

-1-

Specification

Title of the Invention:

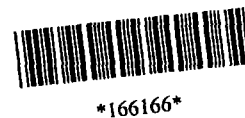
Error Correction System, Error Correction Method, and Data Memory with Error Correction Function

Detailed Description of the Invention:

[0001]

[Field of the invention]

The present invention relates to an error correction system and a data memory with the error correction function, and in particular, to a technique for reducing the time necessary for error correction processing by increasing data transfer efficiency in an error correction system using error correction codes in a product code format such as found in a DVD (Digital Versatile Disk).



-2-

[0002]

[Background art]

Generally, a DVD typified by an optical disk uses an error correction technique using error correction codes in a product code format, and a Reed-Solomon code is used as the error correction code. Figure 1 shows a basic data format using error correction codes in the product code format. Data bytes are arranged in k_1 rows $\times$ k_2 columns. As is well-known in the art, in the row (horizontal) direction, row error correction codes called PI codes (Parity-Inner Codes) or Inner codes are added, and in the column (vertical) direction, column error correction codes called PO codes (Parity-Outer Codes) or Outer codes are added. An array 1 of data bytes with the PI codes and PO codes being added is called "ECC block." As is well-known, the PI codes and PO codes are generated in accordance with code generation polynomials based on row data and column data.

[0003]

A format of the ECC block is defined in the DVD standard: k_1 is 192 rows and k_2 is 172 bytes; the PI of each row is 10 bytes and the PO is 16 rows. Therefore, length Y of the ECC block in the row direction is 182 bytes, and length Z of the ECC block in the column direction is 208 rows. Thus, each of 208 rows of the ECC block includes 182 bytes, and each of 182 columns of the ECC block includes 208 bytes. The ECC block includes 16 sectors S1-S16, and each sector includes 12 byte-rows L1-L12. Each of the byte-rows L1-L12 includes 172 data bytes and 10 PI bytes. The PO includes 16 PO byte-rows PO1-PO16.

[0004]

Figure 2 schematically shows the ECC block recording format in which the ECC blocks are recorded on data tracks of a DVD. 16 PO byte-rows PO1-PO16 in Figure 1 are distributed to 16 sectors

-3-

respectively. For example, the sector S1 includes combination of 12 byte-rows L1-L12 and the associated PO byte-row PO1, and the sector S16 includes combination of 12 byte-rows L1-L12 and the associated PO byte-row PO16 (not shown).

[0005]

The ECC block in Figure 1 is converted into the recording format in Figure 2, and is recorded on a DVD. Generally, the ECC block in Figure 1 is converted into the recording format in Figure 2, and a frame synchronization signal is added to the recording format, which is 8/16-modulated (EFM plus modulation) and is recorded on the DVD. When it is read, a signal read from the DVD is 8/16-demodulated, the frame synchronization signal is removed, and the read signal is reassembled in the ECC block format of Figure 1.

[0006]

In error correction processing, first, error correction is performed in the row direction, and erasure correction is performed in the column direction if error bytes the number of which exceeds a predetermined number are detected in the row direction. In a case of a DVD, Hamming distance of the Reed-Solomon code in the row direction is 11, and Hamming distance of the Reed-Solomon code in the column direction is 17. The maximum number of error correction in the row direction is 5 bytes, and the maximum number of error correction in the column direction is 8 bytes. Nevertheless, the maximum number of error correction in the row direction is generally limited to a number less than 5, for example, 3 bytes, and, if more than 3 error bytes are detected, erasure processing in the column direction is performed.

[0007]

Next, error correction procedure in a prior DVD system will be described with reference to Figure 3. ECC block data is stored

on a DVD 11 such as an optical disk in the recording format described in regard to Figure 2. The disk 11 is rotated by a spindle motor 12. A read head 13 moving on a guide bar 14 reads recorded data and servo information from the DVD 11, and sends these to a read circuit 16. The read circuit 16 sends the recorded data and servo information, which are read, to a formatter 20. Rotation of the spindle motor 12 and movement of the read head 13 on the guide bar 14 are controlled by a digital servo processor 18 responding to the servo information.

