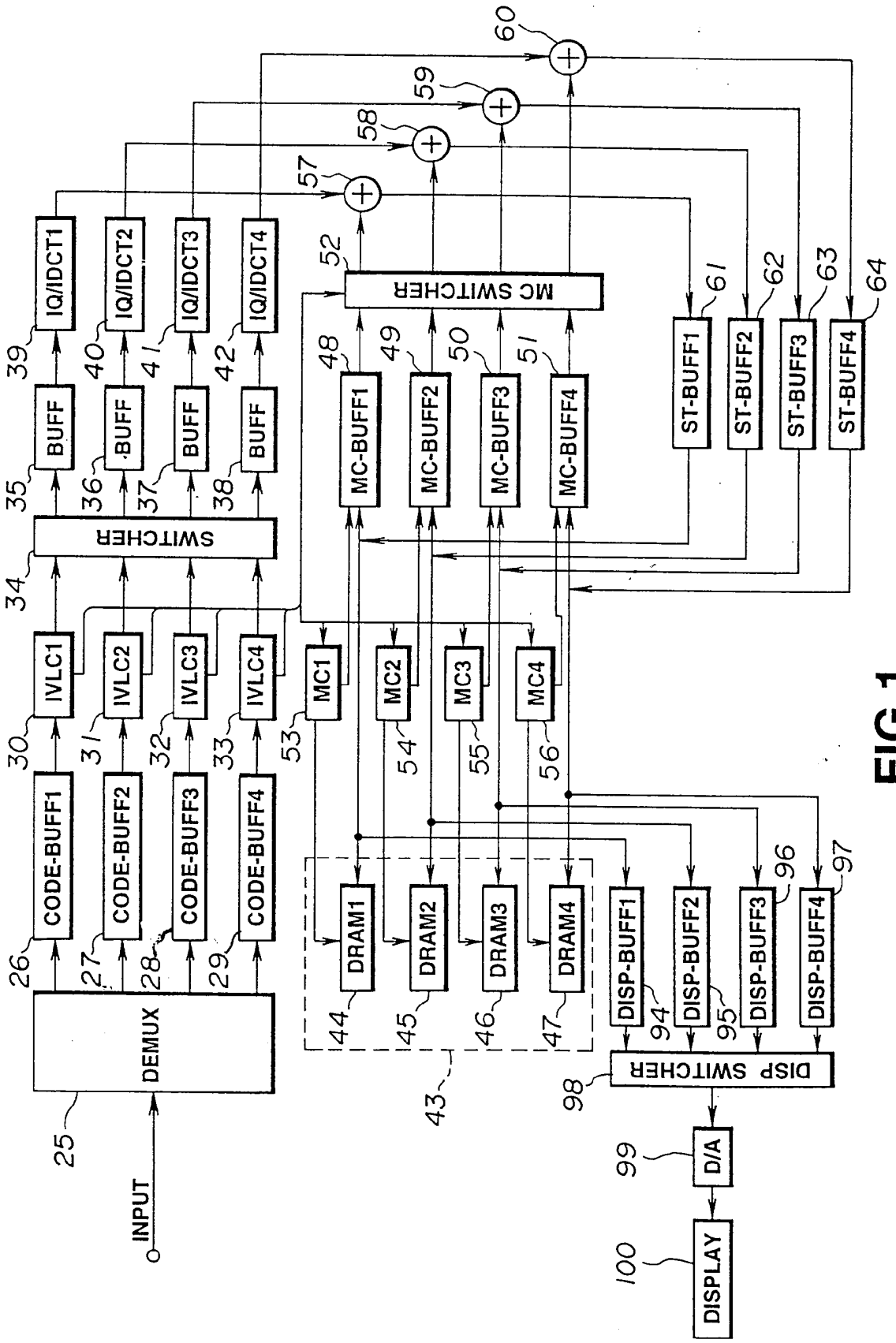


FIG.1

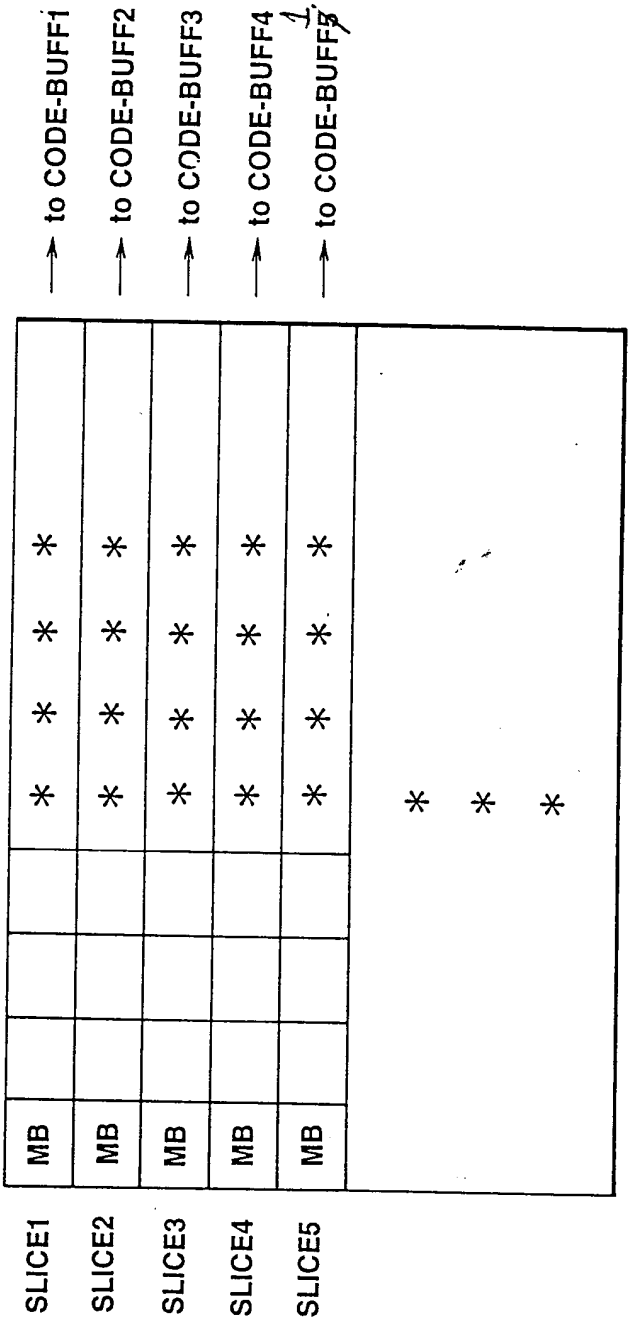


FIG.2

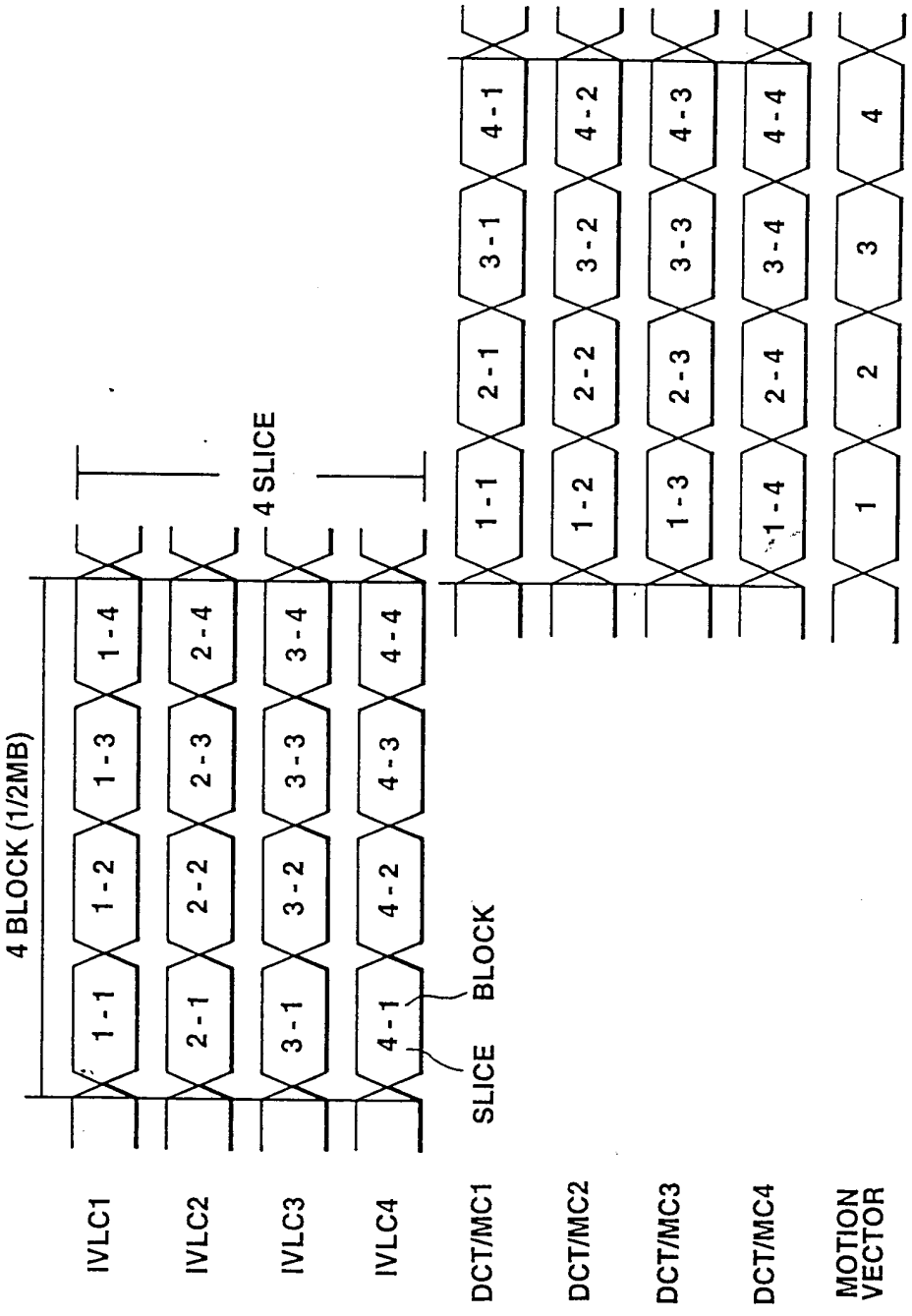


FIG.3

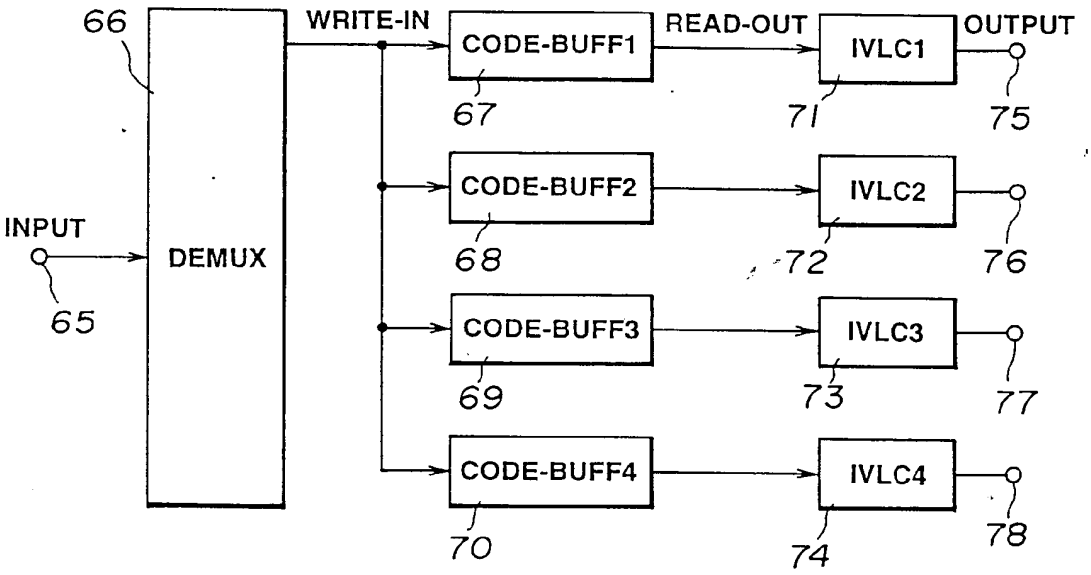


FIG.4

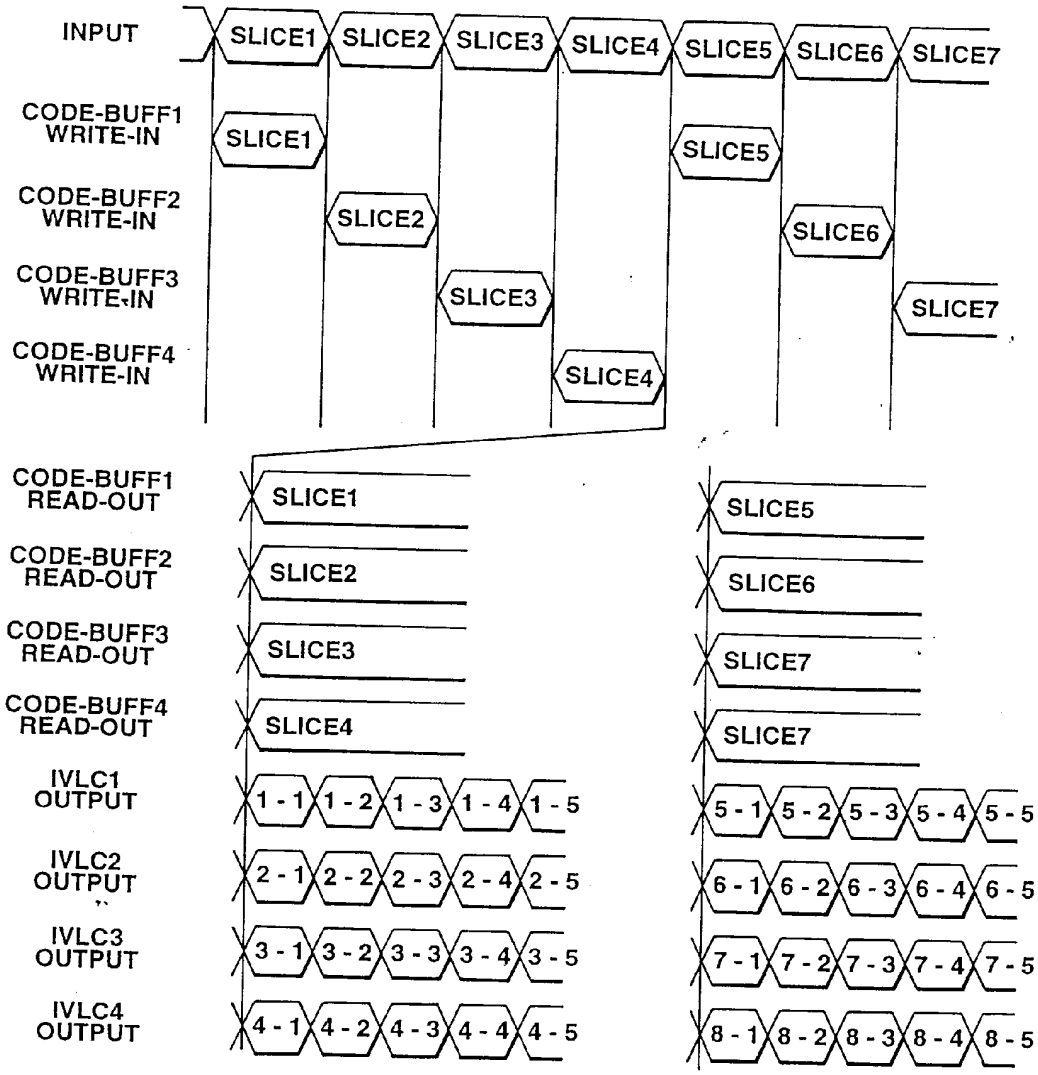


FIG.5

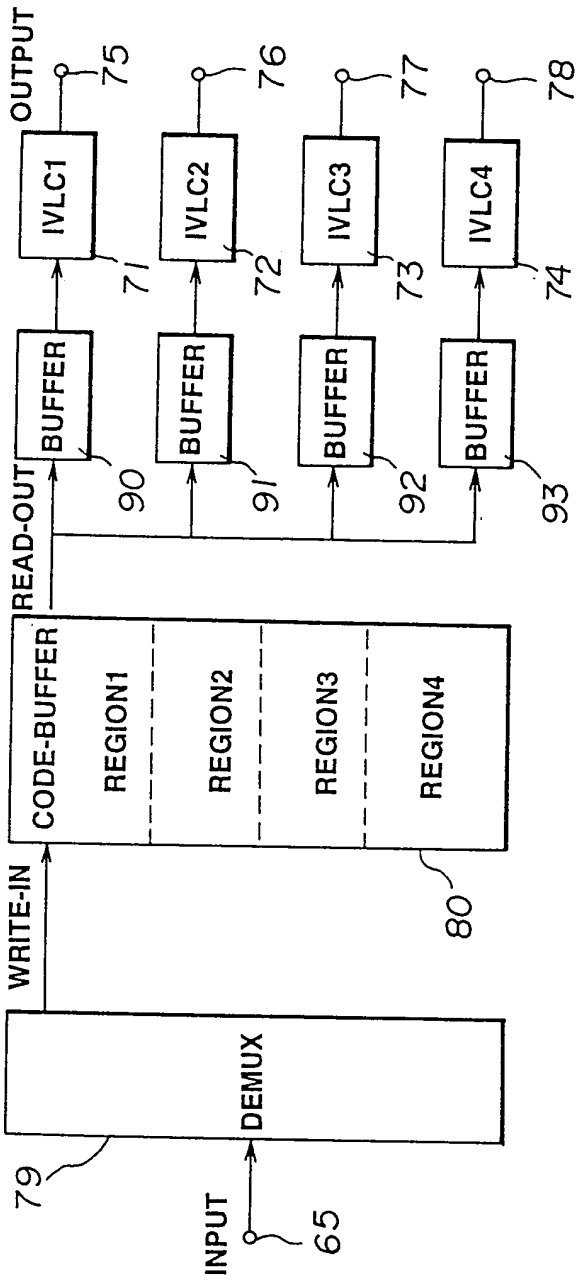


FIG.6

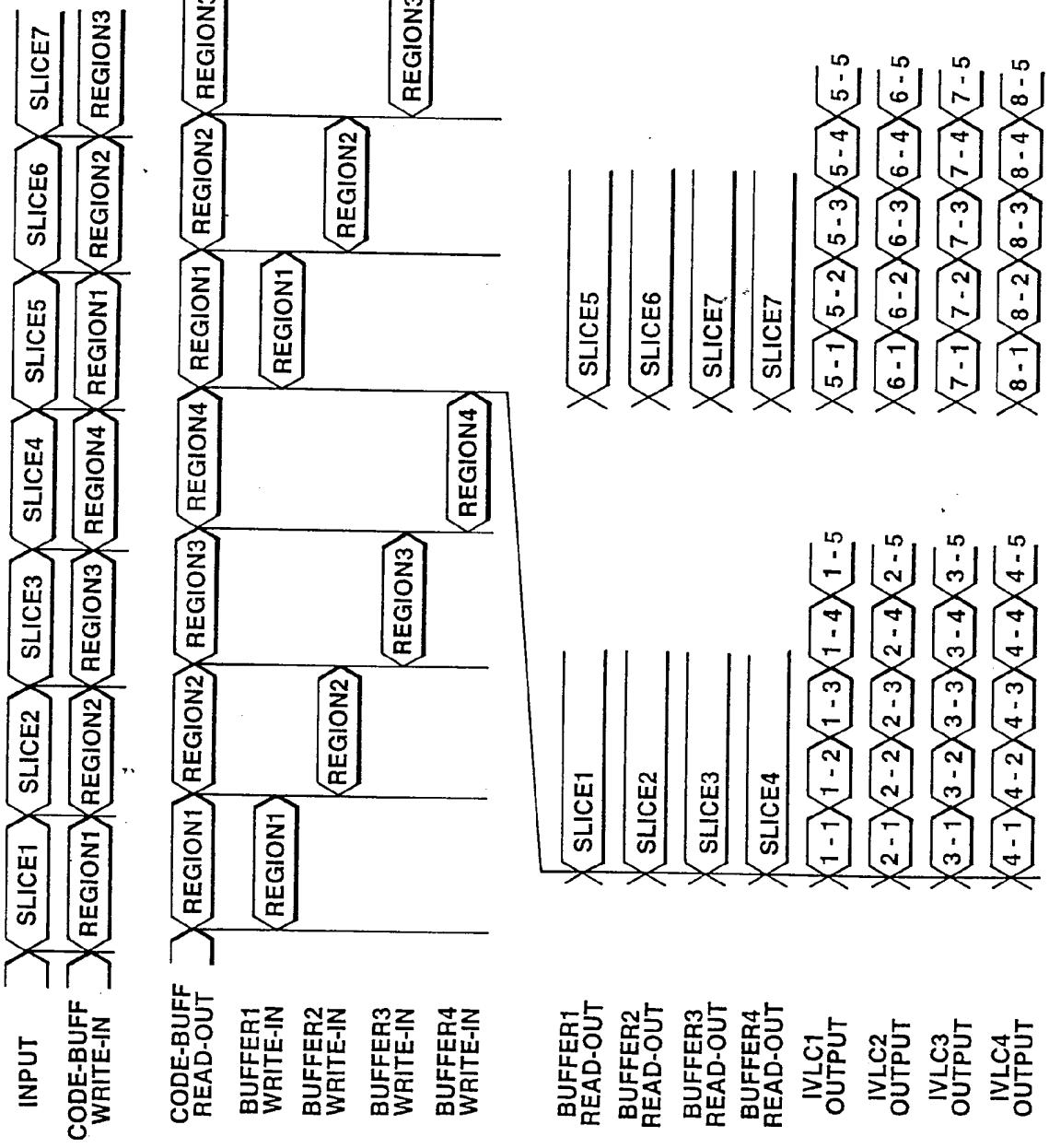


FIG.7

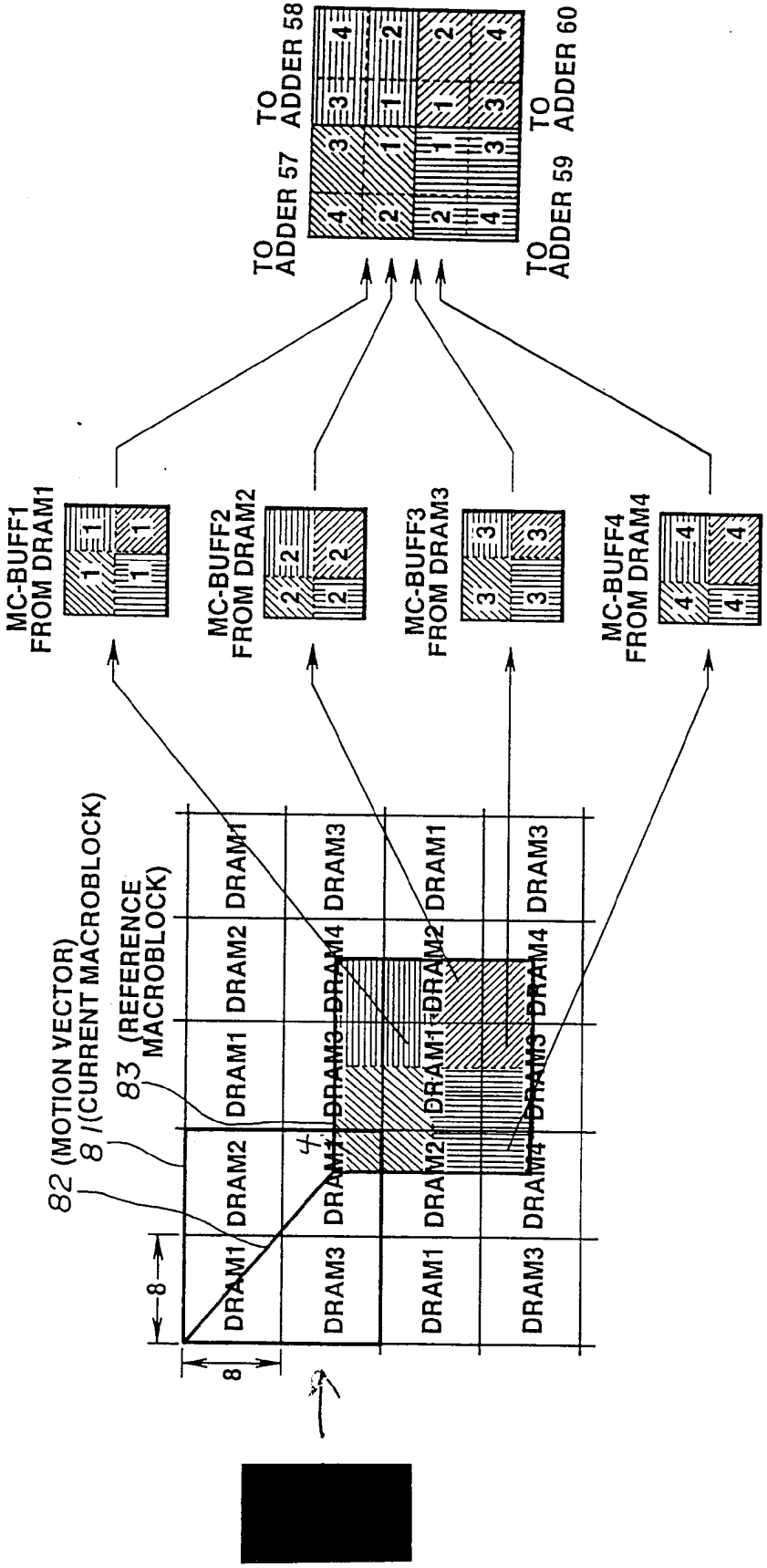


FIG.8(A) FIG.8(B) FIG.8(C)

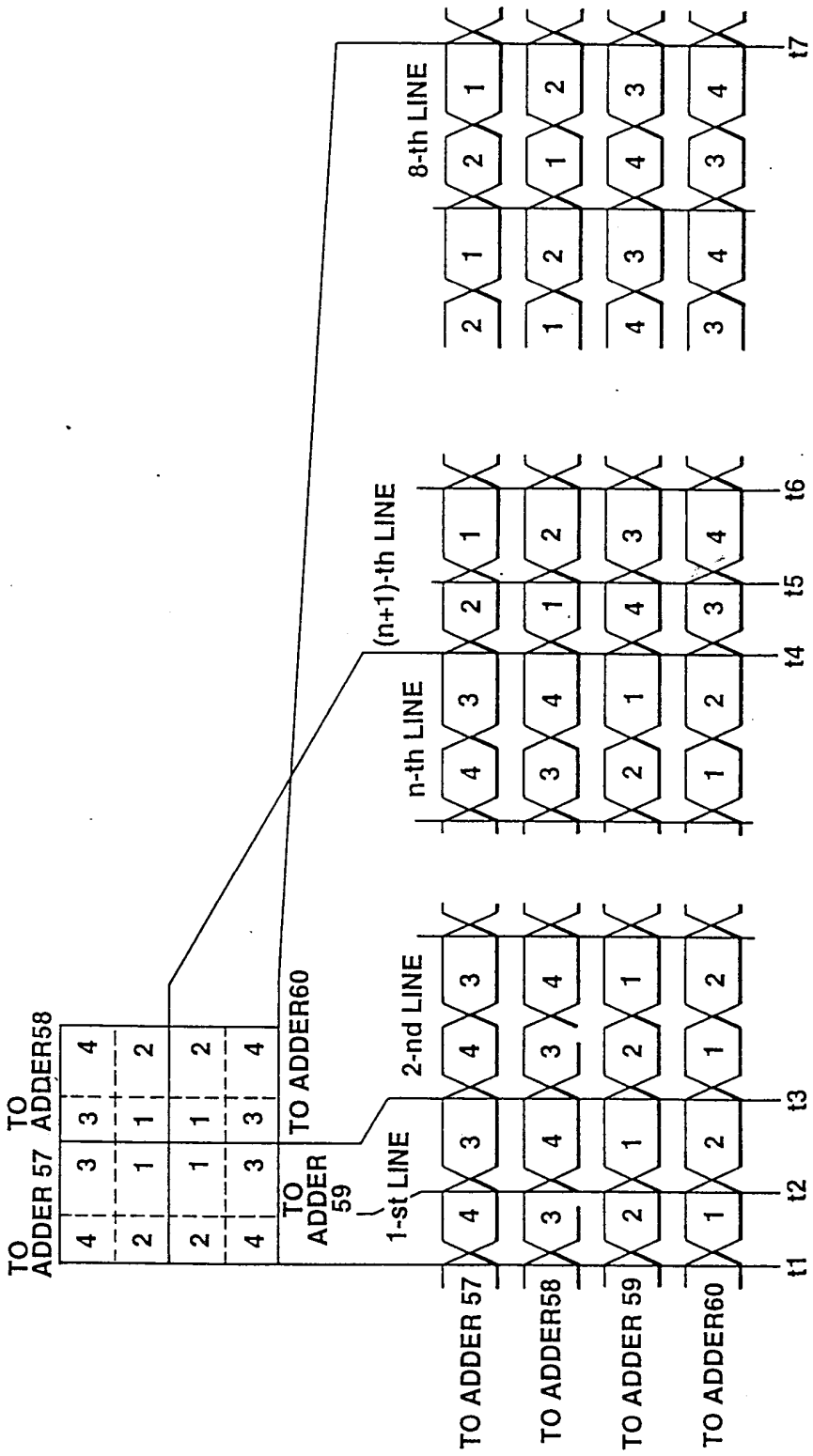


FIG.9

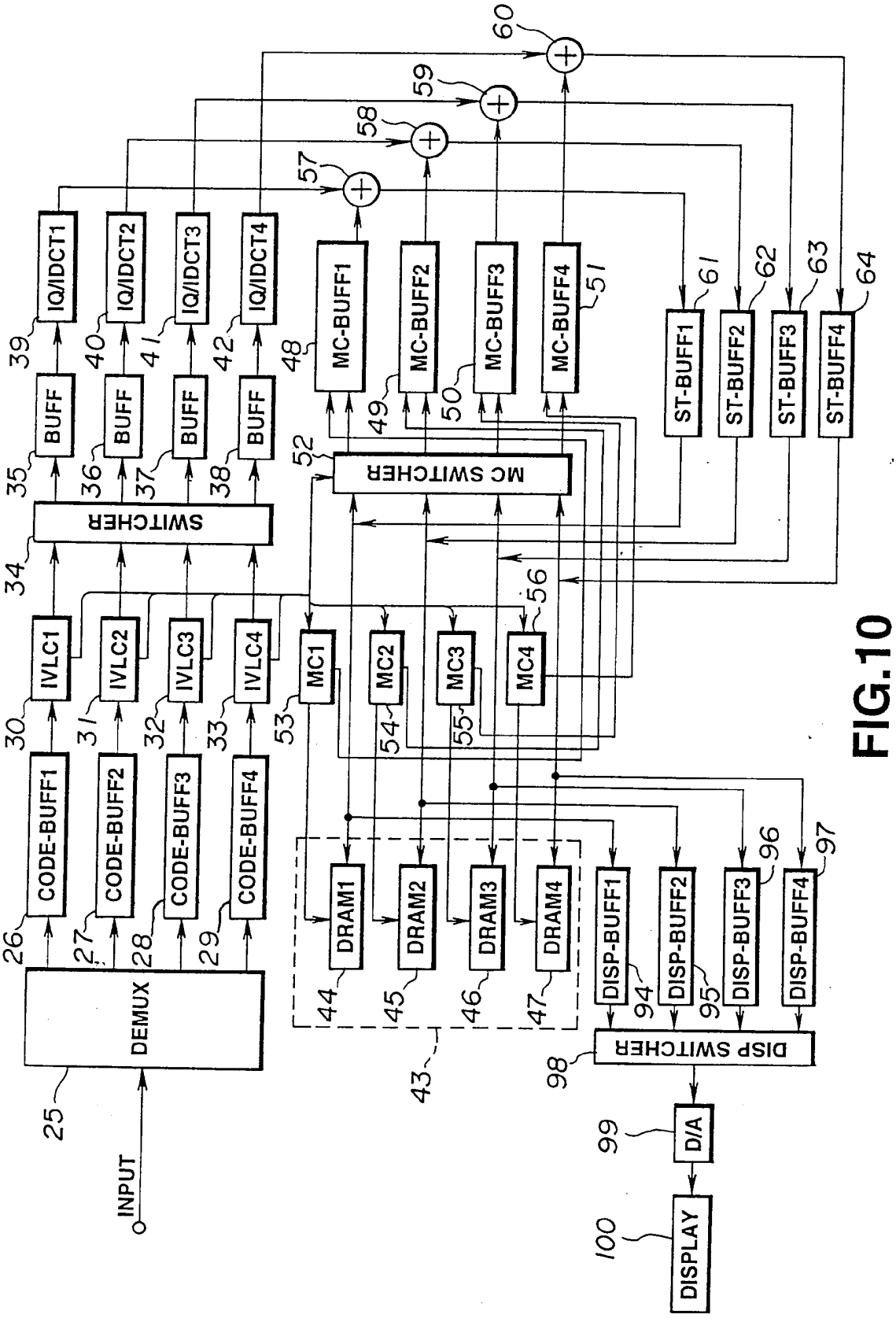


FIG.10

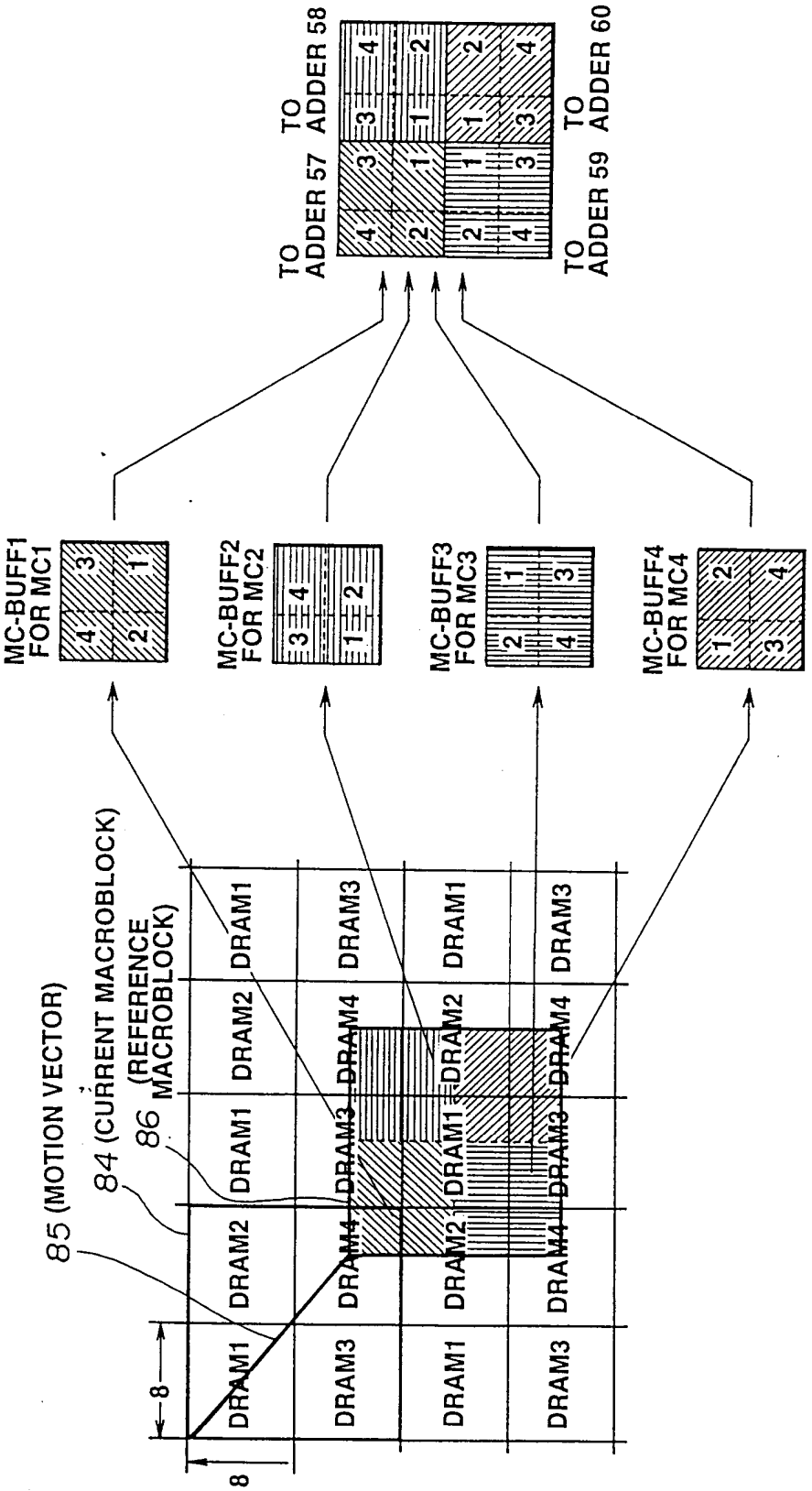


FIG.11(A)

FIG.11(B)

FIG.11(B)

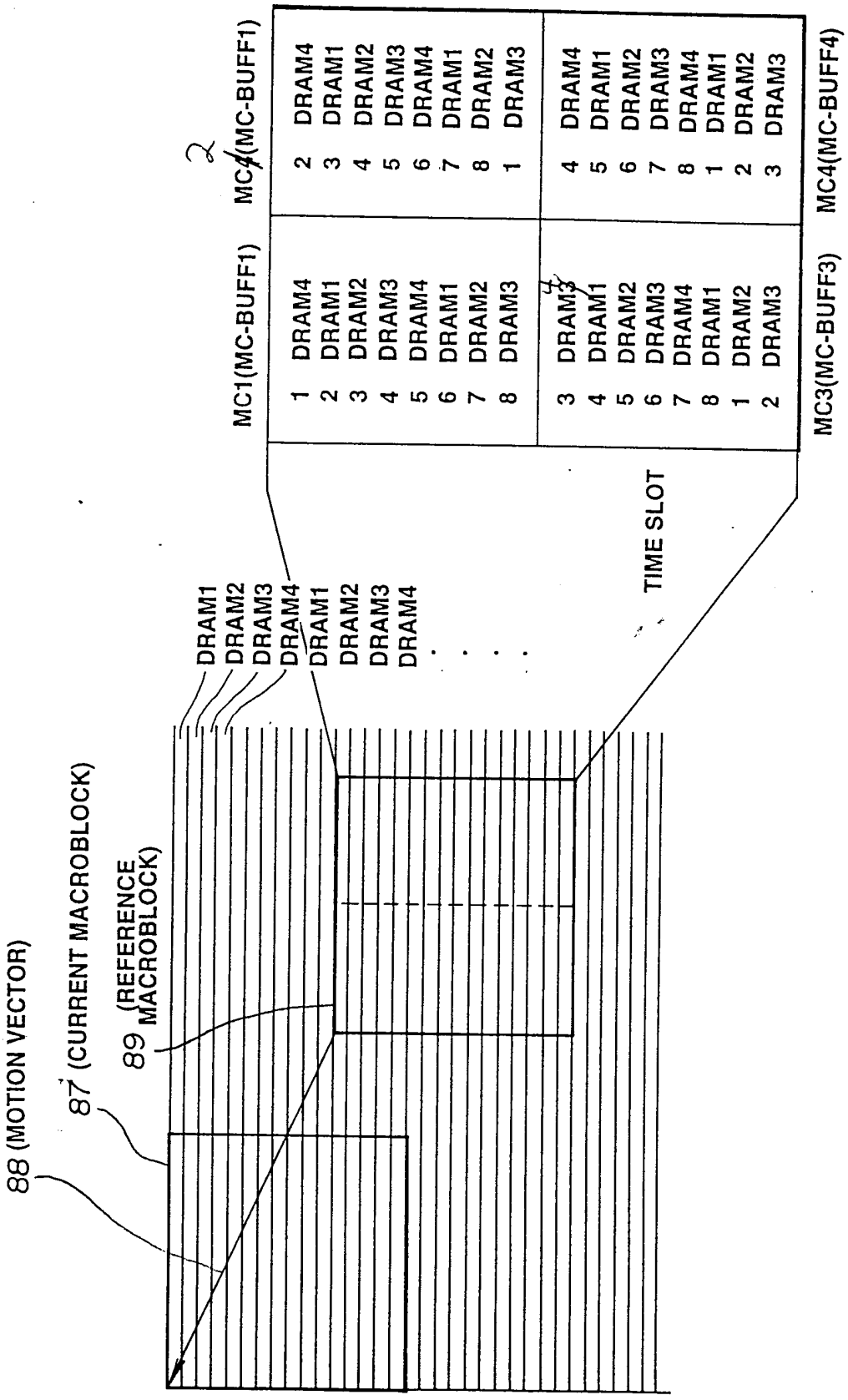


FIG.12(A)

FIG.12(B)

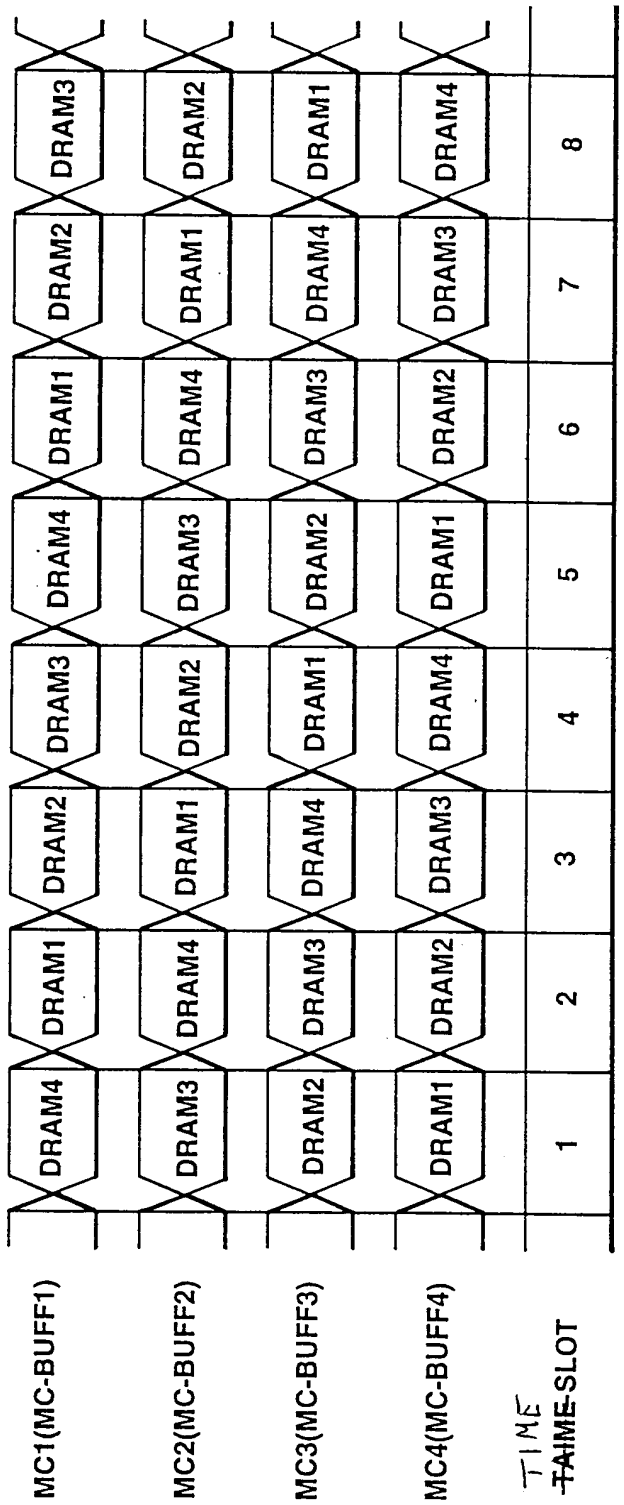


FIG.13

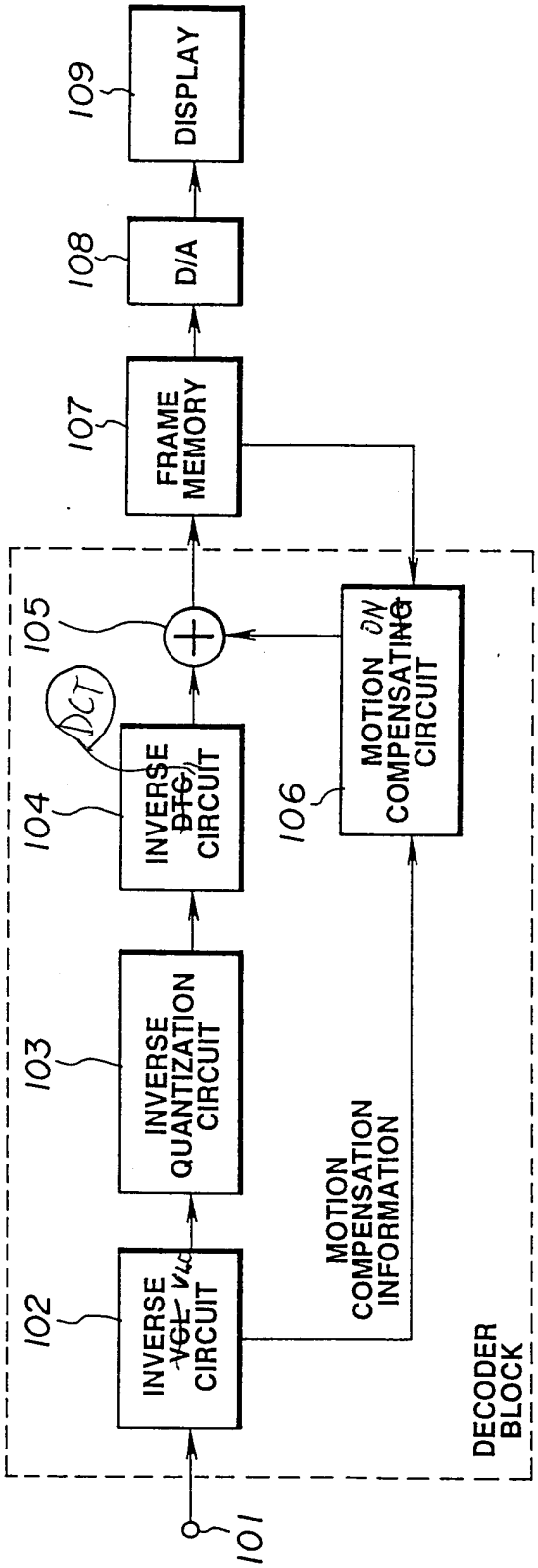


FIG.14

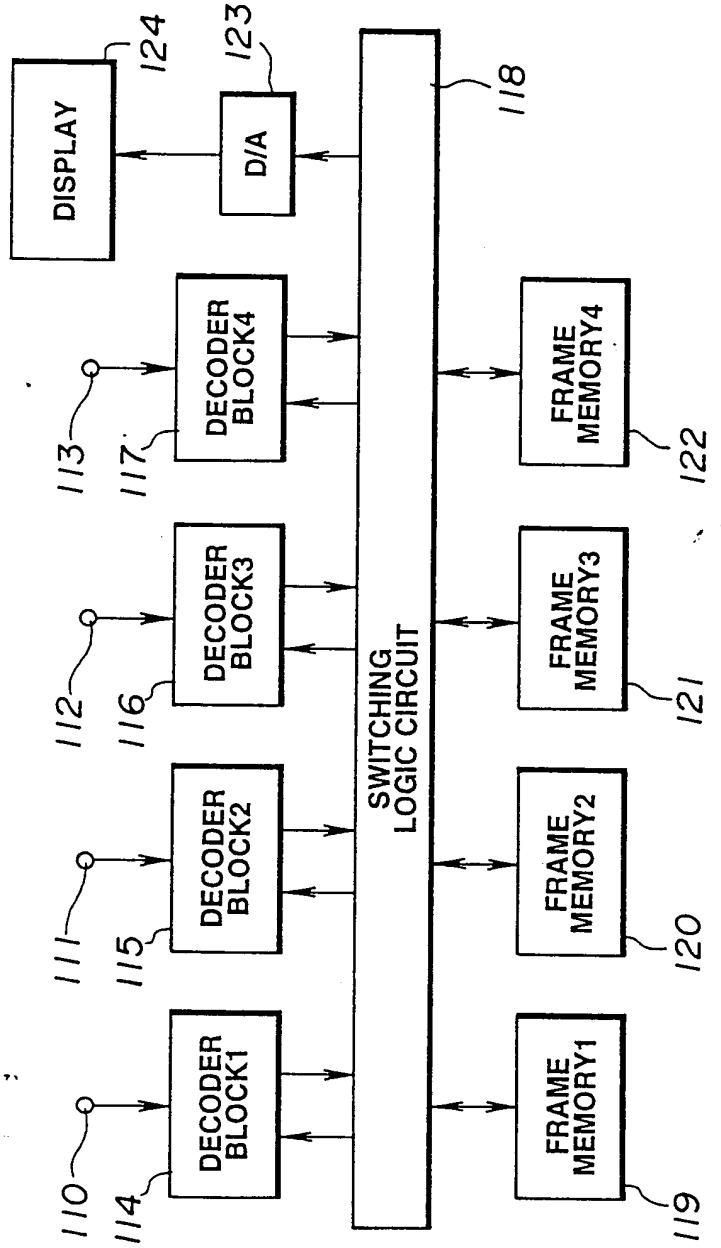


FIG.15

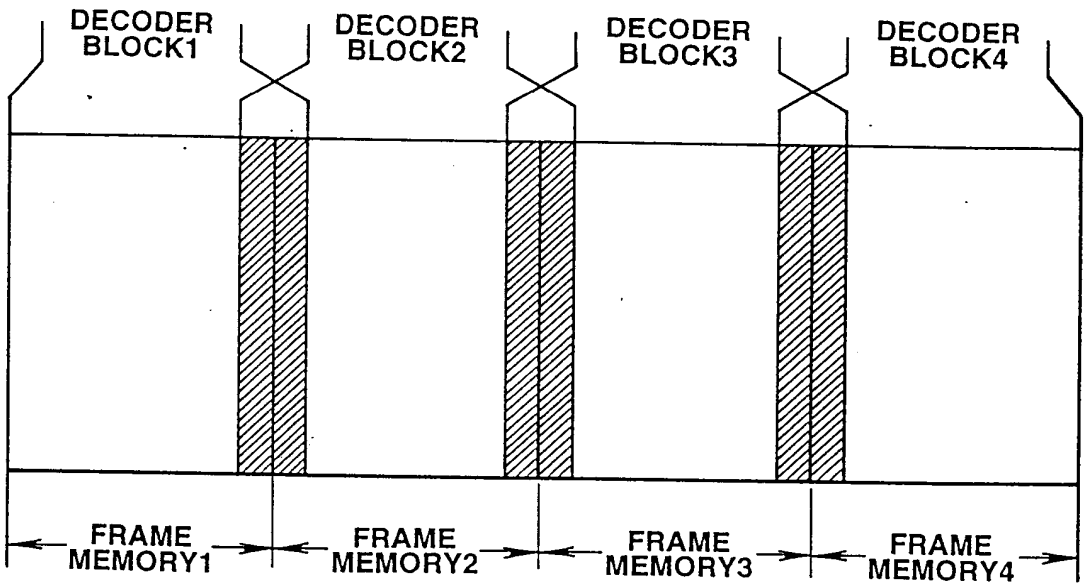


FIG.16

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 614 317 A3**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94301252.6**

(51) Int. Cl.[5] : **H04N 7/13,** H04N 7/137

(22) Date of filing : **22.02.94**

(30) Priority : **05.03.93 JP 45112/93**

(43) Date of publication of application :
07.09.94 Bulletin 94/36

(84) Designated Contracting States :
DE FR GB IT NL

(88) Date of deferred publication of search report :
25.01.95 Bulletin 95/04

(71) Applicant : **SONY CORPORATION**
7-35 Kitashinagawa 6-chome
Shinagawa-ku
Tokyo 141 (JP)

(72) Inventor : **Koyanagi, Hideki, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)
Inventor : **Sumihiro, Hiroshi, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)
Inventor : **Emoto, Seiichi, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)
Inventor : **Wada, Tohru, c/o Intellectual Property Div.**
Sony Corporation,
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo 141 (JP)

(74) Representative : **Robinson, Nigel Alexander Julian et al**
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Video signal decoding.