[0008]

In Figure 3, broken lines and numbers inside parentheses that are shown near the broken lines show the sequence of data transfer and processing.

[0009]

(1) The formatter 20 performs 8/16 demodulation of the signal that is read, removes the frame synchronization signal, and generates an ECC block in the format of Figure 1. The formatter 20 sequentially transfers byte-rows of the ECC block of Figure 1 (hereinafter, these byte-rows are called ECC block rows) L1, L2, L3, ..., P015, P016 to DRAM 22 functioning as a first buffer memory. The DRAM 22 is used as a working memory for holding the ECC block, storing the ECC block after correction if an error is detected, and transferring the ECC block after correction to a host PC (personal computer).

(2) When one or more ECC block rows are transferred to the DRAM 22, the first ECC block row is transferred to an SRAM 24 functioning as a second buffer memory. The SRAM 24 is a high-speed buffer having comparatively small capacity, and is used for efficiently performing error correction processing. The ECC block row transferred to the SRAM 24 includes 172 data bytes and 10 PI bytes.

(3) Next, the ECC block row transferred to the SRAM 24 is

-5-

transferred to an error correction code decoder 26. The decoder 26 includes a syndrome generator 28 and an error correction circuit 30. The syndrome generator 28 generates a 10-byte syndrome based on the data of the ECC block row supplied from the SRAM 24. The error correction circuit 30 obtains error positions and error values of the ECC block row based on the generated syndrome, and corrects the error bytes if the number of the error bytes is three or less.

(4) Then, the decoder 26 sends the error-corrected bytes to an MPU 32.

(5) The MPU 32 sends the error-corrected bytes, sent from the decoder 26, to the DRAM 22 to rewrite the error bytes of the corresponding ECC block row in the DRAM 22 with the corrected bytes.

[0010]

The above procedure is sequentially performed in regard to all the ECC block rows. If error bytes the number of which exceeds three are detected in the decoder 26, an erasure pointer is set. If the erasure pointer is set, byte-columns of the ECC block (hereinafter, these are called ECC block columns) in the DRAM 22 are sequentially transferred to the SRAM 24 after completion of error correction in the row direction. In the column direction, a 16-byte syndrome is generated by the syndrome generator 28, and similarly to the error correction in the row direction, error correction processing is performed in the column direction. In order to perform the error correction processing in the column direction, it is necessary to store the entire ECC block in the DRAM 22 beforehand.

[0011]

(6) When all the error correction processing is completed, the ECC block that does not include an error exists in the DRAM 22. The data of the ECC block in the DRAM 22 is sent to the host PC

-6-

(personal computer) via a cable 34.

[0012]

[Problems to be solved by the invention]

A syndrome for error correction in the row direction is generated based on a 182-byte ECC block row including 172 data bytes and 10 PI bytes. Since the prior system described above has the syndrome generator 28 in the downstream of the SRAM 24, it is necessary to transfer the entire 182-byte ECC block row from the DRAM 22 to the SRAM 24, and next, to decoder 28. Therefore, this system has a problem that the system needs much time to transfer the data such that error correction processing cannot be efficiently performed.

[0013]

In the error correction processing, it is preferable to execute the processing in an on-the-fly mode, that is, in a real time mode without making the data transfer to the host waited. Although the on-the-fly could be realized even with the prior system described above by increasing processing efficiency with the use of, for example, pipeline-structured parallel processing, it is preferable to basically reduce processing time. If the time required for the data transfer between the DRAM 22 and decoder 26 can be reduced, it becomes possible to reduce the occupied time of an internal bus 36 due to data transfer, and hence it becomes possible to increase the time when the MPU 32 can use the internal bus 36, increasing the processing efficiency of the entire data memory.

[0014]

In addition, the prior system has a problem that, since the ECC block format is fixed and the length of the ECC block row (182 bytes) is not a power of 2, unused areas are left in the DRAM buffer 22, and hence the use efficiency of the buffer memory is

low.

[0015]

:

Therefore, an object of the present invention is to provide an error correction system and an error correction method that can reduce the time necessary for error correction processing by increasing the data transfer efficiency.

[0016]

Another object of the present invention is to provide an error correction system that can reduce the time necessary for error correction processing and can increase the use efficiency of a buffer memory.

[0017]

Still another object of the present invention is to provide a data memory that has error correction capability with using error correction codes in a product code format and can reduce the time necessary for error correction processing.

[0018]

[Summary of the invention]

According to an aspect of the present invention, an error correction system of the present invention includes a formatter generating an ECC block that includes at least data arranged in rows and columns and row error correction codes for the respective rows, a syndrome generator generating a syndrome based on each of the ECC block rows supplied from the formatter, a first buffer memory receiving and storing the ECC block rows and associated syndromes from the syndrome generator, a second buffer memory receiving and storing said syndromes from said first buffer memory and an error correction code decoder that receives syndromes from the second buffer memory and performs error correction processing for the ECC block rows in the first

-8-

buffer memory.

[0019]

The first buffer memory has a first and second banks, and each row of the banks has $2^n \times (2m+1)$ (here, n and m are positive integers) of byte length and is divided into $(2m+1)$ byte-blocks. The byte-length Y of the ECC block rows is $2^n \times 2m < Y < 2^n \times (2m+1)$, and each of the ECC block rows is stored by a block of 2^n byte-length alternately in the byte-blocks of the first and second banks in an interleave mode. A syndrome corresponding to each of the ECC block rows is stored in $[2^n \times (2m+1) - Y]$ bytes of empty area of a bank byte-block where the $(2m+1)$ block in the corresponding ECC block row is stored.

[0020]

According to another aspect of the present invention, a data memory of the present invention includes a storage medium storing sector data including data and error correction codes in a product code format, a formatter that formats sector data read from the storage medium and generates an ECC block including data arranged in rows and columns, a row error correction inner-code for each row, and a column error correction outer-code for each column, a syndrome generator that sequentially receives the ECC block rows from the formatter and generates a syndrome based on each of the ECC block rows, a first buffer memory receiving the ECC block rows and associated syndromes from the syndrome generator, a second buffer memory receiving the syndromes from the first buffer memory, and a decoder that receives the syndromes from the second buffer memory and performs error correction processing for the ECC block rows stored in the first buffer memory.

[0021]

[Preferred embodiment]

Next, the present invention will be described with reference to Figure 4. The same reference numbers will be assigned to the parts in Figure 4 that are the same as ones in Figure 3. A feature of the present invention is that arrangement is modified so that a syndrome generator 38 similar to the syndrome generator 28 included in the decoder 26 in the prior system of Figure 3 is added between the formatter 20 and DRAM buffer 22. The syndrome generator 38 sequentially receives ECC block rows from the formatter 20, transfers the received ECC block row to the DRAM buffer 22, and generates a syndrome for the ECC block row. Each ECC block row and its associated syndrome are stored together in the DRAM 22. The DRAM 22 transfers only the syndromes to the SRAM 24. The SRAM 24 transfers the syndromes, received from the DRAM 22, to the decoder 26. The error correction circuit 30 in the decoder 26 performs error correction processing in the row direction based on the syndrome received from the SRAM 24. The syndrome generator 28 in the decoder 26 is used for generating syndromes in the column direction, and the generator 28 is not used for generating syndromes in the row direction.

[0022]

Broken lines in Figure 4 and numbers inside parentheses that are shown near the broken lines show the sequence of data transfer and processing.

[0023]

(1) The formatter 20, similarly to the prior system, generates an ECC block in the format of Figure 1. The formatter 20 sequentially transfers the ECC block rows L1, L2, L3, ..., P015, P016 to a syndrome generator 38.

-10-

(2) Every time the syndrome generator 38 receives an ECC block row from the formatter 20, the syndrome generator 38 generates a 10-byte syndrome for the ECC block row. The syndrome generator 38 generates a syndrome for the received ECC block row while the syndrome generator 38 transfers the received ECC block row to the DRAM buffer 22 functioning as a first buffer memory. The generated syndrome is stored with the associated ECC block row in the DRAM 22.