(57) A digital video signal that has been encoded using motion- compensated prediction, transform encoding, and variable-length coding, is decoded using parallel processing. Frames of the video signal are divided into slices (1, 2, 3, 4) made up of a sequence of macroblocks (MB). The signal to be decoded is slice-wise divided for parallel variable-length decoding. Each variable-length-decoded macroblock is divided into its constituent blocks for parallel inverse transform processing. Resulting blocks of difference data are added in parallel to corresponding blocks of reference data. The blocks of reference data corresponding to each macroblock are read out in parallel from reference data memories (44, 45, 46, 47) on the basis of a motion vector (83) associated with the macroblock. Reference data corresponding to each macroblock is distributed for storage among a number of reference data memories.

EP 0 614 317 A3

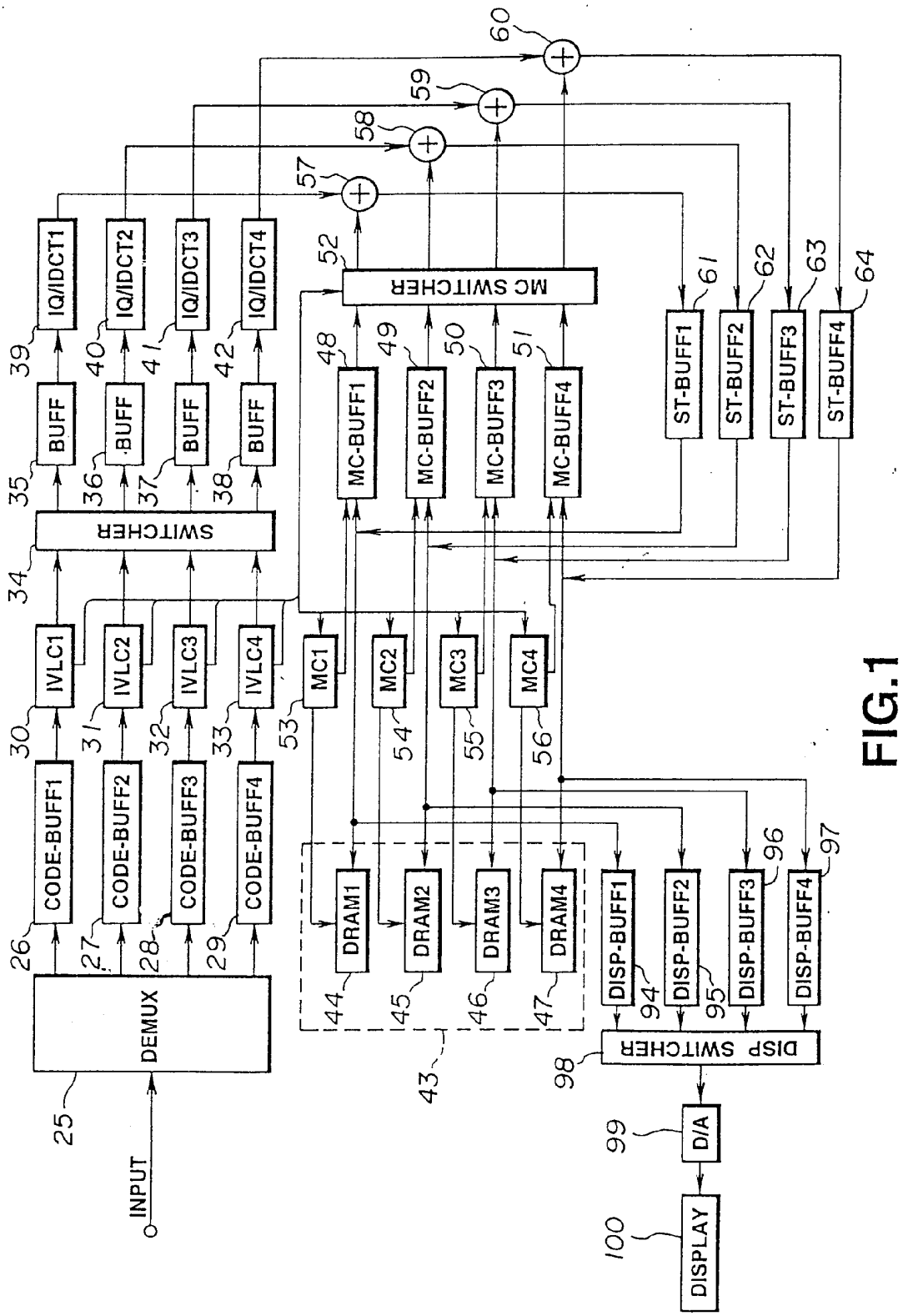


FIG.1

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 30 1252

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
X	EBU TECHNICAL REVIEW, no.251, 1 April 1992, BRUSSEL, BE pages 22 - 33, XP275388 M. BARBERO ET AL. 'DCT source coding and current implementations for HDTV'	1-4, 14-20	H04N7/13 H04N7/137
Y	* paragraph 3.2-3.3; figures 3-5 *	5,6	
A		7-13, 21-30	

Y	EP-A-0 442 548 (LABORATOIRES D'ELECTRONIQUE PHILIPS ET AL.)	5,6	
A	* the whole document *	1-4,7-30	

A	PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON APPLICATION SPECIFIC ARRAY PROCESSORS, 5 September 1990, PRINCETON, NEW JERSEY, US pages 226 - 234, XP245104 T. NISHITANI ET AL. 'A Real-Time Software Programmable Processor for HDTV and Stereo Scope Signals' * paragraph 3; figure 3 *	1-30	
	---		TECHNICAL FIELDS SEARCHED (Int.Cl.5)
X	EP-A-0 479 511 (VICTOR COMPANY OF JAPAN, LTD.)	25,26	H04N
A	* the whole document *	1-24, 27-30	

X	IEEE MICRO, vol.12, no.5, 1 October 1992, LOS ALAMITOS, CA, US pages 22 - 27, XP330853 O. DUARDO ET AL. 'Architecture and Implementation of ICs for a DSC-HDTV Video Decoder System'	1-4, 14-20	
A	* paragraph 1 -paragraph 2; figure 4 *	5-13, 21-30	

	-/--		

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	16 December 1994	Foglia, P

EPO FORM 1503 03.82 (P04C01)

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 30 1252

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
P,A	US-A-5 212 742 (NORMILE ET AL.) * column 9, line 22 - column 14, line 11; figures 4-6 *	1-30	
X	RTM RUNDFUNKTECHNISCHE MITTEILUNGEN, vol.36, no.5, 1 September 1992, NORDERSTEDT, DE pages 196 - 205, XP321766 H. HOFMANN ET AL. 'Ein Codec für die digitale Übertragung von HDTV-Signalen' * paragraph 5; figures 6A-8 *	1,14	
P,A	EP-A-0 535 746 (PHILIPS ELECTRONICS UK LTD. ET AL.)		
A	TRANSPUTER/OCCAM JAPAN 3, 17 May 1990, TOKYO,JP pages 91 - 100 M. ICHIKAWA ET AL. 'Design and implementation of software based real-time video codec using multi-transputer architecture' * paragraph 3.1 -paragraph 4.3.5 *	1-30	TECHNICAL FIELDS SEARCHED (Int.Cl.5)
A	US-A-5 130 797 (MURAKAMI ET AL.)		

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	16 December 1994	Foglia, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 680 185 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **95105803.1**

(51) Int. Cl.[6]: **H04L 29/06**

(22) Date of filing: **19.04.95**

(30) Priority: **28.04.94 US 233908**

(43) Date of publication of application:
02.11.95 Bulletin 95/44

(84) Designated Contracting States:
DE ES FR GB IT

(71) Applicant: **THOMSON CONSUMER ELECTRONICS, INC.**
10330 North Meridian St.
Indianapolis, IN 46206 (US)

(72) Inventor: **Joseph, Kuriacose**
818 Ravens Crest Drive
Plainsboro,
New Jersey 08543 (US)
Inventor: **Dureau, Vincent**
219 Sherman Canal
Venice,
California 90291 (US)
Inventor: **Jessup, Ansley Wayne, Jr.**
22 Elmwood Lane
Willingboro,
New Jersey 08046 (US)
Inventor: **Delpuch, Alain**
2221 Parnell Avenue
Los Angeles,
California 90064 (US)

(74) Representative: **Wördemann, Hermes, Dipl.-Ing.**
Deutsche Thomson-Brandt GmbH,
Patent Dept.,
Göttinger Chaussee 76
D-30453 Hannover (DE)

(54) A distributed computer system.

(57) A distributed computer system is disclosed which comprises a source (30) of a continuous data stream repetitively including data representing a distributed computing application and a client computer (22), receiving the data stream, for extracting (207) the distributed computing application representative data from the data stream, and executing (224) the extracted distributed computing application.

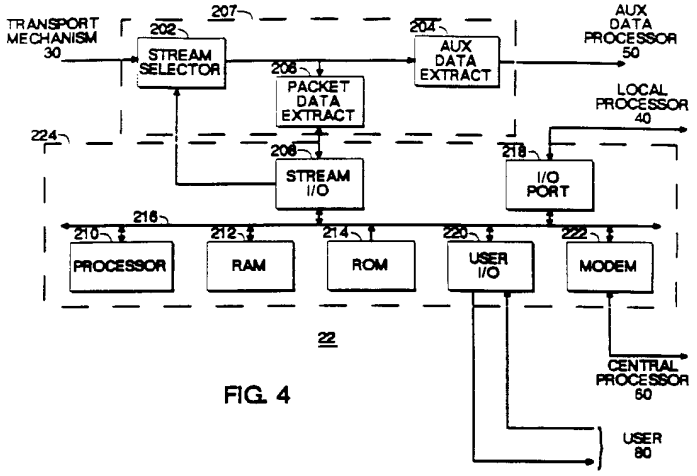


FIG. 4

EP 0 680 185 A2

The present invention relates to a client-server distributed computer system. Such a computer system has application in broadcast multimedia applications.

Early computer systems were standalone systems, consisting generally of mainframe computers. Later, several mainframe computer systems were closely connected, or clustered, to handle larger computing jobs, such as a large number of time sharing users. With the advent of personal computers, large numbers of relatively low power standalone computer systems were controlled directly by their users. Soon these large numbers of personal computers were coupled together into networks of computers, providing shared resources and communications capabilities to the users of the individual personal computers and between those users and the preexisting mainframe computers.

One form of such a network includes a central computer, called a server, which generally includes a large amount of mass storage. Programs used by the network users are centrally stored in the mass storage on the server. When a user desires to run a program, the user's computer requests that a copy of that program be sent to it from the server. In response to that request, the server transfers a copy of the program from its mass storage to the main memory of the personal computer of that user, and the program executes on that personal computer. Data also may be centrally stored in the server and shared by all the users on the network. The data is stored on the mass storage of the server, and is accessible by all the network users in response to a request. The server also serves as a hub for communications of messages (electronic mail) between network users The server in such a system handles the storage and distribution of the programs, data and messages, but does not contribute any processing power to the actual computing tasks of any of the users. I.e. a user cannot expect the server computer to perform any of the processing tasks of the program executing on the personal computer. While such networks perform a valuable function, they are not distributed computing systems, in which interconnected computers cooperate to perform a single computing task.

In an improvement to such networks, the network may be configured in such a manner that a user on the network may request that the server, or other personal computer connected to the network, execute a program. This is termed remote execution because a computer (server or other personal computer) remote from the requester is executing a program in response to a request from the requester. In such a system, the program of which remote execution is requested is either sent from the requester to the remote computer, or retrieved from the server in response to a request by the remote computer. When the program is received, it is executed. In this manner several computers may be enlisted to cooperate in performing a computing function.

Recently, there have been programs which distribute the actual computing tasks necessary for performing a single computing function. For example, in such a data base program, where the data base is stored in the mass storage of the server, if a user desires to make a query of the data base, the portion of the data base management program on that user's personal computer will generate a query request, which is forwarded to the server. The portion of the data base management program on the server performs the query processing, e.g. parsing the query request, locating where the data specified in the query request resides on its mass storage device, accessing that data, and sending the results back to the requesting personal computer over the network. The portion of the data base management program on the personal computer then processes the data received from the server, e.g. formatting it, and displaying it on the screen or printing it on a printer. While the server is processing the query request, the personal computer is free to perform other processing, and while the personal computer is generating the query request, and processing the resulting data received from the server, the server is free to process query requests from other personal computers.

Other types of programs are also amenable to this type of distributed computing, termed client-server computing. The sharing of the processing tasks between the personal computer and the server improves the overall efficiency of computing across the network. Such client-server computer systems, and remote execution networks, may be termed distributed computing systems because several computers (the server and/or the respective peripheral computers) cooperate to perform the computing function, e.g. data base management.

Recently, broadcast multimedia programs, more specifically, interactive television (TV) programs, have been proposed. Interactive TV programs will allow a viewer of a television program to interact with that program. In an interactive TV system, the central broadcast location (TV network, local TV studio, cable system, etc.) will have a central computer, corresponding to the server computer, which will produce signals related to the interactive TV program to be broadcast simultaneously with the TV (video and audio) signals. These signals carry data representing the interactive TV program and may include commands, executable program code and/or data for controlling the viewer interaction. Each viewer location will have a computer, corresponding to the client computer, which will receive the commands, executable

code and/or data from the central computer, execute the executable code, process the received data, accept input from the user and provide data to the user by means of the TV screen. The input from the user may be sent back to the computer at the broadcast location, allowing the user to interact with the interactive TV program.

U.S. Patent 4,965,825, SIGNAL PROCESSING APPARATUS AND METHODS, issued Oct. 23, 1990 to Harvey et al., describes an interactive TV system in which a central broadcast location includes signals carrying commands, executable code and data in, for example, the vertical blanking interval of the television signal for receipt by the computer systems at the viewer locations. A computer at the viewer location extracts the commands, executable code and data and executes the code to process the data and interact with the user. Such a system is comparable to the remote execution function of distributed computer systems, described above, in that the viewer computer is enlisted into the interactive TV program, and is controlled by the central location.

In all of the above systems, a central computer controls or responds to requests from peripheral computers attached to it through a network. I.e. the peripheral computer (personal computer) requests remote execution of a program, requests a file or message from, or sends a query request to, another computer. Only in response to a request does the other computer provide a response, e.g. remote execution, the requested file, message or retrieved data. In addition, in general, the peripheral computer is required to have all the resources necessary to completely, or almost completely, execute the desired program, with the server acting only as another storage mechanism. or at most sharing a portion of the computing tasks.

The inventors herein propose a distributed computing system in which a server computer continuously produces a data stream. This data stream acts a mass storage device for the client computers receiving it. This data stream repetitively includes data representing a distributed computing application in which the client computer may participate, including executable code and data. A transport mechanism, including a high speed, one-way, communication path, carries the data stream from the server to the client. The client receives the data stream, extracts the distributed computing representative data and executes the distributed computing application.

In accordance with principles of the present invention, a distributed computer system comprises a source of a continuous data stream repetitively including data representing a distributed computing application and a client computer, receiving the data stream, for extracting the distributed computing application representative data from the data stream, and executing the extracted distributed computing application.

In a distributed computing system according to the invention, the client computer system need not include all the resources, in particular, main memory and mass storage, necessary to perform the entire program. Instead, no mass storage is required because the data stream provides the function of the mass storage device, and the main memory requirement is modest because only the currently executing portion of the program need be stored in memory. When the currently executing portion has completed, its memory space is freed up, and the next executing portion is extracted from the data stream, stored in the freed memory space, and that portion begins execution.

In addition, a distributed computing system according to the present invention allows the user of the client computer to have the option of participating in the distributed computing task. If it is desired to participate, the client computer extracts the data representing the distributed computing application and executes the distributed computing application, as described above. If it is desired not to participate, the data stream is merely ignored, and the processing desired by the user, or none at all, is performed. Such a distributed computing system also allows each participating client computer to join the distributed computing function at any time and to proceed at its own pace in performing its own computing function.

A distributed computing system according to the present invention is particularly amenable to interactive TV applications because it allows a viewer to tune into an interactive TV channel at any time, join in the interactivity whenever desired (or not at all), and allows all the viewers to proceed at their different paces. This is especially advantageous in an environment when an interactive commercial, with its own executable code and data, may be presented within an interactive program, or when the viewer wishes to change channels.

BRIEF DESCRIPTION OF THE DRAWING

In the drawing:

FIGURE 1 is a block diagram of a distributed computing system according to the present invention;

FIGURE 2 is a block diagram of a server computer as illustrated in FIGURE 1;

FIGURE 3 is a timing diagram illustrating the data streams produced by a server computer in a distributed computing system as illustrated in FIGURE 1;

FIGURE 4 is a block diagram of a client computer as illustrated in FIGURE 1.

FIGURE 1 is a block diagram of a distributed computing system according to the present invention. In FIGURE 1, a server computer 10, which may include a large computer system, is coupled to a plurality of client computers 20 through a transport mechanism 30. The server computer 10 may be coupled to more than the three client computers 20 illustrated in FIGURE 1, and the client computers 20 may be geographically widely dispersed. Client computer 22 is bidirectionally coupled to a local computer 40, to an auxiliary data processing system 50 and to a central processing facility 60. The central processing facility 60 is bidirectionally coupled to the server computer 10. The central processing facility 60 may also be connected to facilities other than the server computer 10 illustrated in FIGURE 1. The local computer 40 is further bidirectionally coupled to a mass storage device 70. The client computer 22 interacts with a user 80 by providing information to the user via a display screen or other output device (not shown) and by accepting information from the user via a keyboard or other input device (also not shown).

Client computers 24 and 26 also interact with their users, (not shown in order to simplify the drawing). In addition, client computers 24 and 26 are bidirectionally coupled to the central processing facility 60. Such links are optional, however. The only requirements for any client computer 20 is a way to interact with a user, and a connection to the transport mechanism 30. Links to local computers, auxiliary data processing systems, and the central processing facility 60 are all optional, and need not be present in every one of the client computers 20.

The transport mechanism 30 includes a unidirectional high speed digital data link, such as a direct fiber optic or digital satellite link from the server 10 to the client computers 20. The data may be transported over the transport system 30 by a packet data system. In such a system, a stream of data packets, each including identification information indicating, among other things, the type of data contained in that packet and the actual data, is transmitted through the data link. Such a packet data system allows several independent streams of data, each identified by identification information in their packets, to be time multiplexed within a single stream of packets.

In addition, it is possible to multiplex a plurality of such packet data streams over respective channels on the same physical medium (fiber optic or satellite radio link) making up the transport mechanism 30. For example, different data streams may be modulated on carrier signals having different frequencies. These modulated carriers may be transmitted via respective transponders on a satellite link, for example. Further, if a particular transponder has sufficient capacity, it is possible to time multiplex several data streams on a single modulated carrier.

The client computers 20 each contain a data receiver for selecting one of the streams of packets being transported over the transport mechanism 30, receiving the selected stream of packets and extracting the data contained in them. Continuing the above example, the data receiver may include a tunable demodulator for receiving one of the respective modulated carriers from the satellite link. In addition, the data receiver may include circuitry for time demultiplexing the respective data streams being carried by that modulated carrier.

In operation, the server 10 produces a continuous data stream in the form of a stream of packets for the client computers 20. The server 10 repetitively inserts a packet, or successive packets, containing data representing the distributed computing application, including at least one executable code module, into the data stream. This code module contains executable code for the client computers 20. The data receiver in, for example, client computer 22, continuously monitors the packets in the data stream on transport mechanism 30. When a packet including identification information indicating that it contains the code module (or a portion of the code module) required by the client computer 22 is present in the data stream, the client computer 22 detects its presence, extracts the code module (or the portion of the code module) from that packet and stores it in the main memory. When the code module is completely received, the client computer 22 begins to execute it.

There may be more than one code module placed in the continuous data stream, each containing a different portion of the distributed computing application. For example, it is possible to divide the distributed computing application into small portions in such a manner that only one portion at a time need be executed at a time. The portion of the distributed computing application currently needed to execute is loaded into the memory of the client computer 22. When that portion has completed its execution, then a code module containing the executable code for the next portion of the distributed computing application is extracted from the data stream, stored in memory and executed. Each portion is extracted from the data stream as needed. If there is sufficient memory in the client computer 22, it is possible to load several code modules into the memory and switch between them, without extracting them from the data flow, but this is not necessary. By structuring a distributed computing application in this manner, the required memory size of the client computer 22 may be minimized

The server 10 may also repetitively include a packet or packets containing one or more data modules in the data stream. The data modules contain data to be processed by the executable code in the code module. Prior to, or during the execution of the code from a previously extracted code module, the client computer 22 may require access to the data in the data module or modules. If so, the client computer 22 monitors the data stream for the required data module or modules. When packets containing the data module or modules (or portions of the data module or modules) are present in the data stream, they are extracted, and the contents stored in the main memory of the client computer 22. When all the required data modules have been completely received, the client computer 22 begins or continues execution of the code from the code module to process the data from the received data module or modules. As is the case for code modules, it is possible for more than one data module to be stored in memory, if there is sufficient memory in client computer 22.

The server 10 may further repetitively include in the data stream a packet or packets containing a directory of the code and data modules currently being included in the data stream. The directory includes a list of all the code and data modules which are present in the data stream, along with information about those modules. If a directory is present in the data stream, then, prior to extraction of any code or data modules from the data stream, the client computer 22 monitors the data stream for the directory. When packets containing the directory (or portions of the directory) are present in the data stream, they are extracted, and their data stored in the main memory of the client computer 22. When the directory has been completely received, the client computer 22 evaluates the entries in the directory, then requests the first code and/or data module from the data stream and execution proceeds as described above.

Any of the client computers 20 may join the distributed computing function represented by the packet stream at any time, and each of the client computers 20 may operate at its own speed, generally in response to the user 80. In order to allow for this, the server 10 repetitively places the directory and all the code and data modules which the client computers 20 may require to perform their portion of the distributed computing function into the data stream on the transport mechanism 30. Whenever one of the client computers 20 joins the distributed computing function, it monitors the newly selected packet stream on the transport mechanism 30 for the directory module, extracts it, and processes it as described above. During execution, whenever one of the client computers 20 requires the a new code and/or data module, it monitors the data stream on the transport mechanism 30 for the newly required code and/or data module, extracts it and either executes it, if it is a code module, or processes it if it is a data module, as described above.

The packet data stream may also include packets of auxiliary data. This data is not required by the client computer 22 for execution of the code, although it may be related to the execution because the user 80 may interact with the executing program on the client computer 22 based on received auxiliary data. The data stream receiver in the client computer 22 recognizes the auxiliary data packets in the data stream on the transport mechanism 30 and passes them directly to the auxiliary data processor 50. The auxiliary data processor 50 processes its packets independently of the client computer 22. If the auxiliary data must be presented to the user 80, the auxiliary data processor 50 may provide its own display device (not shown) which may be shared with the client computer 22, or the display device (not shown) associated with the client computer 22 may be shared with the auxiliary data processor 50, to provide a single information display to the user 80. The auxiliary data processor 50 may have links to other illustrated elements (not shown), but that is dependent upon the type of data.

In an interactive TV system, for example, the auxiliary data includes the video and audio portions of the underlying television signal. For example, the auxiliary data would include video packets containing MPEG, or MPEG-like, encoded data representing the television image and audio packets containing digitally encoded audio. Further, there may possibly be several different audio packet streams carrying respective audio channels for stereo, second audio program (SAP) or multilanguage capability. In an auxiliary data processor 50 in such a system, the video packets would be supplied to a known MPEG (or similar) decoder (not shown) which would generate standard video signals, which would be supplied to a television receiver or video monitor (not shown). The audio packets would be supplied to a known audio decoder (not shown) which would generate standard audio signals for the television receiver or speakers (not shown).

In such an interactive TV system, the client computer 22 may, in response to execution of the executable code module, generate graphic displays to supply information to the user 80. These graphic displays may be combined with the standard video signal from the MPEG decoder in a known manner, and the combined image displayed on the television receiver or video monitor. The client computer 22 may also generate sounds to provide other information to the viewer. The generated sounds

may be combined, in known manner, with the standard audio signals from the audio decoder, and the combined sound played through the television receiver or speakers.

Furthermore, time code data may be included in either or both of the television auxiliary packet data stream and the packet data stream representing the interactive TV application. This permits synchronization of any graphic images or sounds generated by the client computer 22 with the television signal from the auxiliary data. In this case, the client computer 22 would have access to the time code data, and would control the generation of the graphic image and/or sound to occur at the desired time, as supplied by the time code data.

In such an interactive TV system, both the client computer 22 and the auxiliary data processor 50 may be contained in a single enclosure, such as a television receiver, or television set-top decoder box. A television receiver, or decoder box would include connectors for attaching to a local computer or other equipment.

The user 80 provides input to the program running on the client computer 22 during its execution. This data may be required by the server 10 in order to effect the distributed computing function. In an interactive TV system, for example, user 80 may provide input to the client computer through a handheld remote control unit.

The user data is transferred to the server computer 10 via the central processing facility 60. In one embodiment, data is sent from the client computers 20 to the server computer 10 via modems through the telephone system acting as the central processing facility 60. The server computer 10 receives and processes the data received from the client computers 20 during execution of its portion of the distributed computing function.

Server computer 10 may generate new, or modify existing, code and/or data modules in the data stream on the transport mechanism 30, in a manner described below, based on that received data. Alternatively, the server computer 10 may immediately return information to the client computers 20 in the other direction through the central processing facility 60. The information in newly generated code and/or data modules is processed by all client computers 20 participating in the distributed computing function, while information passed from the server computer 10 to the client computers 20 through the central processing facility 60 is specifically related to the client computer (22, 24, 26) to which that information was sent.

In another embodiment, the central processing facility 60 may include its own computer system, separately connected by modem to both the client computers 20 and the server computer 10 through the telephone system. In either of the above embodiments, the central computing facility 60 provides access to other computers or processing facilities (not shown) via the telephone system. Thus, if information from other computer systems is needed to perform the distributed computing function, those computer systems may be accessed via modem through the telephone system by either the client computers 20 or the server computer 10.

An input/output (I/O) port on the client computer 22 is coupled to a corresponding port on the local computer 40. Local computer 40 is collocated with the client computer 22. Local computer 40 may be a personal computer used by the user 80 of the client computer 22, or may be a larger computer, or computer network located at the same site as the client computer 22. This allows the client computer 22 to access data on the attached mass storage 70 of the personal computer or a computer on the network located at the client computer 22 site. In addition, the client computer 22 may use the mass storage 70 of the local computer 40 for storage of data to be retrieved later. It is likely that the local computer 40 will include both an output device (not shown) such as a computer monitor and an input device (also not shown) such as a computer keyboard. Both of these may be shared with the client computer 22 and/or the auxiliary data processor 50, as described above.

For example, the distributed computing system illustrated in Figure 1 may be part of a widespread corporate computing system, and the server 10 may be located at a central location of that corporation. The client computer 22 may be located at a remote location, and the local computer 40 may be coupled to the personal computer network at that location. Workers at that location may store shared data (e.g. financial information) on the server connected to that network. The distributed computing function may include gathering local financial data from the client computers at the remote locations, processing that financial data and returning overall financial results to the client computers. In such an application, the executable code executed on the client computer 22 accesses the data from the local computer 40 (either from its attached mass storage 70 or through the network) through the I/O port, and sends it to the server computer 10 through the central processing facility 60. The server computer 10 continues its processing based on the information received from client computer 22 (and other client computers 20), and returns the results of that processing to the client computers 20 either through the central processing facility 60 or via the data stream on the transport mechanism 30.

In another example, the distributed computing system may be an interactive television system, broadcasting a home shopping show as the distributed computing application. In such a case, the auxiliary data carries the video and audio portion of the television signal, which may show and describe the items being offered for sale, and may include both live actors and overlaid graphics generated at the central studio. Code and data modules making up the interactive television application may include data about the products which will be offered for sale during this show, or portion of the show, and executable code to interact with the user in the manner described below.

When a viewer wishes to order an item, a button is pressed on the TV remote control. This button signals the client computer 22 to display a series of instructions and menus necessary to solicit the information necessary to place the order, e.g. the item number, name and address of the viewer, the method of payment, the credit card number (if needed), etc. These instructions are generated in the client computer as graphics which are overlaid on the television video image. It is also possible for a computer generated voice to be generated and combined with the television audio either by voice-over, or by replacing the television audio. The viewer responds to the instruction by providing the requested information via the TV remote control. When the information requested by the on-screen display and/or voice instructions has been entered by the viewer, it is sent to a central computer via the modem in the client computer. An order confirmation may be sent in the other direction from the central computer.

It is also possible that permanent information about the viewer (i.e. the name, address, method of payment and credit card number) may be preentered once by the viewer, so it is not necessary to solicit that information each time an order is placed. The information is stored in permanent memory in the client computer. In such a case, when an order is placed, that information is retrieved from the permanent memory, appended to the item number and transmitted to the central computer. It is further possible that, by means of time codes, or other commands, inserted into the data stream, the client computer will know which item is currently being offered for sale. In such a case, the viewer will be able to order it by simply pressing one button on the TV remote control. In response, the client computer can combine the previously received information related to the item currently being offered for sale with the previously stored personal information related to the viewer, and transmit the order to the central computer and receive the confirmation in return.

Because the code and data modules related to the home shopping program are repetitively inserted into the data stream, a viewer may tune into the program at any time and be able to participate interactively. Similarly, it is not necessary for the viewer to participate interactively, but may simply ignore the interactive portion of the show.

It is also possible for the client computer 22 to receive control information from the local computer 40. For example, the user 80, using the local computer 40, could control the client computer 22 via the I/O port to select a desired one of the data streams on transport mechanism 30, and process the program currently being broadcast on that data stream, with interaction with the user 80 through the input and output devices (not shown) connected to the local computer 40.

It is further possible for the user 80 to cause the client computer 22 to access the server computer 10 through the central processing facility 60, instead of via the data stream on transport mechanism 30, and receive code and data modules via this bidirectional link.

FIGURE 2 is a block diagram illustrating a server computer 10 as illustrated in FIGURE 1. In FIGURE 2, a source of distributed computing application code and data 101 includes an application compiler, and software management module (not shown) and has an output terminal coupled to an input terminal of a flow builder 102. An output terminal of flow builder 102 is coupled to an input terminal of a transport packetizer 104. An output terminal of transport packetizer 104 is coupled to a first input terminal of a packet multiplexer 106. An output terminal of packet multiplexer 106 is coupled to an input terminal of a transport multiplexer 110. An output terminal of transport multiplexer 110 is coupled to the physical medium making up the transport mechanism 30 (of FIGURE 1). A second input terminal of packet multiplexer 106 is coupled to a source of auxiliary data packets 107. A clock 109 has respective output terminals coupled to corresponding input terminals of the transport packetizer 104 and auxiliary data source 107. A data transceiver 103 has an first bidirectional terminal coupled to the central processing facility 60 (of FIGURE 1) and a second bidirectional data coupled to the application code and data source 101.

Application code and data source 101, flow builder 102, transport packetizer 104, auxiliary data source 107, clock 109 and packet multiplexer 106, in combination, form a channel source 108 for the transport mechanism, illustrated by a dashed box in . Other channel sources, including similar components as those illustrated in channel source 108 but not shown in FIGURE 1, are represented by another dashed box 108a. The other channel sources (108a) have output terminals coupled to other

input terminals of the transport multiplexer 110, and may have input terminals coupled to central processing facilities through data transceivers.

In operation, data representing the distributed computing application program, and data related to the transmission of the program over the transport mechanism 30 are supplied to the flow builder 102 from the application source 101. This data may be supplied either in the form of files containing data representing the code and data modules, or by scripts providing information on how to construct the code and data modules, or other such information. The code and data modules may be constant or may change dynamically, based on inputs received from the client computers 20 via the central computing facility 60 and/or other sources. The executable code and data module files may be generated by a compiler, interpreter or assembler in a known manner in response to source language programming by an application programmer. The data file related to the transmission of the modules includes such information as: the desired repetition rates for the directory and the code and data modules to be included in the data stream; the size of main memory in the client computers 20 required to store each module, and to completely execute the application program; a priority level for the module, if it is a code module, etc.

Flow builder 102 processes the data from the application source 101. In response, flow builder 102 constructs a directory module, giving an overall picture of the application program. The information in the directory module includes e.g. the identification of all the code and data modules being repetitively transmitted in the data stream, their size and possibly other information related to those modules. Then the application program representative data is processed to generate the code and data modules. The directory, code and data modules thus constructed are formatted by adding module headers and error detection and/or correction codes to each module. A transmission schedule is also generated. After this processing is complete, the data representing the directory module and the code and data modules are repetitively presented to the transport packetizer 104 according to the schedule previously generated.

The transport packetizer 104 generates a stream of packets representing the directory module and the code and data modules as they are emitted from the flow builder 102. Each packet has a constant predetermined length, and is generated by dividing the data stream from the flow builder into groups of bits, and adding a packet header with information identifying the information contained in the packet, and an error detection and/or correction code, etc., to each group, such that each packet is the same predetermined length. (If there is insufficient data from the flow builder 102 to completely fill a packet, the packet is padded with null data.) These packets are time multiplexed with the auxiliary data packets, in a known manner, to form a single packet stream in the packet multiplexer 106. It is also possible for the generated packets to have varying lengths. In this case, the packet header for each packet will contain the length of that packet. In addition, time code data packets are placed in the data stream packets and/or the auxiliary data packets based on data received from the clock 109.

Packet streams from all of the channel sources (108,108a) are multiplexed into a single transport channel, which is transmitted through transport mechanism 30. As described above, the packet streams may be frequency multiplexed by having each packet stream modulate a carrier signal at a different frequency, with all of the carriers being carried by a satellite link to the client computers 20, in a known manner. In addition, if there is sufficient capacity within one carrier channel, several packet streams may be statistically time multiplexed, and used to modulate a single carrier, also in a known manner. For example, it has been proposed to time multiples up to eight interactive television data streams through a single satellite link.

Data from the client computers 20 via the central processing facility 60 (of FIGURE 1) is received at the server computer 10 by the data transceiver 103, which may include its own processor (not shown). If an immediate response is generated, the transceiver 103 processor returns that response via the central processing facility 60 to a specific client computer (22-26), a specific set of the client computers 20 or to all client computers 20 in their turn. If, however, a common response to all client computers 20 is desired, the application programmer may amend the code and data files in the application code and data source 101 using the application compiler. These amended files are then processed by the flow builder again to generate another flow. It is further possible that the code and data files in the application source 101 may be amended automatically and dynamically (i.e. in real time) in response to data received from the transceiver 103, and the flow updated as the data is being received from the client computers 20.

FIGURE 3 is a timing diagram illustrating the data streams produced by the server computer 10 in a distributed computing system as illustrated in FIGURE 1. In FIGURE 3 server computer 10 is shown as simultaneously producing a plurality of packet streams 32-38. Each packet stream (32-38) is shown as a horizontal band divided into packets having the same duration and number of bits. As described above, it is possible that the size of the

packets within any packet stream vary with the amount of data to be carried. In FIGURE 3 it can be seen that the starting times of the packets are not synchronized. It is possible to synchronize the packets, but it in not necessary. In FIGURE 3, packets carrying data representing directories are designated DIR, packets carrying data representing code modules are designated CM, packets carrying data representing data modules are designated DM, and packets carrying auxiliary data are designated AUX.

In the top series of packets 32, the leftmost packet contains data representing a code module, CM. This is followed by three packets containing auxiliary data, AUX, followed by another packet containing data representing the code module, CM. From the series of packets 32 it can be seen that the code module is repetitively produced. There may be more or fewer packets in between successive repetitions of the code module packets CM. The rate of repetition may be specified by the programmer when the application is programmed, and may be varied during the execution of the application.

In the next series of packets 34, the leftmost packet contains auxiliary data, AUX. The next two packets contain respective portions of a code module (CM1,CM2). The last packet contains auxiliary data, AUX. From the series of packets 34 it can be seen that if a code module is too large to be contained in a single packet, it may be carried by more than one, with each packet containing a portion of the code module. Although two packets are illustrated in the series of packets 34 as containing the code module (CM1,CM2), any number of packets may be used to carry the code module, depending upon its size. The two packets carrying the code module, (CM1,CM2) are repetitively transmitted (not shown) in the series of packets 34, as described above.

In the series of packets 36, the leftmost packet contains data representing a code module (CM). The next packet (DM1) is a first packet containing data representing a data module. The next packet contains auxiliary data, AUX. The next packet (DM2) is a second packet containing the remaining data representing the data module. From the series of packets 36 it may be seen that a data module (DM1,DM2), associated with the code module (CM), may also be included in the packet stream. Both the code module (CM) and the data module (DM1,DM2) are repetitively transmitted (not shown) in the series of packets 36. The rate of repetition of the code module (CM) may be different from that of the data module (DM1,DM2), and both rates may be specified by the application programmer and varied during the execution of the application.

It may further be seen that if the data module is too large to be contained in a single packet, it may be carried by more than one packet, with each packet containing a portion of the data module. Although two packets are illustrated in the series of packets 36 as containing the data module (DM1,DM2), any number of packets may be used to carry the data module, depending upon its size. It may be further seen that the packets carrying the data module need not be transmitted sequentially, but may have intervening packets in the packet stream. The same is true for multiple packets carrying a code module or directory module (not shown).

In the bottommost series of packets 38, the leftmost packet contains data representing the directory (DIR). The next packet contains data representing a code module (CM), followed by a packet containing auxiliary data (AUX) and a packet containing data representing a data module (DM). In the series of packet 38 all of a directory module (DIR), a code module (CM) and a data module (DM) in a single packet stream may be seen. The respective repetition rates of these three modules may be different, as specified by the programmer of the application, and may be varied during the execution of the application.

FIGURE 4 is a block diagram of a client computer 22 as illustrated in FIGURE 1. In FIGURE 4, transport mechanism 30 (of FIGURE 1) is coupled to an input terminal of a stream selector 202. An output terminal of stream selector 202 is coupled to respective input terminals of an auxiliary data extractor 204 and a packet data extractor 206. An output terminal of auxiliary data extractor 204 is coupled to the auxiliary data processor 50 (of FIGURE 1). A bidirectional terminal of packet data extractor 206 is coupled to a corresponding terminal of a stream I/O adapter 208. A control output terminal of stream I/O adapter 208 is coupled to a corresponding control input terminal of stream selector 202. The combination of stream selector 202, auxiliary data extractor 204 and packet data extractor 206 form a data stream receiver 207 for client computer 22, illustrated by a dashed line in FIGURE 4.

Stream I/O adapter 208 forms a part of a processing unit 224 in client computer 22, illustrated by a dashed line in FIGURE 4. In addition to the stream I/O adapter 208, processing unit 224 includes a processor 210, read/write memory (RAM) 212 and read-only memory (ROM) 214 coupled together in a known manner via a system bus 216. Further input and output facilities are provided by an I/O port 218, coupled to the local processor 40 (of FIGURE 1); user I/O adapter 220, for communicating with user 80; and modem 222, coupled to the central processing facility 60 (of

FIGURE 1); all also coupled to the system bus 216 in a known manner. Other adapters (not shown) may be coupled to system bus 216 to provide other capabilities to the processing unit 224.

As described above, auxiliary data extractor 204, I/O port 218 and modem 222 are not required in a client computer 20 according to the present invention. They are illustrated in FIGURE 1 and FIGURE 4 to show optional additional functionality.

In operation, processor 210 of processing unit 224 retrieves program instructions permanently stored in ROM 214, or temporarily stored in RAM 212, and executes the retrieved instructions to read data from ROM 212 and/or RAM 214, write data to RAM 212 and/or receive data from or supply data to outside sources via the I/O port 218, user I/O adapter 220 and/or modem 222, in a known manner. Under program control, processor 210 may also request a code and/or data module from the data stream supplied to the client computer 22 via the transport mechanism 30 (of FIGURE 1). To retrieve this data, processor 210 first instructs stream I/O adapter 208 to send a selection control signal to the stream selector 202, possibly in response to user input from user I/O adapter 220. Then processor 210 issues a request for a specific code or data module to the stream I/O adapter 208. Stream I/O adapter 208 relays this request to the packet data extractor 204.

Transport mechanism 30 (of FIGURE 1) supplies all of the plurality of packet streams (32-38 of Figure 3) it carries to the stream selector 202, which passes only the selected packet stream. Auxiliary data extractor 204 monitors the selected packet stream, extracts the auxiliary data packets from it and supplies them directly to the auxiliary data processor 50 (of FIGURE 1). Packet data extractor 206 similarly monitors the selected packet stream, extracts the directory, code and/or data module packets requested by the stream I/O adapter 208 and supplies them to the stream I/O adapter 208. The data in the packets returned to the stream I/O adapter 208 is supplied to the RAM 212. When the entire module has been retrieved from the packet stream (which may require several packets, as described above), processor 210 is notified of its receipt by the stream I/O adapter 208. Processor 210 may then continue execution of its program.