(3) The DRAM 22 sequentially transfers only the syndromes to the SRAM 24 functioning as a second buffer memory.

(4) The SRAM 24 transfers the syndrome, received from the DRAM 22, to the error correction code decoder 26. The error correction circuit 30 of the decoder 26 obtains error positions and error values of the associated ECC block row on the basis of the received syndrome, and the circuit 30 corrects the error bytes if the number of the error bytes is three or less.

(5) Then, the decoder 26 sends the error-corrected bytes to an MPU 32.

(6) The MPU 32 sends the error-corrected bytes, sent from the decoder 26, to the DRAM 22 to rewrite the error bytes of the corresponding ECC block row in the DRAM 22 with the corrected bytes.

[0024]

If more than three error bytes are detected in the decoder 26, erasure correction processing is performed similarly to the prior processing of Figure 3. Thus, if the erasure pointer is set, byte-columns of the ECC block in the DRAM 22 are sequentially transferred to the SRAM 24 after completion of error correction in the row direction. In the column direction, a 16-byte syndrome is generated by the syndrome generator 28, and similarly to the error correction in the row direction, error correction processing is performed in the

-11-

column direction.

[0025]

(6) When all the error correction processing is completed, the ECC block that does not include an error exists in the DRAM 22. The data of the ECC block in the DRAM 22 is sent to the host PC (personal computer) via a cable 34.

[0026]

As is evident from the above description, in the present invention, syndromes in the row direction are generated at a position between the formatter 20 and DRAM buffer 22. The syndromes are stored with respective ECC block rows in the DRAM 22. Only the 10-byte syndrome is transferred from the DRAM 22 to SRAM 24 and decoder 26, and hence it becomes possible to largely reduce the data transfer time in comparison with the prior system where the entire 182-byte ECC block row is transferred from the DRAM 22 to SRAM 24 and decoder 26. It is to be noted that, since erasure correction processing in the column direction is performed in the same processing as the prior one, the time necessary for erasure correction is basically the same as that in the prior system.

[0027]

In the present invention, both of the ECC block and syndromes are stored in the DRAM buffer 22. According to the present invention, it becomes possible to increase use efficiency of the DRAM buffer 22. This is an additional feature of the present invention. Next, how the DRAM buffer 22 stores the ECC block and syndromes in the present invention will be described.

[0028]

An ECC block has a fixed data format that is defined in the standard, and each ECC block row has 182 bytes of length. On

-12-

the other hand, in the DRAM, it is usual that the length in the row direction is set in a power of 2. Therefore, there is a problem that the use efficiency of memory decreases since an empty or unused area is left if one ECC block row is stored in each row of the DRAM. In addition, addresses of the DRAM are continuous in the row direction, but the addresses are discontinuous in the column direction. Therefore, there is a problem that access speed in the column direction cannot be increased.

[0029]

Japanese Patent Application No. 10-15740 (1998) assigned to the same assignee as the present patent application proposes a technique wherein an ECC block is stored in two memory banks in an interleave mode so as to increase the use efficiency of the DRAM buffer and to make it possible to continuously access the DRAM buffer in both row and column directions. As the two banks, two SDRAM (synchronous DRAM) chips are used. Figure 5 shows memory data mapping disclosed in this Japanese Patent Application. An upper part of Figure 5 shows logical mapping of the ECC block in Figure 1, and a lower part shows physical mapping to the two memory banks (bank 0 and bank 1). L1, L2, L3,... in Figure 5 corresponds to the ECC block rows L0, L1, L3,... of the sector S1 in Figures 1 and 2. For simplification of description, only the ECC block rows L1-L4 are shown.