The data stream in a distributed computing system illustrated in FIGURE 1 is similar to a mass storage system in prior art systems. An application program executing on the processor 210 makes a request for a module listed in the directory in the same manner that such a program would make a request for a file containing a code or data module previously stored on a mass storage device in a prior art system. The data stream receiver 207 is similar to a mass storage device, and stream I/O 208 acts in a similar manner to a mass storage adapter on a prior art system by locating the desired data, transferring it to a predetermined location (I/O buffer) in the system memory and informing the processor of the completion of the retrieval. However, the stream I/O adapter 208 can only retrieve code and data from the data stream; data cannot be written to the data stream.

As described above, the distributed computing application may be divided into more than one code module, each containing executable code for a different portion of the distributed computing application. When a particular code module is desired, processor 210 requests that code module from stream I/O adapter 208. When execution of that module has completed, processor 210 requests the next module from stream I/O 208. Because code and data modules are repetitively carried on the data stream, a module may be deleted from RAM 212 when it is not currently needed without the necessity of temporarily being stored, because if it is required later, it may again be retrieved from the data stream when needed. However, if RAM 212 has sufficient capacity, processor 210 may request stream I/O adapter to simultaneously load several code modules into RAM 212. If this can be done, then processor 210 may switch between code modules without waiting for stream I/O adapter 208 to extract them from the data stream.

As described above, other I/O adapters may be coupled to the system bus 216 in a known manner. For example, in an interactive TV system, a graphics adapter may be coupled to system bus 216. The graphics adapter generates signals representing graphical images, in a known manner, in response to instructions from the processor 210. Further, these signals may be combined with the standard video signal produced by the video decoder (described above) in the auxiliary data processor 50 of an interactive TV system. When the graphical image representative signal and the standard video signal are combined, the resulting signal represents an image in which the image generated by the graphics adapter is superimposed on the image represented by the broadcast video signal. It is also possible to selectively combine these two image representative signals under the control of the processor 210.

An interactive TV system, may also include a sound adapter coupled to the system bus 216. The sound adapter generates a signal representing a computer generated sound (such as music, synthesized voice or other sound), in a known manner, in response to instructions from the processor 210. Further, these signals may be combined with the standard audio signal produced by the audio de-

coder (described above) in the auxiliary data processor 50 of an interactive TV system. When the sound representative signal and the standard audio signal are combined, the resulting signal represents the combination of the sound generated by the sound adapter and the broadcast audio signal. It is also possible to selectively combine these two sound representative signals under the control of the processor 210.

The timing of the generation and display of the graphical image and sound representative signals, may be controlled by receipt of the time code data from the data stream. This enables an executable code module to synchronize the display of processor generated image and presentation of processor generated sound to the broadcast video and audio. It is further possible to synchronize the operation of the interactive TV application by the insertion of specialized packets into the data stream which cause an interrupt of the code currently executing in processor 210. Stream I/O 208 monitors the data stream for such specialized packets, and generates an interrupt, in a known manner, for the processor 210. Processor 210 responds to that interrupt, also in known manner, by executing an interrupt service routine (ISR). This ISR may be used for synchronization of the interactive TV application, or other purposes.

A client computer 22 in a distributed computing system as illustrated in FIGURE 1 does not need a mass storage device, nor a large amount of RAM 212. Such a system decreases the cost of a client computer, and increases the functionality of the lower cost client computers. In addition, such a client computer has the option of participating in a distributed computing function, may join in the distributed computing function at any time (or may drop out and return later), and may participate at its own pace.

Claims

1. A distributed computer system characterized by:

a source (10) of a continuous data stream repetitively including data representing a distributed computing application; and

a client computer (20), receiving (207) the data stream, extracting (206) the distributed computing application representative data from the data stream, and executing (224) the extracted distributed computing application.

2. The computer system of claim 1, further characterized by an auxiliary data processor; wherein:

the data stream source produces the data stream further including auxiliary data; and

the client computer extracts the auxiliary data from the data stream and supplies it to the auxiliary data processor.

3. The computer system of claim 2, characterized in that:

the data stream source produces the data stream in the form of a series of packets;

a first one of the series of packets contains data representing the distributed computing application and includes identification information indicating that the first one of the series of packets contains data representing the distributed computing application; and

a second one of the series of packets contains auxiliary data and includes identification information indicating that the second one of the series of packets contains auxiliary data.

4. The computer system of claim 1, characterized in that:

the data stream source simultaneously produces a plurality of continuous data streams, each repetitively including data representing a respective distributed computing application; and

the client computer further includes a data receiver for selectively receiving one of the plurality of data streams, and extracting the distributed computing application representative data included in the selected one of the data streams.

5. The computer system of claim 4, further characterized by an auxiliary data processor; wherein:

the data stream source produces the data stream further including auxiliary data; and

the client computer extracts the auxiliary data from the data stream and supplies it to the auxiliary data processor.

6. The computer system of claim 4, characterized in that:

the data stream source produces the data stream in the form of a series of packets;

a first one of the series of packets contains data representing the executable code module and includes identification information indicating that the first one of the series of packets contains data representing the executable code module;

a second one of the series of packets contains data representing the data module and includes identification information indicating that the second one of the series of packets contains data representing the data module; and

a third one of the series of packets contains auxiliary data and includes identification information indicating that the third one of the series of packets contains auxiliary data.

7. The computer system of claim 6, characterized in that:

the data stream source produces the data stream further including a directory module containing information related to the code module; and

the client computer first extracts the directory module from the data stream, then extracts the code module in response to the information related to the code module in the extracted directory module, and executes the extracted code module.

8. The computer system of claim 1, characterized in that:

the data stream source produces the data stream in the form of a series of packets;

a first one of the series of packets contains data representing the executable code module and includes identification information indicating that the first one of the series of packets contains data representing the executable code module;

a second one of the series of packets contains data representing the data module and includes identification information indicating that the second one of the series of packets contains data representing the data module;

a third one of the series of packets contains data representing the directory module and includes identification information indicating that the second one of the series of packets contains data representing the directory module; and

a fourth one of the series of packets contains auxiliary data and includes identification information indicating that the third one of the series of packets contains auxiliary data.

9. The computer system of claim 8, characterized in that:

the data stream source produces the data stream further including a data module and a directory module further contains information related to the data module; and

the client computer further extracts the data module from the data stream in response to the information related to the data module in the extracted directory module and executes the extracted code module to process the extracted data module.

10. In a distributed computer system, a client computer (22), characterized by:

an input terminal (30), for receiving a continuous data stream repetitively including data representing a distributed computing application

a data stream receiver (207), coupled to the input terminal, for receiving the data stream and extracting (206) the distributed computing application representative data; and

a processing unit (224), coupled to the data stream receiver, for receiving and executing (210) the distributed computing application.

11. The client computer of claim 10, characterized in that the processing unit comprises:

a system bus;

read/write memory, coupled to the system bus;

a data stream input/output adapter, coupled between the data stream receiver and the system bus, for receiving the extracted distributed computing application representative data from the data stream receiver, and storing it in the read/write memory; and

a processor, coupled to the system bus for executing the distributed computing application stored in the read/write memory.

12. The client computer of claim 10, characterized in that:

the input terminal receives the data stream as a series of packets containing packets carrying the distributed computing application representative data; and

the data stream receiver comprises a packet data extractor, coupled to the input terminal, for extracting the packets carrying the distributed computing application representative data.

13. The client computer of claim 12, characterized in that:

the series of packets in the data stream further include packets carrying auxiliary data;

the client computer further includes an auxiliary data processor; and

the data stream receiver comprises an auxiliary data packet extractor, coupled to the auxiliary data processor, for extracting the packets carrying the auxiliary data from the data stream and supplying them to the auxiliary data processor.

14. The client computer of claim 13, characterized in that the distributed computing system is an interactive television system, and the auxiliary data is television video and audio.

15. The client computer of claim 10, characterized in that:

the input terminal receives a plurality of data streams, each including data representing a respective distributed computing application; and

the data stream receiver comprises:

a data stream selector, coupled to the input terminal, for producing a selected one of the plurality of data streams in response to control signals from the processing unit; and

a distributed computing representative data extractor, coupled between the data stream selector and the processing unit, for extracting the distributed computing application representative data from the selected one of the plurality of data streams.

16. The client computer of claim 15, characterized in that:

the data stream selector comprises a selection control input terminal, and produces the selected one of the plurality of data streams in response to a control signal at the selection control input terminal;

the processing unit comprises:

a system bus;

read/write memory, coupled to the system bus;

a data stream input/output adapter, coupled between the data stream receiver and the system bus, for receiving the extracted distributed computing application representative data from the data stream receiver, and storing it in the read/write memory, and having a control output terminal coupled to the selection control input terminal of the data stream selector, for producing the selection control signal; and

a processor, coupled to the system bus, for controlling the data stream input/output device to generate a selection control signal selecting a specified one of the plurality of data streams, and for executing the distributed computing application stored in the read/write memory.

17. The client computer of claim 10, characterized in that:

the input terminal receives the distributed computing application representative data including an executable code module;

the data stream receiver extracts the executable code module; and

the processing unit executes the extracted code module.

18. The client computer of claim 17, characterized in that:

the input terminal receives the distributed computing application representative data further includes a directory module containing information related to the executable code module; and

the data stream receiver first extracts the directory module from the data stream;

the processing unit then processes the information related to the executable code module in the directory module;

the data stream receiver then extracts the executable code module from the data stream based on the information related to the executable code module in the extracted directory module; and

the processing unit then executes the extracted executable code module.

19. The client computer of claim 18, characterized in that:

the distributed computing application representative data further includes a data module and the directory module further contains information related to the data module;

the processing unit further processes the information related to the data module in the directory module;

the data stream receiver further extracts the data module from the data stream based on the information related to the data module in the extracted directory module; and

the processing unit executes the extracted code module to process the extracted data.

20. The client computer of claim 10 characterized in that the distributed computing application is divided into a plurality of modules, representing portions of the application, and the processing unit stores only modules of said plurality of modules, necessary to execute the current portion of the application.

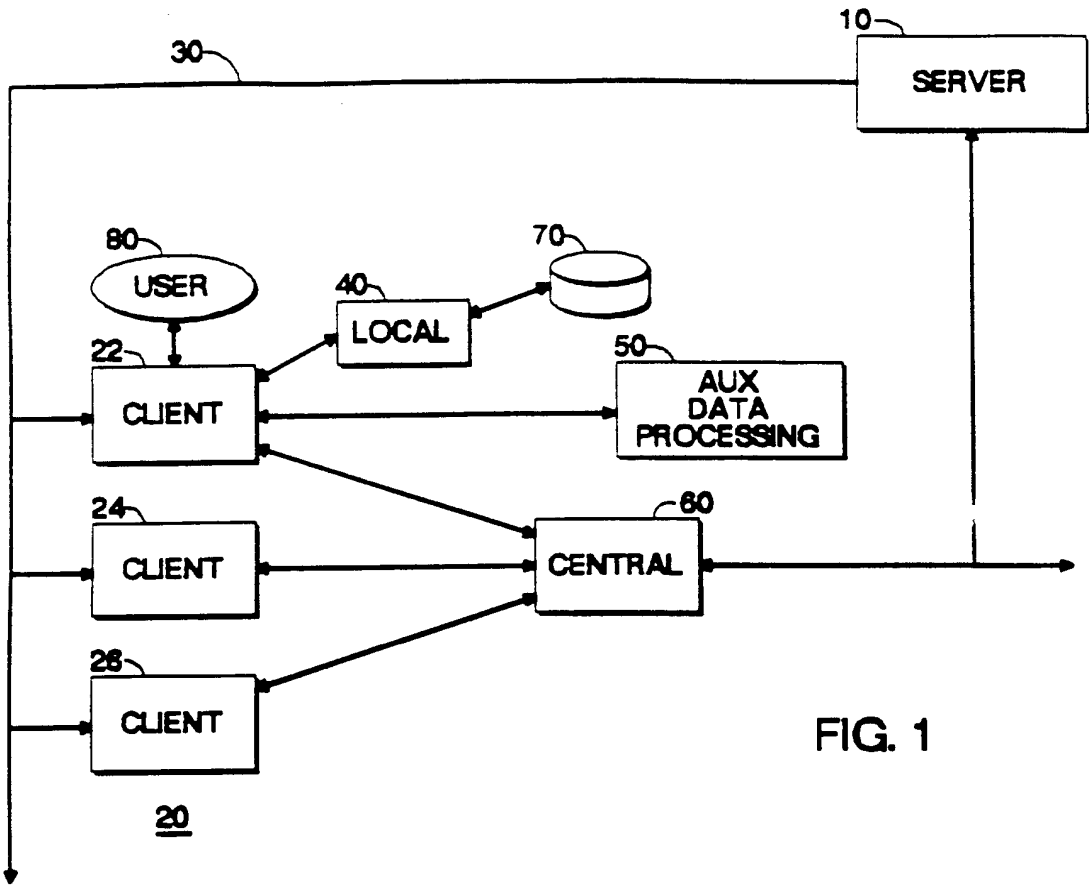


FIG. 1

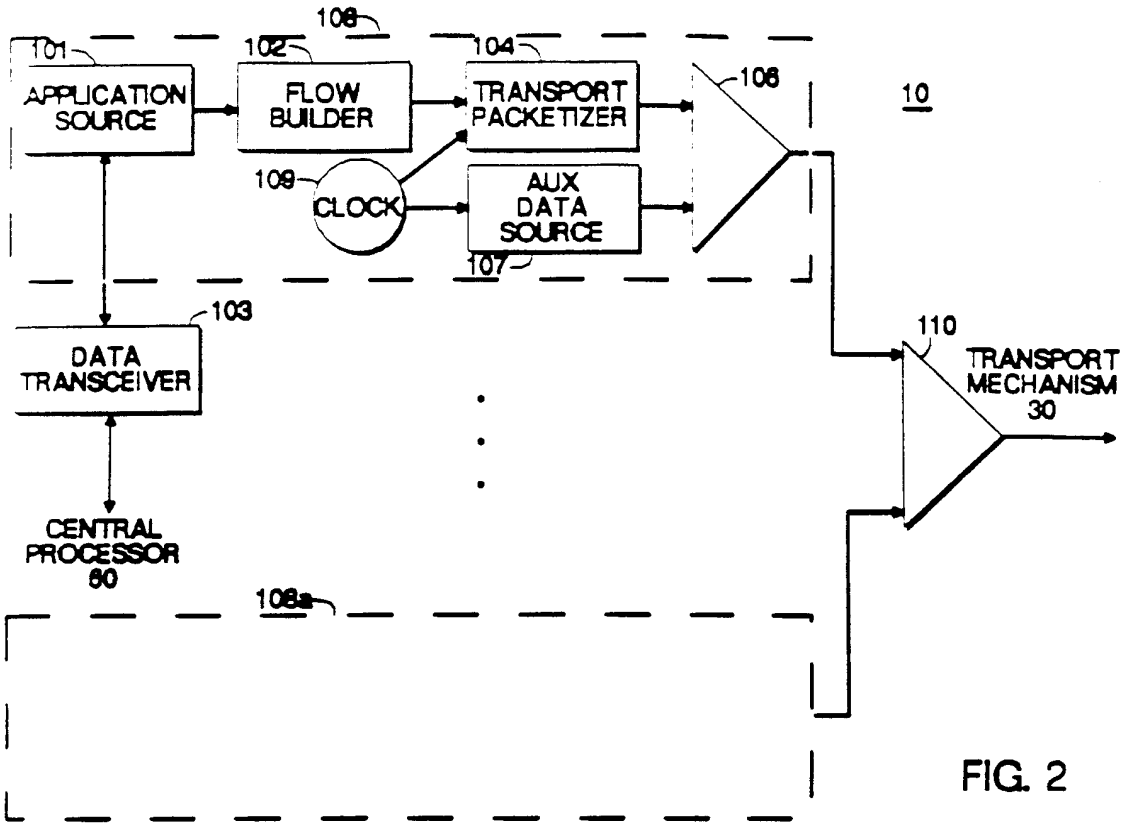


FIG. 2

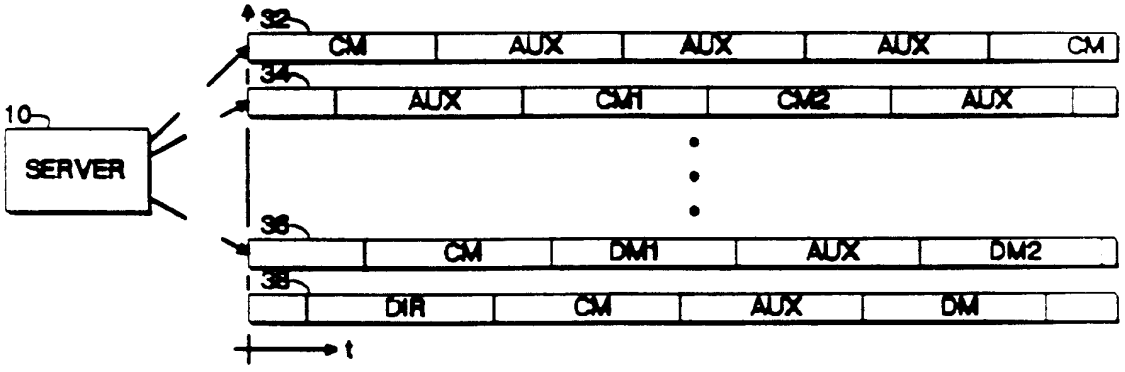


FIG. 3

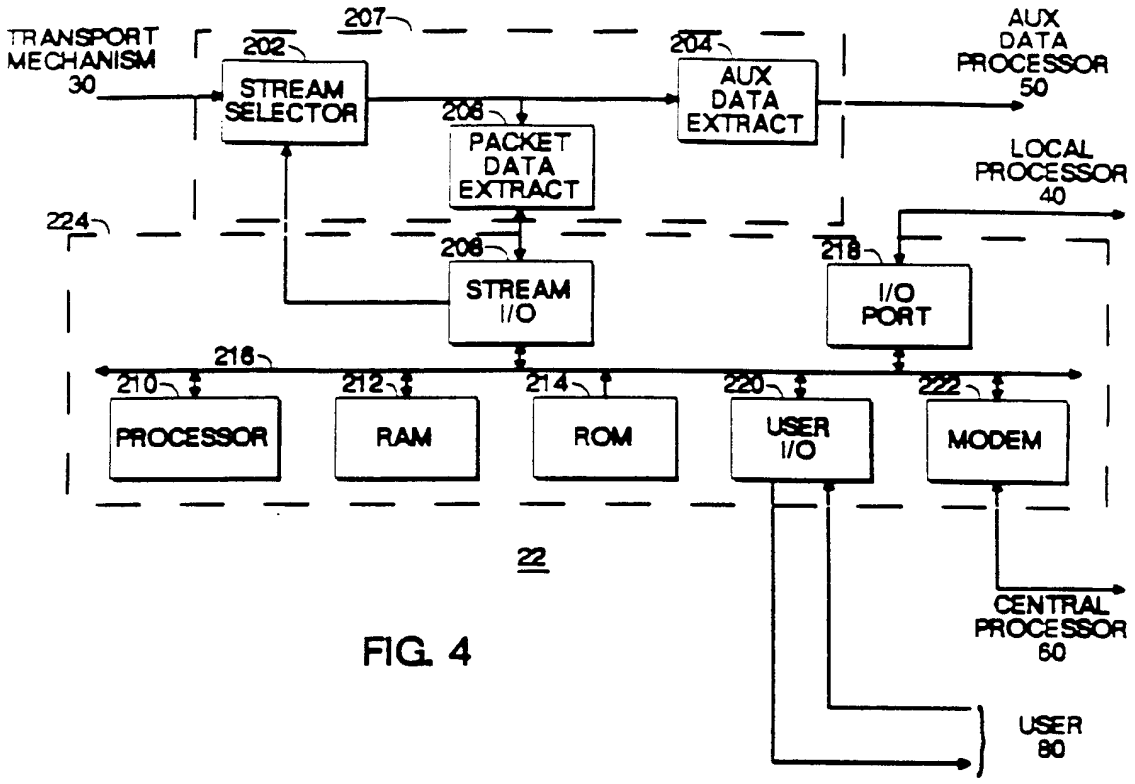


FIG. 4

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 680 185 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
02.05.2002 Bulletin 2002/18

(43) Date of publication A2:
02.11.1995 Bulletin 1995/44

(21) Application number: **95105803.1**

(22) Date of filing: **19.04.1995**

(51) Int Cl.[7]: **H04L 29/06**, H04N 7/24, G06F 9/445

(84) Designated Contracting States:
DE ES FR GB IT

(30) Priority: **28.04.1994 US 233908**

(71) Applicant: **OpenTV, Inc.**
Mountain View, CA 94043 (US)

(72) Inventors:
- **Joseph, Kuriacose**
  Plainsboro, New Jersey 08543 (US)
- **Dureau, Vincent**
  Venice, California 90291 (US)
- **Jessup, Ansley Wayne, Jr.**
  Willingboro, New Jersey 08046 (US)
- **Delpuch, Alain**
  Los Angeles, California 90064 (US)

(74) Representative: **Freeman, Jacqueline Carol et al**
W.P. THOMPSON & CO.
Celcon House
289-293 High Holborn
London WC1V 7HU (GB)

(54) A distributed computer system

(57) A distributed computer system is disclosed which comprises a source (30) of a continuous data stream repetitively including data representing a distributed computing application and a client computer (22), receiving the data stream, for extracting (207) the distributed computing application representative data from the data stream, and executing (224) the extracted distributed computing application.

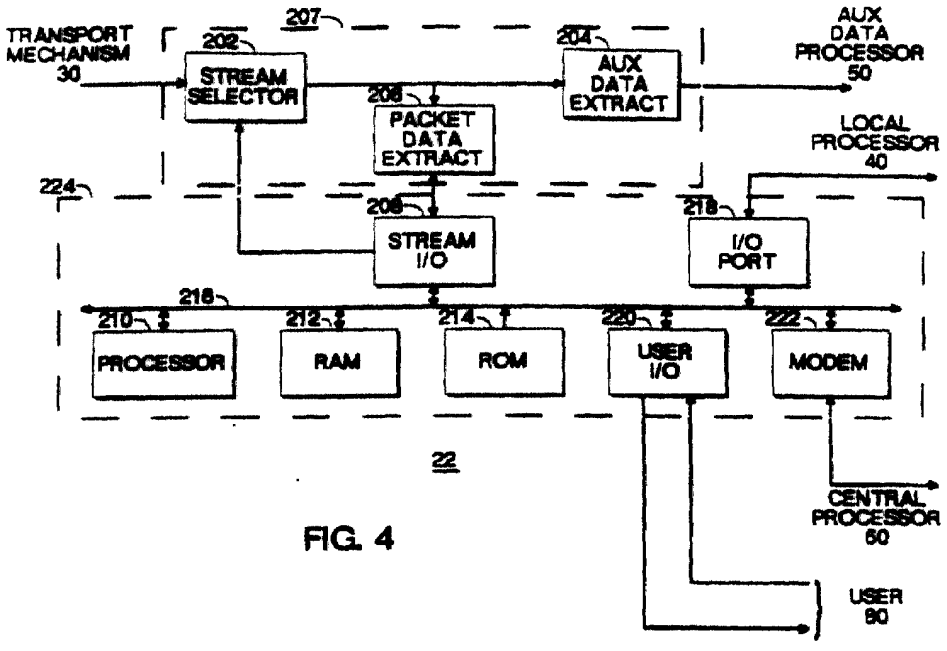


FIG. 4

EP 0 680 185 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 10 5803

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	US 5 251 301 A (COOK GARY M) 5 October 1993 (1993-10-05) * abstract * * column 1, line 62 - column 2, line 54 * * column 3, line 5 - line 48 *	1-6, 10-17,20	H04L29/06 H04N7/24 G06F9/445
A	* column 4, line 57 - column 8, line 11; figure 3 * * claims 1,6-10 *	7-9,18, 19	
X	EP 0 306 208 A (OLIVETTI & CO SPA ;RAI RADIOTELEVISIONE ITALIANA (IT)) 8 March 1989 (1989-03-08) * abstract * * page 2, column 1, line 53 - column 2, line 26 * * page 4, column 5, line 45 - column 6, line 45 * * page 5, column 8, line 30 - page 6, column 9, line 28 * * page 7, column 11, line 3 - column 12, line 4 * * page 9, column 15, line 5 - line 16 * * page 11, column 19, line 55 - column 12, line 22; figure 51 * * figure 3 *	1,2,4,5, 10-13, 15-20	
X	US 5 299 197 A (SCHLAFLY ROGER) 29 March 1994 (1994-03-29) * column 2, line 34 - line 57 * * column 3, line 34 - line 42 * * column 4, line 11 - line 47 *	1-6, 10-13, 15-17,20	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

H04L
H04N
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	6 March 2002	Karavassilis, N

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 95 10 5803

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

06-03-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5251301 A	05-10-1993	US 4920503 A	24-04-1990
EP 0306208 A	08-03-1989	IT 1211278 B	12-10-1989
		EP 0306208 A2	08-03-1989
US 5299197 A	29-03-1994	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 720 374 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.07.1996 Bulletin 1996/27

(51) Int. Cl.[6]: **H04N 7/26**, H04N 7/50

(21) Application number: **94120967.8**

(22) Date of filing: **30.12.1994**

(84) Designated Contracting States:
DE FR GB NL

(71) Applicant: **DAEWOO ELECTRONICS CO., LTD**
Jung-Gu, Seoul 100-095 (KR)

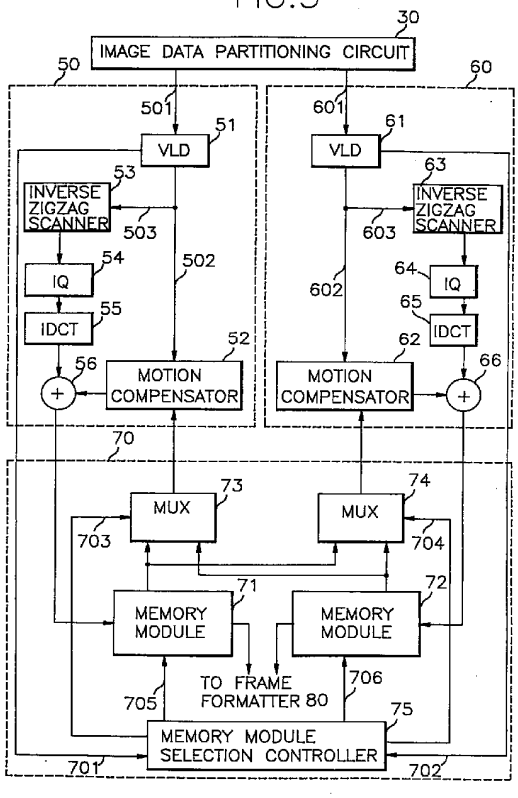
(72) Inventor: **Kwon, Oh-Sang**
Seoul (KR)

(74) Representative: **von Samson-Himmelstjerna, Friedrich R., Dipl.-Phys. et al**
SAMSON & PARTNER
Widenmayerstrasse 5
D-80538 München (DE)

(54) Apparatus for parallel decoding of digital video signals

(57) A novel apparatus for decoding an encoded digital video signal is able to carry out a parallel processing, without using a shared memory system. This apparatus comprises detector for detecting slice start codes from the encoded digital video signal and generating a slice start detection signal, control unit for counting the number of the slice start codes of the encoded bit stream, and for generating a control signal in response to the counted number of the slice start codes, switching block for dividing video frame data into two subframes, two first-in first-out(FIFO) buffers for storing the divided video frame data, a image processing device for decompressing the encoded digital video signal and reproducing the original video image signal, and frame formatter for coupling the reproduced original video image signal.

FIG.3



EP 0 720 374 A1

Description

Field of the Invention

The present invention relates to a video imaging system; and, more particularly, to an improved video image decoding apparatus having two decoding modules for decompressing incoming compressed video image data in parallel.

Description of the Prior Art

In various electronic/electrical applications such as high definition television and video telephone systems, an image signal may need be transmitted in a digitized form. When the image signal is expressed in a digitized form, there is bound to occur a substantial amount of digital data. Since, however, the available frequency bandwidth of a conventional transmission channel is limited, in order to transmit the image signal therethrough, the use of an image signal encoding apparatus becomes necessary to compress the substantial amounts of digital data.

Accordingly, most image signal encoding apparatus employ various compression techniques (or coding methods) built on the idea of utilizing or reducing spatial and/or temporal redundancies inherent in the input image signal.

Among the various video compression techniques, the so-called hybrid coding technique, which combines temporal and spatial compression techniques together with a statistical coding technique, is known to be most effective.

Most hybrid coding techniques employ a motion compensated DPCM(differential pulse code modulation), two-dimensional DCT(discrete cosine transform), quantization of DCT coefficients, and VLC(variable length coding). The motion compensated DPCM is a process of determining the movement of an object between a current frame and its previous frame, and predicting the current frame according to the motion flow of the object to produce a differential signal representing the difference between the current frame and its prediction. This method is described, for example, in Staffan Ericsson, "Fixed and Adaptive Predictors for Hybrid Predictive/Transform Coding", IEEE Transactions on Communications, COM-33, No. 12(December 1985); and in Ninomiya and Ohtsuka, "A Motion Compensated Interframe Coding Scheme for Television Pictures", IEEE Transactions on Communications, COM-30, No. 1(January 1982).

The two-dimensional DCT, which reduces or removes spatial redundancies between image data such as motion compensated DPCM data, converts a block of digital image data, for example, a block of 8x8 pixels, into a set of transform coefficient data. This technique is described in Chen and Pratt, "Scene Adaptive Coder", IEEE Transactions on Communications, COM-32, No. 3(March 1984). By processing such transform coefficient data with a quantizer, zigzag scanner and VLC circuit, the amount of data to be transmitted can be effectively compressed.

Specifically, in the motion compensated DPCM, current frame data is predicted from previous frame data based on an estimation of the motion between the current and the previous frames. Such estimated motion may be described in terms of two dimensional motion vectors representing the displacement of pixels between the previous and the current frames.

In order to compress the image signals with the above mentioned technique, the use of a processor capable of carrying out a high speed processing becomes necessary, and this is usually achieved by employing a parallel processing technique. Generally, in the image signal decoding apparatus having the parallel processing capability, one video image frame area is divided into a plurality of subframes, and image data within the video image frame area is processed on a subframe-by-subframe basis.

On the other hand, to determine a motion vector for a search block in the current frame, a similarity calculation is performed between the search block of the current frame and each of a plurality of equal-sized candidate blocks included in a generally larger search region within a previous frame, wherein the size of the search block typically ranges between 8x8 and 32x32 pixels. Consequently, the search region containing a boundary portion of any subframe also includes a boundary portion of a neighboring subframe. Accordingly, the motion estimation carried out by each processor requires a shared memory system with a multiple random access capability.

Summary of the Invention

It is, therefore, a primary object of the present invention to provide an improved video image decoding apparatus capable of carrying out a parallel processing, without having to use a shared memory system with a multiple random access capability.

In accordance with the present invention, there is provided an apparatus for decoding an encoded digital video signal in an encoded bit stream for the reproduction of an original video image signal, wherein the encoded digital video signal includes a plurality of video frame data, each video frame data having a number of slice start codes representing a start of each slice therein, said apparatus comprising: means for detecting the slice start codes from the encoded digital video signal and generating a slice start detection signal; control means for counting the number of the slice start codes of the encoded bit stream in response to the slice start detection signal, and for generating a control signal in response to the counted number of the slice start codes; means, responsive to the control signal, for dividing video frame data into two subframes; two first-in first-out(FIFO) buffers for storing the divided video frame data; image processing means for decompressing the

encoded input data and reproducing the original video image signal; and means for coupling the reproduced original video image signal. The image processing means includes two decoder modules and a frame memory part for reproducing the original video image signal, wherein each decoder module reproduces each of the two subframes, and the memory part includes two memory modules for storing the divided video frame data, a memory module selection controller for generating first and second selection signals, and first and second address data, and selecting means for producing pixel data stored in the corresponding memory modules in response to the first and second selection signals.

Brief Description of the Drawings

The above and other objects and features of the present invention will become apparent from the following description of preferred embodiments given in conjunction with the accompanying drawings, in which:

Fig. 1 is a schematic representation of a video image frame area divided into two subframes;
Fig. 2 represents a block diagram of the inventive decoding apparatus comprising an image data partitioning circuit and an image processing device;
Fig. 3 presents a more detailed block diagram of the image processing device coupled to the image data partitioning circuit shown in Fig. 2; and
Fig. 4A and 4B describe a timing diagram representing the order of processing for each subframe.

Detailed Description of the Preferred Embodiments

The present invention provides for the communication of high definition television(HDTV) signals from a transmitter location to a population of receivers. At the transmitter of an encoder" end of the communication link, digital video signals for successive frames of a television picture are broken down into subframes for processing by multiple processors. The inventive decoding apparatus includes two decoder modules, each of which is dedicated to the decoding of video data from a particular subframe.

Referring to Fig. 1, there is shown a video image frame area 10 divided into two subframes. The total frame area encompasses M horizontal picture lines, each picture line containing N pixels. For example, a single HDTV frame comprises 960 picture lines, each picture line including 1408 pixels. In other words, a single HDTV frame comprises 60 slices, each slice including 16 horizontal picture lines.

In accordance with the present invention, a video image frame area is divided into two subframes, e.g., subframes 13, 16 as illustrated in Fig. 1.

In order to process the two subframes, a processor is assigned to each subframe for decompressing compressed digital data bounded by the subframe in a video frame. In an encoding apparatus, data redundancies between a current video frame and one or more of its prior video frames is reduced using a motion estimation/compensation technique.

Referring to Fig. 2, there is illustrated a block diagram of the inventive parallel image decoding apparatus, which comprises an image data partitioning circuit 30 and an image processing device 40.

The image data partitioning circuit 30 which includes a slice start code(SSC) detector 31, a control unit 32, a switching block 33, and two first-in first-out(FIFO) buffers 34, 35 is coupled to the image processing device 40; and serves to divide the encoded digital data into two subframes for the processing thereof on a subframe-by-subframe basis. The image processing device 40 includes two decoder modules 50, 60, each of the decoder modules having variable length decoding(VLD) circuits 51, 61, motion compensators 52, 62, inverse zigzag scanners 53, 63, inverse quantizers(IQs) 54, 64, inverse discrete cosine transform(IDCT) circuits 55, 65 and adders 56, 66 decompresses compressed input digital data in connection with a frame memory part 70, respectively.

As shown in Fig. 2, a variable length encoded digital video signal received from an encoding apparatus(not shown) is inputted via terminal 20 to the SSCs detector 31. The encoded digital video signal includes a plurality of video frame data, each of said video frame data occupying a video image frame area has variable length coded transform coefficients, motion vectors and a number of SSCs, wherein each SSC represents a start of a slice included in the encoded bit stream. The SSC detector 31 detects slice start codes from the encoded digital video signal and generates a slice start detection signal to the control unit 32 which serves to control the switching block 33. The control unit 32 counts the number of SSCs in response to the slice start detection signal provided from the SSC detector 31. Whenever the counted number of the SSCs reaches a predetermined value, e.g., 30, a control signal for alternately switching the encoded digital video signal supplied from the SSC detector 31 between S1 and S2 is generated by the control unit 32, thereby dividing each frame of the incoming encoded image signal into two subframes and storing them in two FIFO buffers 34, 35. The FIFO buffers output the subframe data to corresponding decoder modules 50, 60, incorporated in the image processing device 40, each of said decoder modules is dedicated to the processing of video image data bounded by a particular subframe and substantially identical each other. The image processing device 40 reconstructs a discrete cosine transform(DCT) coefficients, performs a motion compensation based on a motion vector, and constitutes representative image data of a given block in the current frame. The decoded subframe data from the image processing device 40 is sent to a frame formatter 80 and combined therein to form a single data stream representing the original video image signal to be, e.g., displayed on a display unit(not shown).

Referring now to Fig. 3, there is shown a more detailed block diagram of the image processing device

40 coupled to the image data partitioning circuit 30 shown in Fig. 2. The decoder modules 50 and 60 contained in the image processing device 40 are made of identical elements, each element serving a same function.

As shown in Fig. 3, video image data bounded by a particular subframe is provided from the image data partitioning circuit 30 to variable length decoding(VLD) circuits 51, 61 through lines 501, 601, respectively. Each VLD circuit processes the video image data bounded by a corresponding subframe. That is, each VLD circuit decodes the variable length coded transform coefficients and the motion vectors to send the transform coefficient data to respective inverse zigzag scanners 53, 63 and the motion vector data to each of the motion compensators 52, 62, incorporated in the decoder modules. The VLD circuits are basically a look-up table: that is, in the VLD circuits, a plurality of code sets is provided to define respective relationships between variable length codes and their run-length codes or motion vectors. The output from each VLD circuit is then distributed to a corresponding processor. Each processor processes video image data bounded by a corresponding subframe.

Video image data bounded by the first subframe 13 shown in Fig. 1 is provided from the VLD circuit 51 to the inverse zigzag scanner 53 through a line 503. In the inverse zigzag scanner 53, the quantized DCT coefficients are reconstructed to provide an original block of quantized DCT coefficients. A block of quantized DCT coefficients is converted into DCT coefficients in the inverse quantizer(IQ) 54 and fed to the inverse discrete cosine transform(IDCT) circuit 55 which transforms the DCT coefficients into difference data between a block of the current subframe and its corresponding block of the previous subframe. The difference data from the IDCT circuit 55 is then sent to the adder 56.

In the meanwhile, the variable length decoded motion vector from the VLD circuit 51 is fed to the motion compensator 52 and a memory module selection controller 75 within the frame memory part 70 via lines 502 and 701. The motion compensator 52 extracts corresponding pixel data from the previous subframe stored in the frame memory part 70 based on the motion vector and sends the corresponding pixel data to the adder 56. The corresponding pixel data derived from the motion compensator 52 and the pixel difference data from the IDCT circuit 55 are summed up at the adder 56 to constitute representative image data of a given block of the current subframe and written onto the first memory module 71 and transmitted to the frame formatter 80 as shown in Fig. 2.

Also, the decoder module 60 is similar to the decoder module 50 in structure and operation. In other words, video image data bounded by the second subframe 16 shown in Fig. 1 is provided from the VLD circuit 61 to the inverse zigzag scanner 63 via a line 603, and the quantized DCT coefficients are reconstructed therein. The quantized DCT coefficients are converted into DCT coefficients in the IQ 64 and fed to the IDCT circuit 65, thereby transforming the DCT coefficients into difference data between a block of the current subframe and its corresponding block of the previous subframe. The difference data from the IDCT circuit 65 is then sent to the adder 66.

In the meanwhile, the motion vector from the VLD circuit 61 is fed to the motion compensator 62 and the memory module selection controller 75 via lines 602 and 702. The motion compensator 62 extracts corresponding pixel data from the previous subframe stored in the frame memory part 70 based on the motion vector and provides the corresponding pixel data to the adder 66. The corresponding pixel data derived from the motion compensator 62 and the pixel difference data from the IDCT circuit 65 are summed up at the adder 66 to constitute representative image data of a given block in the current subframe and written onto the second memory module 72 and transmitted to the frame formatter 80 as shown in Fig. 2.

In accordance with the present invention, one video image frame area is divided into two subframes, each of the subframe data being processed through the use of a corresponding decoder module. In this case, when the boundary portion between the two subframes, e.g., slice 30 or slice 31 shown in Fig. 1, is processed, the motion compensator 52 or 62 may access one of the memory modules 71, 72. That is, if the first motion vector provided from the VLD circuit 51 is found in the subframe 16 during the processing of the slice 30 within the subframe 13, the motion compensator 52 should access the memory module 72. Similarly, if the second motion vector applied from the VLD circuit 61 is in the subframe 13 during the processing of the slice 31, the motion compensator 62 should access the memory module 71. At this time, the motion compensation process performed by each of the two decoder modules is controlled to prevent the two motion compensators from concurrently attempting to access a same memory module. In other words, the two memory modules are made to have an appropriate deadlock so that the two motion compensators do not access a same memory module, simultaneously. A more detailed description of the above mentioned operation will be provided with reference to Fig. 4.

As shown in Fig. 3, for this mutually exclusive memory module access, the frame memory part 70 includes two memory modules 71, 72, the two multiplexer circuits 73, 74 and the memory module selection controller 75. At the memory module selection controller 75, it is checked whether the motion vectors are in an adjacent subframe.

The memory module selection controller 75 receives first and second motion vectors from the VLD circuits 51, 61 via lines 701, 702 and generates first and second selection signals through lines 703, 704 to the multiplexer circuits 73, 74. Also, the memory module selection controller 75 simultaneously produces first and second address data via lines 705, 706 to the memory modules 71, 72.

When each of the motion vectors provided from the VLD circuits to the memory module selection controller 75 is in each of the corresponding subframes, the memory module selection controller 75 generates the first and second selection signals, e.g., logic "low", to multiplexer circuits 73, 74. Each multiplexer circuit outputs corresponding pixel data from the previous subframe stored in the corresponding memory module based on the motion vectors, in response to the first and second selection signals with a said logic "low". That is, when the first selection signal is a logic "low", the multiplexer circuit 73 furnishes the pixel data supplied from the memory module 71 to the motion compensator 52. Similarly, when the second selector signal is a logic "low", the multiplexer circuit 74 offers the pixel data supplied from the memory module 72 to the motion compensator 62.

When each of the motion vectors is in another adjacent subframe, the first and second selection signals produced by the memory module selection controller 75 are logic "high". In this case, the multiplexer circuits 73 and 74 output the pixel data from the memory modules 72 and 71, respectively. As noted above, the mutually exclusive memory accessing operation between the two memory modules 71, 72 is performed under the control of the memory module selection controller 75.

Referring now to Figs. 4A and 4B, there is shown a timing diagram representing the order of processing for each subframe.

As indicated in Fig. 4, the decoder module 50 starts the processing of the video image data occupying the subframe 13. After processing all of the slices contained in the subframe 13, the processing of the subframe 16 is commenced by the decoder module 60. At this time, the decoder module 50 has a deadlock until the decoder module 60 completes the processing of the slice 31 in the subframe 16 in order to prevent the two motion compensators 52, 62 shown in Fig. 3 from accessing a same memory module. When the slice 31 is processed by the decoder module 60, the decoder module 50 begins the processing of next subframe data, e.g., slice 1', in a next video image frame area. The decoder module 60 has a deadlock until the decoder module 50 completes the processing of the slice 30' in said next video image frame area. In this manner, each of the decoder modules 50, 60 repeats the decoding operation until all of the incoming video image data is processed.

While the present invention has been described with respect to certain preferred embodiments only, other modifications and variations may be made without departing from the spirit and scope of the present invention as set forth in the following claims.

Claims

1. An apparatus for decoding an encoded digital video signal in an encoded bit stream for the reproduction of an original video image signal, wherein the encoded digital video signal includes a plurality of video frame data, each video frame data having a number of slice start codes representing a start of each slice therein, which comprises:

means for detecting the slice start codes from the encoded digital video signal and generating a slice start detection signal;

control means for counting the number of the slice start codes in the encoded bit stream in response to the slice start detection signal, and for generating a control signal in response to the counted number of the slice start codes;

means, responsive to the control signal, for dividing video frame data into two subframes;

two first-in first-out(FIFO) buffers for storing the divided video frame data;

image processing means for decompressing the encoded digital video signal and reproducing the original video image signal; and

means for coupling the reproduced original video image signal.

2. The apparatus of claim 1, wherein the image processing means includes two decoder modules and a frame memory part for reproducing the original video image signal, wherein each of the decoder modules generates the decompressed digital video signal, and the frame memory part includes two memory modules for storing the decompressed digital video signal, a memory module selection controller for generating first and second selection signals and first and second address data, and selecting means for producing pixel data stored in the corresponding memory modules in response to the first and second selection signals.