[0030]

Byte-rows of banks 0 and 1 are divided into an odd number of byte-blocks respectively. Each byte-block has the same length in the row direction. The length (byte number) of the banks 0 and 1 in the row direction is expressed in $2^n \times (2m+1)$. Here, n and m are positive integers. 2^n is a byte number of one byte-block, and it is 64 ($n=6$) in this example. $2m+1$ is a number of byte-blocks, and it is 3 ($m=1$) in this example. The length of the banks 0 and 1 in the row direction is 192 bytes. Assuming that a byte number (182 bytes in this example) of one

-13-

ECC block row is Y , the following relation is obtained.

$$2^n \times 2^m < Y < 2^n \times (2^m + 1)$$

[0031]

Respective rows $L_1, L_2, L_3, \dots$ of an ECC block are also divided into three byte-blocks each having 64 bytes of length. A first byte-block A_0 of the ECC block row L_1 is stored in a byte-block A_0 of the bank 0, a second byte-block B_0 of the ECC block row L_1 is stored in a byte-block B_0 of the bank 1, and a third byte-block A_1 of the ECC block row L_1 is stored in a byte-block A_1 of the bank 0. A first byte-block B_1 of the ECC block row L_2 is stored in a byte-block B_1 of the bank 1, and a second byte-block A_2 of the ECC block row L_2 is stored in a byte-block A_2 of the bank 0, and a third byte-block B_2 of the ECC block row L_2 is stored in a byte-block B_2 of the bank 1. Similarly, respective subsequent ECC block rows are stored in the interleave mode in the byte-blocks of the banks 0 and 1.

[0032]

When the ECC block row L_1 is fetched, byte-blocks A_0, B_0 , and A_1 are accessed between the banks 0 and 1 alternately, and when the ECC block row L_2 is fetched, byte-blocks B_1, A_2 , and B_2 are accessed between the banks 1 and 0 alternately. When an ECC block column in the left side is fetched, byte-blocks $A_0, B_1, A_3, B_4, \dots$ are alternately accessed between the banks 0 and 1, and, when an ECC block column in the center is fetched, byte-blocks $B_0, A_2, B_3, A_5, \dots$ are alternately accessed between the banks 1 and 0. It is to be noted that, in actual error correction processing in the column direction, since the ECC block is processed by the byte-column, the most left byte-column of the ECC block is fetched by selecting the most left byte of the byte-block columns $A_0, B_1, A_3, B_4, \dots$.

-14-

[0033]

In the memory data mapping in Figure 5, each bank has continuous addresses in the row direction. In addition, adjacent rows are stored in different banks, and each bank can be accessed in the bank interleave mode so that internal circuit delay caused by precharge may not be seen from the outside, and hence, even if addresses are changed to in the column direction (addition of 192), column data can be continuously fetched. Therefore, according to this method, it is possible to realize high-speed access in both row and column directions.

[0034]

In addition, length of each bank in the row direction is 192 bytes, and an ECC block row is 182 bytes of length. Therefore, only the bank blocks, storing the right-most byte-block columns A1, B2, A4, B5,... of the ECC block, stores 54-byte data. The use efficiency of the storage area in this mapping is $182/192=0.948$, and hence high use efficiency can be obtained.

[0035]

Therefore, according to the bank interleave mode of Figure 5, it is possible to obtain high-speed access and high use efficiency of a storage area. Nevertheless, each bank row has an empty or unused area whose length is $2^n \times (2m+1) - Y = 192 - 182 = 10$ bytes.

[0036]

The present invention not only uses the bank interleave mode of Figure 5 in the DRAM buffer 22 of Figure 4, but also stores syndromes in the unused areas whose length is 10 bytes per bank row. The syndrome generator 38 generates a 10-byte syndrome per ECC block row, and hence, by storing each ECC block row in the banks in an interleaved fashion and by storing an

-15-

associated 10-byte syndrome in a 10-byte unused area, it becomes possible to increase the use efficiency of the DRAM buffer 22 at 100%.

[0037]

Although specific embodiments are described above, it will be apparent for persons skilled in the art that various modifications can be performed within the scope of the present invention. For example, although error correction processing in the column direction is performed after error correction processing in the row direction in embodiments, the present invention can be applied also to a case that the error correction processing in the column direction is performed ahead. In this case, however, the error correction processing cannot be started until the entire ECC block is stored in the DRAM buffer 22. Since, in the case that the error correction processing in the row direction is performed ahead, the error correction processing can be started when one or more ECC block rows are stored in the DRAM buffer, this is more advantageous. Furthermore, although the specific data format structure is used in the embodiment, other data formats can be used.