FIG. 1

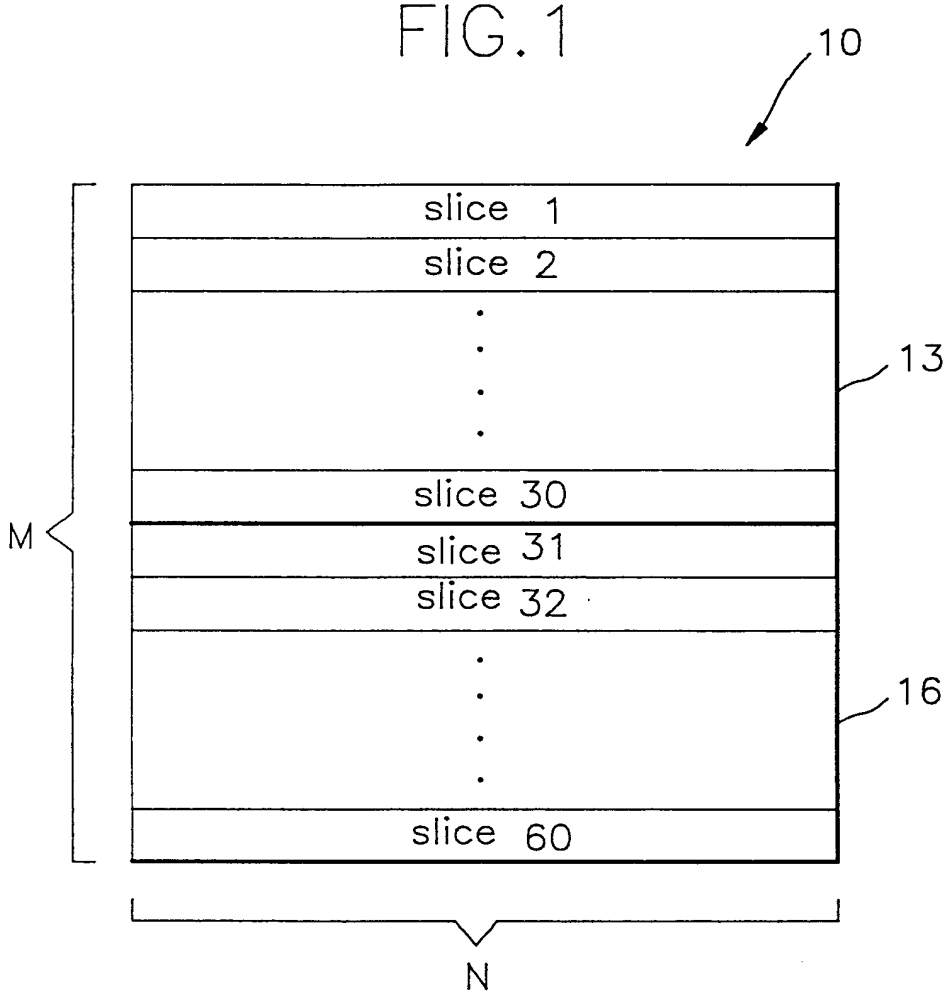


FIG.2

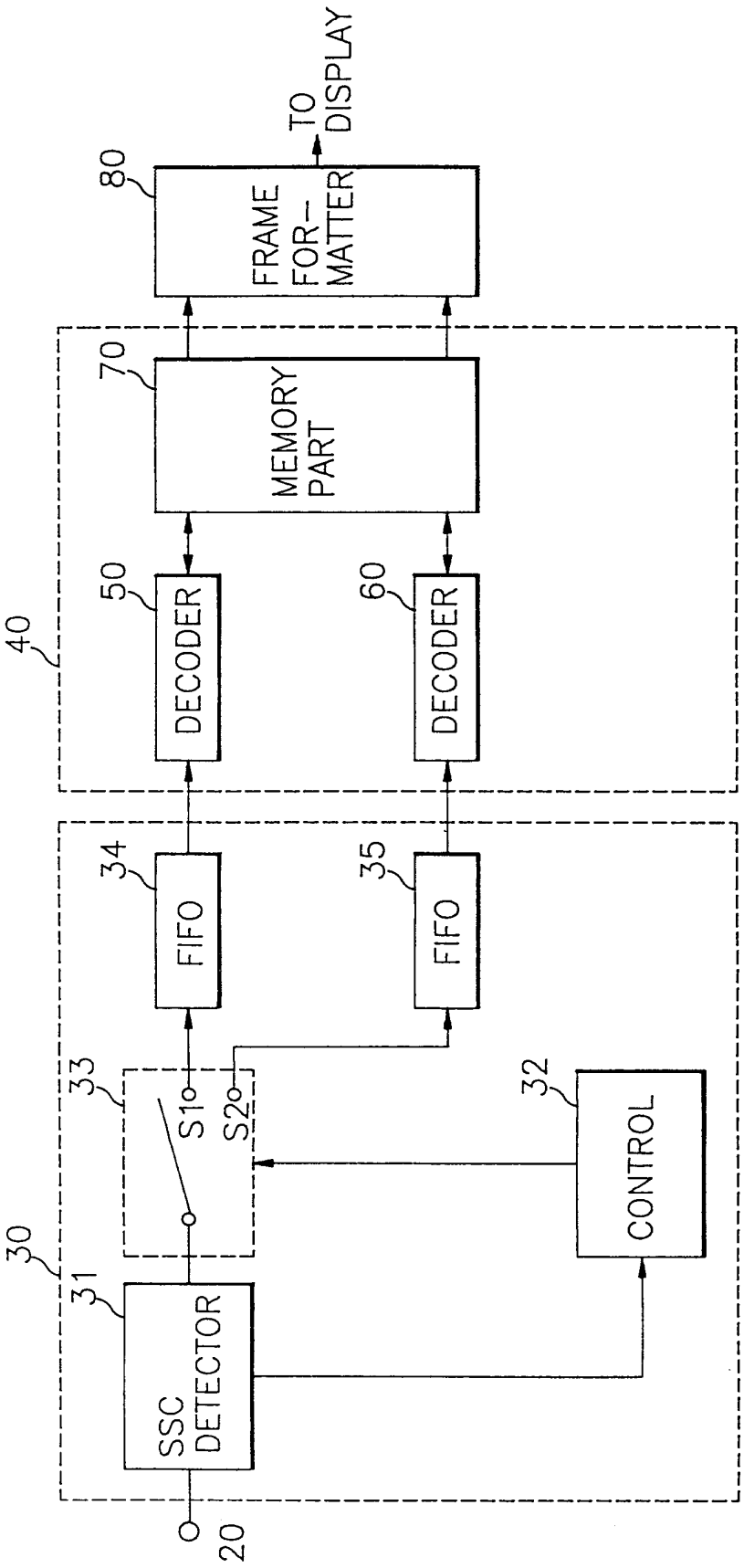


FIG.3

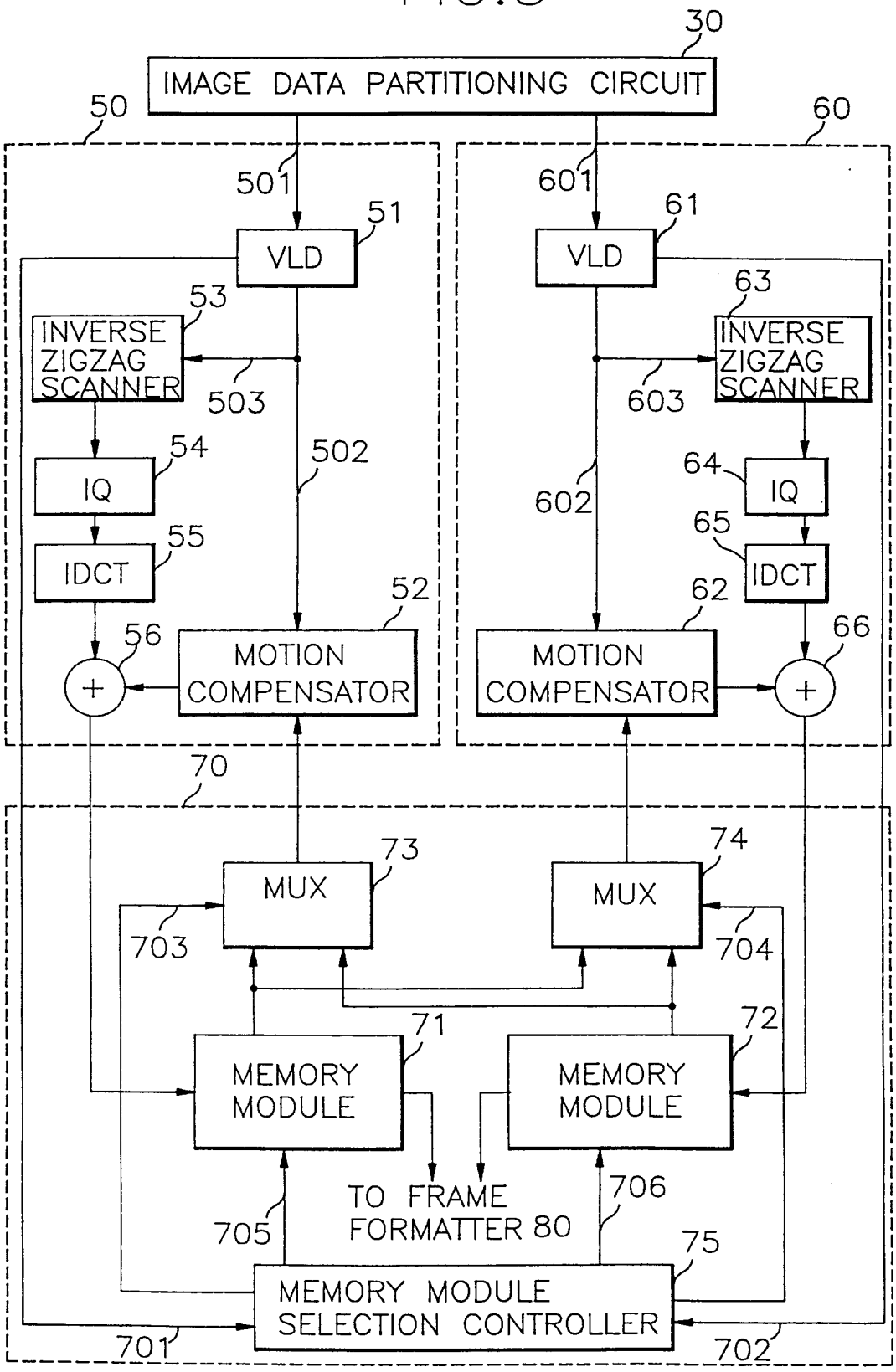
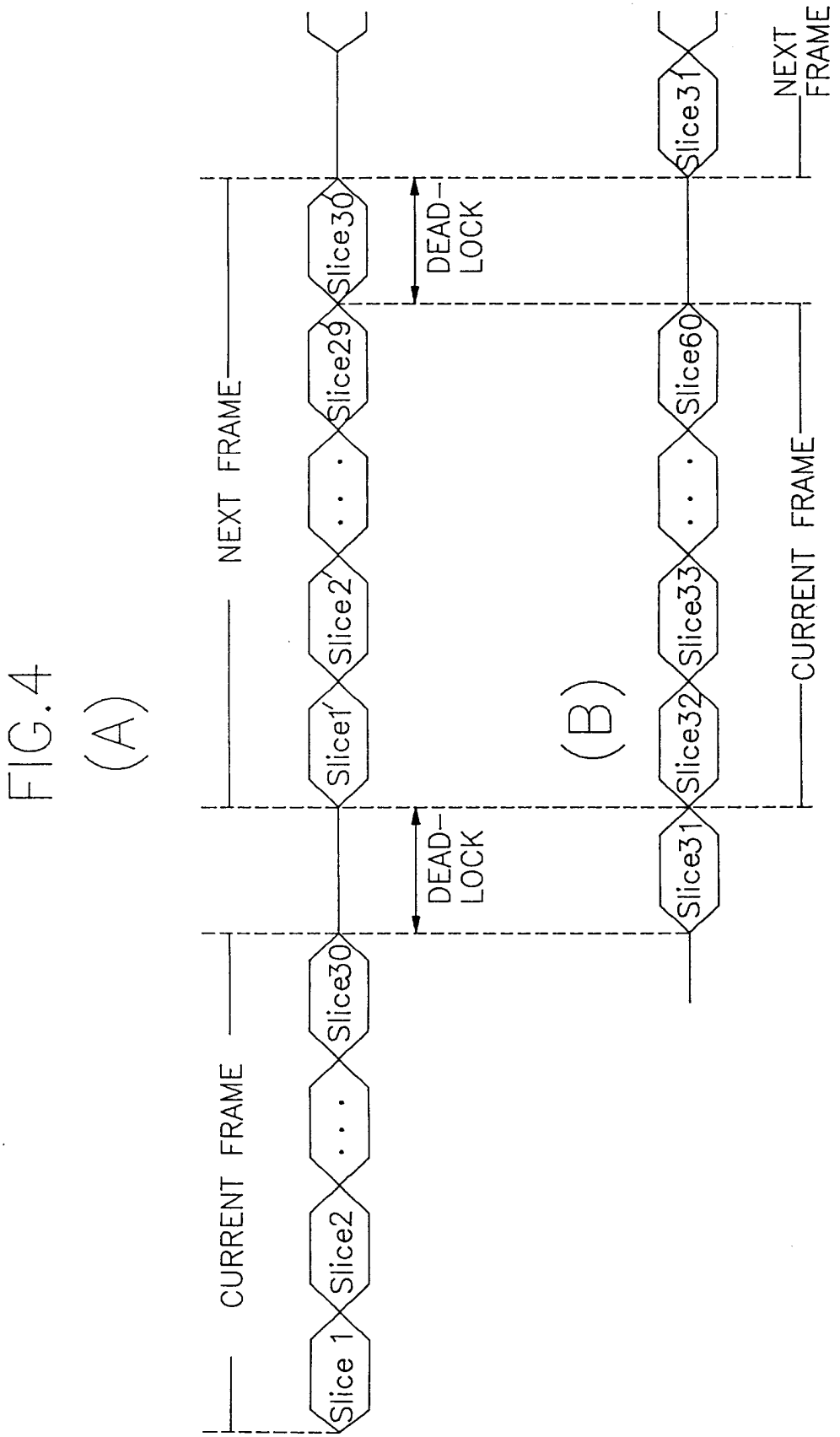


FIG.4



European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 12 0967

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP-A-0 479 511 (VICTOR COMPANY OF JAPAN, LTD.) * the whole document * ---	1,2	H04N7/26 H04N7/50
A	WO-A-91 11074 (BRITISH BROADCASTING CORP.) * the whole document * ---	1	
X	EP-A-0 577 310 (CANON K.K.) * the whole document * ---	1,2	
X	EP-A-0 614 317 (SONY CORP.) * the whole document * ---	1,2	
A	US-A-5 212 742 (NORMILE ET AL.) ---		
A	6TH MEDITERRANEAN ELECTROTECHNICAL CONFERENCE, vol. I, 22 May 1991 LJUBLJANA, SLOVENIA, pages 428-431, XP 000289486 E.J. LALOYA-MONZON ET AL. 'DSP Parallel Architecture for Image Compression' * paragraph 3 * -----	1,2	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

H04N

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	8 June 1995	Foglia, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 762 300 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
12.03.1997 Bulletin 1997/11

(51) Int. Cl.[6]: **G06F 17/30**, H04N 7/173, G06F 3/06

(21) Application number: **96114630.5**

(22) Date of filing: **12.09.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **12.09.1995 JP 234404/95**

(71) Applicant: **KABUSHIKI KAISHA TOSHIBA**
Kawasaki-shi, Kanagawa-ken 210 (JP)

(72) Inventors:
- **Yao, Hiroshi,**
  338 Toshiba-Shinkoyasu-Daiichi-ryo
  Yokohama-shi, Kanagawa-ken (JP)
- **Kanai, Tatsunori**
  Yokohama-shi, Kanagawa-ken (JP)
- **Kizu, Toshiki**
  Yokohama-shi, Kanagawa-ken (JP)
- **Maeda, Seiji**
  Fuchu-shi, Tokyo (JP)

(74) Representative: **Zangs, Rainer E., Dipl.-Ing. et al**
Hoffmann, Eitle & Partner
Arabellastrasse 4/VIII
81925 München (DE)

(54) Real time stream server for handling a plurality of real time stream data with different data rates

(57) A real time stream server capable of realizing a supply of a plurality of real time stream data with different data rates by a scheduling scheme using constant time-slot interval and transfer start timing period. A number of unit streams to be used and a block transfer time for each real time stream data are determined according to a data rate of each real time stream data. Each real time stream data is divided into a plurality of blocks, each block being in a size to be transferred within the block transfer time, and the blocks are sequentially distributed among the unit streams to be used. The blocks of each unit stream data are sequentially stored into a plurality of disk devices. In response to a request for each real time stream data from a client, the blocks constituting each real time stream data are read out from disk devices to a buffer memory, and each real time stream data is read out from a buffer memory and transferred to the client through a network, according to an appropriately scheduled transfer start timing for each unit stream.

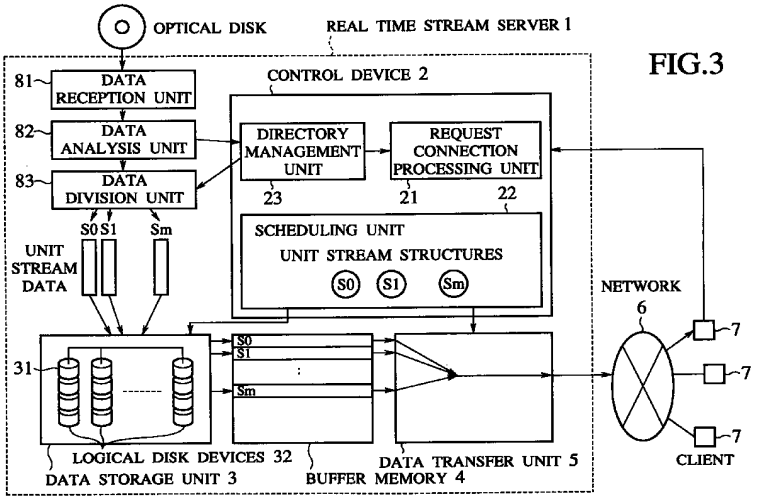


FIG.3

EP 0 762 300 A2

Description

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a real time stream server for supplying a plurality of real time stream data in different data rates simultaneously to clients, and a method for operating the real time stream server.

Description of the Background Art

Data to be sequentially transferred in real time such as video data and audio data are generally called "real time stream data". For a real time stream server for handling such real time stream data, a necessary condition is that it should be able to transfer the real time stream data stored in disk devices to each client while guaranteeing a continuity in real time.

In order to satisfy this condition, in the prior art, the real time stream data are stored in disks by being divided into blocks of a size to be transferred in a prescribed period of time, and the server makes accesses to disks periodically for each stream. The read out blocks are stored in a buffer memory once, and periodically transferred to corresponding clients through a communication network.

A stream scheduling device is a device which realizes a management of timings for issuing disk access commands and data transfer commands in this operation.

In addition, the scheduling device establishes a new stream channel upon receiving a connection request from a user. In order to manage disk accesses and data transfer timings, the scheduling device provides time-slots as shown in Fig. 1 which are partitioned at constant time interval. One disk access is allocated to one time-slot, and one disk access reads out one block of the real time stream data. A period of disk access in one stream is constant, so that by allocating disk accesses of different streams to different time-slots, it becomes possible to share the same stored data among a plurality of streams.

In an example shown in Fig. 1, four time-slots are provided in one period. That is, three disk devices with a read multiplexing level 4 are provided, and blocks are striped over these three disk devices. Consequently, an access period for one disk device is $4 \times 3 = 12$ time-slots, and a maximum number of simultaneously connectable streams is 12.

As shown in Fig. 1, time-slots by which one stream makes accesses to the disk devices are distanced each other by the access period of four time-slots. For example, after a read block A1 (a block-1 of the real time stream data A) is read out, it is transferred to a client-0 and processed (e.g., reproduced) in real time, and then a block A2 is read out and transferred before the processing of a block A1 is finished. In this manner, the scheduling is made so that each stream does not influence the continuity of the other streams.

Fig. 1 shows an exemplary case of a scheme in which the disk access allocated to the time-slot is fixed, but there is also a scheme in which an allocation position of a disk access is made variable among time-slots within a tolerable jitter range, by noting the fact that a time-slot to which a disk access is to be allocated can be changed during a period since a buffer memory becomes available until a transfer start timing (see, Japanese Patent Application No. 7-57384 (1995). Here, a range of time-slots to which a disk access of one stream can be allocated is called tolerable jitter range of that stream.

Fig. 2 shows an exemplary tolerable jitter range. In a case where all the real time stream data to be handled have the same data rate, the scheduling becomes easier by a scheme utilizing the tolerable jitter range.

Now, in a case of handling real time stream data with a higher data rate, a block size becomes larger, and there can be a case in which a reading from a disk cannot be completed within one time-slot.

In order to avoid such a situation, if a time-slot interval and a block size are fixedly set in accordance with the real time stream data with the maximum data rate, there arises a problem in a case where real time stream data with a low data rate are to be supplied at the same time because it would become impossible to take a full advantage of a transfer capacity of the disk devices. This is due to the fact that, for the same period, the block size changes in proportion to the data rate.

On the other hand, when a time-slot interval is varied according to a block size, it becomes difficult to realize a flexible disk access scheduling in which orders of disk accesses can be interchanged. In addition, the buffer memory management becomes complicated in such a case, because there is a need to secure a continuous region in a size of each block size as the buffer memory.

Moreover, when a block size is fixed regardless of a data rate, a period of access to one disk device is going to be different for different streams, and for this reason, it becomes difficult to judge whether it is possible to connect a new stream while guaranteeing the continuity of the already connected streams.

Thus, in a case of supplying a plurality of real time stream data with different data rates, it has conventionally been difficult to realize both a scheme for making the disk access scheduling easier by fixing a period of access to one disk

device and a time-slot interval, and a scheme for taking a full advantage of a transfer capacity of disk devices by changing a number of data supply streams according to a data rate.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a real time stream server and a method for operating a real time stream server, capable of realizing a supply of a plurality of real time stream data with different data rates by a scheduling scheme using constant time-slot interval and transfer start timing period, without wasting a transfer capacity of disk devices.

According to one aspect of the present invention there is provided a real time stream server, comprising: entering means for entering real time stream data to be stored in the real time stream server; determining means for determining a number of unit streams to be used and a block transfer time for the real time stream data, according to a data rate of the real time stream data; dividing means for dividing the real time stream data into a plurality of blocks, each block being in a size to be transferred within the block transfer time, and sequentially distributing the blocks among as many unit streams as the number of unit streams to be used; a plurality of disk devices for sequentially storing the blocks of each unit stream data; a buffer memory for temporarily storing the blocks read out from said plurality of disk devices; control means for reading out the blocks constituting the real time stream data from said plurality of disk devices to the buffer memory, and reading out the real time stream data from the buffer memory, according to a request for the real time stream data from a client; and transfer means for transferring the real time stream data read out from the buffer memory to the client through a network.

According to another aspect of the present invention there is provided a method for operating a real time stream server having a plurality of disk devices and a buffer memory, comprising the steps of: entering real time stream data into the real time stream server; determining a number of unit streams to be used and a block transfer time for the real time stream data, according to a data rate of the real time stream data; dividing the real time stream data into a plurality of blocks, each block being in a size to be transferred within the block transfer time, and sequentially distributing the blocks among as many unit streams as the number of unit streams to be used; sequentially storing the blocks of each unit stream data into said plurality of disk devices; controlling the real time stream server to read the blocks constituting the real time stream data from said plurality of disk devices to the buffer memory, and read out the real time stream data from the buffer memory, according to a request for the real time stream data from a client; and transferring the real time stream data read out from the buffer memory to the client through a network.

Other features and advantages of the present invention will become apparent from the following description taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a timing chart showing one conventional scheme for managing disk accesses and data transfer timings in a stream scheduling device.

Fig. 2 is a timing chart showing another conventional scheme for managing disk accesses and data transfer timings in a stream scheduling device.

Fig. 3 is a block diagram of one embodiment of a real time stream server according to the present invention.

Fig. 4 is a flow chart of an operation for storing real time stream data into memory devices from external in the real time stream server of Fig. 3.

Fig. 5 is a flow chart of an operation for supplying real time stream data according to a request from external in the real time stream server of Fig. 3.

Fig. 6 is a diagram showing exemplary transfer start timings for blocks read out from logical disk devices in a concrete example of the real time stream server of Fig. 3, in a case of using one unit stream.

Fig. 7 is a diagram showing one exemplary transfer start timings for blocks read out from logical disk devices in a concrete example of the real time stream server of Fig. 3, in a case of using two unit streams.

Fig. 8 is a diagram showing another exemplary transfer start timings for blocks read out from logical disk devices in a concrete example of the real time stream server of Fig. 3, in a case of using two unit streams.

Fig. 9 is a diagram showing exemplary transfer start timings for blocks read out from logical disk devices in a concrete example of the real time stream server of Fig. 3, in a case of using three unit stream.

Fig. 10 is a diagram showing exemplary transfer start timings for blocks read out from logical disk devices in a concrete example of the real time stream server of Fig. 3, in a case of using four unit stream.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

First, the main features of a real time stream server according to the present invention will be summarized briefly.

In the present invention, a unit stream having a reference data rate is defined. Then, in a case of supplying real time

stream data in excess of the data rate of the unit stream, as many unit streams as necessary for the required data rate are used to supply this real time stream data.

At first, the real time stream data is divided into unit streams, and stored into a plurality of disk devices. Namely, according to a data rate of entered real time stream data, a number of unit streams to be used and a block transfer time are determined. Then, the real time stream data is divided into a plurality of blocks, each in a size to be transferred with the block transfer time, and these blocks are sequentially allocated to unit stream data. Then, for each unit stream data, allocated blocks are stored by distributing them over a plurality of disk devices sequentially from a top block. Also, for each real time stream data, a number of unit streams to be used, an ID number of a disk device which stores the top block of each unit stream, and a recording position of each block on a disk device which stores that block are recorded.

When a request for this real time stream data so stored in the disk devices is received from a client, stream recourses for as many unit streams as a recorded number of unit streams to be used for the requested real time stream data are secured (reserved), and a transfer start timing of each unit stream is scheduled so that the blocks are transferred continuously in an order as in the original real time stream data. At this point, the transfer start timings of the unit streams are scheduled to be displaced one another by a block transfer time part.

In transferring each block, a corresponding block on the buffer memory is transferred within the block transfer time starting from the transfer start timing scheduled for each unit stream. Here, the transfer start timings of the unit streams are displaced one another by a block transfer time part, so that when the block transfer of a preceding unit stream is finished, the block transfer of a subsequent unit stream is started immediately, and the blocks of the original real time stream data are transferred while guaranteeing the continuity.

In this manner, according to the present invention, the real time stream data with various data rates can be handled by using an appropriate number of independent unit streams according to a data rate of each real time stream data, and therefore it becomes possible to realize the disk access scheduling based on a single scheme regarding a time-slot interval, a block size, a block transfer period, and a buffer memory management, without distinguishing different data rates.

Also, according to the present invention, the stream resources are divided into amounts corresponding to the unit streams, and allocated as much as necessary according to a data rate of each real time stream data to be supplied, so that it becomes possible to utilize the stream resources efficiently without wasting any resource.

Also, in a case of the real time stream server which has a capacity to supply Cmax sets of real time stream data with data rates not greater than R, where R is a reference data rate of a unit stream, a number of supplies Cm (m = 1, 2, • • • • • • • •) to be allocated to each real time stream data with a data rate in a data rate range greater than (m-1)×R and not greater than m×R can be set arbitrarily within a range of:

$$\sum_{m}(m \times Cm) \leqq Cmax$$

where m is a number of unit streams to be used for each real time stream data.

Referring now to Fig. 3 to Fig.10, one embodiment of a real time stream server according to the present invention will be described in detail.

Fig. 3 shows a configuration of a real time stream server in this embodiment, Fig. 4 shows a flow chart of an operation for storing real time stream data into memory devices from external, and Fig. 5 shows a flow chart of an operation for supplying real time stream data according to a request from external.

As shown in Fig. 3, the real time stream server 1 of this embodiment comprises a control device 2, a data storage unit 3 connected with the control device 2, a buffer memory 4 connected with the data storage unit 3, a data transfer unit 5 connected with the control device 2 and the buffer memory 4 and connected to an external network 6, a data reception unit 81 for receiving externally entered real time stream data, a data analysis unit 82 connected with the data reception unit 81 and the control device 2, and a data division unit 83 connected with the control device 2 and the data storage unit 3.

The control device 2 has a request connection processing unit 21, a scheduling unit 22, and a directory management unit 23. The data storage unit 3 has a plurality of disk devices 31 for storing real time stream data in units of blocks.

First, with references to Fig. 3 and Fig. 4, one aspect of this real time stream server 1 regarding a phase for storing real time stream data acquired at the data reception unit 81 into the data storage unit 3 will be described.

Here, the "unit stream" in the real time stream server is defined as a stream which is scheduled by using a block transfer period T, a block size L, and a time-slot interval I which serve as references for the operation of the real time stream server.

The "unit data rate R" is defined as a maximum data rate that can be scheduled as a unit stream. Consequently, the following relationship holds among the block transfer period T, the block size L, and the unit data rate R.

$$L \leqq R * T$$

A value of this unit data rate R is to be determined appropriately in view of optimizing the disk reading efficiency and the data transter/storing efficiency.

Now, at a time of storing real time stream data in the real time stream server, the real time stream data to be stored is entered from the data reception unit 81 (step S11). Here, the data reception unit 81 may be in a form of reading data from an external memory device such as optical disk, floppy disk, CD-ROM, etc., or in a form of receiving data transferred through a network.

At the data analysis unit 82, a maximum data rate of the entered real time stream data is checked (step S12). To realize this checking, the maximum data rate may be obtained by analyzing a header of the real time stream data, or the maximum data rate known in advance may be entered by a human operator through an input device such as keyboard.

When the maximum data rate is obtained, a number of unit streams to be used for that real time stream data is determined (step S13). Here, a number of unit streams to be used is a numerical value indicating a capacity and resources of the real time stream server required for supplying that real time stream data, which is expressed in terms of a required number of the unit streams. Actually, a number of unit streams to be used can be determined as a minimum integer m which satisfies the following relationship.

$$(\text{Maximum data rate}) \leqq m \times (\text{Unit data rate R})$$

When the number of unit streams to be user is determined, the block transfer time which serves as a reference size for dividing the real time stream data into blocks is obtained according to the following formula (step S14).

$$(\text{Block transfer time}) = (\text{Block transfer period T})/m$$

An amount of data to be transferred within this block transfer time obtained according to this formula never exceed the block size L of the unit stream.

The number of unit streams to be used and the block transfer time calculated at the data analysis unit 82 are then sent to the data division unit 83 along with the real time stream data.

At the data division unit 83, the real time stream data is divided into blocks, each in a size to be transferred within the block transfer time T/m (step S15). Then, these blocks are distributed among as many unit stream data as the number of unit streams to be used (step S16). Here, the distribution should preferably be done by using a fixed rule so that the schedules will not interfere among the unit stream data.

More specifically, for example, a set of blocks as a whole can be divided into m pieces of non-dense subsets Bj given by:

$$Bj = \{b(m \times k+j) \mid k = 0, 1, \cdots\cdots\cdots\} \; (j = 0, \cdots\cdots\cdots, m-1)$$

and each such non-dense subset Bj can be identified as a unit stream data. For instance, in a case where m = 4 and there are sixteen blocks b0 to b15, the blocks are distributed among four unit stream data as follows.

$$B0 = \{b0, b4, b8, b12\}$$

$$B1 = \{b1, b5, b9, b13\}$$

$$B2 = \{b2, b6, b10, b14\}$$

$$B3 = \{b3, b7, b11, b15\}$$

These m pieces of unit stream data (B0 to Bm-1) are then sent to the data storage unit 3, and each unit stream data is stored into N sets of disk devices, in a similar manner as used for a unit stream data in a case of m = 1. More specifically, a top block bj of the unit stream data B0 is stored into a disk device of a disk device ID number Hj, while each block b(m×k+j) is stored into a disk device of a disk device ID number [(k+Hj) modulo N].

At this point, the disk device ID number Hj of the disk device which stores the top block of each unit stream, and a recording position of each block on a disk device which stores each block are recorded as a directory information in the directory management unit 23 within the control device 2 (step S18).

Here, a recording position on a disk may be specified by physical cylinder number, track number, and sector number, or by a logical number that can identify a specific position.

Also, a manner of sending m pieces of unit stream data obtained at the data division unit 83 to the data storage unit

3 can be in a form of once storing all the unit stream data on another memory device provided separately from the data storage unit 3 of the real time stream server and then sending each unit stream to the data storage unit 3 one by one, or in a form of sending all m pieces of unit stream data to the data storage unit 3 in parallel.

Next, with references to Fig. 3 and Fig. 5, another aspect of this real time stream server 1 regarding a phase for transferring the real time stream data to a client 7 upon receiving a request for supply of the real time stream data will be described.

Note that, generally speaking, there are two types of a scheduling scheme for allocating a time-slot to a disk access, including a fixed allocation type and a variable allocation type, and the present invention is applicable to either type of the scheduling scheme. In the following, it is only assumed that the scheduling is to be made in such a manner that the continuity of streams is guaranteed, without limiting the scheduling scheme to either one of these two types.

When a request for supply of the real time stream data is received from a client 7 through the network 6 (step S21), the request connection processing unit 21 first obtains a directory information for the requested real time stream data from the directory management unit 23 (step S22).

Then, the number m of unit streams to be used is obtained from the directory information (step S23), and as many stream structures in the scheduling unit 22 as necessary for holding information required in managing m pieces of unit streams are secured (reserved), while a necessary amount of regions in the buffer memory 4 are secured (reserved) (step S24).

Next, the scheduling unit 22 carries out the scheduling including a selection of transfer start timings for the unit streams S0 to Sm-1 to be used (step S25). Here, by the real time stream data storing procedure described above, m pieces of blocks b(m×k+j) (j = 0, • • • • • • • • • , m-1) which are continuous in the original real time stream data are sequentially distributed among the unit streams S0 to Sm-1. Consequently, in order to carry out the transfer of these blocks continuously, the transfer start timings of the unit streams S0 to Sm-1 are displaced one another by the block transfer time T/m part.

Here however it is necessary for each one of the unit streams S0 to Sm-1 to select a time-slot for carrying out the disk access, so that it becomes possible to read out the respective top block from the disk device 31 which stores that top block, before the selected transfer start timing, without affecting the continuity of the other already connected unit streams. Note that the ID number of the disk device 31 which stores the top block of each unit stream can be obtained from the directory information obtained at the step S22. This time-slot selection operation will be described in detail below. When this condition is not satisfied, it is necessary to select different transfer start timings anew.

Each one of the unit streams S0 to Sm-1 so connected is then scheduled as an independent unit stream for which the block transfer time for one block is T/m, that is, scheduled according to the block transfer period T, the block size L, the time-slot interval l, and the block transfer time T/m. Then, according to this scheduling, each block is read out from the buffer memory 4 from the recording position of each block on a disk device which is indicated by the directory information, and supplied to the client 7 from the data transfer unit 5 (step S26).

According to this embodiment of the present invention, the real time stream data with a data rate not greater than m×R are handled as m pieces of independent unit streams, so that it becomes possible to realize the disk access scheduling based on a single scheme regarding a time-slot interval, a block size, a block transfer period, and a buffer memory management.

Also, according to this embodiment of the present invention, in a case of the real time stream server which has a capacity to supply Cmax sets of real time stream data with data rates not greater than R, a number of supplies Cm (m = 1, 2, • • • • • • • •) to be allocated to each real time stream data with a data rate in a data rate range greater than (m-1)×R and not greater than m×R can be set arbitrarily within a range of:

$$\sum_{m}(m \times Cm) \leqq Cmax$$

where m is a number of unit streams to be used for each real time stream data, as should be apparent from the above description.

Now, with references to Fig. 6 to Fig. 10, concrete examples of the real time stream server of the present invention will be described.

First, an exemplary case of the real time stream server which stores and supplies the real time stream data by using a unit stream with a unit data rate R = 1.6 (Mbps) will be described.

When the block transfer time for one block is selected to be 2400 (ms), the maximum size of one block is 500 (kB). Assuming the the transfer rate of the disk device to be used is 20 (Mbps), there is a need to set the time-slot interval to be not less than 200 (ms) in order to read out one block by one time-slot.

In order to raise the transfer rate higher, one block can be divided and stored over a plurality of disk devices 31. For example, considering four disk devices 31 as one set, one block can be divided into four and stored into four disk

devices 31 respectively, and at a time of reading, all of these four disk devices 31 of the same set can be accessed simultaneously. In this manner, the effective maximum block size on a disk becomes 125 (kB), and the time-slot interval of not less than 50 (ms) becomes necessary. In the following, a set of these four disk devices 31 is regarded as one logical disk device 32.

When the time-slot interval is set to be 50 (ms) by regarding a set of four disk devices as one logical disk device as described above, a number of unit streams that can be supplied by one logical disk device is 2400/50 = 48 pieces. In addition, when eight of such logical disk devices are provided, it becomes possible to supply $48 \times 8 = 384$ pieces of the unit streams.

In a case of handling the unit streams in such a real time stream server, the block bk is to be stored in the logical disk device with a logical disk device ID number [(k+H) modulo 8], where H is a logical disk device ID number of the logical disk device which stores the top block b0.

In order to supply these unit streams in accordance with the request from the client 7, the disk devices storing the block to be transferred next are accessed at a period of 2400 (ms), and the block of not greater than 500 (kB) in total is read out to the buffer memory 4, 125 (kB) from each one of four disk devices corresponding to one logical disk device, within the time-slot interval 50 (ms). This block is then transferred to the client 7 at the period of 2400 (ms). in the transfer time of 2400 (ms).

Fig. 6 shows exemplary transfer start timings for blocks read out from the logical disk devices storing blocks of one unit stream in this case. In Fig. 6, the top block is stored in the logical disk device-0 (disk-0). Note however that there is no need for a disk device to store the top block to be the logical disk device-0, and it is also possible to store the top block of different real time stream data in different logical disk devices.

Next, in the real time stream server of the concrete example described above, a case of handling the real time stream data with a data rate of 3 (Mbps) will be considered. Here, the minimum integer m which satisfies $3 \leqq m \times R = m \times 1.6$ (Mbps) is 2, so that it is possible to store and supply the real time stream data with a date rate of 3 (Mbps) by using 2 pieces of unit streams with the block transfer time for one block equal to 1200 (ms). In order to store the real time stream data with a date rate of 3 (Mbps), this data is divided into blocks, each in a size to be transferred in 1200 (ms), and a set of these blocks are divided into m = 2 subsets:

$$B0 = \{b(2k) \mid k = 0, 1, \cdots\cdots\cdots\}$$

$$B1 = \{b(2k+1) \mid k = 0, 1, \cdots\cdots\cdots\}$$

Then, the block b(2k+j) is stored into the logical disk device with a logical disk device ID number [(k+Hj) modulo 8]. In other words, for B0, the top block b0 is stored into the logical disk device with a logical disk device ID number H0, the next block b2 is stored into the logical disk device with a logical disk device ID number [(H0+1) modulo 8], the next block b4 is stored into the logical disk device with a logical disk device ID number [(H0+2) modulo 8], and so on.

In order to supply this stream in response to the request from the client 7, two unit streams S0 and S1 for supplying B0 and B1 respectively are prepared.

Then, the reference time-slots for the unit stream S0 and the unit stream S1 are selected such that the transfer start timings of the block b0 and the block b1 are displaced by 2400/2 = 1200 (ms). Then, the disk access is allocated to the time-slot in a similar manner as used for a unit stream in a case of m = 1, but here it must be possible to realize the disk access allocation for the unit stream S0 and the disk access allocation for the unit stream S1 simultaneously. If the simultaneous disk access allocations are impossible, the reference time-slots of the unit streams S0 and S1 are to be selected anew.

When the reference time-slots of the unit streams S0 and S1 are determined in this manner, it suffices to schedule the unit streams S0 and S1 as independent unit streams. In other words, for each one of the unit stream S0 and S1, a block to be transferred next by using the time-slot to which the disk access is allocated is read out from the logical disk device which stores this block to the buffer memory 4, and then this block is transferred to the client 7 at the period of 2400 (ms), in the transfer time of 1200 (ms). Here, only the transfer time differs from a case of a usual unit stream. A size of one block is approximately 460 (kB), which is less than the maximum block size of 500 (kB) for a usual unit stream, so that the time-slot interval and the buffer memory management can be exactly the same as in a case of a usual unit stream.

Fig. 7 shows exemplary transfer start timings for blocks read out from the logical disk devices storing blocks of the unit streams S0 and S1 in this case. In Fig. 7, the top blocks of the unit streams S0 and S1 are stored in the logical disk devices with the logical disk device ID numbers H0 = 0 and H1 = 4. Note however that any other combination of the logical disk devices to store the top blocks may be used. Fig. 8 shows another exemplary transfer start timings in this case, where H0 = H1 = 0 , so that the block b0 and the block b1 are stored in the same logical disk device in this case.

Next, in the real time stream server of the concrete example described above, a case of handling the real time stream data with a data rate of 4.5 (Mbps) will be considered. Here, the minimum integer m which satisfies $4.5 \leqq m \times R = m \times 1.6$ (Mbps) is 3, so that it is possible to store and supply the real time stream data with a date rate

of 4.5 (Mbps) by using 3 pieces of unit streams with the block transfer time for one block equal to 800 (ms).

In order to store the real time stream data with a date rate of 4.5 (Mbps), this data is divided into blocks, each in a size to be transferred in 800 (ms), and a set of these blocks are divided into m = 3 subsets:

$$B0 = \{b(3k) \mid k = 0, 1, \cdots\cdots\cdots\}$$

$$B1 = \{b(3k+1) \mid k = 0, 1, \cdots\cdots\cdots\}$$

$$B2 = \{b(3k+2) \mid k = 0, 1, \cdots\cdots\cdots\}$$

Then, the block b(3k+j) is stored into the logical disk device with a logical disk device ID number [(k+Hj) modulo 8].

In order to supply this stream in response to the request from the client 7, three unit streams S0, S1 and S2 for supplying B0, B1 and B2 respectively are prepared.

Then, the reference time-slots for the unit streams S0, S1 and S2 are selected such that the transfer start timings of the respective blocks are displaced by 2400/3 = 800 (ms).

Fig. 9 shows exemplary transfer start timings for blocks read out from the logical disk devices storing blocks of the unit streams S0, S1 and S2, for an exemplary case of setting H0 = 0, H1 = 3, and H2 = 6.

Similarly, in a case of handling the real time stream data with a data rate of 6 (Mbps), the minimum integer m which satisfies $6 \leqq m \times R = m \times 1.6$ (Mbps) is 4, so that it is possible to store and supply the real time stream data with a date rate of 6 (Mbps) by using 4 pieces of unit streams with the block transfer time for one block equal to 600 (ms).

Fig. 10 shows exemplary transfer start timings for blocks read out from the logical disk devices storing blocks of the unit streams S0, S1, S2 and S3, for an exemplary case of setting H0 = 0, H1 = 2, H2 = 4, and H3 = 6.

Note that, in the real time stream server of the concrete example described above, a number of unit streams that can be supplied is set as 384, but it is possible to change a number of unit streams to be used arbitrarily according to the request from the client 7 as long as a total number of unit streams to be used is within 384. For example, it is possible to supply 96 sets of streams with a data rate of 6 (Mbps) (corresponding to a total number of unit streams equal to 384) alone, or 48 sets of streams with a data rate of 6 (Mbps) (corresponding to a number of unit streams equal to 192) and 192 sets of streams with a data rate of 1.5 (Mbps) (corresponding to a number of unit streams equal to 192) together.

Next, a manner of selecting the time-slots at the connection request processing unit 21 in the real time stream server 1 of Fig. 3 will be described.

In order to supply the real time stream server with a data rate of m×R, it is necessary to allocate m pieces of disk accesses to the time-slots so that the m pieces of unit streams can be supplied continuously as described above.

Now, as already mentioned above, generally speaking, there are two types of a scheduling scheme for actually allocating a time-slot to a disk access, including a fixed allocation type and a variable allocation type.

The former is the scheduling scheme in which the disk access is fixed to the time-slot which is positioned a certain period of time away relatively from the reference time-slot as shown in Fig. 1. In other words, it is the scheduling scheme which fixes a positional relationship between the transfer timing and the disk access timing on the time-slots.

The latter is the scheduling scheme in which the allocation position of the disk access is made variable among the time-slots within the tolerable jitter range as shown in Fig. 2, by noting the fact that a time-slot to which a disk access is to be allocated can be changed during a period since a buffer memory becomes available until a transfer start timing.

Note here that, in Fig. 2, J indicates the maximum jitter number which is determined according to the following formula.

$$J \leqq BM - D - T - 1$$

where B is a ratio of a size of a buffer memory that can be used by one stream and a size of one block of the real time stream data, M is a time (a number of slots) for reproducing one block at a client, T is a time (a number of slots) for transferring one block to a client, and D is an estimated maximum delay time (a number of slots) in a case where the disk access end timing extends beyond the end timing of the allocated time-slot. In an example shown in Fig. 2, a time required for transferring one block to the client side and a time required for reproducing one block at the client side are assumed to be equal to each other, as a time equivalent to 4 time-slots.

Now, in the scheduling scheme in which the disk access is fixed to the time-slot which is positioned a certain period of time away relatively from the reference time-slot, depending on a combination of the reference time-slots for already connected unit streams. there can be a case in which the reference time-slots for m pieces of new unit streams overlap with the reference time-slots for the already connected unit streams, no matter how these reference time-slots for m pieces of new unit streams are selected. In such a case, it is impossible to allocate the disk access to the time-slot, and it is impossible to connect a new stream.

In contrast, in the scheduling scheme in which the allocation position of the disk access is made variable among

the time-slots within the tolerable jitter range, as long as there is a vacant time-slot within the tolerable jitter range as in a case shown in Fig. 2, it suffices to allocate the disk access of a new unit stream to that vacant time-slot. Also, even when there is no vacant time-slot within the tolerable jitter range, by moving the disk access of the already connected unit stream to another time-slot within its tolerable jitter range, it is possible to create a vacant time-slot and allocate the disk access of a new unit stream to that vacated time-slot.

Even in this case, however, a new stream cannot be connected when it is impossible to create a vacant time-slot no matter how the disk accesses are moved. In order to lower a probability for such a situation to arise, it is preferable to select the reference time-slots such that the reference time-slots of the unit streams are not concentrated in a short period of time as much as possible, that is, the vacant time-slots are as widely spread as possible over all the time-slots.

As described, according to the present invention, the real time stream data with various data rates can be handled by using an appropriate number of independent unit streams according to a data rate of each real time stream data, and therefore it becomes possible to realize the disk access scheduling based on a single scheme regarding a time-slot interval, a block size, a block transfer period, and a buffer memory management, without distinguishing different data rates.

Also, according to the present invention, the stream resources are divided into amounts corresponding to the unit streams, and allocated as much as necessary according to a data rate of each real time stream data to be supplied, so that it becomes possible to utilize the stream resources efficiently without wasting any resource.

It is to be noted that, besides those already mentioned above, many modifications and variations of the above embodiments may be made without departing from the novel and advantageous features of the present invention. Accordingly, all such modifications and variations are intended to be included within the scope of the appended claims.

Claims

1. A real time stream server, comprising:

entering means for entering real time stream data to be stored in the real time stream server;
determining means for determining a number of unit streams to be used and a block transfer time for the real time stream data, according to a data rate of the real time stream data;
dividing means for dividing the real time stream data into a plurality of blocks, each block being in a size to be transferred within the block transfer time, and sequentially distributing the blocks among as many unit streams as the number of unit streams to be used;
a plurality of disk devices for sequentially storing the blocks of each unit stream data;
a buffer memory for temporarily storing the blocks read out from said plurality of disk devices;
control means for reading out the blocks constituting the real time stream data from said plurality of disk devices to the buffer memory, and reading out the real time stream data from the buffer memory, according to a request for the real time stream data from a client; and
transfer means for transferring the real time stream data read out from the buffer memory to the client through a network.

2. The real time stream server of claim 1, wherein the control means includes:

management means for managing a directory information for each real time stream data stored in said plurality of disk devices, the directory information indicating the number of unit streams to be used, a disk device ID number of a disk device which stores a top block of each unit stream, and a recording position of each block on a disk device which stores each block, for said each real time stream data.

3. The real time stream server of claim 1, wherein the control means carries out operations with respect to each real time stream data according to the directory information for each real time stream data managed in the management means.

4. The real time stream server of claim 1, wherein the control means includes:

scheduling means for securing stream resources for as many unit streams as the number of unit streams to be used, scheduling a transfer start timing for each unit stream in order to transfer the blocks continuously in a sequential order to form the real time stream data, and controlling the transfer means to read out each block of each unit stream on the buffer memory within the block transfer time since the transfer start timing for each unit stream.

5. The real time stream server of claim 4, wherein the scheduling means schedules the transfer start timing for each

unit stream such that scheduled transfer start timings of the unit streams are displaced one another by the block transfer time.

6. The real time stream server of claim 1, wherein the unit stream is defined as a stream which is scheduled by using a prescribed block transfer period T, a prescribed block size L, and a prescribed time-slot interval l.

7. The real time stream server of claim 1, wherein the determining means detects a maximum data rate of the real time stream data, and determines the number m of unit streams to be used as a minimum integer which satisfies:

$$\text{the maximum data rate} \leqq m \times R$$

where R is a prescribed maximum data rate of each unit stream.

8. The real time stream server of claim 7, wherein the determining means determines the block transfer time as:

$$\text{the block transfer time} = T/m$$

where T is a prescribed block transfer period, and m is the number of unit streams to be used.

9. The real time stream server of claim 1, wherein the real time stream server has a capacity to supply Cmax sets of real time stream data with data rates not greater than R, where R is a prescribed maximum data rate of each unit stream, and the control means sets a number of supplies Cm (m = 1, 2, • • • • • • • •) to be allocated to each real time stream data with a data rate in a data rate range greater than (m-1)×R and not greater than m×R, arbitrarily within a range of:

$$\sum_{m}(m \times Cm) \leqq Cmax$$

where m is the number of unit streams to be used for said each real time stream data.

10. A method for operating a real time stream server having a plurality of disk devices and a buffer memory, comprising the steps of:

entering real time stream data into the real time stream server;
determining a number of unit streams to be used and a block transfer time for the real time stream data, according to a data rate of the real time stream data;
dividing the real time stream data into a plurality of blocks, each block being in a size to be transferred within the block transfer time, and sequentially distributing the blocks among as many unit streams as the number of unit streams to be used;
sequentially storing the blocks of each unit stream data into said plurality of disk devices;
controlling the real time stream server to read the blocks constituting the real time stream data from said plurality of disk devices to the buffer memory, and read out the real time stream data from the buffer memory, according to a request for the real time stream data from a client; and
transferring the real time stream data read out from the buffer memory to the client through a network.

11. The method of claim 10, wherein the controlling step includes the step of:

managing a directory information for each real time stream data stored in said plurality of disk devices, the directory information indicating the number of unit streams to be used, a disk device ID number of a disk device which stores a top block of each unit stream, and a recording position of each block on a disk device which stores each block, for said each real time stream data.

12. The method of claim 11, wherein the controlling step carries out operations with respect to each real time stream data according to the directory information for each real time stream data managed by the management step.

13. The method of claim 10, wherein the controlling step includes steps of:

securing stream resources for as many unit streams as the number of unit streams to be used;

scheduling a transfer start timing for each unit stream in order to transfer the blocks continuously in a sequential order to form the real time stream data; and

controlling the real time stream server to read out each block of each unit stream on the buffer memory within the block transfer time since the transfer start timing for each unit stream.

14. The method of claim 13, wherein the scheduling step schedules the transfer start timing for each unit stream such that scheduled transfer start timings of the unit streams are displaced one another by the block transfer time.

15. The method of claim 10, wherein the unit stream is defined as a stream which is scheduled by using a prescribed block transfer period T, a prescribed block size L, and a prescribed time-slot interval l.

16. The method of claim 10, wherein the determining step detects a maximum data rate of the real time stream data, and determines the number m of unit streams to be used as a minimum integer which satisfies:

$$\text{the maximum data rate} \leqq m \times R$$

where R is a prescribed maximum data rate of each unit stream.

17. The method of claim 16, wherein the determining step determines the block transfer time as:

$$\text{the block transfer time} = T/m$$

where T is a prescribed block transfer period, and m is the number of unit streams to be used.

18. The method of claim 10, wherein the real time stream server has a capacity to supply Cmax sets of real time stream data with data rates not greater than R, where R is a prescribed maximum data rate of each unit stream, and the controlling step sets a number of supplies Cm (m = 1, 2, • • • • • • • • •) to be allocated to each real time stream data with a data rate in a data rate range greater than (m-1)×R and not greater than m×R, arbitrarily within a range of:

$$\sum_{m}(m \times Cm) \leqq Cmax$$

where m is the number of unit streams to be used for said each real time stream data.

FIG.1
PRIOR ART

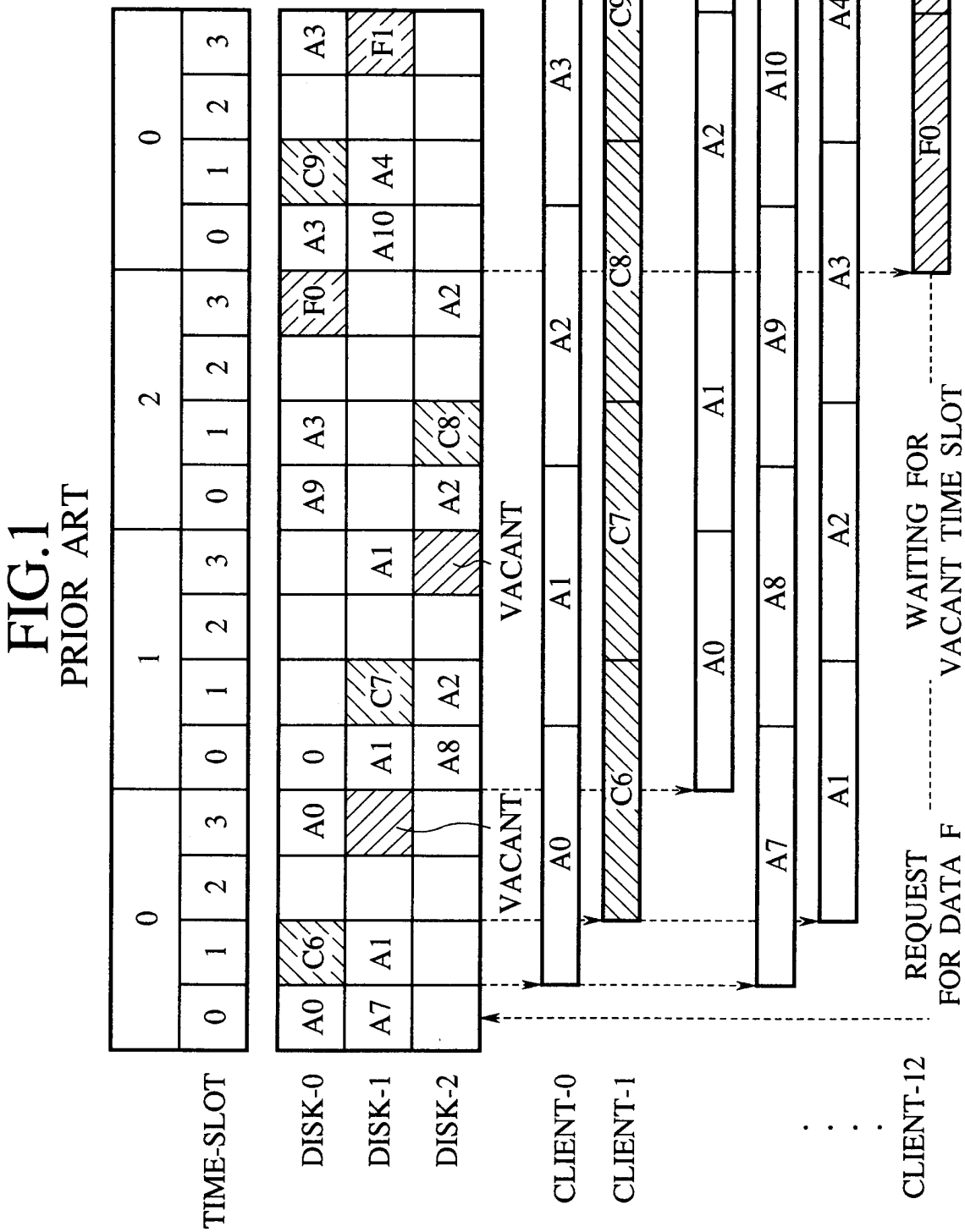


FIG.2
PRIOR ART

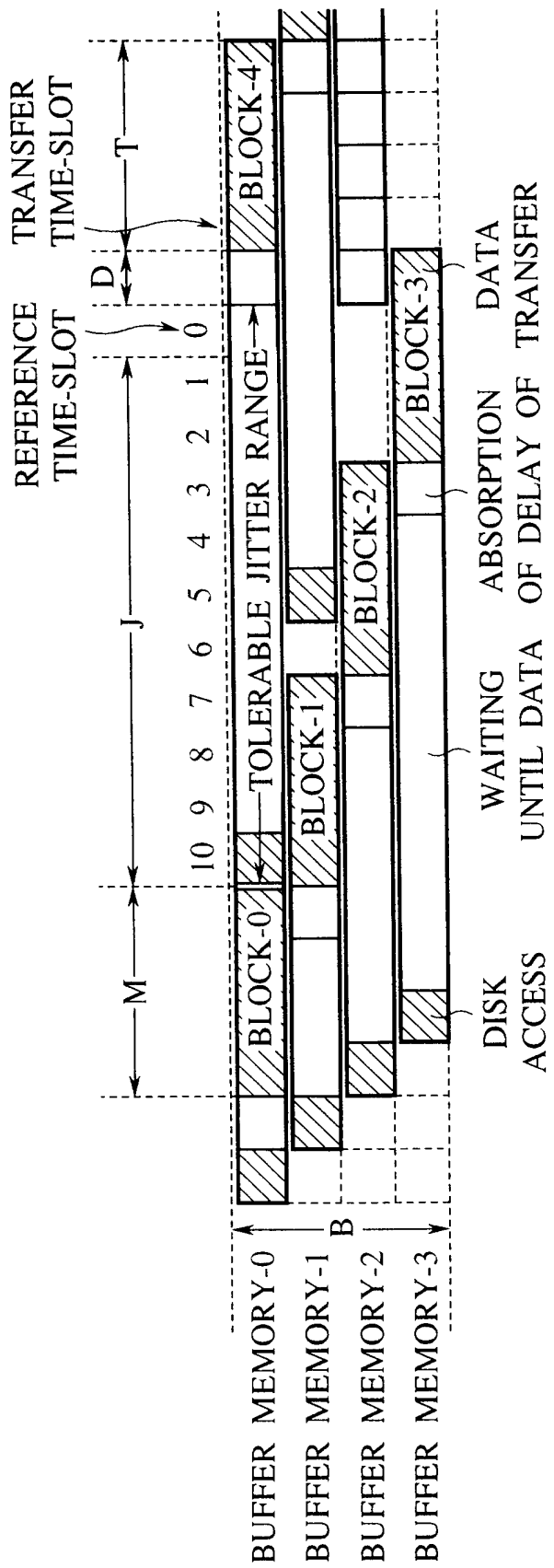


FIG.3

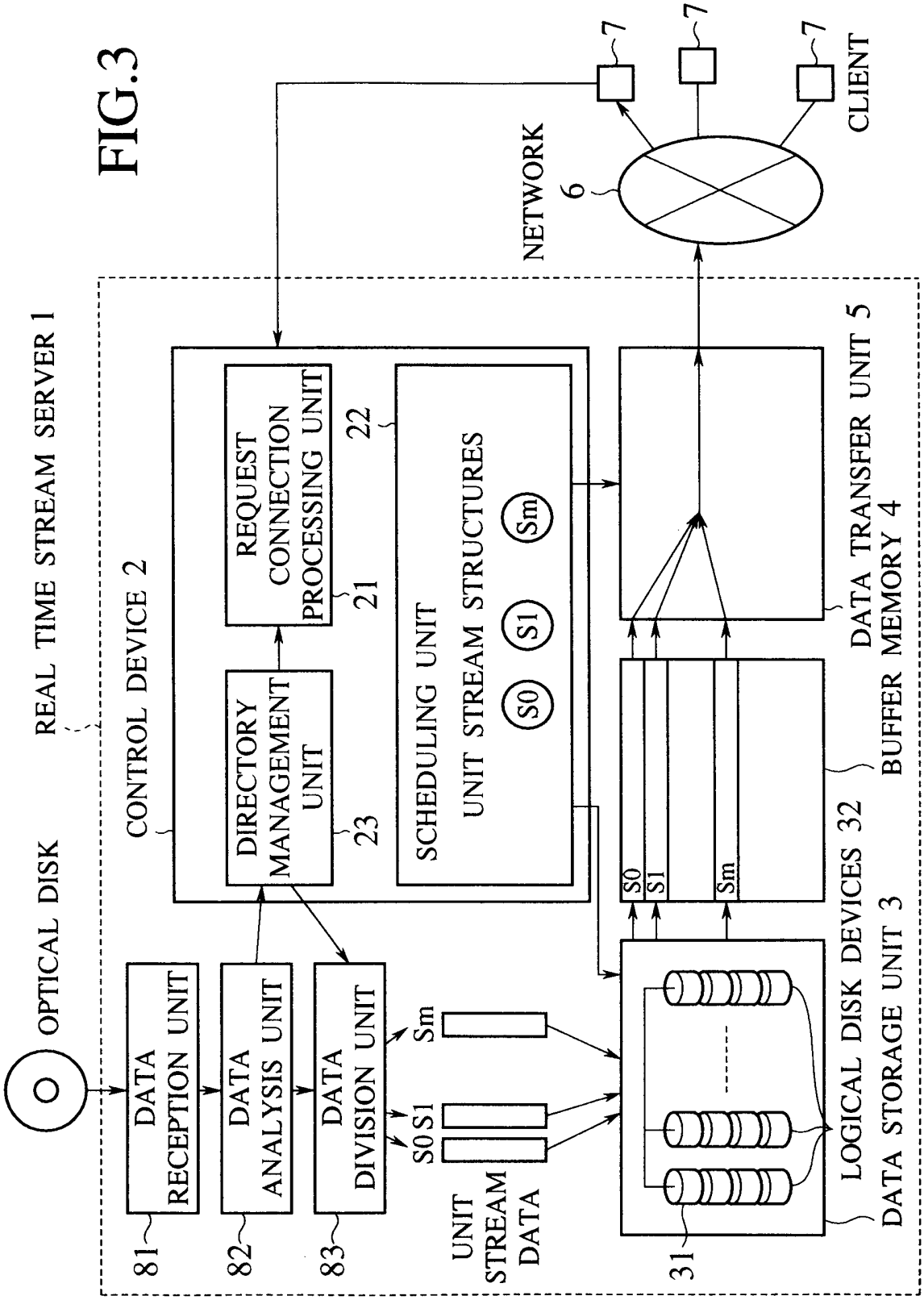


FIG.4

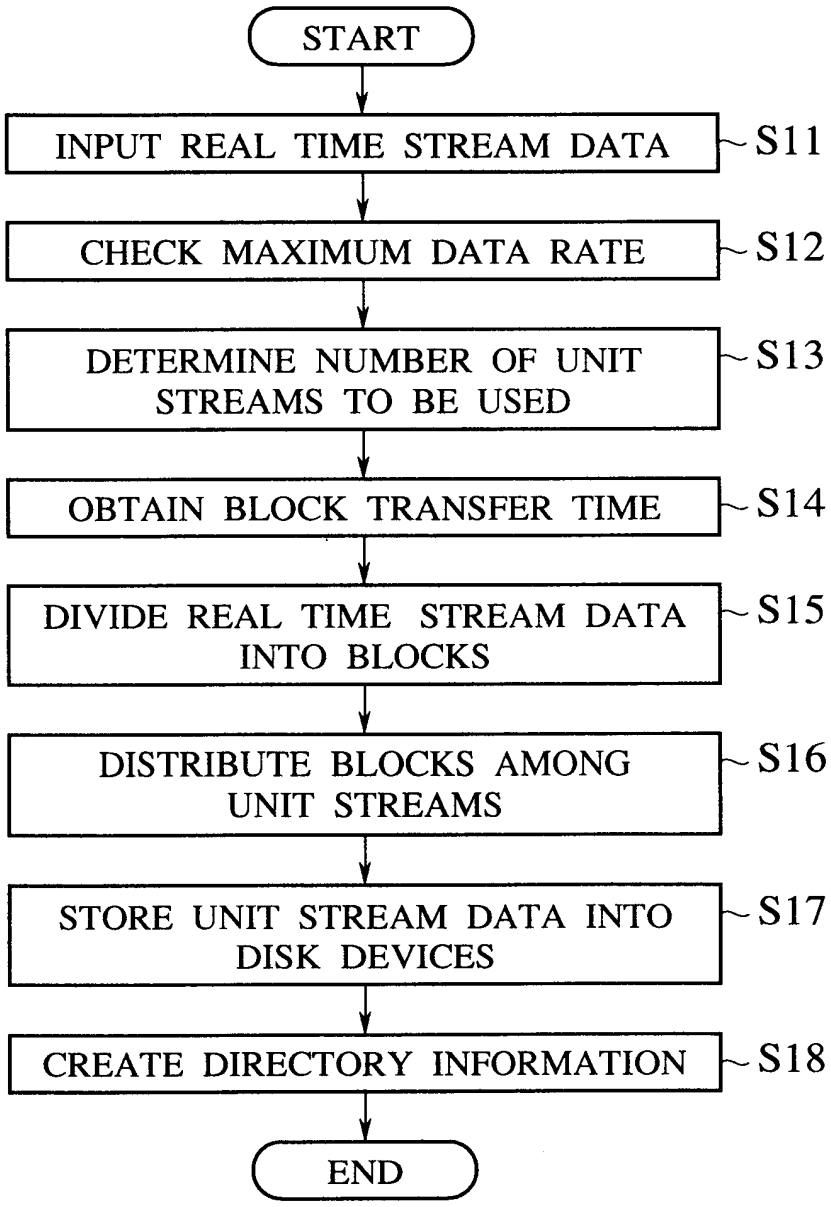


FIG.5

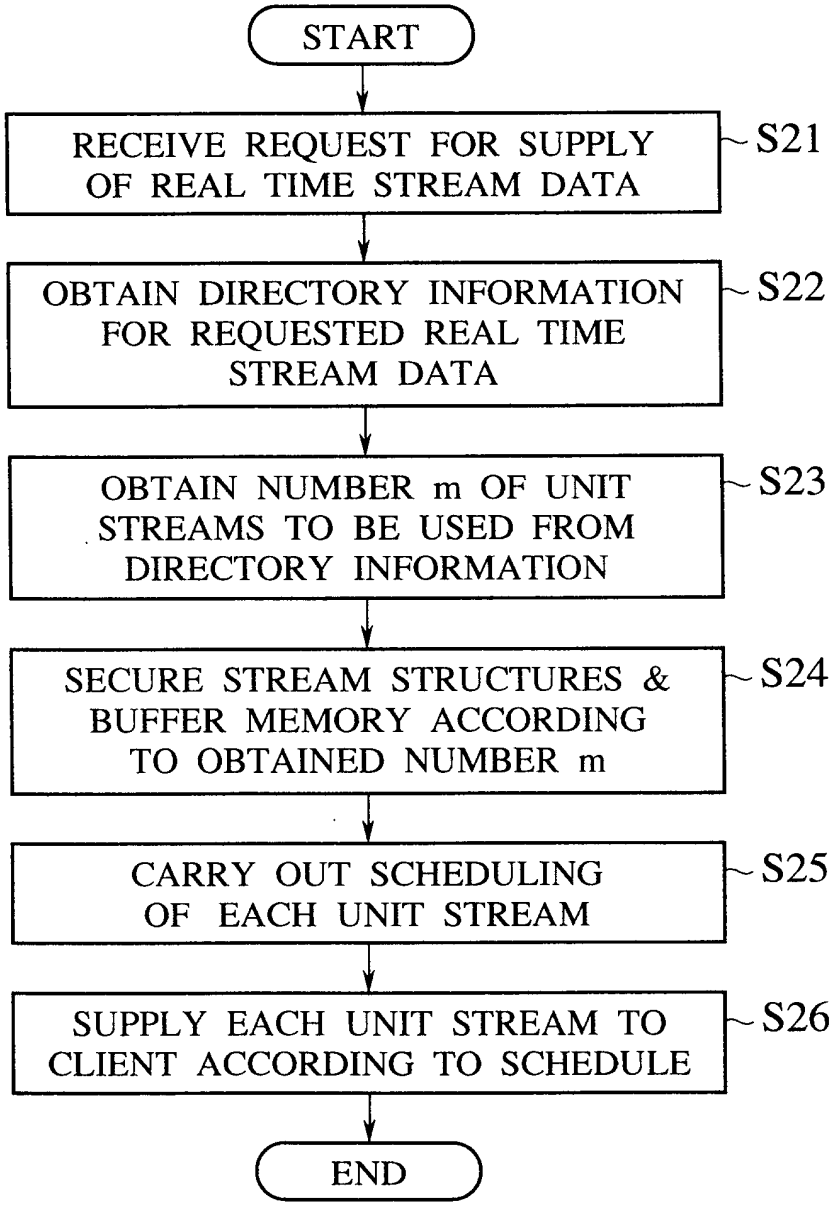


FIG.6

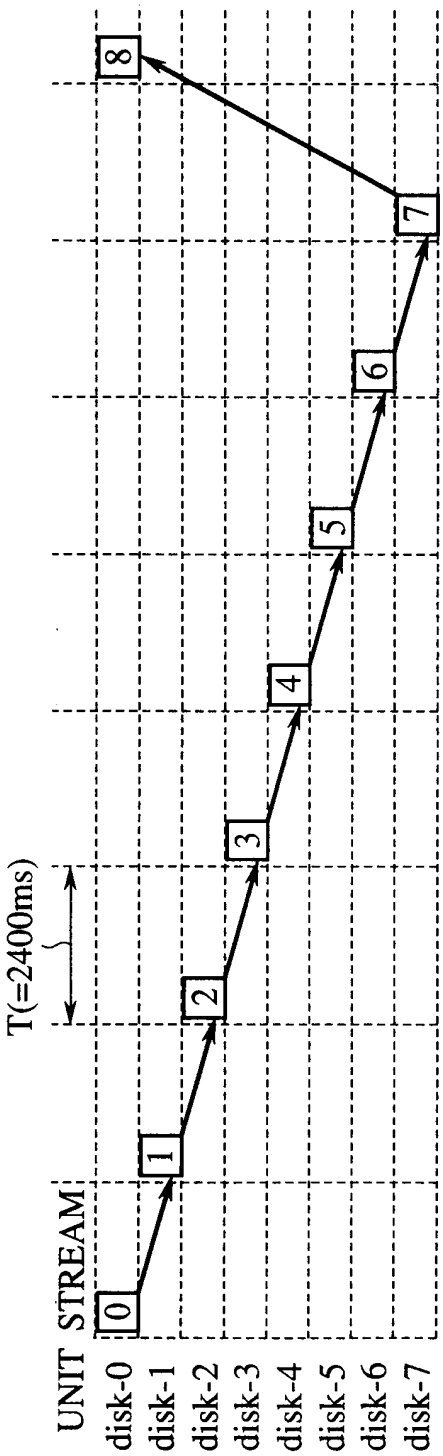


FIG.7

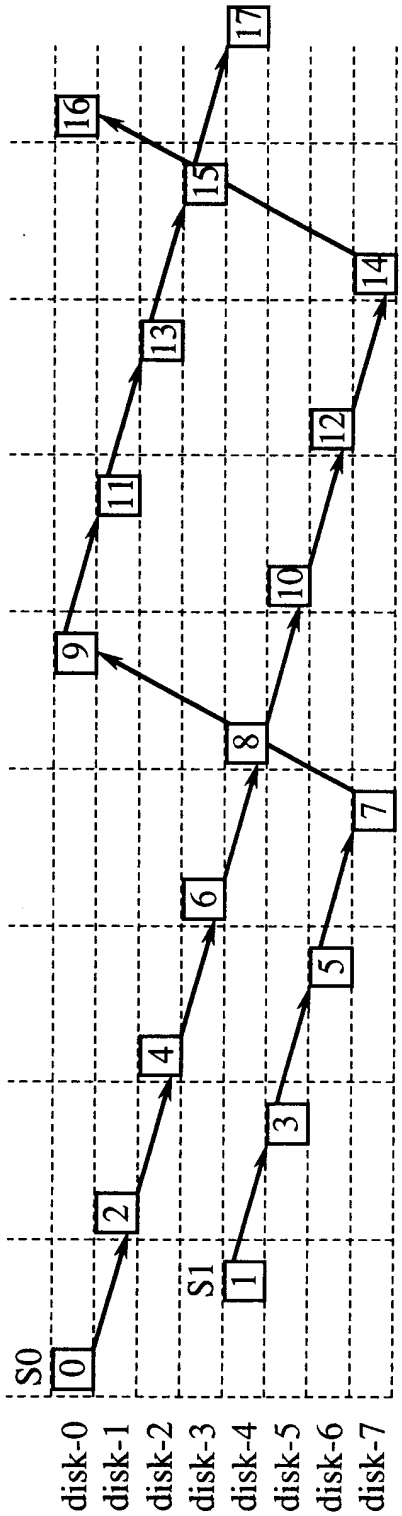


FIG.8

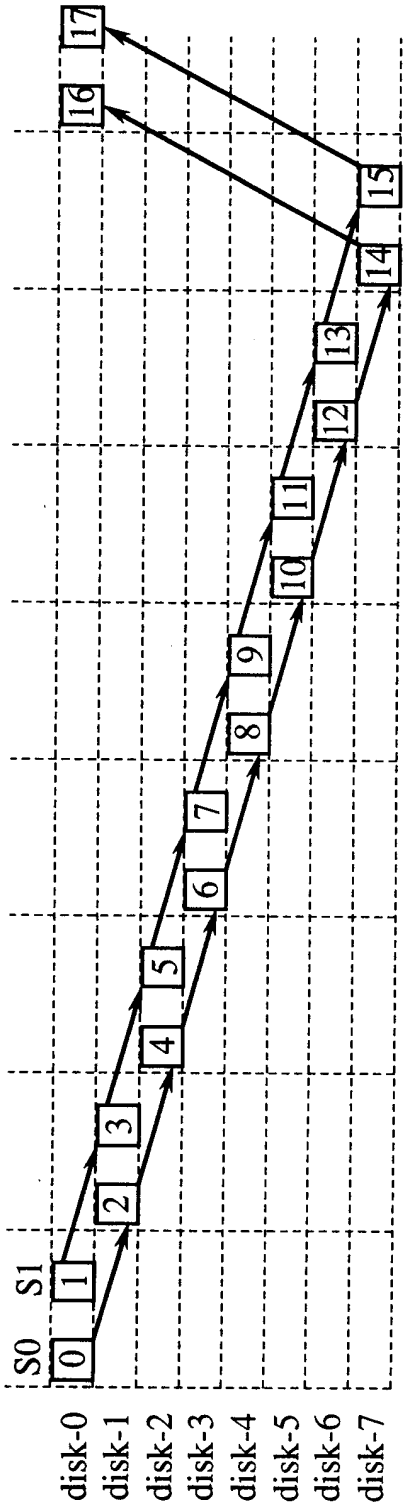


FIG.9

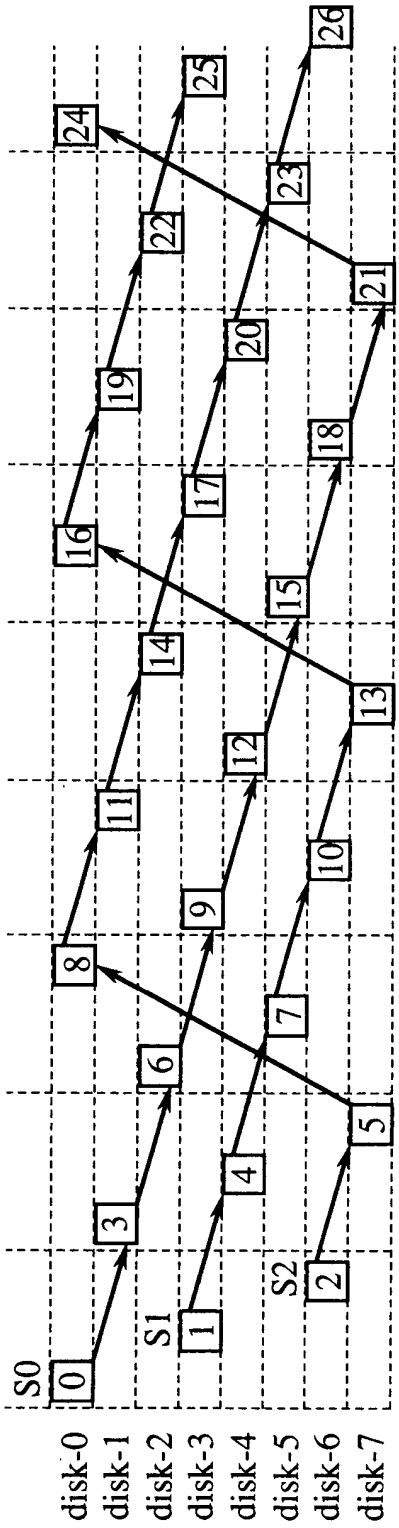
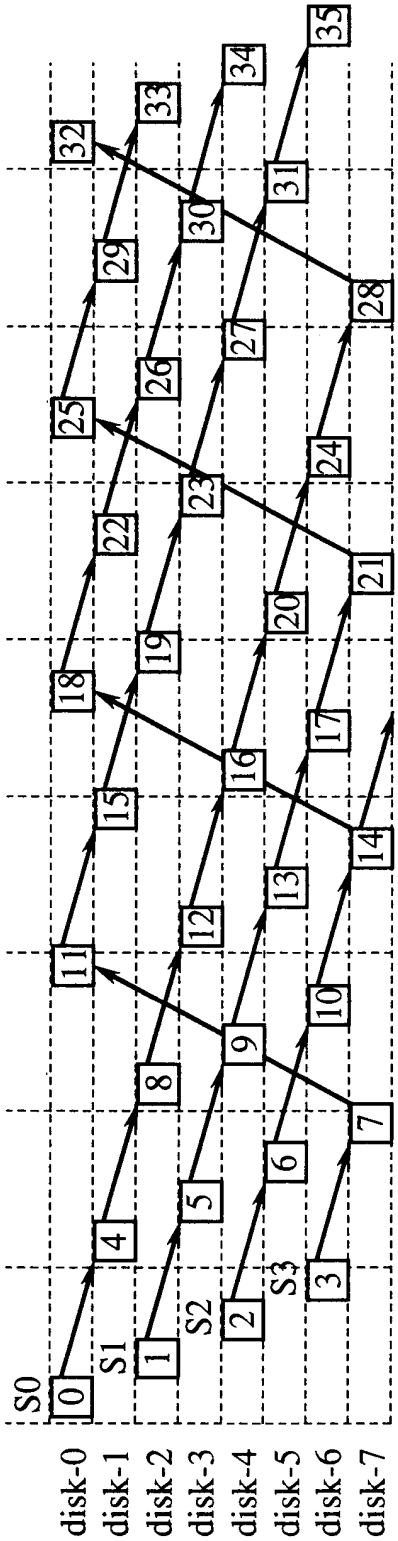


FIG.10



(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 817 017 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
07.01.1998 Bulletin 1998/02

(51) Int. Cl.[6]: **G06F 9/46**

(21) Application number: **97110675.2**

(22) Date of filing: **30.06.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: **03.07.1996 US 676859**

(71) Applicant:
SIEMENS AKTIENGESELLSCHAFT
80333 München (DE)

(72) Inventors:
- **Dorn, Karlheinz, Dipl.-Inf.**
  90562 Kalchreuth (DE)
- **Becker, Detlef, Dipl.-Ing.**
  91096 Möhrendorf (DE)
- **Quehl, Dietrich, Dipl.-Ing.**
  91052 Erlangen (DE)
- **Reinfelder, Hans-Erich, Dr.**
  91054 Erlangen (DE)

(54) Application program interface system

(57) An object oriented communication system supporting external data representation without an interface definition language, propagating events in both push and pull communication modes, fully distributing events, client/server-RPC-like mode and server processing pattern management. An applications program interface for the communication system having two macro routines for building classes which make the classes transferrable by the communication system.

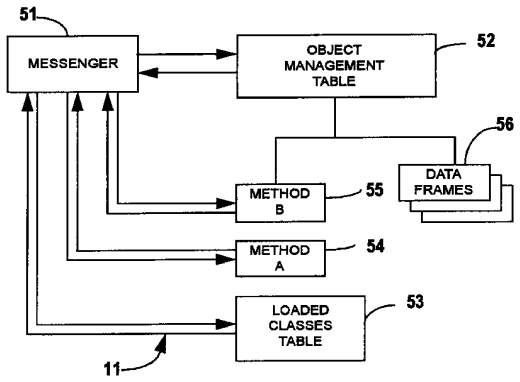


FIG 1

EP 0 817 017 A2

Description

BACKGROUND OF THE INVENTION

The present application is directed to application programmers interfaces (API) for programmer applications for communications systems.

As set forth in U.S. Patent No. 5,499,365, full incorporated herein by reference, object oriented programming systems and processes, also referred to as "object oriented computing environments," have been the subject of much investigation and interest. As is well known to those having skill in the art, object oriented programming systems are composed of a large number of "objects." An object is a data structure, also referred to as a "frame," and a set of operations or functions, also referred to as "methods," that can access that data structure. The frame may have "slots," each of which contains an "attribute" of the data in the slot. The attribute may be a primitive (such as an integer or string) or an object reference which is a pointer to another object. Objects having identical data structures and common behavior can be grouped together into, and collectively identified as a "class."

Each defined class of objects will usually be manifested in a number of "instances". Each instance contains the particular data structure for a particular example of the object. In an object oriented computing environment, the data is processed by requesting an object to perform one of its methods by sending the object a "message". The receiving object responds to the message by choosing the method that implements the message name, executing this method on the named instance, and returning control to the calling high level routine along with the results of the method. The relationships between classes, objects and instances traditionally have been established during "build time" or generation of the object oriented computing environment, i.e., prior to "run time" or execution of the object oriented computing environment.

In addition to the relationships between classes, objects and instances identified above, inheritance relationships also exist between two or more classes such that a first class may be considered a "parent" of a second class and the second class may be considered a "child" of the first class. In other words, the first class is an ancestor of the second class and the second class is a descendant of the first class, such that the second class (i.e., the descendant) is said to inherit from the first class (i.e., the ancestor) The data structure of the child class includes all of the attributes of the parent class.

Object oriented systems have heretofore recognized "versions" of objects. A version of an object is the same data as the object at a different point in time. For example, an object which relates to a "work in progress", is a separate version of the same object data which relates to a completed and approved work. Many applications also require historical records of data as it existed at various points in time. Thus, different versions of an object are required.

Two articles providing further general background are E.W. Dijkstra, The Structure of "THE" Multi programming System, Communications of the ACM, Vol. 11, No. 5, May 1968, pp. 341-346, and C.A.R. Hoare, Monitors: Operating Systems Structuring Concepts, Communications of the ACM, Vol. 17, No. 10, October, 1974, pp. 549-557, both of which are incorporated herein by reference. The earlier article describes methods for synchronizing using primitives and explains the use of semaphores while the latter article develops Brinch-Hansen's concept of a monitor as a method of structuring an operating system. In particular, the Hoare article introduces a form of synchronization for processes and describes a possible method of implementation in terms of semaphores and gives a proof rule as well as illustrative examples.

As set forth in the Hoare article, a primary aim of an operating system is to share a computer installation among many programs making unpredictable demands upon its resources. A primary task of the designer is, therefore, to design a resource allocation with scheduling algorithms for resources of various kinds (for example, main store, drum store, magnetic tape handlers, consoles). In order to simplify this task, the programmer tries to construct separate schedulers for each class of resources. Each scheduler then consists of a certain amount of local administrative data, together with some procedures and functions which are called by programs wishing to acquire and release resources. Such a collection of associated data and procedures is known as a monitor.

The adaptive communication environment (ACE) is an object-oriented type of network programming system developed by Douglas C. Schmidt, an Assistant Professor with the Department of Computer Science, School of Engineering and Applied Science, Washington University. ACE encapsulates user level units and WIN32 (Windows NT and Windows 95) OS mechanisms via type-secured, efficient and object-oriented interfaces:

- IPC mechanisms - Internet-domain and UNIX-domain sockets, TLI, Named pipes (for UNIX and Win 32) and STREAM pipes;
- Event multiplexing - via select() and poll() on UNIX and WaitForMultipleObjects on Win 32;
- Solaris threads, POSIX Pthreads, and Win 32 threads;
- Explicit dynamic linking facilities - e.g., dlopen/dlsym/dlclose on UNIX and LoadLibrary/GetProc on Win 32;
- Memory-mapped files;

- System VIPC - shared memory, semaphores, message queues; and
- Sun RPC (GNU rpc++).

In addition, ACE contains a number of higher-level class categories and network programming frameworks to integrate and enhance the lower-level C++ wrappers. The higher-level components in ACE support the dynamic configuration of concurrent network daemons composed of application services. ACE is currently being used in a number of commercial products including ATM signaling software products, PBX monitoring applications, network management and general gateway communication for mobile communications systems and enterprise-wide distributed medical systems. A wealth of information and documentation regarding ACE is available on the worldwide web at the following universal resource locator: http://www.cs.wustl.edu/...schmidt/ACE-overview.html.

The following abbreviations are or may be utilized in this application:

- Thread - a parallel execution unit within a process. A monitor synchronizes, by forced sequentialization, the parallel access of several simultaneously running Threads, which all call up functions of one object that are protected through a monitor.
- Synchronizations-Primitive - a means of the operating system for reciprocal justification of parallel activities.
- Semaphore - a Synchronizations-Primitive for parallel activities.
- Mutex - a special Synchronizations-Primitive for parallel activities, for mutual exclusion purposes, it includes a critical code range.
- Condition Queue - an event waiting queue for parallel activities referring to a certain condition.
- Gate Lock - a mutex of the monitor for each entry-function, for protection of an object, for allowing only one parallel activity at a time to use an Entry-Routine of the object.
- Long Term Scheduling - longtime delay of one parallel activity within a condition queue or event waiting queue for parallel activities.
- Broker - a distributor.

In addition, the following acronyms are or may be used herein:

AFM	Asynchronous Function Manager
SESAM	Service & Event Synchronous Asynchronous Manager
PAL	Programmable Area Logic
API	Application Programmers Interface
IDL	Interface Definition Language
ATOMIC	Asynchron Transport Optimizing observer-pattern-like system supporting several Modes (client/server - push/pull) for an IDL-less Communication subsystem, described herein
XDR	External Data Representation
I/O	Input/Output
IPC	Inter Process Communication
CSA	Common Software Architecture (a Siemens AG computing system convention)
SW	Software

SUMMARY OF THE INVENTION

The present invention provides a location and protocol transparent object oriented communication system that implicitly encodes and decodes transferred data, if connected peers reside on host with different internal data representation. In that regard, the invention provides an Asynchronous Transport Optimizing Observer- Pattern-Like system Supporting Several Modes for an Interface Definition Language- less Communication Subsystem (ATOMIC) as well as an application programming interface therefor. However, the data structure must be identical to that expected by the supplier.

In an embodiment, the invention provides an object oriented communication system on a computer platform, comprising:

means for supporting external data representation without an interface definition language; means for propagating events in both push and pull communication modes and selecting which mode is used for a given connection; means for distributing events; and means for server processing pattern management.

In an embodiment, the means for supporting external data representation without an interface definition language comprises means for implicitly coding and decoding transferred data.

In an embodiment, all communication end points that use the same address are logically connected.

In an embodiment, there is provided a hook routine which called at the supplier side before data is sent and a hook routine which is called before data is stored in a target object, both hook routines called with an environment string as an argument, both hook routines influencing data transfer.

In an embodiment, the invention further provides means for performing XDR encoding and decoding.

In an embodiment, the invention further provides a macro routine which makes a class accessible to a communication end-point.

In an embodiment, the macro routine makes the class accessible via the communication end point by declaring inserter and extractor operators of the communication systems internal encoder/decoder class as friends, and implementing short member functions and one member function pointer into the class.

In an embodiment, the invention further provide a macro routine which defines a subset of data members that are to be transferred and informs the underlying system as to how to deal with pointers and vectors.

In an embodiment, the macro routine has two arguments, a class name and a list of white space separated macro routines, one such macro routine for each transferrable data member.

In an embodiment, the invention provides a supplier class associated with a pattern string in order to transfer component classes to consumers associated with the same pattern string residing on a host.

In an embodiment, the supplier class is a template class and can only exist in conjunction with a concrete component class.

In an embodiment, the invention further provides a consumer class associated with a pattern string in order to receive component classes in PUSH mode or PULL mode from suppliers associated with the same pattern string residing on hosts.

In an embodiment, the consumer class is a template class and can only exist in conjunction with a concrete component class.

In an embodiment, the invention provides an object oriented communication system on a computer platform, comprising:

means for supporting external data representation without any interface definition language said means for supporting external data representation without an interface definition language comprises means for implicitly encoding and decoding transfer data; means for propagating events in both push and pull communication modes and selecting which mode is used for a given connection, including a hook routine called at the supplier side before data is sent and a hook routine called before data is stored in a target object, both hook routines called within an environment string as an argument; means for distributing events; and means for server processing pattern management, wherein all communication endpoints that use the same address are logically connected.

In an embodiment, the invention provides an object oriented communication system programmer interface on a computer platform, comprising: a first macro routine which makes a class accessible to a communication endpoint by declaring inserter and extractor operators of a communication systems internal encoder/decoder class as friends and implementing short member functions and one member function pointer into the class; and a second macro routine which defines a subset of data members that are to be transferred and informs the underlying system as to how to deal with pointers and vectors, a second macro routine having two arguments, a class name and a list of white space separated macro routines, one such white space separate macro routine for each transferrable data member.

In an embodiment, the invention provides a supplier class associated with a pattern string in order to transfer component classes to consumers associated with the same pattern string residing on a host.

In an embodiment, the supplier class is a template class and can only exist in conjunction with a concrete component class.

In an embodiment, the invention further provides a consumer class associated with a pattern string in order to receive component classes in PUSH mode or PULL mode from suppliers associated with the same pattern string residing on hosts.

In an embodiment, the consumer class is a template class and can only exist in conjunction with a concrete component class.

These and other features of the invention are discussed in greater detail below in the following detailed description of the presently preferred embodiments with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a hardware and software environment.

Figure 2 illustrates the main components of an object-oriented program from Figure 5 of U.S. Patent No.

5,313,629.

Figure 3 illustrates an example of an inheritance hierarchy to an object oriented platform.

Figure 4 illustrates use of a setValue() using event propogation or client/server communication without reply.

Figure 5 illustrates blocking of a setValue() using client/server communication with reply.

Figure 6 illustrates a nonblocking setValue() using client/server communication with reply - waitFor...().

Figure 7 illustrates a nonblocking setValue() using client/server communication with reply - callback.

Figure 8 illustrates blocking getValue() without dataChanged() enabled.

Figure 9 illustrates sending no reply with dataChanged() enabled.

Figure 10 illustrates sending reply without dataChanged() enabled.

Figure 11 illustrates sending reply with dataChanged() enabled.

Figure 12 illustrates a nonblocking getValue() using waitForMultipleObjects.

Figure 13 illustrates a nonblocking getValue() using call-back function.

Figure 14 illustrates dispatching dataChanged() to handle incoming data.

Figure 15 illustrates blocking a PULL mode getValue() using NOWAIT flag.

Figure 16 illustrates dispatching dataChanged() to handle pulled data.

COPENDING APPLICATIONS

The following commonly assigned copending applications are fully incorporated herein by reference:

Title	Application NUMBER	Filing Date	Attorney Docket No.
MONITOR SYSTEM FOR SYNCHRONIZATION OF THREADS WITHIN A SINGLE PROCESS			GR 96 P 3106 E
SERVICE AND EVENT SYNCHRONOUS/ASYN-CHRONOUS MANAGER			GR 96 P 3107 E
SOFTWARE ICS FOR HIGH LEVEL APPLICATION FRAMEWORKS			GR 96 P 3109 E

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENTS

As stated above, the present invention (ATOMIC) provides a communication system application programmers interface (API) as well as basic mechanisms of the system itself.

Again referring to U.S. Patent No. 5,499365, in an object oriented computing environment, work is accomplished by sending action request messages to an object which contains data. The object will perform a requested action on the data according to its predefined methods. Objects may be grouped into object classes which define the types and meanings of the data, and the action requests (messages) that the object will honor. The individual objects containing data are called instances of the class.

Object classes can be defined to be subclasses of other classes. Subclasses inherit all of the data characteristics and methods of the parent class. They can add additional data and methods and they can override or redefine any data elements or methods of the parent class. An object may be represented schematically, and generally is represented herein by a rectangle. The upper rectangle contains the data structure represented by a frame having slots, each of

which contains an attribute of the data in the slot. The lower rectangle indicates the object's methods which encapsulate the frame and which are used to perform actions on the data encapsulated in the frame of the upper rectangle.

Figures 1,2 and 3 are reproduced herein from U.S. Patent No. 5,499,365. The following description relating thereto is derived from that patent.

Referring now to Figure 1, a hardware and software environment in which the present invention operates will now be described. As shown in Figure 1, the present invention is a method and system within an object oriented computing environment 11 operating on one or more computer platforms 12. It will be understood by those having skill in the art that computer platform 12 typically includes computer hardware units 13 such as a central processing unit (CPU) 14, a main memory 15 and an input/output (I/O) interface 16, and may include peripheral components such as a display terminal 21, an input device 22 such as a keyboard or a mouse, nonvolatile data storage devices 23 such as magnetic or optical disks, printers 24 and other peripheral devices. Computer platform 12 also typically includes microinstruction codes 26 and an operating system 28.

As shown in Figure 1, object oriented computing environment 11 operates on computer platform 12. It will be understood by those having skill in the art that object oriented computing environment may operate across multiple computer platforms. Object oriented computing environment 11 is preferably written in the C++ computer programming language. The design and operation of computer platforms and object oriented computing environments including that of an object manager, are well known to those having skill in the art and are described, for example, in U.S. Patent No. 5,265,206 issued November 23, 1993 to Abraham, et al., entitled "A Messenger and Object Manager to Implement an Object Oriented Environment"; and U.S. Patent No. 5,161,225 to Abraham, et al., entitled "Persistent Stream for Processing Time Consuming and Reusable Queries in an Object Oriented Database Management System; U.S. Patent No. 5,151,987 to Abraham, et al., entitled "Recovery Objects in an Object Oriented Computing Environment"; and U.S. Patent No. 5,161,223 to Abraham, entitled "Resumeable Batch Query for Processing Time Consuming Queries in an Object Oriented Database Management System", the disclosures of which are hereby incorporated herein by reference, and in numerous textbooks such as Object Oriented Software Construction by Bertrand Meyer, published by Prentice Hall in 1988, the disclosure of which is incorporated herein by reference and the publication referred to above in the Background section.

Referring now to Figure 2, which is a reproduction of Figure 5 of U.S. Patent No. 5,313,629, the main components of an object oriented program (11, Figure 1) will be described. A detailed description of the design and operation of an object oriented program is provided in "Object Oriented Software Construction", by Bertrand Meyer, published by Prentice Hall in 1988, the disclosure of which is incorporated herein by reference.

Referring to Figure 2, a typical object oriented computing environment 11 includes three primary components: a Messenger 51, and Object Management Table 52 and a Loaded Classes Table 53. The Messenger 51 controls communication between calling and called messages, Object Management Table 52 and Loaded Classes Table 53. Object Management Table 52 contains a list of pointers to all active object instances. The Loaded Classes Table 53 contains a list of pointers to all methods of active object classes.

Operation of the Object Oriented Program 11 will now be described for the example illustrated in Figure 2, in which Methods A (block 54) of an object sends a message to method B (block 55) of an object. Method A sends a message to Method B by calling Messenger 51. The message contains (1) an object reference of the instance to receive the message, (2) the method the object instance is requested to perform on the data it encapsulates, and (3) any parameters needed by the receiving method. Messenger 51 obtains a pointer to the data frame 56 of the instance object specified by Method A, by searching Object Management Table 52 for the instance object. If the specified instance object cannot be found, Object Management Table 52 adds the instance object to the table and calls the instance to materialize its data from the database.

Once in the instance table, Object Management Table 52 returns the pointer to the materialized instance object.

Messenger 51 then obtains the address of Method B from the Loaded Classes Table 53. If the instance's class is not loaded, the Loaded Classes Table 53 will load it at this time to materialize its data. The Loaded Classes Table 53 searches for the specified method (Method B) and returns the address of the method to Messenger 51.

The Messenger 51 then calls Method B, passing it a system data area and the Parameters from the call made by Method A including the pointer. Method B accesses the data frame 56 using the pointer. Method B then returns control to the Messenger 51 which returns control to Method A.

Figure 3 illustrates an example of an inheritance hierarchy to an object oriented computing platform. As shown, three object classes are illustrated for "salesperson", "employee" and "person", where a salesperson is a "kind of" employee, which is a "kind of" person. In other words, salesperson is a subclass of employee and employee is the superclass of salesperson. Similarly, employee is the subclass of person and person is the superclass of employee. Each class shown includes three instances. B. Soutter, W. Tipp and B.G. Blue are salespersons. B. Abraham, K. Yates and R. Moore are employees. J. McEnroe, R. Nader and R. Reagan are persons. In other words, an instance is related to its class by an "is a" relation.

Each subclass "inherits" the frames and methods of its superclass. Thus, for example, a salesperson frame inherits

age and hire date objects from the employee's superclass as well as promote methods from the employee superclass. Salesperson also includes a unique quota attribute and a pay commission method. Each instance can access all methods and frames of its superclass, so that, for example, B.G Blue can be promoted.

The ATOMIC communication system is a location and protocol transparent object oriented communication system that implicitly encodes and decodes transferred data, if the connected peers reside on hosts with different internal data representation.

To that end, all communication endpoints (a/k/a peers) that use the same address - a character string (pattern) - are logically connected . The patterns are valid with the network segment the host is connected to. Different name spaces may be realized by using a name service for the pattern strings (e.g., by adding the host name and/or the process name to the pattern string).

The communication system provides one way communication between supplier and consumer peers.

COMMUNICATION MODES

The ATOMIC communication system supports two communication modes: an event propogation mode, which is preferred; and a classic client/server communication mode, which is known from RPC based communication toolkits.

Table 1 below summarizes a comparison of the event propogation and classic client/server communication modes

Event Propagation Modes m (suppliers) to n (consumers) Connections	PUSH Mode	Consumer Callback
	PULL Mode	Supplier Selection
Client/Server Communication Modes n (suppliers) to 1(consumer) Connections	Single Push without reply	
	Supplier with Callback and/or waitFor...() (... [REPLY [,Callback] [,SynchHandle]])	
	Consumer with Callback and/or waitFor...() (... [WAIT [,Callback] [,SynchHandle]])	

Table 1: Event Propogation and Client/Server Modes

Event Propogation Mode

In the event propogation mode one or more suppliers make events known to zero (0) or more consumers, which may be interested in this event by using the same pattern string as the supplier(s). Neither acknowledgments nor replies are supported in this mode because of the arbitrary number of consumers. This mode supports two transfer modes described next, the PUSH mode and the PULL mode.

The PUSH Mode

The PUSH mode - the most common event propogation mode - is a supplier triggered communication. The supplier of an event propagates an event that causes the delivery of a dataChanged() method (which is a callback function, action routine) - if enabled by the consumer - in the consumer context. It is up to the consumers whether to allow PUSH mode events to be queued (such that no event is lost) or not to be queued.

The Pull Mode

The PULL mode is a consumer triggered communication. The consumer fetches incoming events independently of the supplier's timing in propagating them. There is no queuing at the consumer's side because every consumer read request causes the communication system to get a copy of the latest version of the supplier's data. The internal handling depends on the queuing flag set (or not set) in the consumer's CsaRemote object. In case of queuing, a "getValue()" call blocks until the next data structure is provided by the supplier; if queuing is switched off, the "getValue()" call returns the contents of the last data structure that were sent by the selected supplier (if any, otherwise an error is reported).

To avoid multiple queries into the same receiver object as the result of a consumer read request, one supplier must be selected to get a unique object read.

Client/Server Communication Mode

The classic supplier triggered client/sever communication allows one or more clients (suppliers) to connect to one (and only one) server (consumer). This n-to-one relationship allows the server to send a reply back to the client on an incoming event (message), if this reply was requested by the client (supplier).

A significant add-on to the standard client/server communication, as known from, RPC is the consumer triggered client/server communication. Every event received at the consumer side is queued into the consumer's input queue and can be retrieved by calling the getValue() method (see description below) without getting any callback() routines dispatched. This feature allows the consumer to process a new event when appropriate without taking care of the restrictions that go along with asynchronous dispatching.

Location Transparency

The location of the communication partner (supplier as well as consumer) is fully transparent (i.e. as to whether it is located within the same process, on the same host, or on a remote host).

The ATOMIC communication system decides which protocol provides the best performance for the particular connection.

The user can specify a shared memory flag as an attribute to the constructors of the CsaConnectable (supplier) and CsaRemote (consumer) objects, and it is treated as a hint to the communication system.

ENVIRONMENT AND HOOKS

The Environment String

The ATOMIC communication system (Msc) transfers data together with additional header information containing the sender's peer address, the addressee's peer information, and an optional user specified environment string. The data type of the environment is defined in the header file CsaMscOptions.hh as follows:

```
const int theCsaMscEnvSize = 32;
typedef char CsaMscEnvType[theCsaMscEnvSize];
```

The environment string can be passed to sender/receiver methods (see CsaConnectable's setValue()/getValue() and CsaRemote's getValue()/dataChanged() descriptions below).

The semantics of the environment string are application specific and defined. The ATOMIC communication system passes the environment data without interpretation.

Adding and Removing Hook Routines

The ATOMIC communication system provides an interface to implement two hook routines, one at the supplier side that is called before the data are sent and one at the consumer side, that is called before the received data are stored in the target object. The hook routines are of type bool (i.e., boolean) and are called with one argument, the environment string. The hook routines are implemented once per process and are intended to be used by applications that modify/interpret implicitly the environment string (e.g. copy thread specific data into the environment string or store the environment string as thread specific data).

The value ("true" or "false") returned by the hook routines influences the data transfer. In that regard, the value "true" doesn't change the behavior while value "false":

- at the supplier side, aborts a setValue() call without sending the data to the consumer(s)

- at the consumer side, aborts a getValue() call without copying the data to the consumer(s) target object(s) and without dispatching/notifying the consumer.

These hook routines may be used for event filtering depending on (implicitly or explicitly specified) environment string contents.

The following sample code shows how the hook routines can be inserted, removed or changed. This sample code shows setting supplier and consumer hook routines :

```
// include options header file
#include        <CsaMscOptions.hh>
// the input (consumer) hook routine
static bool inHook (CsaMscEnvType & theEnv)  {
     return (true);
}
// an alternate input (consumer) hook routine


static book inHook2 (CsaMscEnvType & theEnv)  {
     return (true);
}
// the output (supplier) hook routine
static bool outHook (CsaMscEnvType & theEnv)  {
     return (true);
}
CsaMscOptions  theHooks = {inHook, outHook};
// set the hook routines
   CsaOsOptDb: :setOptions(CsaMscOptionName, (void  *)&
      theHook);
   // read the hook routines
   CsaOsptDb: :getOptions(CsaMscOptionName, (void  *) &
      theHooks);
   // modify and update the consumer hook routine
   theHooks.theInputHook = inHook2;
   CsaOsOptDb: :setOptions(CsaMscOptionName, (void  *) &
      theHooks);
```

Building Classes and Structures

Some goals for the design of a communication are:

- the communication should be protocol transparent,

- the communication should be location transparent,
- the communication should be able to transfer all generic data types supported by the compiler,
- the application programmer should not have to deal with data representation details such as XDR routines
- the communication systems restrictions to the class design should be as few as possible.

To achieve these goals, two macros, discussed below, are provided to the class designer, which macros make the class transferrable by the communication system. These macros are referred to herein as the IMPLEMENT_MSC and DECLARE_MSC macros. The class definition must be identical for both the supplier and the consumer. Therefore, the same header file is included by both communication endpoints; changes of the header file do not cause inconsistencies because they are not done in different files.

The XDR encoding/decoding is performed internally by a communication subsystem (the IMPLEMENT_MSC macro must be present and specify all data members to be transferred), if the corresponding communication endpoint is located on a host with different internal data representation (different processor architecture).

The short component class example below shows how to use these macros in nested classes (structures are handled identically to classes; the DECLARE_MSC macro is inserted in the public (default) section of the structure):

```
const int theFloatDimension = 333;
/ /  user class example 1
class XyzSimpleClass {
     public:
          XyzSiimpleClass () {}
          -XyzSimpleClass () {}
          DECLARE_MSC(XyzSimpleClass)
     protected:
          int            aIntVar;
          float          aFloatArray[theFloaDimension];
};
IMPLEMENT_MSC(SyzSimpleClass,  V(aIntVar)  V(aFloatArray) )
/ /user class example 2
class AbcWithPointers {
     public:
          AbcWithPointers(XyzSimpleClass *thePointer=0):
               myPointer (thePointer)
               {   dsblDataChanged () ;  }
          -AbcWithPointers () {}
          bool dataChanged (CsaMscRcvdFrom from_in,
                            CsaMscEnvType &theEnv)
          {   return (true)  ;  }
```

```
        DECLARE_MSC(AbcWithPointers)
    protected:
        double              myDoubleVar;
        XyzSimpleClass      mySimpleClass;
        XyzSimpleClass      *myPointer;
};
IMPLEMENT_MSC(AbcWithPointers,  V(myDoubleVar)
              V(mySimpleClass)  P(myPointer)
```

There is no restriction in the number or size of the data members that are to be transferred. Some compilers-preprocessors, however, limit the size of macro expansions.

The DECLARE_MSC macro

The DECLARE_MSC macro makes the class accessible by a communication endpoint (CsaConnectable = supplier or CsaRemote = consumer) by declaring the inserter/extractor operators of the communication system's internal encoder/decoder class as friends, and implementing a few very short member functions (the enable/disable dataChanged() method), and one member function pointer (the dataChanged() method itself) into the class.

The DECLARE_MSC macro must be added to the public section of the class as it inserts the member function pointer into the protected section of the class and the member functions into the public section of the class.

The IMPLEMENT_MSC Macro

The IMPLEMENT_MSC macro defines the subset of data members that are to be transferred and tells the underlying system how to deal with pointers/vectors.

The IMPLEMENT_MSC macro must be placed after the class definition (it implements the inserter/extractor operators of the communication system's internal encoder/decoder class.

The IMPLEMENT_MSC macro has two arguments - the class name and a list of white space separated macros; one macro for each transferrable data member. The V(datamember) macro tells the communication system to treat the variable in the argument as a scalar or vector that is to be transferred.

The P(datamember) macro tells the communication system to dereference the pointer specified in the argument and transfer the contents of the class/structure/variable the pointer points to.

It should be noted that:

- All variable specification macros (e.g., V() and P() ...) build a white space separated list.
- The user classes may be derived from other classes. The data members of the base class must be specified in the IMPLEMENT_MSC macro of the derived class.
- Classes may be nested (container classes).
- The transfer is restricted to data members (no VMT's ...).

CsaConnectable (the supplier)

A CsaConnectable is the supplier class associated with a pattern string in order to transfer component classes (specified as templates) to consumer(s) associated with the same pattern string residing on local or remote hosts.

The class CsaConnectable is a template class and therefore can only exist in conjunction with a concrete component class.

A more detailed interface description is provided below and sample code is provided under the heading "Examples."

The Constructor

The constructor takes two arguments:

- a pattern string which specifies the name of the communication endpoint,
- an attribute mask (local/shared memory has to be used for message buffering)

The CsaConnectable establishes the connection to the underlying basic communication system and allocates a generic SESAM (reference should be made to Application Attorney Docket No. GR 96 P 3107, incorporated herein by reference) slot for event notification.

Data Transfer

Data transfer is initialized by a call to member function setValue(). The user object specified in the argument list contains the data to be transferred.

In most cases, the event propagation mode will be used. In this mode, only one argument must be supplied - a reference to the user class object that contains the data to be transferred.

Both, the PUSH mode and the PULL mode interface do not differ from the suppliers point of view. Reference should be made to Figure 4 wherein setValue() using event propogation or client/server communication without reply is illustrated.

In case of client/server communication, some more information must be passed to setValue(). Because only one server can be connected to the supplier, one of the existing consumers must be selected as the server. This can be performed by calling getConsumers(), selecting the appropriate consumer and passing the consumer informations (class CsaMscPeerInfo) as an argument to setValue.

In the client/server mode, a reply from the server (consumer) might be expected. If the reply argument is specified, the call to setValue() blocks until the reply is received. Reference should be made to Figure 5 wherein blocking of setValue() using client/server communication with reply is illustrated.

The last data set transferred through the CsaConnectable can be reread via getValue(). The getValue(), unlike the getValue() method of CsaRemote, never blocks because the requested data are already present (or not; in this case an error status will be returned). Therefore, not asynchrony is provided in the CsaConnectable's getValue() interface.

If blocking calls (client/server mode only) to setValue() are not acceptable, the setValue() method can be performed in a separate thread. This is done by implicitly using the SESAM's dynamic slot mechanism. The synchronisation (again, reference should be made to Application Attorney Docket No. GR 96 P 3107 E for a more detailed description of these aspects of SESAM) can be realized in two different ways - waiting for a SynchHandle (returned by setValue()) (see Figure 6 illustrating nonblocking setValue() using client/server communication with reply - waitFor...() and/or getting a callback method (must be passed to setValue()) dispatched after completion of setValue() (see Figure 7 illustrating nonblocking setValue() using client/server communication with reply -callback.

It should be noted that asynchronous setValue() calls are only supported if a reply was requested.

In the constructor of CsaConnectable, a generic SESAM slot is allocated, and the SynchHandle associated with this slot is stored as a CsaConnectable's private data. This SynchHandle - in this context called notification handle - can be obtained by calling the method getNotificationHandle(). This handle can be used for example in watchdog threads that keep track of replies that are initiated by setValue() calls of other threads without knowledge of the setValue()'s arguments.

Data Processing

The data members of the user class object are copied by an i/o stream-like encoder/decoder into a message buffer, which is passed to the underlying communication system. The CsaConnectable holds always the latest message buffer for subsequent getValue() calls and to grant requests from a PULL mode consumer. There is no 1 to 1 relationship between this message buffer in the output queue and a specific user class object, if more than one object has been transferred through this CsaConnectable by one or more threads.

Design Restrictions

CsaConnectables (suppliers) may be located on a stack, allocated from a heap or stored in a global address space. CsaConnectables in shared memory are not supported.

There must not be classes derived from class CsaConnectable. Containment can be used instead.

There are no restrictions on the lifetime of the CsaConnectable.

CsaRemote (Consumer)

CsaRemote is the consumer class associated with a pattern string in order to receive component classes (specified as templates) in PUSH mode or PULL mode from supplier(s) associated with the same pattern string residing on local or remote hosts.

The class CsaRemote is a template class and therefore can only exist in conjunction with a concrete component class.

The Constructor

The constructor takes two arguments: a pattern string which specifies the name of the communication endpoint; and an attribute mask specifying:

a) whether a shared/local memory has to be used for message buffering,
b) whether or not an incoming message must be queued, and
c) the CsaRemote (consumer) to select the PUSH/PULL mode.

The CsaRemote establishes the connection to the underlying basic communication system and allocates a generic SESAM (see SESAM API description, copending Application Attorney Docket No. GR 96 P 3107 E) slot for event notification.

Data Transfer

For the consumer side there are two modes of operation, the event propagation containing the PUSH and PULL modes as well as the client/server communication (supplier and consumer triggered).

The supplier triggered modes - event propagation PUSH mode and the client/server mode - are very similar from the consumer's point of view; the only difference is the reply that will be returned to the supplier (if requested) in client/server mode. Common to both modes is the dispatching scheme and the blocking/nonblocking getValue() (receive) calls.

Consumer triggered mode - event propagation PULL mode - is different from the supplier triggered mode in copying the last data set (that will always be kept by the supplier) by every call to getValue() - regardless whether the supplier's data changed or didn't change between two calls to getValue().

Data Filter Method dataChanged

The DECLARE_MSC macro adds a data filtering and event dispatching mechanism to the user's component class.

The designer of the user component class can add a method (in this document always named dataChanged()) to his class, that can be enabled or disabled at runtime. This method is - if enabled - implicitly called after copying the received data into the target object - regardless whether the data are received by a synchronous/asynchronous call to getValue() or by enabling the dispatching with setCallbackObject(). In the latter case, the action routine that will get dispatched is the dataChanged() member function. There are two arguments passed to the dataChanged() method, a mask of type CsaMscRcvdFrom which specifies the location of the sender (same thread, same process but different thread, other process on same host or process on a remote host) (see SESAM API description) and the environment string.

In client/server mode, the return value of dataChanged() is returned to the supplier (client) together with the thread specific error object, if a reply was requested.

The great advantage of dispatching a member function of the user class is the accessibility of all data members by the dispatched function.

The dataChanged() method is enabled by invoking the user class method:

```
void enblDataChanged(
bool (userclass::*f)(CsaMscRcvdFrom,
                     CsaMscEnvType &
     )
);
```

The dataChanged() method is disabled by invoking the user class method void dsblDataChanged(void).

Both methods are implemented by the macro DECLARE_MSC.

It should be noted that the dataChanged() method always should explicitly disabled or enabled in the constructor of the user class to avoid uninitialized member function pointer. Toggling between enabled and disabled state is possible at runtime.

Supplier Triggered Event Processing

The most simple case is just calling getValue() with one argument, the reference to a user object as the receiver buffer without enabling the dataChanged() method.

The object getValue() blocks until data are available for reading. Reference should be made to Figure 8 which illustrates blocking getValue without dataChanged enabled.

If the dataChanged() method is implemented and enabled, it is invoked after reading the incoming data and before returning to the caller of getValue(). Reference should be made to Figure 9 which illustrates blocking getValue with dataChanged enabled.

In client/server mode, the supplier may request a reply. If no dataChanged() method is implemented and enabled, the reply will be delivered with and have the status of "success" after copying the incoming data into the target object. In that regard, reference should be made to Figure 10 which illustrates sending a reply without dataChanged() enabled. If the dataChanged() method is implemented and enabled, the reply will be delivered after return from dataChanged() passing the return status and, if dataChanged() returned 'false', the thread specific error object back to the supplier. In that regard, reference should be made to Figure 11 which illustrates sending a reply with dataChanged() enabled.

As described for CsaConnectable, the blocking invocation can be performed in a separate thread implicitly using SESAM's dynamic slot mechanism. The synchronization (see detailed description in commonly assigned and copending application Attorney Docket No. GR 96 P 3107 E) can be realized in two different ways - waiting for a SynchHandle (returned by getValue()) (see Figure 12 which illustrates nonblocking getValue() using waitForMultipleObjects) and/or getting a call-back method (must also be passed to setValue ()) dispatched after completion of setValue() (see Figure 13 which illustrates nonblocking getValue() using a callback function).

Many applications are event driven or have more than one input event to wait for. These applications cannot block in a single getValue(); they need to get dispatched after arrival of data in one or more CsaRemote objects. This applications can declare an object as the receiver object for the specified CsaRemote object using the dataChanged() method as the callback method. The dataChanged() method is dispatched from the main dispatcher as long as the input queue contains unread data, similar to the RPC action routine (see Figure 14 which illustrates dispatching dataChanged() to handle incoming data). At invocation time of dataChanged() the data are already stored in the specified object. Reply handling is similar as described for getValue() calls with dataChanged() enabled.

It should be noted that the dataChanged() method must be enabled before invoking setCallbackObject().

After enabling the dataChanged() method as the dispatcher for incoming events, no further getValue() calls are possible for this CsaRemote object.

In some cases it may be of interest to be notified every time data on one or more CsaRemote objects arrive. The application process then would call the method waitForMultipleObjects() on the notification handle(s) of the CsaRemote object(s) of interest and invoke for every signaled CsaRemote object the getValue() method with the flag "NOWAIT", as long as data are available.

Consumer Triggered Event Processing

In event propagation PULL mode, the consumer triggers the receiving of messages from supplier(s). To get only one data set for the pull request, one specific supplier must be selected. The selection is done by calling the method getSuppliers(), selecting one of the suppliers and calling the method getValue() for the selected supplier.

The CsaRemote class provides two different ways of pulling data from the consumer - request a data set regardless if it was yet read by a previous call to getValue() (by calling getValue() with the flag "NOWAIT") or request a new version of the data set (by calling getValue() with the flag "WAIT" which means wait for a new update by the supplier) (see Figure 15 which illustrates blocking PULL mode getValue() using the NOWAIT flag).

In the latter case the request for a new update is queued at the supplier's CsaConnectable until the next setValue(). This set Value() causes all queued requests to be granted, regardless if they are queued by one or more CsaRemotes (i.e. if more than one request from one CsaRemote is pending at the same CsaConnectable, the setValue() method grants all requests!).

The asynchronous functionality - passing the getValue() invocation to SESAM's dynamic slots and waiting for completion and/or forcing a callback function to be dispatched, respectively - is similar to that of the PUSH mode1.

The dispatching of the dataChanged() method enabled by a previous call to setCallbackObject() is slightly different by means of initiator of the callback. In PULL mode the dataChanged is dispatched due to the supplier's response on a consumer's getvalue() call (see Figure 13 which illustrates dispatching datachanged() to handle pulled data).

Replies in Client/Server mode

As described above, replies are possible in client/server mode only. For the processing of replies, see Table 2 below. In Table 2, the entry of an "X" means "not of concern."

PULL mode	QUEUED	setCallback -Object()	Reply - Behavior
NO	NO	NO	Implicit reply after the message is stored in the input message queue. The input queue has the length of 1 message.
NO	NO	YES[b]	see above
NO	YES[c]	NO [1)]	Get_value() calls the dataChanged() method, wich returns an error status passed as a reply status to the supplier. Each message can trigger one reply.
NO	YES [3)]	YES [2)]	The return status of the dataChanged() method is passed to the supplier of the message as a reply status (implicitly)
YES	X	X	No reply possible

Notes: (1) Consumer triggered event event processing.
(2) Supplier triggered event event processing.
(3) If the queue is full, the supplier will block until the consumer dequeues at least one event.

Table 2: Reply Behavior

Data Processing

All incoming data are queued into the input queue of the consumer. In the case of PUSH mode consumers that specify the attribute "NOTQUEUED" to the constructor, the input queue has a maximum length of 1 message buffer, which will be overwritten by a new incoming event.

The data members of the user class object are copied by an i/o stream-like encoder/decoder from a message buffer, which is queued to the input queue of the CsaRemote, to the user class object.

There is no 1 to 1 relationship between this message buffer in the output queue and a user class object, if more than one object has been transferred through this CsaRemote.

Design Restrictions

CsaRemote objects (consumers) may be located on a stack, allocated from a heap or stored in a global address space. CsaRemotes in shared memory are not supported.

There must be no classes derived from class CsaRemote. Instead, one must use containment.

There are no restrictions on the lifetime of a CsaRemote object.

The user class object's lifetime must not be less than the lifetime of the CsaRemote.

In summary, the principal new approach of the invention is the novel and inventive combination of all the following features within a single homogenous package:

object oriented
supports external data representation without the need of an Interface Definition Language
Event-Propagation for Push&Pull-Modes
Client-Server Communication with reply
full asynchronous Support
multithreaded and multithreadsafe
Layering between Application-View and Implementation-View
transparency of locations and protocols and according optimizations
use of a server process for pattern-management only in the registration phase, but never in the Transport phase
a fully distributed (with local optimizations) event propagation mechanism, so no further event propagation mechanism is necessary throughout a software system.

Examples

The following examples illustrate typical usages of CsaConnectable (supplier) and CsaRemote (consumer) objects. Both, the supplier and the consumer, use the same header file with class definitions.

The Header File

```
const int theFloatDimension = 333;
// user class example 1
class XyzSimpleClass {
public:
     XyzSimpleClass(){}
     ~XyzSimpleClass(){}
     DECLARE_MSC(XyzSimpleClass)
protected:
     int       aIntVar;
     float     aFloatArray[theFloatDimension];
};
IMPLEMENT_MSC(XyzSimpleClass, V(aIntVar) V(aFloatArray))
//user class example 2
class AbcWithPointers {
     public:
AbcWithPointers(XyzSimpleClass *thePointer=0):
                    myPointer(thePointer)
     { dsblDataChanged(); } ~AbcWithPointers(){}
bool dataChanged(CsaMscRcvdFrom from_in,
                         CsaMscEnvType &theEnv)
{ return(true); } DECLARE_MSC(AbcWithPointers)
     protected:
          double    myDoubleVar;
XyzSimpleClass  mySimpleClass; XyzSimpleClass  *myPointer;
};
IMPLEMENT_MSC(AbcWithPointers, V(myDoubleVar)
V(mySimpleClass)
P(myPointer))
```

The Supplier Program

```
#include  <CsaConnectable.hh>   // communication classes
#include  <user.hh>             // user class(es)
// Callback function that notifies the completion of a
// blocking call to setValue() with reply
void * callbackFunc(void *){
          return ((void *) 0);
}
/*
 * The main program
 */
main(int argc, char **argv)
{
XyzSimpleClass   sc1;    // a simple user class
    AbcWithPointers wp1(&sc1); // a
container user class CsaMscPeerInfo peers;  // information
    about consumers
      bool     status;   // return status for method calls
CsaSesam::SynchHandleType Synch; // SESAM's synchronization
                                                  // handle
// Event Propagation (PUSH mode)
CsaConnectable <AbcWithPointers> con1 ("push_mode_conn");
status = con1.setValue(wp1);
// Event Propagation (PULL mode)
CsaConnectable <AbcWithPointers> con2 ("pull_mode_conn");
status = con2.setValue(wp1);
// Client/Server mode (no reply, synchronous completion) Csa-
Connectable
<AbcWithPointers> con3 ("clsv_mode_conn"); status =
con3.getConsumers(&peers);
for ( peers.reset(); peers++ ; ) {
  // ... select appropriate consumer
  break;
}
status = con3.setValue(wp1, &peers);
// Client/Server mode (reply, synchronous completion)
```

```
status = con3.setValue(wp1, &peers, CsaMscPeer::Reply);
// Client/Server mode (reply, callback function)
status = con3.setValue(wp1, &peers, CsaMscPeer::Reply,
                                    callbackFunc);
// Client/Server mode (reply, wait for completion)
status = con3.setValue(wp1, &peers, CsaMscPeer::Reply,
                                    0, &Synch);
// some code ...
// AFM's WaitForMultipleObjects(1,&Synch,LOG_AND,60000);
return 0;
}
```

The Consumer Program

```
#include  <CsaRemote.hh>    // communication classes
          #include<user.hh>
     // user class(es)

// A allback function that notifies the asynchronous comple-
tion
// of a call to getValue().
void * callbackFunc(void *){
  return ((void *) 0);
}
/*
 * The main program
 */
main(int argc, char **argv)
{
CsaMscPeerInfo   peers;     // information about consumers
CsaSesam::SynchHandleType Synch; // SESAM's synchronization
                                                 // handle
bool      status;   // return status of method calls
/*
* Event Propagation (PULL mode)
*/
```

```
CsaRemote <AbcWithPointers> rem1("pull_mode_conn",
                                    CsaMscPeer::PullMode);
XyzSimpleClass    sc1;   // a simple user class AbcWith-
    Pointers      wp1(&sc1); // a
container class
// first select a supplier
status = rem1.getSuppliers(&peers);
for ( peers.reset(); peers++ ; ) {
  // ... select appropriate supplier
  break;
}
// enable the dataChanged method
wp1.enblDataChanged(AbcWithPointers::dataChanged);
// get data using synchronous getValue() call
while (1) {
  status = rem1.getValue(wp1, &peers);
  // ... do something
}
// get data using asynchronous getValue() call
status = rem1.getValue(wp1, &peers, CsaMscPeer::Wait,
                                        callbackFunc);
while (1) {
status = rem1.getValue(wp1, &peers, CsaMscPeer::Wait,
                                        0, &Synch);
     // ... do something
     // SESAM's  WaitForMultipleOb-
jects(1,&Synch,LOG_AND,60000);
}
     /*
     * Event propagation - PUSH mode1
     */
CsaRemote <AbcWithPointers> rem2("pull_mode_conn",
     CsaMscPeer::PushMode);
XyzSimpleClass sc2; // a simple user class AbcWithPointers
     wp2(&sc2); // a
container user class
// Enable the dataChanged method
```

```
wp2.enblDataChanged(AbcWithPointers::dataChanged);
// First get some data using synchronous getValue() call
status = rem2.getValue(wp2);
// Now let dataChanged method be dispatched on every
// incoming event. From now on every getValue() call
// on this Remote will be rejected.
status = rem2.setCallbackObject(wp2);
// call an appropriate main loop
*/
  * Client/server communication
  */
CsaRemote <AbcWithPointers> rem3("clsv_mode_conn",
                                 CsaMscPeer::PushMode);
XyzSimpleClass   sc3;    // a simple user class
AbcWithPointers     wp3(&sc3); // a container user class
// Enable the dataChanged method
wp3.enblDataChanged(AbcWithPointers::dataChanged);
// First get some data using synchronous getValue() call
status = rem3.getValue(wp3);
// Now let dataChanged method be dispatched on every
// incoming event. From now on every getValue() call
// on this Remote will be rejected.
status = rem3.setCallbackObject(wp3);
// call an appropriate main loop
}
```

Although modifications and changes may be suggested by those skilled in the art, it is the intention of the inventors to embody within the patent warranted hereon all changes and modifications as reasonably and properly come within the scope of their contribution to the art.

Claims

1. An object oriented communication system on a computer platform, comprising:

   means for supporting external data representation without an interface definition language;
   means for propagating events in both push and pull communication modes and selecting which mode is used for a given connection;
   means for distributing events; and
   means for server processing pattern management.

2. The object oriented communication system according to claim 1, wherein the means for supporting external data representation without an interface definition language comprises means for implicitly coding and decoding trans-

ferred data.

3. The object oriented communication system according to claim 1 or 2, wherein all communication end points that use the same address are logically connected.

4. The object oriented communication system according to claim 1 to 3, wherein there is provided a hook routine which called at the supplier side before data is sent and a hook routine which is called before data is stored in a target object, both hook routines called with an environment string as an argument, both hook routines influencing data transfer.

5. The object oriented computing system programmer interface according to claim 1 to 4, further comprising means for performing XDR encoding and decoding.

6. The object oriented communication system according to claim 1 to 5, further comprising a macro routine which makes a class accessible to a communication endpoint.

7. The object oriented communication system according to claim 6, wherein the macro routine makes the class accessible via the communication end point by declaring inserter and extractor operators of the communication systems internal encoder/decoder class as friends, and implementing short member functions and one member function pointer into the class.

8. The object oriented communication system according to claim 1 to 7, further comprising a macro routine which defines a subset of data members that are to be transferred and informs the underlying system as to how to deal with pointers and vectors.

9. The object oriented communication system according to claim 8, wherein the macro routine has two arguments, a class name and a list of white space separated macro routines, one such macro routine for each transferrable data member.

10. The object oriented communication system according to claim 1 to 9, comprising a supplier class associated with a pattern string in order to transfer component classes to consumers associated with the same pattern string residing on a host.

11. The object oriented communication system according to claim 10, wherein the supplier class is a template class and can only exist in conjunction with a concrete component class.

12. The object oriented communication system according to claim 1 to 11, further comprising a consumer class associated with a pattern string in order to receive component classes in PUSH mode or PULL mode from suppliers associated with the same pattern string residing on hosts.

13. The object oriented communication system according to claim 12, wherein the consumer class is a template class and can only exist in conjunction with a concrete component class.

14. An object oriented communication system programmer interface on a computer platform, comprising:

a first macro routine which makes a class accessible to a communication endpoint by declaring inserter and extractor operators of a communication systems internal encoder/decoder class as friends and implementing short member functions and one member function pointer into the class; and
a second macro routine which defines a subset of data members that are to be transferred and informs the underlying system as to how to deal with pointers and vectors, a second macro routine having two arguments, a class name and a list of white space separated macro routines, one such white space separate macro routine for each transferrable data member.

15. The object oriented communication system programmer interface according to claim 14, comprising a supplier class associated with a pattern string in order to transfer component classes to consumers associated with the same pattern string residing on a host.

16. The object oriented communication system programmer interface according to claim 14 or 15, wherein the supplier

class is a template class and can only exist in conjunction with a concrete component class.

17. The object oriented communication system programmer interface according to claim 14 to 16, further comprising a consumer class associated with a pattern string in order to receive component classes in PUSH mode or PULL mode from suppliers associated with the same pattern string residing on hosts.

18. The object oriented communication system programmer interface according to claim 14 to 17, wherein the consumer class is a template class and can only exist in conjunction with a concrete component class.

19. A storage medium including object oriented code for an object oriented communication system on a computer platform, comprising:

means for supporting external data representation without an interface definition language;
means for propagating events in both push and pull communication modes and selecting which mode is used for a given connection;
means for distributing events; and
means for server processing pattern management.

20. The storage medium according to claim 19, wherein the means for supporting external data representation without an interface definition language comprises means for implicitly coding and decoding transferred data.

21. The storage medium according to claim 19 or 20, wherein all communication end points that use the same address are logically connected.

22. The storage medium according to claim 19 to 21, wherein there is provided a hook routine which called at the supplier side before data is sent and a hook routine which is called before data is stored in a target object, both hook routines called with an environment string as an argument, both hook routines influencing data transfer.

23. The storage medium according to claim 19 to 22, further comprising means for performing XDR encoding and decoding.

24. The storage medium according to claim 19 to 23, further comprising a macro routine which makes a class accessible to a communication endpoint.

25. The storage medium according to claim 24, wherein the macro routine makes the class accessible via the communication end point by declaring inserter and extractor operators of the communication systems internal encoder/decoder class as friends, and implementing short member functions and one member function pointer into the class.

26. The storage medium according to claim 19 to 25, further comprising a macro routine which defines a subset of data members that are to be transferred and informs the underlying system as to how to deal with pointers and vectors.

27. The storage medium according to claim 26, wherein the macro routine has two arguments, a class name and a list of white space separated macro routines, one such macro routine for each transferrable data member.

28. The storage medium according to claim 19 to 27, comprising a supplier class associated with a pattern string in order to transfer component classes to consumers associated with the same pattern string residing on a host.

29. The storage medium according to claim 28, wherein the supplier class is a template class and can only exist in conjunction with a concrete component class.

30. The storage medium according to claim 19 to 29, further comprising a consumer class associated with a pattern string in order to receive component classes in PUSH mode or PULL mode from suppliers associated with the same pattern string residing on hosts.

31. The storage medium according to claim 30, wherein the consumer class is a template class and can only exist in conjunction with a concrete component class.

32. A storage medium including object oriented code for an object oriented communication system on a computer platform, comprising:

a first macro routine which makes a class accessible to a communication endpoint by declaring inserter and extractor operators of a communication systems internal encoder/decoder class as friends and implementing short member functions and one member function pointer into the class; and
a second macro routine which defines a subset of data members that are to be transferred and informs the underlying system as to how to deal with pointers and vectors, a second macro routine having two arguments, a class name and a list of white space separated macro routines, one such white space separate macro routine for each transferrable data member.

33. The object oriented communication system programmer interface according to claim 32, comprising a supplier class associated with a pattern string in order to transfer component classes to consumers associated with the same pattern string residing on a host.

34. The storage medium according to claim 33, wherein the supplier class is a template class and can only exist in conjunction with a concrete component class.

35. The storage medium according to claim 33 or 34, further comprising a consumer class associated with a pattern string in order to receive component classes in PUSH mode or PULL mode from suppliers associated with the same pattern string residing on hosts.

36. The storage medium according to claim 33 to 35, wherein the consumer class is a template class and can only exist in conjunction with a concrete component class.

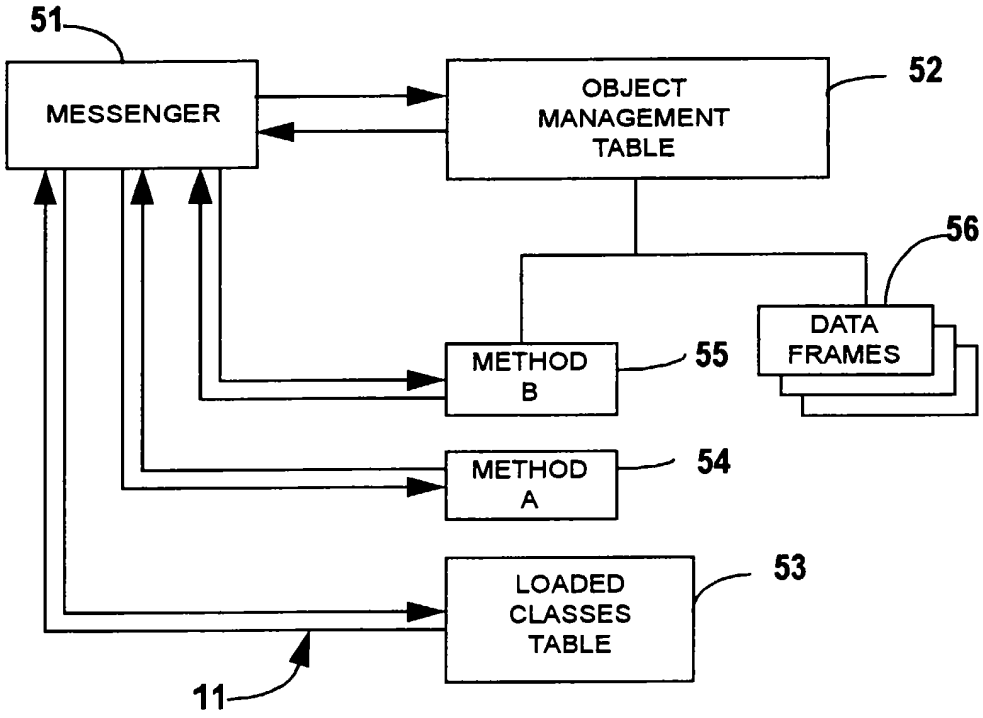


FIG 1

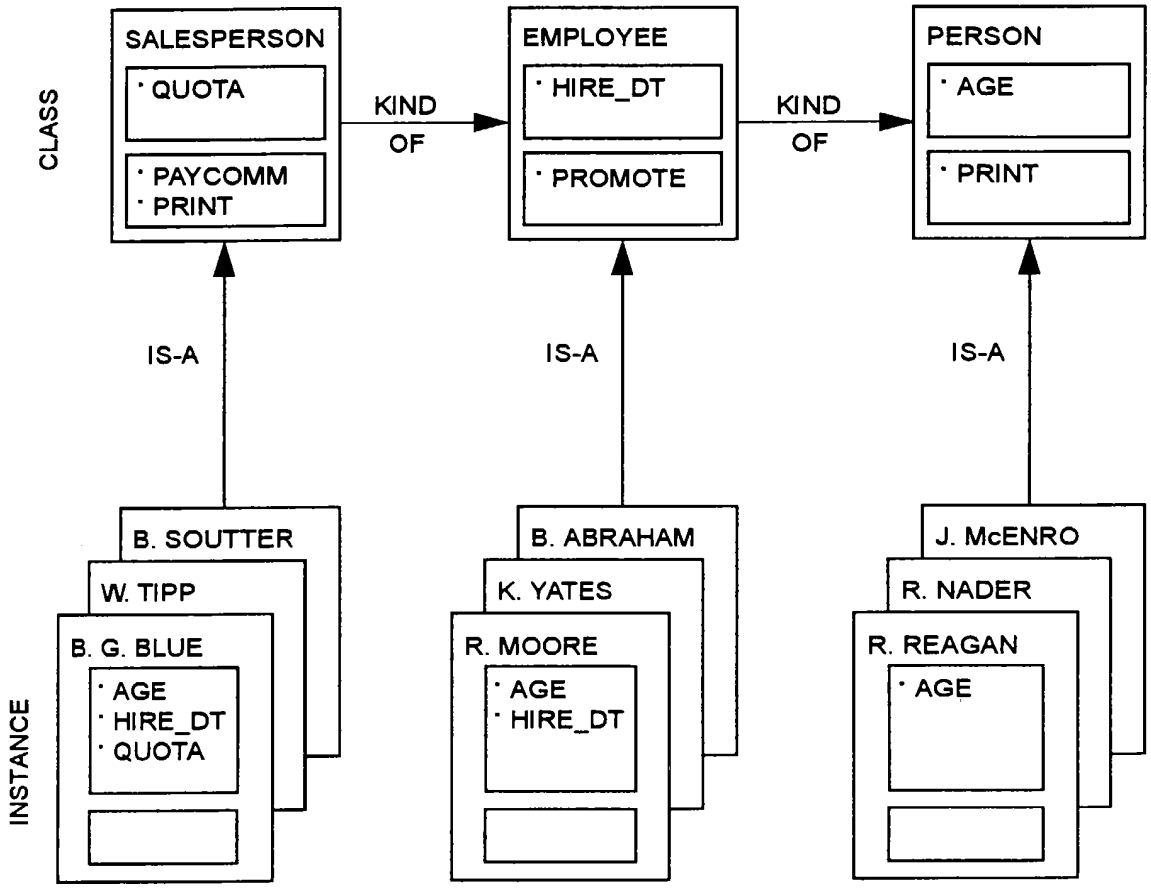


FIG 2

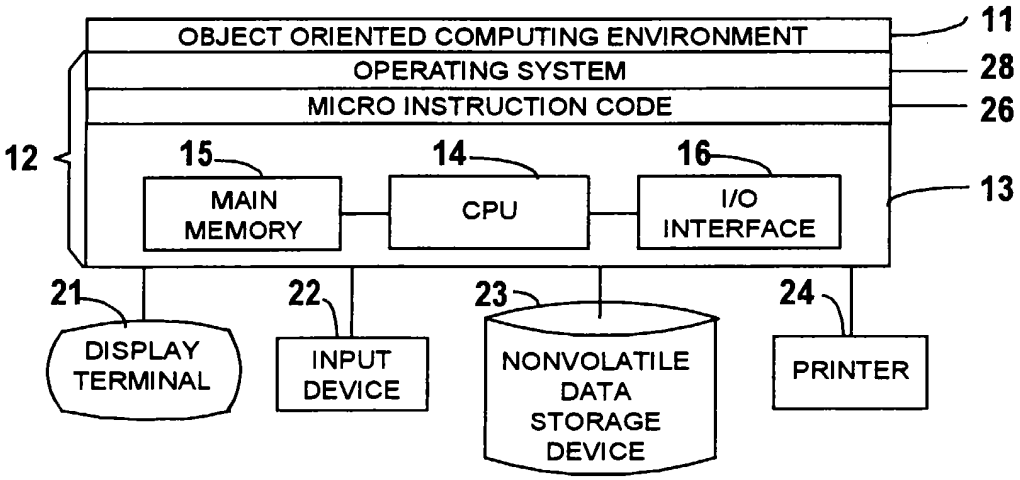


FIG 3

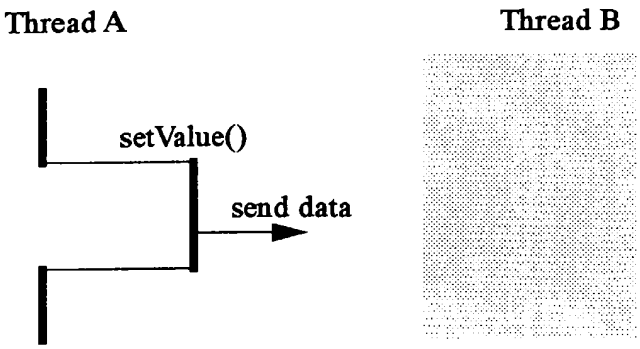


FIG 4

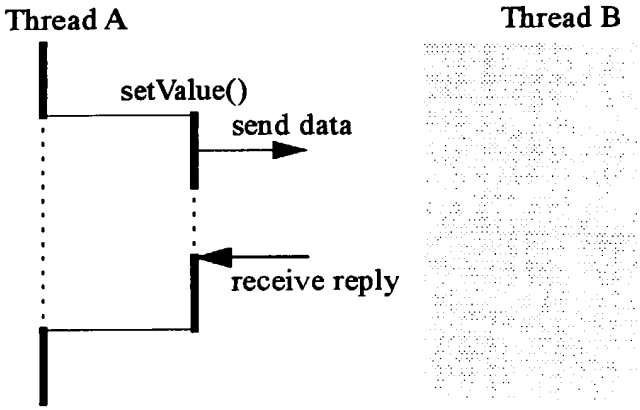


FIG 5

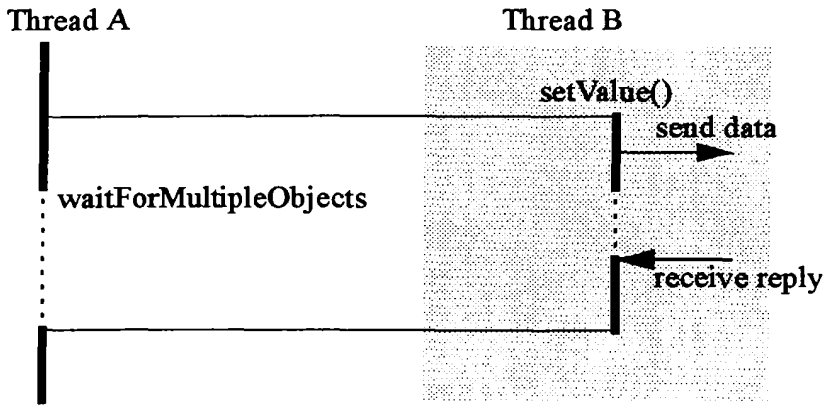


FIG 6

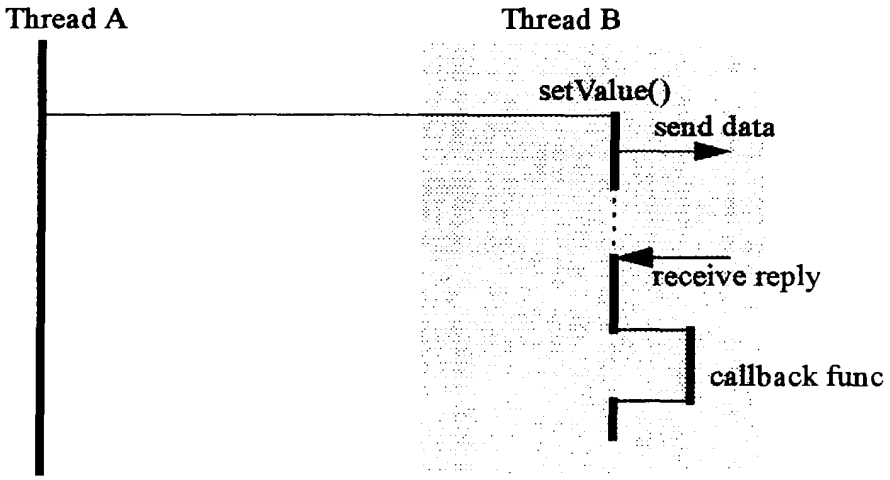


FIG 7

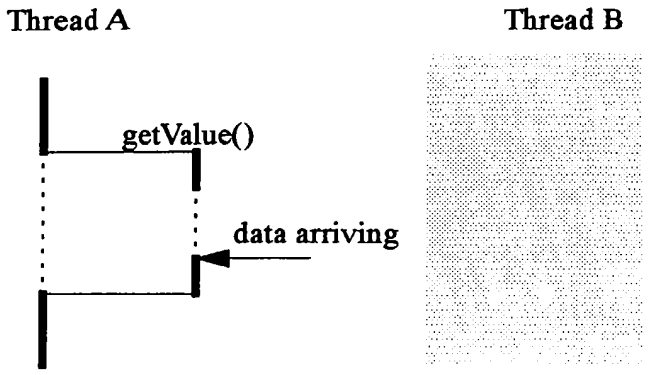


FIG 8

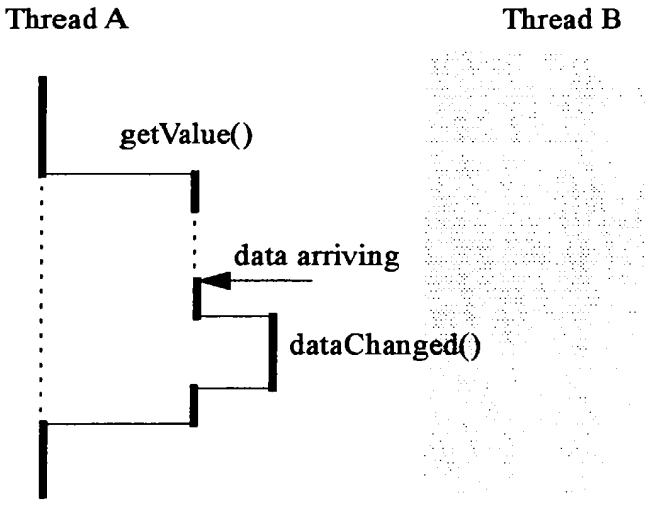


FIG 9

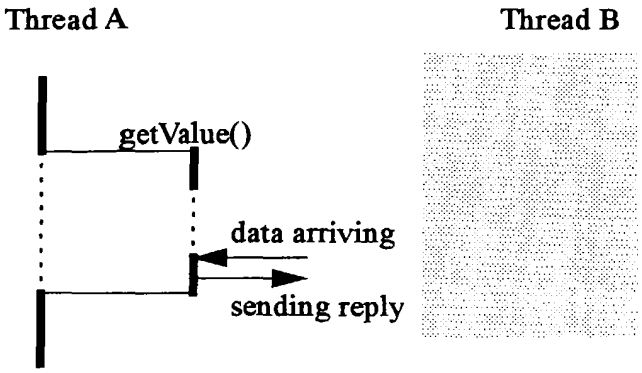


FIG 10

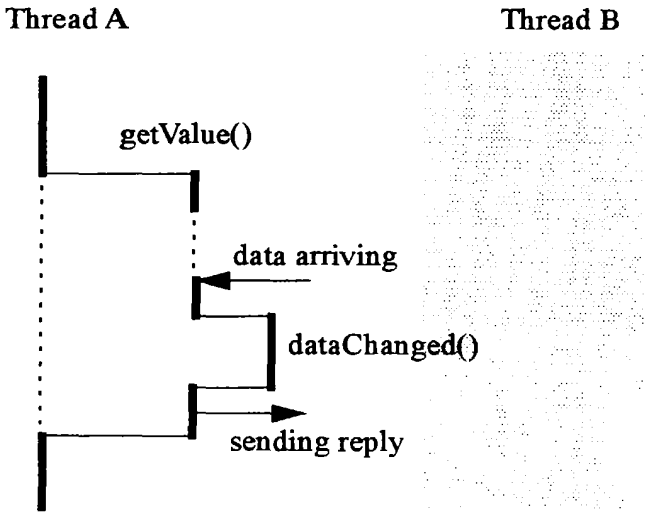


FIG 11

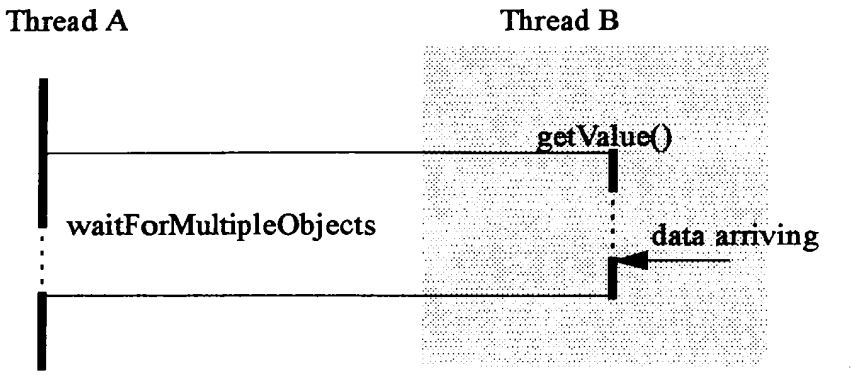


FIG 12

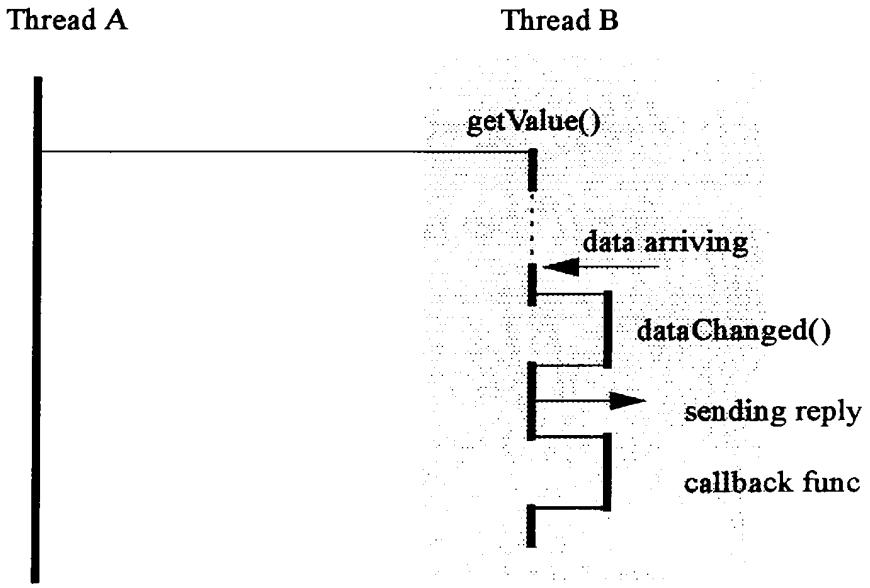


FIG 13

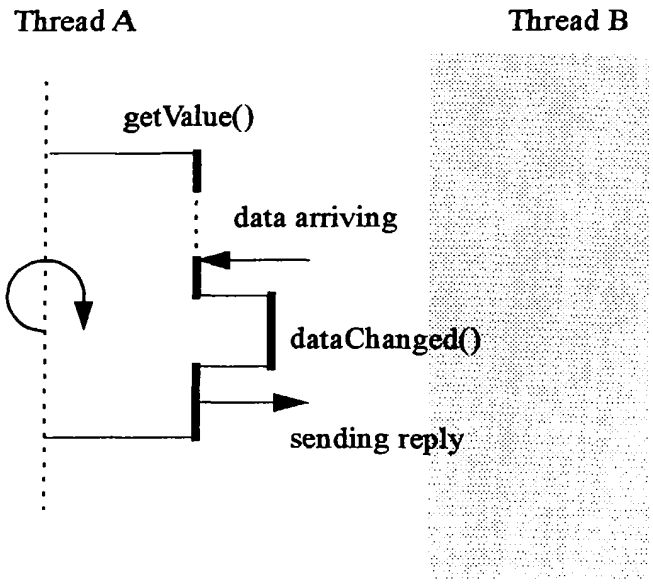


FIG 14

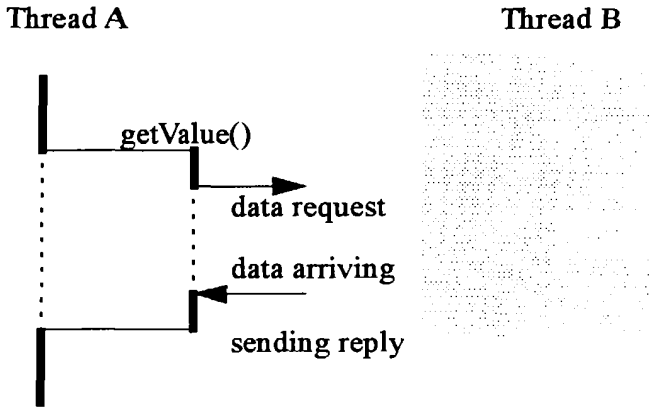


FIG 15

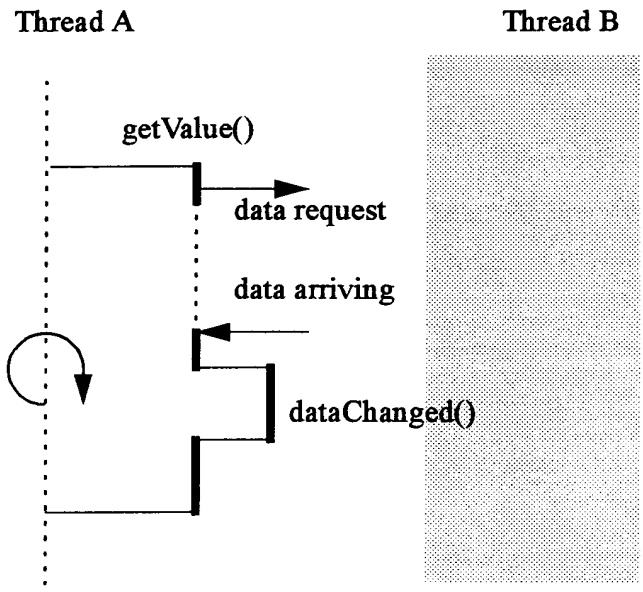


FIG 16

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 827 336 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.03.1998 Bulletin 1998/10

(51) Int Cl.[6]: **H04N 5/44**

(21) Application number: **97306679.8**

(22) Date of filing: **29.08.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: **30.08.1996 JP 230015/96**
11.03.1997 JP 56687/97

(71) Applicant: **MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.**
Kadoma-shi, Osaka-fu, 571 (JP)

(72) Inventors:
- **Shimoji, Tatsuya**
  Neyagawa-shi, Osaka-fu 572 (JP)
- **Okamura, Kazuo**
  Hirakata-shi, Osaka-fu 573 (JP)
- **Hirai, Junichi**
  Suita-shi, Osada-fu, 564 (JP)
- **Oashi, Masahiro**
  Hirakata-shi, Osaka-fu 573-01 (JP)
- **Kakiuchi, Takashi**
  Toyonaka-shi, Osaka-fu 561 (JP)
- **Kusumi, Yuki**
  Kashiba-shi, Nara-ken 639-02 (JP)
- **Miyabe, Yoshiyuki**
  Osaka-shi, Osaka-fu 532 (JP)
- **Minakata, Ikuo**
  Souraku-gun, Kyoto-fu 619-02 (JP)
- **Kozuka, Masayuki**
  Neyagawa-shi, Osaka-fu 572 (JP)
- **Mimura, Yoshihiro**
  Hirakata-shi, Osaka-fu 573 (JP)
- **Inoue, Shinji**
  Neyagawa-shi, Osaka-fu 572 (JP)
- **Mori, Toshiya**
  Settsu-shi, Osaka-fu 566 (JP)
- **Takao, Naoya**
  Kadoma-shi, Osaka-fu 571 (JP)

(74) Representative: **Crawford, Andrew Birkby et al**
A.A. THORNTON & CO.
Northumberland House
303-306 High Holborn
London WC1V 7LE (GB)

(54) Digital broadcasting system, digital broadcasting apparatus, and associated receiver therefor

(57) A broadcasting system which includes a broadcasting apparatus and a reception apparatus and which achieves interactiveness using a broadcast wave. The broadcasting apparatus includes a content storing unit for storing the plurality of contents, each content including a set of video data and a set of control information that indicates another content that is a link destination for a present content, and a transmitting unit for multiplexing a set of video data and a plurality of sets of the same control information included in a same content as the set of video data, and for transmitting the multiplexed sets of video data and control information. The reception apparatus includes an extracting unit for extracting a set of video data and a set of control information in a same content as the set of video data, a storing unit for storing the extracted set of control information, a reproducing unit for reproducing the extracted set of video data and outputting an image signal, an operation unit for receiving a user operation that indicates a content switching, and a control unit for controlling the extracting unit to extract another content indicated by the set of control information stored in the storing unit, in accordance with the user operation.

EP 0 827 336 A2

Description

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a digital broadcasting system.

2. Description of the Related Art

Since the development of digital satellite broadcasting in recent years, there has been a great increase in the number of programs being provided on an ever greater number of channels. For digital satellite broadcasting, large numbers of channels are possible by multiplexing the channels together into a single frequency band. This multiplexing is performed using what is called a "transport stream" under MPEG2 (Moving Pictures Experts Group2) standard. This technique is described in detail in the documentation for IS/IEC Standard 13818-1 (MPEG2 system standard).

Digital satellite broadcasting has a drawback in that image information is transmitted one-directionally from a transmitter, so that no interaction between the receiver and the transmitter is possible. However, users would be able to enjoy a greater variety of programs if they were able to make interactive selections of image information in accordance with the content of the image information received by the receiver terminal.

SUMMARY OF THE INVENTION

It is a primary object of the present invention to provide a digital broadcasting apparatus that one-directionally broadcasts a broadcast wave which, when received by a reception apparatus, gives the user the impression of interaction that is achieved through bidirectional communication. Here, it is the object of the present invention to further provide a reception apparatus, a digital broadcasting system, and a recording medium for recording a program to be used by a reception apparatus.

The above object can be achieved by a broadcasting apparatus for broadcasting an interactive program composed of a plurality of contents that are linked to one another, the broadcasting apparatus including: a content storing unit for storing the plurality of contents, each content including a set of video data and a set of control information that indicates another content that is a link destination for a present content, and a transmitting unit for multiplexing a set of video data and a plurality of sets of the same control information included in a same content as the set of video data, and for transmitting the multiplexed sets of video data and control information.

Here, the content storing unit may include: a first storing unit for storing the sets of video data included in the plurality of contents; a second storing unit for storing the sets of control information included in the plurality of contents; and a construction table storing unit for storing a construction table showing correspondence between the sets of video data stored in the first storing unit and the sets of control information stored in the second storing unit.

Here, the transmitting unit may include: a multiplexing unit for reading the plurality of sets of video data stored in the first storing unit and the plurality of sets of control information stored in the second storing unit as respective digital data streams, and multiplexing the digital data streams to generate a multiplexed stream; a multiplexing control unit for referring to the construction table and controlling the multiplexing unit to multiplex the plurality of sets of video data and to repeatedly multiplex a set of control information corresponding to a set of video data; and a broadcasting unit for placing the multiplexed stream generated by the multiplexing unit onto a digital broadcast wave and broadcasting the digital broadcast wave.

Here, the content storing unit may further include: a third storing unit for storing sets of audio data that correspond to the sets of video data, wherein the construction table storing unit stores correspondence between a set of video data, a set of audio data, and a set of control information included in each of the plurality of contents, and wherein the multiplexing unit also multiplexes the sets of audio data stored in the third storing unit into the multiplexed stream.

With the above construction, control information is repeatedly multiplexed with the video data and is transmitted, so that the reception apparatus can perform reproduction while interactively switching between contents in accordance with user operations. This is to say, the present digital broadcasting apparatus can provide interactive programs using a one-directional broadcast.

Here, each content may include a plurality of sets of control information, each set of control information including a set of link information showing contents that are link destinations and a set of time information indicating a valid period for the present control information within the reproduction period of the set of video data corresponding to the present set of control information, and the multiplexing control unit may control the multiplexing unit to repeatedly multiplex each set of control information with the corresponding set of video data during the valid period of the set of control information.

With the stated construction, the digital broadcasting apparatus can repeatedly and freely multiplex different sets of control information for different periods within the reproduction time of sets of video data, so that the link destination contents and number of potential links can be dynamically changed in accordance with the content of the video data. As a result, each content can be linked to appropriate link destination contents for the content of each scene in the video data.

Here, the multiplexing control unit may control the multiplexing unit to repeatedly multiplex each set of control information with the corresponding video data starting from a predetermined time before the valid period of the set of control information, the predetermined time being sufficiently long to enable a reception apparatus to process a set of control information.

With the stated construction, control information is repeatedly multiplexed with the video data starting from a predetermined time before the valid period of the control information, so that when a content is being reproduced by the reception apparatus, the reception apparatus will have enough time to process new control information which has a different valid period.

Here, the multiplexing control unit may append a version number, reflecting the valid period of each set of control information, to each set of control information in a given content.

With the stated construction, the reception apparatus can obtain new control information with a different valid period for a same content using the version numbers.

Here, each set of control information stored by the second storing unit may include a set of link information showing contents that are link destinations and supplementary images representing menu items for each link destination.

With the stated construction, menu items for assisting user operations in the reception apparatus can be freely set in each set of control information.

Here, at least one set of control information may include: a plurality of sets of additional information representing one of text and a graphic image that is to be displayed superimposed onto the corresponding video data; and a set of script information that validates one of the sets of additional information within a reception apparatus, in accordance with a user operation.

With the stated construction, sets of control information can be provided with script information and a plurality of sets of additional information, with the reception apparatus being able to interactively switch between the sets of additional information. As one example, when the video data expresses a weather forecast and the sets of additional information provide a plurality of supplementary explanations (text or graphics) relating to the content of the video data, switching of the display of supplementary explanations can be performed using the script information in accordance with the user operations. By doing so, one pair of a set of video data and a set of control information can be used to express what are effectively a plurality of contents.

Here, at least one set of control information may include: at least two groups of a set of link information and supplementary images; a set of initial information showing a group of a set of link information and supplementary images that is valid at a start of reproduction by a reception apparatus for a content including the present set of control information; and a set of script information that changes a valid setting in the reception apparatus in accordance with a user operation.

With the stated construction, sets of control information are provided with a plurality of groups which may be switched according to the script information, so that an effective increase in the number of link destination contents can be achieved.

Here, each group of a set of link information and supplementary image may further include a set of additional information representing one of text and a graphic image that is to be displayed superimposed onto the corresponding video data.

With the stated construction, the groups are provided with additional information, so that one pair of a set of video data and a set of control information can be used to effectively express a plurality of contents that have different link destination contents according to the additional information.

The object of the present invention can also be achieved by a broadcasting apparatus for broadcasting an interactive program composed of a plurality of contents that are linked to one another, the broadcasting apparatus including an image storing unit storing a plurality of sets of video data and a plurality of sets of still image data; a control information storing unit for storing sets of type 1 control information and sets of type 2 control information, the sets of type 1 control information being elements of contents including video images, the sets of type 2 control information being elements of contents including still images, and the sets of type 1 control information and sets of type 2 control information including sets of link information that indicate contents which are link destinations for a present content; a construction table storing unit storing a first construction table showing correspondence between sets of video data and sets of type 1 control information and a second construction table showing correspondence between sets of still image data and sets of type 2 control information; a first multiplexing unit for generating a first multiplexed stream by multiplexing a set of video data in the first construction table and repeatedly multiplexing a set of type 1 control information corresponding to the set of video data; a second multiplexing unit for generating a second multiplexed stream by repeatedly multi-

plexing a plurality of sets of still image data in the second construction table with a set of type 2 control information; and a broadcasting unit for placing the multiplexed stream generated by the multiplexing unit onto a digital broadcast wave and broadcasting the digital broadcast wave.

With the stated construction, interactive programs that are made up of two types of contents, which is to say video-based contents and still-based contents, can be broadcasted. Type 1 and type 2 control information can have both kinds of contents as link destination contents, so that a reception apparatus can perform reproduction switching between both kinds of content in accordance with user operations. In this way, very impressive interactive programs can be realized.

The above object can also be achieved by a reception apparatus for receiving a broadcast wave including an interactive program composed of a plurality of contents that are linked to one another, wherein the broadcast wave includes a multiplexed stream into which different sets of video data have been multiplexed with a plurality of sets of control information showing a link to another content, the sets of control information being repeatedly multiplexed, the reception apparatus including: an extracting unit for extracting a set of video data and a set of control information in a same content as the set of video data; a storing unit for storing the extracted set of control information; a reproducing unit for reproducing the extracted set of video data and outputting an image signal; an operation unit for receiving a user operation that indicates a content switching; and a control unit for controlling the extracting unit to extract another content indicated by the set of control information stored in the storing unit, in accordance with the user operation.

With the stated construction, the reception apparatus can behave interactively as if two-way communication were being performed, despite only using a one-directional broadcast of image information, meaning that users can enjoy interactive programs. Since the control information is repeatedly transmitted, the storing unit only requires enough storage capacity to store the control information for one content.

Here, first identification information may be appended to each set of video data and second identification information is appended to each set of control information, and wherein the sets of control information include first identification information and second identification information which express a content of a link destination, the extracting unit may include: a first judging unit for judging the first identification information appended to sets of video data in the broadcast wave; a second judging unit for judging the second identification information appended to sets of control information in the broadcast wave; an obtaining unit for obtaining a set of video data and when the first judging unit judges that the first identification information coincides with specified identification information indicated by the control unit and obtaining a set of control information when the second judging unit judges that the second identification information coincides with specified identification information, the reproducing unit may reproduce the set of video data obtained by the obtaining unit, and the storing unit may store the set of control information obtained by the obtaining unit.

With the stated construction, the reception apparatus judges the sets of video information and control information and obtains the appropriate data, so that the only the data to be reproduced is obtained, thereby improving the reception efficiency.

Here, a set of entry information giving first identification information and second identification information for the content to be reproduced first may be multiplexed into the multiplexed stream, the control unit may send an indication to the extracting unit to extract the set of entry information when the operation unit has received a selection operation for a multiplexed stream from a user, the extracting unit may further include: an entry information extracting unit for receiving the indication from the control unit and extracting the set of entry information from the multiplexed stream; and an entry information storing unit for storing the set of entry information extracted by the entry information extracting unit, wherein the control unit may give the obtaining unit an indication of the first identification information and second identification information included in the entry information as the specified identification information.

With the stated construction, the reception apparatus can extract the content to be reproduced first in accordance with the entry information, so that contents which contain important information, such as a main menu, can definitely be reproduced.

Here, the link information may include an identifier of a set of video data and an identifier of a set of control information which show a content of a link destination, the first identification information and second identification information may be IDs (identifiers) of digital data streams which represent a set of video data and a set of control information in the multiplexed stream, a correspondence table, showing correspondence between the identifiers for sets of video data and the first identification information and correspondence between the identifiers for sets of control information and the second identification information, may be multiplexed into the multiplexed stream and repeatedly transmitted, and the extracting unit may extract the correspondence table and the control unit may refer to the correspondence table, convert an identifier of the set of video data included in the link information into first identification information and an identifier of the set of control information into second identification information and inform the extracting unit of the converted first and second identification information.

With the stated construction, the interactive programs of the present invention are broadcast using a digital satellite broadcast according to MPEG2 standard, so that the present invention can be achieved by modifying a conventional digital satellite broadcast tuner.

Here, at least one set of control information may include link information showing a content of a link destination and supplementary images that include a menu item image for each link destination, the reproducing unit may include: a video data reproducing unit for reproducing the set of video data obtained by the obtaining unit; and an image reproducing unit for reproducing supplementary images stored by the storing unit superimposed onto the video data, wherein the operation unit may receive a user selection of a menu item image, and wherein the control unit may determine the first identification information and the second identification information of a link destination content in accordance with the link information and the menu item image selected by the user.

With the stated construction, menu item images are displayed by the reception apparatus, assisting the interactive operations of the user and enabling the achievement of impressive interactive programs.

Here, at least one set of control information may include additional information which expresses one of a text image and a graphics image, and wherein the reproducing unit may additionally reproduce one of the text image and graphics image stored in the storing unit superimposed onto the video data.

With the stated construction, additional images such as text or graphics are displayed in addition to the video data, making the interactive programs even more impressive.

Here, each set of control information may include valid period information showing a valid period of the set of control information, wherein each content may have a plurality of sets of control information which have different valid periods, and wherein the reproducing unit may reproduce supplementary images stored in the storing unit only during a valid period of the set of control information stored in the storing unit.

With the stated construction, sets of control information with different valid periods within the reproduction time of video data are repeatedly multiplexed, so that the link destination contents and number of potential links can be dynamically changed in accordance with the content of the video data. As a result, each content can be linked to appropriate link destination contents for the content of each scene in the video data. As a result, the user of the reception apparatus can gain greater enjoyment from the interactive programs which have link destinations that correspond to the video scenes.

Here, each of the plurality of sets of control information for a same content has a version number that reflects the valid period, and wherein the control unit controls the extracting unit to extract a set of control information which has a next version number, when one set of control information has been extracted by the extracting unit.

With the above construction, the reception apparatus can use the version numbers for the content presently being reproduced and so obtain following sets of control information with different valid periods.

Here, at least one set of control information may include a plurality of sets of additional information which each express one of a text image and a graphics image to be displayed superimposed onto the video data, and a set of script information that validates one of the sets of additional information within a reception apparatus, in accordance with a user operation, wherein the control unit may determine a valid set of additional information by interpreting and executing the script information stored in the storing unit, and wherein the reproducing unit may reproduce one of the text image and the graphics image included in the valid set of additional information based on a result of interpreting and executing by the control unit.

With the stated construction, the reception apparatus does not need to obtain new control information, and so can perform content switching in accordance with the script information. Such content switching performed entirely by the execution of script information is much more responsive to user operations.

Here, the multiplexed stream may include sets of audio data corresponding to the sets of video data, wherein the extracting unit may extract a set of audio data corresponding to a set of video data from the broadcast wave, and wherein the reproducing unit may additionally reproduce the extracted set of audio data.

With the stated construction, interactive programs composed of contents including video, stills, and audio can be realized.

The stated object can also be achieved by a recording medium used by a reception apparatus that includes a receiving unit for receiving a broadcast wave including an interactive program composed of a plurality of contents that are linked to one another, an extracting unit for extracting one digital data stream from the broadcast wave, and a reproducing unit for reproducing a set of video data and outputting an image signal, the recording medium storing a program that includes the following steps: an extracting step for extracting a set of video data and a set of control information in a same content as the set of video data from the broadcast wave; a storing step for storing the extracted set of control information into a memory in the reception apparatus; a reproducing step for reproducing the extracted set of video data and outputting an image signal; a judging step for judging whether a user operation indicating a switching of content has been made; and a control step for controlling the extracting unit to extract another content indicated by the set of control information stored in the memory, when the judging step judges that a user operation indicating a switching of content has been made.

With the stated recording medium, the program can be installed into a conventional reception apparatus (satellite broadcast tuner), so that the present invention can be easily realized.

BRIEF DESCRIPTION OF THE INVENTION

These and other objects, advantages and features of the invention will become apparent from the following description thereof taken in conjunction with the accompanying drawings which illustrate a specific embodiment of the invention. In the drawings:

Fig. 1 shows a plurality of examples of contents which are selectively reproduced by a reception apparatus;
Fig. 2 is an expansion of the left side of Fig. 1;
Fig. 3 is an expansion of the right side of Fig. 1;
Fig. 4 shows the construction of the digital broadcasting apparatus and reception apparatus in the digital broadcasting system of the first embodiment of the present invention;
Fig. 5 shows a plurality of examples of contents which compose an interactive program;
Figs. 6A and 6B show example sets of image data which are stored by the presentation information storage unit in the present embodiment;
Figs. 6C and 6D show example sets of audio data which are stored by the presentation information storage unit in the present embodiment;
Fig. 7 shows an example of the navigation information which is stored by the navigation information storage unit in the present embodiment;
Fig. 8 shows another example of the navigation information which is stored by the navigation information storage unit in the present embodiment;
Fig. 9 shows another example of the navigation information which is stored by the navigation information storage unit in the present embodiment;
Fig. 10 shows another example of the navigation information which is stored by the navigation information storage unit in the present embodiment;
Fig. 11 shows another example of the navigation information which is stored by the navigation information storage unit in the present embodiment;
Fig. 12 shows an example of the construction information table which is stored by the construction information storage unit in the present embodiment;
Fig. 13 shows an example of the entry information which is stored by the construction information storage unit in the present embodiment;
Fig. 14 shows an example of the multiplexing information table stored by the multiplexing information storage unit in the present embodiment;
Fig. 15 shows an example of the content identifier assigning table generated by the multiplexing control unit in the present embodiment;
Fig. 16 shows an example of the version number assigning table generated by the multiplexing control unit in the present embodiment;
Fig. 17 shows an example of a navigation information table generated by the navigation information table generating unit in the present embodiment;
Fig. 18 shows another example of a navigation information table generated by the navigation information table generating unit in the present embodiment;
Fig. 19 shows another example of a navigation information table generated by the navigation information table generating unit in the present embodiment;
Fig. 20 shows another example of a navigation information table generated by the navigation information table generating unit in the present embodiment;
Fig. 21 shows another example of a navigation information table generated by the navigation information table generating unit in the present embodiment;
Fig. 22A shows an example of the NIT generated by the system information table generating unit in the present embodiment;
Fig. 22B shows an example of the SDT generated by the system information table generating unit in the present embodiment;
Fig. 22C shows an example of the EIT generated by the system information table generating unit in the present embodiment;
Fig. 23 shows an example of the PAT generated by the system information table generating unit in the present embodiment;
Fig. 24 shows an example of the PMT generated by the system information table generating unit in the present embodiment;
Fig. 25 shows the detailed content of the Entry_Descriptor in the PMT generated by the system information table generating unit in the present embodiment;

Figs. 26A to 26D show the details of the NE_Component_Descriptor in the PMT generated by the system information table generating unit in the present embodiment;
Figs. 27A and 27B show the details of the stream_identifier_descriptor in the PMT generated by the system information table generating unit in the present embodiment;
Fig. 28 is a graphic representation of a transport stream multiplexed by the multiplexing unit in the present embodiment;
Fig. 29 is a graphic representation of a transport stream multiplexed by the transmission unit in the present embodiment;
Figs. 30 to 32 are flowcharts showing the operation of the data transmission apparatus in the present embodiment;
Figs. 33A and 33B are examples of filter conditions stored by the filter condition storage unit in the TS decoder unit of the present embodiment;
Figs. 34A to 34D are examples of display images displayed by the display unit in the present embodiment;
Figs. 35A and 35B are examples of display images displayed by the display unit in the present embodiment;
Fig. 36 is a flowchart showing an overview of the reception processing for an interactive program;
Fig. 37 is a flowchart showing the details of the content switching process shown in Fig. 36;
Fig. 38 is a flowchart showing the details of the image data switching process shown in Fig. 37;
Fig. 39 is a flowchart showing the details of the audio data switching process shown in Fig. 37;
Fig. 40 is a flowchart showing the details of the navigation information switching process shown in Fig. 37;
Fig. 41 is a flowchart for the interactive control processing performed according to the navigation information;
Fig. 42 is a flowchart showing the user I/F processing performed according to the navigation information;
Fig. 43 shows examples of other interactive programs which are composed of the four contents 10 to 13;
Figs. 44 to 47 show four sets of navigation information which correspond to contents 10 to 13;
Fig. 48 shows a set of navigation information which corresponds to all contents 10 to 13;
Fig. 49 shows examples of contents which are each displayed as one frame on the display screen of the reception apparatus;
Fig. 50 is a model representation of the transmission data transmitted by the transmission apparatus;
Fig. 51 a model representation of the transmission of the transmission data from the transmission apparatus;
Fig. 52 shows the construction of the data transmission apparatus and the data reception apparatus in the second embodiment of the present invention;
Fig. 53A shows an example of the image data stored by the presentation information storage unit in the present embodiment;
Fig. 53B shows another example of the image data stored by the presentation information storage unit in the present embodiment;
Fig. 54 shows an example of the navigation information which is stored in the navigation information storage unit of the present embodiment;
Fig. 55 shows another example of the navigation information which is stored in the navigation information storage unit of the present embodiment;
Fig. 56 shows an example of the construction information table and the entry information which are stored in the construction information storage unit of the present embodiment;
Fig. 57 shows an example of the multiplexing information table stored in the multiplexing information storage unit of the present embodiment;
Fig. 58 shows an example of the content identifier assigning table generated by the multiplexing control unit of the present embodiment;
Fig. 59 shows an example of the display image information identifier assigning table which is generated by the multiplexing control unit of the present embodiment;
Fig. 60 shows the state when the identifier information appending unit of the present embodiment has appended the identifier VE_id to the private area of the bitstream for the image data;
Fig. 61 shows an example of a navigation information table generated by the navigation information table generating unit of the present embodiment;
Fig. 62 shows another example of a navigation information table generated by the navigation information table generating unit of the present embodiment;
Fig. 63A shows an example of the stream correspondence table generated by the stream correspondence information table generating unit in the present embodiment;
Fig. 63B shows another example of the stream correspondence table generated by the stream correspondence information table generating unit in the present embodiment;
Fig. 63C shows another example of the stream correspondence table generated by the stream correspondence information table generating unit in the present embodiment;
Fig. 64 is a model representation of a transport stream which has been multiplexed by the multiplexing unit of the

present embodiment;
Fig. 65 is a model representation of another transport stream which has been multiplexed by the multiplexing unit of the present embodiment;
Fig. 66A shows an example of the NIT generated by the system information table generating unit of the present embodiment;
Fig. 66B shows an example of the SDT generated by the system information table generating unit of the present embodiment;
Fig. 66C shows an example of the EIT generated by the system information table generating unit of the present embodiment;
Fig. 67 shows an example of the PAT generated by the system information table generating unit of the present embodiment;
Fig. 68 shows an example of the PMT generated by the system information table generating unit of the present embodiment;
Fig. 69A shows the details of the Entry_Descriptor in the PMT generated by the system information table generating unit of the present embodiment;
Fig. 69B shows the details of the NE_Component_Descriptor in the PMT generated by the system information table generating unit of the present embodiment;
Fig. 69C shows the details of the VE_Information Component_Descriptor in the PMT generated by the system information table generating unit of the present embodiment;
Fig. 69D shows the details of the stream_identifier_descriptor in the PMT generated by the system information table generating unit of the present embodiment;
Fig. 70 is a model representation of the transport stream multiplexed by the transmission unit of the present embodiment;
Fig. 71 is a flowchart showing the operation of the data transmission apparatus of the present embodiment;
Fig. 72 is a flowchart showing the operation of the data transmission apparatus of the present embodiment;
Fig. 73 is a flowchart showing the operation of the data transmission apparatus of the present embodiment;
Fig. 74A shows examples of the filter conditions which are stored by the filter condition storage unit of the TS decoder unit of the present embodiment;
Fig. 74B shows other examples of the filter conditions which are stored by the filter condition storage unit of the TS decoder unit of the present embodiment;
Fig. 75A shows an example display screen which is displayed by the display unit in the present embodiment;
Fig. 75B shows another example display screen which is displayed by the display unit in the present embodiment;
Fig. 75C shows another example display screen which is displayed by the display unit in the present embodiment;
Fig. 76 is a flowchart showing the showing the entire operation of the data receiver apparatus of the present embodiment;
Fig. 77 is a flowchart showing the details of the processing in S2410 of the present embodiment;
Fig. 78 is a flowchart showing the details of the processing in S2504 of the present embodiment;
Fig. 79 is a flowchart showing the details of the processing in S2506 of the present embodiment;
Fig. 80 is a flowchart showing the details of the processing in S2416 of the present embodiment;
Fig. 81 shows the construction of the digital broadcasting apparatus of the third embodiment of the present invention;
Figs. 82A and 82B show examples of the construction information tables stored in the construction information storage unit of the third embodiment;
Fig. 83 shows an example of the navigation information in the third embodiment;
Figs. 84A to 84C are representations of sets of video data in the third embodiment;
Fig. 85 shows the relation between scenes in the video data and the sets of navigation information;
Fig. 86 also shows the relation between scenes in the video data and the sets of navigation information;
Fig. 87 shows a specific example of a set of navigation information;
Fig. 88 shows another specific example of a set of navigation information;
Fig. 89 shows another specific example of a set of navigation information;
Fig. 90 shows another specific example of a set of navigation information;
Fig. 91 shows another specific example of a set of navigation information;
Fig. 92 shows another specific example of a set of navigation information;
Fig. 93 shows another specific example of a set of navigation information;
Fig. 94 shows another specific example of a set of navigation information;
Fig. 95 shows a specific example of a set of navigation information which corresponds to a plurality of contents;
Fig. 96 shows an example of a navigation information table;
Fig. 97 shows an example of a transport stream multiplexed by the transmission unit;

Fig. 98 is a block diagram showing the construction of the data reception apparatus in the present embodiment;
Fig. 99 shows an example of the filter condition table stored by the filter condition storage unit;
Fig. 100 is a flowchart for the control executed by the reception control unit; and
Fig. 101 is also a flowchart for the control executed by the reception control unit.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

0. Outline Description of the Present Invention

An outline description of the interactive programs achieved by digital broadcasting is given below, prior to the description of the construction of the broadcasting apparatus and reception apparatus in the digital broadcasting system of the present invention.

The broadcasting apparatus in the present digital broadcasting system broadcasts interactive programs composed of contents that feature links to one another. The reception apparatus receives these interactive programs and reproduces them, interactively switching between contents as requested by a user.

The term "content" here refers to information which forms each compositional element of an interactive program and so is the unit of information used when interactive switching operations are made by a user. In the present embodiment, there are two types of contents which are namely stream-based contents and page-based contents. Here, stream-based contents are contents which are mainly used for moving pictures (video), while page-based contents are contents which are mainly used for displaying still images.

Fig. 1 shows a plurality of examples of contents which are selectively reproduced by a reception apparatus. Expansions of the left and right side of Fig. 1 are shown in Fig. 2 and Fig. 3, respectively. Here, the line "A-A"' in Figs. 2 and 3 shows the boundary between the two halves of Fig. 1.

In the present figures, numerals 100S-105S, 105S', and 105S" denote stream-based contents, while numerals 100P-106P denote page-based contents.

Content 100S represents video and audio for a world travel guide which gives a succession of introductions of various countries around the world, such as China, Japan, and Egypt, as well as menus composed of a plurality of button images (hereinafter referred to as buttons) which are only displayed during the video display of the corresponding destination. These buttons are used to make user selections of other contents which are linked to the present content. Simplified representations of several scenes (or frames) out of the sets of video introducing different countries are also shown in Figs. 1 to 3 in order of their reproduction times. As one example, scene 100S1 represents a travel guide for China, while scene 100S2 represents a travel guide for Japan. When switching from scene 100S1 to scene 100S2, the displayed menu also changes from the menu for China to the menu for Japan. The link destinations for the displayed buttons also change from the contents for the travel guide for China to the contents for the travel guide for Japan.

Contents 101S, 102S, and 103S include the same video and audio for introductions of various countries around the world as content 100S, but include different menus for the countries being introduced and text information which may be supplementary information for the country being introduced.

Content 104S is composed of video and audio for a travel guide about Japan which successively introduces different areas of Japan, such as Osaka and Nara, with a menu composed of buttons which correspond to the region currently being displayed.

Contents 105S, 105S', and 105" include the same video and audio as content 104S, but have different menus for the various regions being introduced.

Content 100P represents a still image showing a world weather forecast, as well as buttons for different countries, such as Japan, China, or Taiwan, which are link destination contents.

In the same way, contents 101P to 106P represent still images showing weather forecasts for different regions and countries which are the link destinations, as well as buttons for the different link destinations.

The interactive program which contains the contents described above is reproduced by the reception apparatus with interactive switching of contents being performed in accordance with user requests. The arrows in Figs. 1 to 3 show examples of switching between contents.

As one example, when the user makes an operation to select and activate the "Weather" button during the reproduction of scene 100S1, the reproduction apparatus switches the reproduction to the link destination, content 100P, as shown by the arrow in Fig. 2. As a result, the user is shown a still image for a main menu that informs the user of the weather around the world. Here, if the user makes an operation to select and activate the "Return" button during the reproduction of content 100P, the reproduction apparatus switches the reproduction to scene 100S2 at that point, as shown by the arrow in Fig. 2. In this way, switching is performed between stream-based contents and page-based contents.

As another example, if the user makes an operation to select and activate the "Transport" button during the reproduction of scene 101S1 (a scene with a submenu including items such as transport to Japan and accommodation),

the reproduction apparatus switches the reproduction to the link destination, scene 102S1, as shown by the arrow in Fig. 2. As a result, the user is shown text information which gives a supplementary explanation about transport to Japan. Here, if the user makes an operation to select and activate the "Return" button during the reproduction of scene 101S1, the reproduction apparatus switches the reproduction to scene 101S2, as shown by the arrow in Fig. 2. In this way, switching is performed between different stream-based contents.

As another example, if the user makes an operation to select and activate the "Osaka" button during the reproduction of content 104P (a main menu showing the weather in Japan), the reproduction apparatus switches the reproduction to the link destination, content 106P, as shown by the arrow in Fig. 2. As a result, the user is shown a weather forecast for Osaka, Japan. Here, if the user makes an operation to select and activate the "Return" button during the reproduction of content 106P, the reproduction apparatus switches the reproduction to content 104P. In this way, switching is performed between different page-based contents.

This concludes the overview of the present invention. Below, the first embodiment will describe the construction of a digital broadcasting system for realizing an interactive program composed of stream-based contents, while the second embodiment will describe the construction of a digital broadcasting system for realizing an interactive program composed of page-based contents and the third embodiment will describe the construction of a digital broadcasting system for realizing an interactive program composed of both types of contents.

1. First Embodiment

The following explanation will first deal with the interactive programs composed of stream-based contents (in the present embodiment, hereinafter abbreviated to "contents"), before describing the construction of the digital broadcasting system.

The content S100 shown in Figs. 1 and 2 expresses video data and audio data for video and audio which give a world travel guide, as well as a plurality of sets of navigation information which express supplementary explanations and menus composed of a plurality of buttons corresponding to the countries being displayed.

The sets of navigation information referred to here can be provided so as to correspond to the reproduced content of the video data during given time periods. As one example, one set of navigation information (set as "version 1") may be provided for scene 100S1 which includes a travel guide for China, while another set of navigation information (set as "version 2") may be provided for scene 100S2 which includes a travel guide for Japan and another set of navigation information (set as "version N") may be provided for a scene which includes a travel guide for Egypt.

Contents 101S to 103S share the same video data and audio data for the world travel guide as content 100S, and also include a plurality of sets of navigation information which correspond to the reproduced content of the video data during given time periods.

The reproduction apparatus is able to switch from a content which is currently being displayed to a different stream-based content because navigation information is transmitted by the broadcasting apparatus along with the video data and audio data, according to the method described below.

Navigation information is repeatedly transmitted by the broadcasting apparatus during the reproduction time of the image data. When doing so, the transmitted navigation information corresponds to the reproduced content of the video data for the given time period, so that each set of navigation information is only repeatedly transmitted during the time period to which it corresponds.

As one example, during the reproduction time of the travel guide for China included in scene S100S1, the navigation information for version 1 is repeatedly transmitted. Similarly, during the reproduction time of the travel guide for Japan included in scene S100S2, the navigation information for version 2 is repeatedly transmitted. Also, during the. reproduction time of the travel guide for Egypt, the navigation information for version N is repeatedly transmitted.

Here, the reason the same navigation information is repeatedly sent is to enable the reception apparatus to immediately receive the navigation information of a switching destination when content switching is performed or when reception is commenced midway through the broadcast of an interactive program. When the navigation information is dynamically set for different sections of the video data, this means that the latest navigation information can be received at the point where the content switching is made.

By doing so, the user of a reception apparatus in what is a one-directional broadcasting system can make what appear to be interactive operations that switch between stream-based contents.

1-1 Digital Broadcasting System

Fig. 4 is a block diagram showing the construction of the digital broadcasting system of the first embodiment of the present invention.

The present digital broadcasting system includes a digital broadcasting apparatus 5101 and a plurality of reception apparatuses. In Fig. 4, this plurality of reception apparatuses is represented by only one reception apparatus 5121. In

the present digital broadcasting system, interactive programs which are composed of stream-based contents (hereinafter, simply "contents") that are linked to one another are broadcast by the digital broadcasting apparatus 5101 and are reproduced by the reception apparatus 5121 which interactively switches between the different contents.

To simplify the explanation of the present embodiment, an example of an interactive program which is composed of the four contents numbered 0-3 in Fig. 5 will be used to describe the construction of the digital broadcasting apparatus 5101 and the reception apparatus 5121.

Content 0 is composed of scenes 01a to 01h, and expresses video and audio (not illustrated) which gives the viewer of a weather forecast for the Kansai region of Japan.

Scene 01a is the opening scene.

Scene 01b has two buttons (button images) 02b and 03b superimposed over video image. These buttons 02b and 03b are respectively linked to content 1 and content 2 and are used by the reception apparatus 5121 when the user performs a content switching operation. This is also the case for scenes 01c to 01h.

Content 1 is composed of scenes 11a to 11h and, in addition to the video and audio (not illustrated) for the weather forecast used in content 0, includes the text information 13b to 13h. These sets of text information 13b to 13h express supplementary information such as the estimated maximum temperature, estimated minimum temperature, humidity, and probability of rain.

Scene 11b has a button 12b which features a link to content 0 and a text information 13b displayed on top of the video images. This is also the case for scenes 11c to 11h.

Content 2 and expresses video and audio (not illustrated) which gives the viewer of a weather forecast for the Kanto region of Japan. Scene 21b has a button 22b linked to content 3 and a button 23b which is linked to content 0.

Content 3 includes sets of text information, in addition to the video and audio (not illustrated) for the weather forecast used in content 2. As one example, the button 32b in scene 31b is linked to content 2.

The interactive program composed of the four contents described above is reproduced by the reception apparatus 5121 while switching between contents as desired by the user, according to the method described below.

As one example, when the user makes a select and activate operation for the button 02b during the reproduction by the reception apparatus 5121 of scene 01b in content 0, the reception apparatus 5121 switches the reproduction to content 1 which is the link destination of this button. When doing so, since every content is multiplexed with corresponding reproduction times, content 1 is not reproduced from the start and is instead reproduced starting from a scene whose reproduction time corresponds with the part of content 0 where the switching operation was made. Here, since content 1 contains the same video and audio as content 0, the user will not notice a change in the reproduced video or audio and will merely obtain supplementary information for the weather forecast which is provided as text.

1-2 Digital Broadcasting Apparatus 5101

As shown in Fig. 4, the digital broadcasting apparatus 5101 is composed of a transmission data storage unit 5102, a data multiplexing unit 5103, a multiplexing information storage unit 5104, a system information table generating unit 5105, and a transmission unit 5106. This digital broadcasting apparatus 5101 broadcasts the interactive program described above on a digital broadcast wave.

The transmission data storage unit 5102 includes a recording medium such as a magnetic disc and is used to store the data for each content which composes the interactive program. Here, the data for one content is composed of presentation information, such as video data and audio data, and navigation information which expresses hyperlinks to other contents, buttons, and text information. As one example, the video and audio for each content in Fig. 5 are included in this presentation information, while the buttons and text information to be displayed on top of the video images, and the hyperlinks given to each button are included in the navigation information.

The data multiplexing unit 5103 generates a multiplexed stream by multiplexing the content data stored in the transmission data storage unit 5102. More specifically, the data multiplexing unit 5103 multiplexes the presentation information for each content so that the reproduction times of the presentation information coincide, as well as repeatedly multiplexing the navigation information during the reproduction time of the presentation information in the same content. Here, the navigation information is repeatedly multiplexed so that no matter when content switching is performed by the reception apparatus, the reproduction apparatus will definitely be able to receive the navigation information for the content which is the link destination.

The multiplexed stream described above is generated as a part of a transport stream in accordance with DVB-SI (Digital Video Broadcasting) and MPEG system standards. This transport stream is a collection of a plurality of digital data streams which are multiplexed into the bandwidth of one carrier wave on a digital satellite broadcast, and so has a bandwidth which is equivalent to five or six broadcast channels.

The multiplexing information storage unit 5104 stores a variety of parameters that are needed by the data multiplexing unit 5103 to generate the multiplexed stream.

The system information table generating unit 5105 refers to the multiplexing information storage unit 5104 and

generates system information tables (made up of a variety of tables) which are required by the reception apparatus 5121 to select a multiplexed stream.

The transmission unit 5106 multiplexes the multiplexed stream generated by the data multiplexing unit 5103 and the system information tables generated by the system information table generating unit 5105 into a transport stream which it then transmits. The system information tables referred to here are composed of a variety of tables which store information used to identify the multiplexed streams that express the interactive programs of the present invention in the transport stream.

1-2-1 Transmission Data Storage Unit 5102

As shown in Fig. 4, the transmission data storage unit 5102 includes a presentation information storage unit 5107, a navigation information storage unit 5108, and a construction information storage unit 5109. This transmission data storage unit 5102 stores the data for a plurality of contents which compose one interactive program (application) classified into presentation information and navigation information. The transmission data storage unit 5102 also stores a construction information table which shows the correspondence between sets of presentation information and sets of navigation information.

1-2-1-1 Presentation Information Storaae Unit 5107

The presentation information storage unit 5107 stores the presentation information which is the video data and audio data included in each content.

Figs. 6A and 6B show the scenes (frames) in the sets of video data used as the presentation information of the contents shown in Fig. 5.

The video data 5201 shown in Fig. 6A has the filename "Video0.m2v" and expresses video which gives a weather forecast for the Kansai region of Japan. This video data is presentation information which is used by both content 0 and content 1 shown in Fig. 5.

The video data 5202 shown in Fig. 6B has the filename "Video1.m2v" and expresses video which gives a weather forecast for the Kanto region of Japan. This video data is presentation information which is used by both content 2 and content 3 shown in Fig. 5.

These sets of video data 5201 and 5202 are stored in the presentation information storage unit 5107 having been compressed according to IS/IEC 13818-2 (MPEG2 Video) standard. However, other video data formats are also possible.

Figs. 6C and 6D show examples of sets of audio data which are used as presentation information.

Audio data 5203 shown in Fig. 6C has the filename "Audio0.m2a" and is the audio data that is to be reproduced with the video data 5201 shown in Fig. 6A. This audio data is presentation information which is used by both content 0 and content 1 shown in Fig. 5.

Audio data 5204 shown in Fig. 6D has the filename "Audio1.m2a" and is the audio data that is to be reproduced with the video data 5202 shown in Fig. 6B. This audio data is presentation information which is used by both content 2 and content 3 shown in Fig. 5.

These sets of audio data are stored in the presentation information storage unit 5107 having been compressed according to IS/IEC 13818-3 (MPEG2 Audio) standard. However, other video data formats are also possible.

1-2-1-2 Navigation Information Storage Unit 5108

The navigation information storage unit 5108 stores the navigation information for each content. These sets of navigation information include hyperlink information for links to other contents and valid time information for the valid time of the present set of navigation information. The hyperlink information is given as objects to be used by the reception apparatus to enable the user to make interactive operations. The valid time information, meanwhile, is added to enable the content of the valid navigation information to be updated (expressed using the concept of a "version up") in accordance with changes in the content of the video data or other presentation information.

Fig. 7 shows an example of the navigation information corresponding to the scene 01b (or, more correctly, scenes 01b to 01d) shown in Fig. 5. This navigation information has the filename "Navi0-0.nif" and includes the navigation information 5301, the object definition table 5302, the handler definition table 5303, the hyperlink table 5304, the bitmap table 5305, and the time information table 5306.

The object definition table 5302 is a list of information which shows the types and attributes of the objects which are to be displayed superimposed onto the video data included in the presentation information. More specifically, the object definition table includes the following columns.

The "object index" column shows the numbers used to identify each of the objects.

The "type" column shows the type of each object. As examples of types of objects, "button" denotes a button object which used to display a button which has an attached hyperlink, while "picture" denotes a picture object which is used to display a still image or text information. In the present example, the buttons 02b and 03b shown in Fig. 5 have been recorded as button objects with the object index numbers "0" and "1". On the other hand, the text information 13b shown in Fig. 5 has been recorded as a picture object.

The "X" column and "Y" column are used to record the X and Y coordinates of the standard display position of each object, such as buttons or pictures, on the display screen. The values in these columns are used, for example, to determine the display positions of the buttons 02b and 03b shown in Fig. 5.

The "handler" column is used to show the handler index which indicates the handler, out of the handlers stored in the handler definition table 5303, that corresponds to each object. These handlers are scripts, which is to say programs or instruction words which are written in a programming language that is executable by the reception apparatus 5121. In particular, handlers for button objects include content switching instructions that are executed by the reception apparatus 5121 when the corresponding button object is activated by a user operation.

The "normal bitmap" column is used to show the bitmap index number that is used to indicate the bitmap image (button image or picture representing still image information), out of the bitmap images in the bitmap table 5305, which is to be displayed at the standard display position given by the X and Y coordinates described above during the normal (non-selected state) display state. Here, the non-selected state is the state of a given button when it has not been selected by the reproduction apparatus.

The "focused bitmap" column is used to show the bitmap index number that is used to indicate the bitmap image (button image or picture representing still image information), out of the bitmap images in the bitmap table 5305, which is to be displayed at the standard display position given by the X and Y coordinates described above during the selected state. Here, the selected state is the state of a given button when it has been selected by the reproduction apparatus.

The handler definition table 5303 stores the handlers (scripts) which are indicated for each object in the object definition table 5302. More specifically, the handler definition table 5303 includes the following columns. The "handler index" column stores numbers (handler indexes) for identifying each handler. The "script" column shows the handler (script) corresponding to each handler index. In particular, the handlers corresponding to button objects include content switching instructions such as "goto_content (Hyperlink Index 0)" given in Fig. 7.

The hyperlink table 5304 stores the arguments for the content switching instructions in the handler definition table 5303. More specifically, the hyperlink table includes the following columns. The "hyperlink index" column stores values (hyperlink indexes) for identifying each hyperlink. The "content number" column stores hyperlink information which is the content number of the link destination used as the argument in a content switching instruction. As one example, the content switching instruction "goto_content (Hyperlink Index 0)" is effectively the same as the instruction "goto_content (content 1)", with this being the instruction which is executed by the reception apparatus 5121 when the corresponding button object is activated.

The bitmap table 5305 stores bitmap data for bitmap images indicated in the "normal bitmap" and "focused bitmap" columns of the object definition table 5302. More specifically, the bitmap table includes the following columns. The "bitmap index" column is used to store the values ("bitmap numbers") which are used to identify the bitmaps. The "bitmap data" column is used to store the bitmap data used to express buttons and text information which are displayed superimposed onto the presentation information. As one example, the button 02b in scene 01b of Fig. 5 is displayed using the "Details for Osaka" bitmap with the bitmap index 0 in the normal state and using the "Details for Osaka" bitmap (which has a different color density to make it stand out) with the bitmap index 1 in the selected state. In the same way, button 03b is displayed using the "See Kanto" bitmap with the bitmap index 2 in the normal state and using the "See Kanto" bitmap with the bitmap index 3 in the selected state.

The time information table 5306 stores the start_time which denotes the time at which the present navigation information becomes valid and the end_time which denotes the time at which the present navigation information ceases to be valid. These times are expressed as relative times (in units of one second) where the broadcasting start time of the interactive program is set at "0".

Fig. 8 shows an example of the navigation information which corresponds to scene 11b in Fig. 5. This navigation information 5401 has the filename "Navi1-0.nif" and includes the object definition table 5402, the handler definition table 5403, the bitmap table 5404, and the time information table 5405.

On the line of the object definition table 5402 with the object index number "0", the type of the object is "button", and the display coordinates of the top-left corner of the display of the object on the display screen are X=20, Y=400. When this button is activated, the handler with the handler index number "0" is activated. During the non-selected state, this button is displayed using the bitmap with the bitmap index number "0", while during the selected state, the button is display using the bitmap with the bitmap index number "1".

On the line of the object definition table 5402 with the object index number "1", the type of the object is "picture", and the display coordinates of the top-left corner of the display of the object on the display screen are X=300, Y=20. This object is displayed using the bitmap with the bitmap index number "2".

The handler definition table 5403 shows that when the handler with the handler index number "0" is activated, the script "goto_entry" will be executed. This script is an instruction which indicates a switching to a default content for the start of reproduction, which is to say the content which is to be reproduced first by the reproduction apparatus.

The bitmap table 5404 stores the bitmap data for the bitmaps with the bitmap index numbers "0", "1", and "2". Of these, the bitmap with the bitmap index number "2" is the text information 13b shown in Fig. 5.

The time information table 5405 shows that the navigation information 5401 becomes valid five seconds after the start of reproduction and ceases to be valid sixty-five seconds after the start of the reproduction. This is to say, navigation information 5401 will stop being used sixty-five seconds after the start of reproduction.

Other examples of navigation information stored in the navigation information storage unit 5108 are shown in Fig. 9 (navigation information 5501), Fig. 10 (navigation information 5601), and Fig. 11 (navigation information 5701). These sets of navigation information respectively correspond to scene 01e in content 0, to scene lle in content 1, and to scene 21b in content 2.

1-2-1-3 Construction Information Storage Unit 5109

The construction information storage unit 5109 stores a construction information table, which is a list of pairings of presentation information and navigation information which compose each content, and entry information. This entry information shows an entry content number of an entry content that is the content to be reproduced first when the reproduction of the interactive program is commenced by the reception apparatus 5121.

Fig. 12 shows an example of the construction information table stored by the construction information storage unit 5109. This construction information table 5801 shows a combination of video data, audio data, and navigation information for each content identified using content numbers. These content numbers are the numbers which are exclusively assigned for identification purposes to each of the contents stored in the transmission data storage unit 5102. Here, while content numbers are one-to-one assigned to contents, it is also possible in exceptional circumstances for them to correspond to a plurality of contents.

The content number "0" line of the construction information table 5801 shows that the content with the content number "0" is composed of the video data with the filename "Video0.m2v" and the audio data with the filename "Audio0.m2a" stored in the presentation information storage unit 5107, and the sets of navigation information identified by the filenames "Navi0-0.nif", "Navi0-1.nif", "Navi0-2.nif", "Navi0-3.nif", and "Navi0-4.nif" stored in the navigation information storage unit 5108. Here, each set of navigation information stored in the "navigation information" column is given in ascending order of valid start time. This is also the case for the other lines in the construction information table 5801.

Fig. 13 shows an example of the entry information stored in the construction information storage unit 5109. This entry information 5901 shows that the content number of the entry content of the application stored in the transmission data storage unit 5102 is "0".

1-2-2 Multiplexing Information Storage Unit 5104

The multiplexing information storage unit 5104 stores the multiplexing information table for the resource assigning information for identifiers and areas used when multiplexing the interactive program into an MPEG2 transport stream for broadcasting.

Fig. 14 shows an example of the multiplexing information table stored by the multiplexing information storage unit 5104. The multiplexing information table 6001 in this figure is a table which shows the various identifiers for the interactive program and its composite elements, as well as the various bit rates used in transmission.

In Fig. 14, the rows 6002 to 6005 for the "original_network_id", "transport_stream_id", "service_id", and "event_id" show the values of the identifiers assigned to the interactive programs when multiplexing the program into the MPEG2 transport stream used for broadcasting the interactive program. In a standard satellite digital broadcasting system, transmission of one or more MPEG2 transport streams is performed from a single satellite (network) using carrier waves on separate frequency bands. Here, each broadcast program is multiplexed into the MPEG2 transport stream having been assigned its own "original_network_id", "transport_stream_id", "service_id", and "event_id" in accordance with ETS 300 468 Standard (hereinafter, referred to as "DVB-SI Standard").

The original_network_id is a unique identifier which identifies the network.

The transport_stream_ID is a unique identifier which identifies the transport stream in a network.

The event_ID is a unique identifier which identifies one event on a transport stream. Here, an event is a collection of a number of components, and is the equivalent of the concept of a "program" which is used in conventional analog broadcasting.

A component is a stream (program element) identified by a PID (packet identifier) under IS/IEC 13818-1 Standard (MPEG2 system standard), and represents one compositional element of a program, such as video or audio. As one

example, each set of video data shown in Fig. 12, each set of audio data, and each collection of sets of navigation information in each content is a separate component.

In the present embodiment, a service is a collection of sequences of events, which is the equivalent of one channel in conventional analog broadcasting. The interactive program described above is one time segment on such a service.

A transport stream is a collection of a plurality of services. Here, bandwidth can be assigned to transport streams and services in a variety of ways, with, for example, each transport stream being assigned around 30Mbps and each service being assigned around 5Mbps. In such a case, each transport stream is the equivalent of six channels. The transfer rates used for the interactive programs transmitted as events, meanwhile, will greatly differ since different numbers of contents and differing amounts of video data are included.

Each program (event) transmitted using a digital broadcasting system in accordance with DVB-SI standard can be uniquely specified in every digital broadcasting system using a combination of the "original_network_id", "transport_stream_id", "service_id", and "event_id". The details of the "original_network_id", "transport_stream_id", "service_id", and "event_id" are given in the documentation for DVB-SI standard.

The "PMT_PID" column 6006 and "PCR_PID" column 6007 express the values of the PID which are assigned to the PMT (Program Map Table) and the PCR (Program Clock Reference). The PMT referred to here is one of the system information tables multiplexed into the transport stream and is a table that shows the correspondence between the various streams which express the video data and audio data (components) included in an event and the identifiers (packet identifiers: PIDs) of the packets used to transfer these components. The PCR is also one of the system information tables and is time information that is used as a standard in the digital broadcasting apparatus 5101 when data for each content is multiplexed into the multiplexed stream, as well as being used as the standard time information when each event is reproduced by the reception apparatus 5121.

The "NE_component(0)_Bitrate" column 6008 and the "NE_component(0)_pid" column 6009 show the values of the transfer rate and PID which are assigned to each component for transferring navigation information tables which are included in content 0. This is also the case for "NE_component(1)_Bitrate" onwards. Here, "NE" is an abbreviation for "Navigation Element".

The "VE_component(0)_Bitrate" column 6010 and the "VE_component(0)_pid" column 6011 show the values of the transfer rate and PID which are assigned to each component for transferring the video data corresponding to the component_tag number "0x00". This is also the case for "VE_component(1)_Bitrate" onwards. Here, VE is an abbreviation for "Video Element".

The "AE_component(0)_Bitrate" column 6012 and the "AE_component(0)_pid" column 6013 show the values of the transfer rate and PID which are assigned to each component for transferring the audio data corresponding to the component_tag number "0x00". This is also the case for "AE_component(1)_Bitrate" onwards. Here, AE is an abbreviation for "Audio Element".

It should be noted that in the present embodiment, the number of PIDs for transferring navigation information is kept equal to the number of contents so that the PIDs may be used to identify each set of navigation information, although the number of PIDs for transferring navigation information may be less than the number of contents, and may for example be "1". In such a case, a combination of a PID and another parameter (such as a "table_id_extension" under MPEG2 standards) may be used as the information for identifying each set of navigation information. This is also the case for the video data and audio data included in the presentation information which may each be identified by a combination of a PID and another parameter, such as the stream_id under MPEG2 standard. By doing so, even if the number of PIDs that may be used in each transport stream is limited to a given number, a number of contents which exceeds this given number may still be transmitted.

1-2-3 Data Multiplexing Unit 5103

The data multiplexing unit 5103 shown in Fig. 4 first (a) assigns a variety of identifiers to each of the contents stored in the transmission data storage unit 5102 (or in other words, generates a content identifier assigning table), (b) assigns version numbers to each set of navigation information (or in other words, generates a version number assigning table), (c) instructs the navigation information table generating unit 5111 to generate a navigation information table, (d) instructs the system information table generating unit 5105 to generate the system information tables, and (e) multiplexes the presentation information in accordance with these tables so that the reproduction times of the presentation information are aligned, as well as repeatedly multiplexing each set of navigation information corresponding to the presentation information for the period that the navigation information is valid. To do so, the data multiplexing unit 5103 includes a multiplexing control unit 5110, a navigation information table generating unit 5111, and a multiplexing unit 5112.

The process (c) described above is performed by the navigation information table generating unit 5111, while the processes (a), (b), and (e) are performed by the multiplexing control unit 5110. Process (d) meanwhile, is performed by the system information table generating unit 5105.

1-2-3-1 Multiplexing Control Unit 5110

The multiplexing control unit 5110 can be composed of a CPU (Central Processing Unit), a ROM (Read Only Memory) storing a program, and a RAM (Random Access Memory) used as a work area, and generates the content identifier assigning table and the version number assigning table (processes (a) and (b) above), as well as generating a multiplexing instruction for each set of presentation information and each set of navigation information in accordance with these tables, and informing the multiplexing unit 5112 of these multiplexing instructions (process (e) above). These multiplexing instructions include the various identifiers needed for multiplexing, the multiplexing start position in the transport stream, and the transfer rate, for each set of video data and audio data in the presentation information and for each set of navigation information.

In more detail, the process (e) involves the multiplexing control unit 5110 generating multiplexing instructions so that the reproduction times of each set of video data and audio data in the presentation information overlap. As one example, it may generate multiplexing instructions which set the multiplexing start positions of video data and audio data at the same time. For sets of navigation information, the multiplexing control unit 5110 may generate multiplexing instructions so that the sets of navigation information are repeatedly multiplexed during the reproduction period of the presentation information in the same content. This is to say, a plurality of multiplexing start positions are set for each set of navigation information, with multiplexing instructions being generated for each of these multiplexing start positions.

1-2-3-2 Multiplexing Control Unit 5110: (a) Generation of the Content Identifier Assigning Table

On being activated by the transmission unit 5106, the multiplexing control unit 5110 reads the construction information table and multiplexing information table stored in the construction information storage unit 5109 and in the multiplexing information storage unit 5104, and generates the content identifier assigning table.

Fig. 15 shows an example of a content identifier assigning table generated from the construction information table 5801 shown in Fig. 12 and the multiplexing information table 6001 shown in Fig. 14.

This content identifier assigning table 6101 is composed so that the values in the "original_network_id" column 6002, the "transport_stream_id" column 6003, the "service_id" column 6004, and the "event_id" column 6005 of the multiplexing information table 6001 are written into the "orig_nw_id" column 6103, the "ts_id" column 6104, the "VE_svc_id" column 6105, and the "VE_event_id" column 6106. In the same way, the values in the "service_id" column 6003 and the "event id" column 6005 are written into the "AE_svc_id" column 6108 and the "AE_event_id" column 6109. This is also the case for the "NE_svc_id" column 6111 and the "NE_event_id" column 6112.

Each set of video data is assigned a two-digit hexadecimal component tag in order starting from "0x00", with these values being written into the "VE_comp_tag" column 6107. As examples, video data "Video0.m2v" is assigned the component tag "0x00" and video data "Video1.m2v" is assigned the component tag "0x01", with these values being written into the "VE_comp_tag" column 6107.

These component tags are values which are freely one-to-one assigned to each PID, and are used to indirectly refer to each PID. In the present embodiment, the component tags with the value "N" correspond to the PIDs given by the "VE_component (N)_pid" in the multiplexing information storage unit 5104. This correspondence between PIDs and component tags is given in the PMT which is described later in this specification. By doing so, the reception apparatus 5121 is able to refer to the component tags written in the "descriptor" column of the PMT and so determine the PID, before using this PID to obtain the desired video or other data. Here, even if different values for the PIDs are written in the system information tables when the interactive program is multiplexed with other programs by the transmission unit 5106, the reception apparatus 5121 will still definitely be able to obtain the desired video data.

It should be noted here that if a component tag is not used, the values of the PIDs may be directly written into the "VE_comp_tag" and "AE_comp_tag" columns. When doing so, if the PIDs are rewritten using different values in the system information tables during the multiplexing into the transport stream by the transmission unit 5106, the values of the PIDs in these columns in the navigation information table may also be appropriately rewritten.

In the same way, each set of audio data is assigned a component tag which is written into the "AE_comp_tag" column 6110. As examples, audio data "Audio0.m2a" is assigned the component tag "0x00" and audio data "Audio1.m2a" is assigned the component tag "0x01", with these values being written into the "AE_comp_tag" column 6110.

A four-digit hexadecimal value is written into the "NE_id" (navigation information identifier) column 6113 of the content identifier assigning table 6101, with this value being incremented by one for each content number 6102.

It should be noted that the "VE_id" and "AE_id" in Fig. 15 are information which is used to identify page-based contents. In this first embodiment, the interactive program is assumed to be entirely composed of stream-based contents, so that the "VE_id" and "AE_id" columns are not used. The details of these columns are given in the second and following embodiments.

1-2-3-3 Multiplexing Control Unit 5110: (b) Generation of the Version Number Assigning Table

On completing the generation of the content identifier assigning table 6101, the multiplexing control unit 5110 generates the version number assigning table.

More specifically, the multiplexing control unit 5110 refers to the construction information table 5801 and assigns version numbers, which start at "0" and are incremented by one each time, to each set of navigation information with the same content number, starting from the in order from the first set of navigation information. It should be noted here that when the version number exceeds "31", the next assigned version number will be "0", with numbers incremented by "1" being used thereafter.

Fig. 16 shows an example of the version number assigning table. In this example, the sets of navigation information "Navi0-0.nif", "Navi0-1.nif", "Navi0-2.nif" ... are assigned the version numbers "0x00", "0x01", "0x02"... This is also the case for the navigation information in contents 1-3.

On completing the generation of the version number assigning table 6201, the multiplexing control unit 5110 instructs the navigation information table generating unit 5111 to generate the navigation information table.

1-2-3-4 Navigation Information Table Generating Unit 5111: (c)

On being instructed by the multiplexing control unit 5110 to generate the navigation information table, the navigation information table generating unit 5111 generates a navigation information table by replacing the content numbers of the link destinations in the hyperlink table with various identifiers which express each component that includes the contents which are the link destinations.

More specifically, the navigation information table generating unit 5111 reads the navigation information stored in the navigation information storage unit 5108, and, when a hyperlink table is included in the navigation information, refers to the content identifier assigning table generated by the multiplexing control unit 5110 using the information for the link destination given as a content number, changes the content numbers into various identifiers, and by doing so generates the navigation information table.

The navigation information table generating unit 5111 also stores the generated navigation information table in a storage area (not illustrated) as the navigation information table with the filename NVT (content number, version number). The navigation information table generating unit 5111 obtains this content number and version number by referring to the construction information table in the construction information storage unit 5109 and the version number assigning table in the multiplexing control unit 5110. When the read navigation information does not include a hyperlink table, the navigation information table generating unit 5111 stores the navigation information as it is in the storage area, changing only the filename.

Fig. 17 shows the generated navigation information table 6301 with the filename "NVT (0,0)". This navigation information table 6301 has been generated from the navigation information 5301 with the filename "Navi0.nif" shown in Fig. 7, and so corresponds to scene 01b shown in Fig. 5.

The navigation information table 6301 includes the object definition table 6302, the handler definition table 6303, the hyperlink table 6304, the bitmap table 6305, and the time information table 6306. With the exception of the filenames and the hyperlink table 6304, the content is the same as the navigation information 5301 shown in Fig. 7.

The hyperlink table 6304 is such that each content number in the hyperlink table 5304 of Fig. 7 has been converted to the various identifiers given in the content identifier assigning table 6101 shown in Fig. 15. The columns such as "orig_nw_id" in the hyperlink table 6304 are given as "-", with no identifiers having been entered. This shows that the contents belonging to the navigation information table 6301 have the same identifiers as the contents given as the link destinations, so that these do not need to be recorded in the table.

In the present example, the "Hyperlink Index 0" row shows that there is a link between scene 01b of content 0 shown in Fig. 5 to scene 11b of content 1. With the exception of the "NE_id" column, all of the entries in the "Hyperlink Index 0" row of the hyperlink table 6304 are "-", showing that the link destination, content 1, has the same images and audio as content 0, with only the navigation information table (NE_id) being different.

In the present example, the "Hyperlink Index 1" row shows that there is a link between scene 01b of content 0 shown in Fig. 5 to scene 21b of content 2. With the exception of the "VE_comp_tag", the "AE_comp_tag" and the "NE_id" columns, all of the entries in the "Hyperlink Index 1" row of the hyperlink table 6304 are "-", showing that the link destination, content 2, has different images (VE_comp_tag), audio (AE_comp_tag) and a different navigation information table (NE_id) to content 0.

Supposing here that the content which is the link destination belongs to a different service, the appropriate identifiers will be given in the "VE_service_id", the "AE_service_id", and the "NE_service_id". However, by omitting these identifiers when the values for the link destination are the same as those for the current content as in the example above, a reduction in the size of the navigation information table can be achieved.

It should be noted that the "VE_id" and "AE_id" columns in the hyperlink table in Fig. 17 include information used

for identifying page-based contents. For the navigation information table NVT (0,0), all of the link destinations are stream-based contents, so that no entries are made into the "VE_id" and "AE_id" columns. The case where page-based columns are included as link destinations is explained in the second and following embodiments.

Fig. 18 shows the navigation information table 6401 with the filename NVT (1,0). This navigation information table 6401 has been generated from the navigation information 5401 with the filename "navi1-0.nif" shown in Fig. 8, and so corresponds to scene 11b in content 1 shown in Fig. 5.

Since navigation information 5401 does not include a hyperlink table, the content of navigation information table 6401 is the same as navigation information 5401. However, the link from scene 11b of content 1 to content 0 is expressed by the handler definition table 6403 in Fig. 18 and the entry information shown in Fig. 13.

In the same way, the navigation information table 6501 with the filename NVT (0,1) is shown in Fig. 19, the navigation information table 6601 with the filename NVT (1,1) is shown in Fig. 20, and the navigation information table 6701 with the filename NVT (2,0) is shown in Fig. 21. These have been respectively generated from the navigation information 5501 with the filename "navi0-1.nif" shown in Fig. 9, from the navigation information 5601 with the filename "navi1-1.nif" shown in Fig. 10, from the navigation information 5701 with the filename "navi2-0.nif" shown in Fig. 11.

On completing the generation of the navigation information table, the navigation information table generating unit 5111 informs the multiplexing control unit 5110. On receiving notification of the completion of the generation of the navigation information table, the multiplexing control unit 5110 instructs the system information table generating unit 5105 to generate the system information tables. The generation of the system information tables (d) is described later in this specification.

1-2-3-5 Multiplexing Control Unit 5110: Generation of Multiplexing Instructions

On receiving notification of the completion of the generation of the system information tables, the multiplexing control unit 5110 first reads the value of "PCR_PID" from the multiplexing information storage unit 5104 and notifies the multiplexing unit 5112. This action is performed so that the multiplexing unit 5112 can multiplex the time information (PCR), which is set as a standard when multiplexing each set of content data into the multiplexed stream.

Next, the multiplexing control unit 5110 generates multiplexing instructions for the presentation information and sends these instructions to the multiplexing unit 5112.

More specifically, the multiplexing control unit 5110 generates multiplexing instructions for the video data and the audio data included in all of the contents with a multiplexing start position of "0", so that the presentation information in all of the contents will be multiplexed with overlapping reproduction times. This reproduction start time is a relative time with the transmission start time being set at "0".

Each multiplexing instruction for video data and audio data includes a multiplexing start position, a PID, and a bit rate. As one example, for the video data "Video0.m2v" of content 0 in the construction information table 5801, the multiplexing control unit 5110 refers to the content identifier assigning table 6101 and reads the value "0x00" of the "VE_comp_tag" 6107 of this video data. The multiplexing control unit 5110 then refers to the multiplexing information table 6001 and reads the value "0x0096" of the "VE_component(0)_pid" 6011 to obtain the PID of this video data and reads the value "4Mbps" as the "VE_component(0)_Bitrate" 6010. The multiplexing control unit 5110 then informs the multiplexing unit 5112 of this PID and this bit rate in addition to the multiplexing start position.

The multiplexing control unit 5110 next generates multiplexing instructions for navigation information according to the process described below, before notifying the multiplexing unit 5112 of these instructions.

The multiplexing control unit 5110 generates multiplexing instructions for each content so that the navigation information tables included in each content will be repeatedly multiplexed during their valid time periods. As one example, the multiplexing control unit 5110 repeatedly generates multiplexing instructions for the navigation information table 6301 (NVT(0,0)) shown in Fig. 17 during its valid period which, as shown by the time information table 6306 is from the start_time (5 seconds) to the end_time (65 seconds). In the present embodiment, however, the navigation information tables are multiplexed at a predetermined time (such as one second) before their valid start_times. Navigation information tables are multiplexed this predetermined time before their valid start_times to give the reception apparatus 5121 enough of a margin to process the navigation information tables.

The multiplexing instructions for navigation information tables each include a multiplexing start position, a PID, a transfer amount (bit rate), a version number, and a table_id_extension.

As one example, when multiplexing the navigation information table with the filename "NVT(0,0)" shown in Fig. 17, the multiplexing control unit 5110 sets the multiplexing start position the predetermined time before the valid start_time (resulting here in a time of four seconds), reads the value "0x0092" of the "NE_component(0)_pid" 6009 and the value "1Mbps" of the "NE_component(0)_Bitrate" 6008 from the multiplexing information table 6001, and informs the multiplexing unit 5112 of these values as the PID and the bit rate. Also, the multiplexing control unit 5110 reads the value "0x0000" of the "NE_id" 6113 corresponding to content number 0 from the content identifier assigning table 6101 and informs the multiplexing unit 5112 of this value as the table_id_extension.

The multiplexing control unit 5110 then calculates the next multiplexing start position by dividing the transfer rate (bit rate) used for transferring the present navigation information by the size of present navigation information table, and generates the next multiplexing instruction as described above.

The multiplexing control unit 5110 repeats the above process, successively finding the next multiplexing start positions, generating multiplexing instructions, and informing the multiplexing unit 5112 of the multiplexing instructions, until the valid end_time is reached. By doing so, the navigation information table NVT (0,0) is repeatedly multiplexed into the multiplexed stream between the four-second mark and the sixty-five-second mark.

By repeating the processing described above, the multiplexing control unit 5110 generates multiplexing instructions for the other navigation information tables NVT (0,1), NVT(0,2) ... included in content 0, the navigation information tables NVT(1,0), NVT(1,1) ... included in content 1, and so on, and informs the multiplexing unit 5112 of these multiplexing instructions.

1-2-4 System Information Table Generating Unit 5105: (d)

On being instructed by the multiplexing control unit 5110, the system information table generating unit 5105 generates the system information tables. These system information tables are made up of a variety of tables which store information that is used to identify multiplexed streams in the transport stream, which is to say various kinds of information used by the reception apparatus 5121 to select events.

More specifically, the system information table generating unit 5105 refers to the multiplexing information storage unit 5104 and generates the NIT (Network Information Table), the EIT (Event Information Table), the SDT (Service Description Table), and the PAT (Program Association Table), in accordance with ETS 300 468 (DVB-SI) standard and IS/IEC 13818-1 (MPEG2 system) standard.

The NIT referred to here is used to record physical information related to the transfer path for each transport stream transferred on a specified network. Fig. 22A shows an example of an NIT, NIT 6801, which is generated by the system information table generating unit 5105. In this example, the transport stream identified by the transport_stream_id "0x0001" for the original_network_id "0x0001" is transmitted on the network identified by the network_id "0x0001", with the "transfer preface" expressing the frequency and modulation method of the transmission.

The SDT stores information, such as service names, for each service included in a specified transport stream. An example, SDT 6802, of the SDT generated by the system information table generating unit 5105 is shown in Fig. 22B. In this example, the service identified by the service_id value "0x0002" is included in the transport stream with the transport_stream_id "0x0001", with information such as the service names being written into the column headed "Service name and other information".

The EIT stores information, such as event names, start times, and end times, for each of the events on a specified service. An example, EIT 6803, of the EIT generated by the system information table generating unit 5105 is shown in Fig. 22C. In this example, the event identified by the event_id "0x0002" on the service identified by the service_id "0x0002" is included, with information such as the event name being written into the column headed "Event name and other information".

The PAT includes information for the PIDs of the PMT (Program Map Table) for each program included in a specified transport stream. An example, PAT 6901, of the PAT generated by the system information table generating unit 5105 is shown in Fig. 23. In this example, the program identified by the program_no "0x0002" is included in the transport stream with the transport_stream_id "0x0001", with the PID of this PMT being given as "0x0090". Here, the program_no matches the service_id, and a "program" is equivalent to an "event".

The system information table generating unit 5105 refers to the multiplexing information storage unit 5104, the multiplexing control unit 5110, and the construction information storage unit 5109 and generates, in accordance with MPEG2 system standards, the PMTs corresponding to the multiplexed programs which use the transmission data stored in the transmission data storage unit 5102. An example of a PMT generated by the system information table generating unit 5105 from the multiplexing information table 6001 shown in Fig. 14, the content identifier assigning table 6101 shown in Fig. 15, and the entry information 5901 shown in Fig. 13, is shown in Fig. 24.

During the generation of PMT 7001, "program_number" is a value showing the program number of a program (or in other words, event) in which transmission data is multiplexed, with the value "0x0002" of the "service_id" 6004 in the multiplexing information table 6001 being extracted and written in as this "program number".

The "PCR_PID" 20 is a value showing the PID of the packet which includes the clock information (PCR) which is used as the standard for decoding the present program. In the present example, the value "0x0091" of the "PCR_PID" in multiplexing information table 6001 is extracted and is written in as this "PCR_PID".

The "Entry_Descriptor" 7003 is the descriptor which includes information for the identifier of an entry content which is the first content to be reproduced when the present program is selected. Fig. 25 shows the details of the "Entry Descriptor" included in PMT 7001. Here, the descriptor_tag of "Entry_Descriptor" 7003 is an identifier showing the type of descriptor and is set a value such as "0x98" that is predetermined for an entry_descriptor. The "entry_VE_comp_tag",

the "entry_AE_comp_tag" and the "entry_NE_id" columns are used to show the values of the identifiers which are used for the image data, audio data, and navigation information which compose the entry content.

In generating the PMT, the system information table generating unit 5105 refers to the construction information storage unit 5109 and obtains the content number "0" of the entry content. The system information table generating unit 5105 then obtains the value "0x00" of the "VE_comp_tag" 6107, the value "0x00" of the "AE_comp_tag" 6110, and the value "0x0000" of the "NE_id" 6113 of the content whose content number is "0", and writes these values into the "entry_VE_comp_tag", the "entry_AE_comp_tag", and the "entry_NE_id" columns.

The table 7004 in the PMT 7001 shows the "stream_type" 7006 indicating the type of data which is transmitted in each component and a "descriptor" 7007 which expresses additional information, for each value of the "PID" 7005 of the components which compose the present program. The first row of table 7004 is used to record the value "0x0092" of the "NE_component(0)_pid" read from the multiplexing information table 6001, the value "0x05" showing that the data type of the transferred data is section data, and the "NE_Component_Descriptor(0)" 7201 shown in Fig. 26A. This "NE_Component_Descriptor(0)" shows that navigation information which has a value of NE_id which is equal to or above the "min_NE_id" and equal to or less than the "max_NE_id" is transferred using the component to which this descriptor is attached. In the present embodiment, the component identified by the "NE_component(0)_pid" 6009 is used to multiplex the navigation information for the content with the content number 0, so that the value "0x0000" of the "NE_id" 6113 read from the content identifier assigning table 6101 corresponding to the content number 0 is written into the "min_NE_id" and into the "max_NE_id". A value showing the type of descriptor (in this case "0x99"), is written into the "descriptor_tag".

On the second to fourth rows of table 7004, values of the "NE_component(1)_pid", "NE_component(2)_pid", and "NE_component(3)_pid" read from the multiplexing information table 6001 are written into the "PID" column, with the "stream_type" being set at "0x05" and the NE_Component_Descriptor(1) 7202 shown in Fig. 26B, the NE_Component_Descriptor(2) 7203 shown in Fig. 26C, and the NE_Component_Descriptor(3) 7204 shown in Fig. 26D being set in the "descriptor" column.

On the fifth row of table 7004, the value "0x0096" of the "VE_component(0)_pid" 6011 read from the multiplexing information table 6001, the value "0x02" showing that the data type of the transferred data is image data, and the "stream_identifier_descriptor(0)" 7301 shown in Fig. 27A are recorded. The "stream_identifier_descriptor(0)" 7301 shows that the component tag of the component for this PID is "0x00". The value of the "descriptor_tag" is set a value, such as "0x52", showing the type of descriptor.

On the sixth row of table 7004, the value of the "VE_component(1)_pid" read from the multiplexing information table 6001, the value "0x02" for the "stream type", and the "stream_identifier_descriptor(1)" 7301 shown in Fig. 27B are recorded.

On the seventh and eighth rows of table 7004, the values of the "AE_component(0)_pid" and the "AE_component (1)_pid" read from the multiplexing information table 6001, the value "0x03" for the "stream type" showing that the data is audio data, and the "stream_identifier_descriptor(0)" and "stream_identifier_descriptor(1)" are recorded.

1-2-5 Multiplexing Unit 5112

Based on the multiplexing instructions sent from the multiplexing control unit 5110, the multiplexing unit 5112 multiplexes the content data into an MPEG2 transport stream according to a method which is standardized for MPEG2 system standard. The multiplexing unit 5112 then successively outputs the generated transport stream data to the transmission unit 5106.

In more detail, on receiving a multiplexing instruction for image data from the multiplexing control unit 5110, the multiplexing unit 5112 reads the image data from the presentation information storage unit 5107 and converts it into a data stream, before multiplexing this data stream into the transport stream starting from the indicated start position using the indicated PID and bit rate. Similarly, on receiving a multiplexing instruction for audio data from the multiplexing control unit 5110, the multiplexing unit 5112 reads the audio data from the presentation information storage unit 5107 and converts it into a data stream, before multiplexing this data stream into the transport stream starting from the indicated start position using the indicated PID and bit rate.

On receiving a multiplexing instruction for a navigation information table from the multiplexing control unit 5110, the multiplexing unit 5112 reads the navigation information table from the navigation information table generating unit 5111 and converts it into a data stream, before multiplexing this data stream into the transport stream starting from the indicated start position using the indicated PID, table_id_extension, version_no, and bit rate.

As for the PCR, the multiplexing unit 5112 sets the initial value at the start of the generated transport stream at "0", and multiplexes the PCR using the PCR_PID sent from the multiplexing control unit 5110.

Fig. 28 shows an example of the multiplexed stream generated by the multiplexing unit 5112. The horizontal axis in this figure represents elapsed time, while the vertical axis represents the content data and PCR which are multiplexed at the same time.

The element 7401 in Fig. 28 shows the video data stream that is the result of the conversion by the multiplexing unit 5112 of the video data "Video0.m2v" which is common to both content 0 and content 1 into a data stream. This video data stream has been given the PID "0x0096". This video data stream 7401 is shown as one consecutive data stream in Fig. 28, although in reality this data stream is divided into packets of a predetermined length (these packets being called 188-byte transport packets) by the multiplexing unit 5112 and being multiplexed with the allocated bit rate of 4Mbps. In the same way, element 7402 is the video data stream which is shared by content 2 and content 3.

Element 7403 in Fig. 28 shows the audio data stream that is the result of the conversion of the audio data "Audio0.m2a" which is shared by content 0 and content 1 into a data stream by the multiplexing unit 5112. This audio data stream has been additionally assigned the PID "0x0098". This audio data stream is multiplexed using the assigned bit rate (0.5Mbps). In the same way, element 7404 is the audio data stream which is commonly used by the content 2 and content 3.

Element 7405 is a data stream which is used for transmitting the navigation information tables included in content 0. This data has been multiplexed by the multiplexing unit 5112 using the assigned bit rate (1Mbps). Each navigation information table in the data stream 7405 is assigned the PID "0x0092", the table_id_extension "0x0000", and a version number from "0x00" to "0x04". These navigation information tables are multiplexed so that navigation information tables with the same version number are multiplexed a plurality of times, with the version number being progressively incremented as the reproduction time elapses. Here, the PID, table_id_extension, and version no are used by the reception apparatus 5121 to identify each navigation information table in the data stream 7405. In the same way, elements 7406 to 7408 are data streams used to transfer the navigation information included in contents 1 to 3.

Element 7409 in Fig. 28 is the time information (PCR) used as the standard for setting the reproduction time, which is also multiplexed into the transport stream.

It should be noted here that every time the multiplexing unit 5112 receives a multiplexing instruction from the multiplexing control unit 5110, it may perform a multiplexing operation and generate a multiplexed stream in an intermediate state which is then stored in a storage unit (not illustrated). After completing the processing of all of the multiplexing instructions, the multiplexing unit 5112 may output the completed multiplexed stream to the transmission unit 5106. Alternatively, instead of immediately processing the multiplexing instructions, the multiplexing unit 5112 may store these instructions in a storage unit (not illustrated), before sorting the necessary multiplexing instructions and performing multiplexing in order of reproduction time to generate a multiplexed stream, which it may then successively output to the transmission unit 5106.

1-2-6 Transmission Unit 5106

The transmission unit 5106 includes a scheduler, and is activated by the multiplexing control unit 5110 at a predetermined time before the transmission start time of an event, such as five minutes before the start of the transmission. When the transmission start time is reached, the transmission unit 5106 repeatedly multiplexes information such as the NIT, PAT, PMT, SDT, and EIT generated by the system information table generating unit 5105 into the transport stream outputted by the multiplexing unit 5112 at a predetermined interval using predetermined PIDs in accordance with DVB-SI standard and MPEG2 system standard. The transmission unit 5106 then performs modulation and other processes, before transmitting the data to a plurality of data reception apparatuses 5121.

Fig. 29 gives a model representation of a transport stream multiplexed by the transmission unit 5106. In this example, the NIT, the PAT, the PMT, the SDT, and the EIT have been additionally multiplexed into the transport stream multiplexed by the multiplexing unit 5112. In reality, a plurality of events have also been multiplexed into this transport stream by the transmission unit 5106, although only the event (interactive program) shown in Fig. 5 has been shown in Fig. 29.

1-2-7 Operation of the Data Broadcasting Apparatus 5101

The following is a description of the operation of the data transmission apparatus 5101 in the present embodiment, which is constructed as described above.

Fig. 30 is a flowchart showing the entire operation of the digital broadcasting apparatus 5101.

The multiplexing control unit 5110 first generates the (a) content identifier assigning table (S7602) and then generates the (b) version number assigning table (S7604). After this, the multiplexing control unit 5110 gives an indication for the generation of the (c) navigation information tables (S7606) and the generation of the (d) system information tables (S7608). Once the system information tables have been generated by the system information table generating unit 5105, the multiplexing control unit 5110 reads the value of the "PCR_PID" from the multiplexing information storage unit 5104 and notifies the multiplexing unit 5112 of this value (S7610).

After this, the multiplexing control unit 5110 instructs the multiplexing unit 5112 to multiplex the presentation information (S7611), and instructs the multiplexing unit 5112 to multiplex the navigation information (S77-0, S77-1, ... S77-n).