[0038]

[Advantages of the invention]

In the present invention, a syndrome generator is arranged between a formatter and a first buffer memory, and only syndromes are transferred to a second buffer memory and a decoder for error correction. Therefore, it becomes possible to largely reduce the time necessary for data transfer, permitting high-speed error correction processing.

[0039]

In addition, by using the bank interleave mode in the first buffer memory and storing syndromes in unused areas of each

-16-

bank, it becomes possible to improve the use efficiency of the first buffer memory.

[0040]

Furthermore, since an additional memory and memory control is not necessary for storing syndromes, the present invention is economic. Although it is required to add the syndrome generator between the formatter and first buffer memory, the syndrome generator is constructed with comparatively simple gates and is inexpensive, and hence addition of the syndrome generator hardly affects the entire system cost.

4. Brief Description of the Drawings:

Figure 1 shows a data format of an ECC block used in a DVD;

Figure 2 shows a recording format of sector data recorded in a DVD;

Figure 3 shows a schematic diagram of the configuration of a prior DVD memory;

Figure 4 shows a schematic diagram of the configuration of a DVD memory according to an embodiment of the present invention; and

Figure 5 shows data mapping to a DRAM buffer.

[Description of symbols]

1 ... ECC block
11 ... DVD
12 ... Motor
13 ... Read head
16 ... Read circuit
18 ... Digital servo processor

-17-

20 ... Formatter
22 ... DRAM buffer (First buffer memory)
24 ... SRAM buffer (Second buffer memory)
26 ... Decoder
28 ... Syndrome generator
30 ... Error correction circuit
32 ... MPU
36 ... Data bus
38 ... Syndrome generator

Claims:

(1) An error correction system comprising:

a formatter generating an ECC block that includes at least data arranged in rows and columns and row error correction codes for the respective rows;

a syndrome generator generating a syndrome based on each of the ECC block rows supplied from said formatter;

a first buffer memory receiving and storing the ECC block rows and associated syndromes from said syndrome generator;

a second buffer memory receiving and storing said syndromes from said first buffer memory; and

an error correction code decoder that receives said syndromes from said second buffer memory and performs error correction processing for the corresponding ECC block rows in said first buffer memory.

(2) The error correction system according to claim 1, wherein said first buffer memory has first and second banks,

wherein each row of said banks has $2^n \times (2m+1)$ (here, n and m are positive integers) of byte length and is divided into $(2m+1)$ byte-blocks,

wherein byte-length Y of said ECC block rows is $2^n \times 2m < Y < 2^n \times (2m+1)$, and each of said ECC block rows is stored by the block of 2^n byte-length alternatively in said byte-blocks of said first and second banks in an interleave mode.

(3) The error correction system according to claim 2, wherein a syndrome corresponding to each of said ECC block rows is stored in $[2^n \times (2m+1) - Y]$ bytes of empty area of a bank byte-block where the $(2m+1)$ block in the corresponding ECC block row is stored.

(4) The error correction system according to claim 3, wherein said Y is 182 bytes long, n is 6, m is 1, and said syndrome is 10 bytes long.

(5) The error correction system according to any one of claims 1 to 4, wherein said ECC block includes a column error correction code for each column,

wherein said decoder performs error correction processing for ECC block columns in said first buffer memory if necessary, after the error correction processing of ECC block rows.

(6) An error correction method comprising the steps of:

generating a syndrome based on each of the ECC block rows of an ECC block which includes at least data arranged in rows and columns, and row error correction codes for the respective rows;

storing said ECC block rows and associated syndromes in a first buffer memory;

storing said syndromes in a second buffer memory; and

transferring said syndromes in said second buffer memory to an error correction code decoder and performing error correction processing for the ECC block rows in said first buffer memory.

(7) The error correction method according to claim 6, wherein said buffer memory has first and second banks,

wherein each row of said banks has $2^n \times (2m+1)$ (here, n and m are positive integers) of byte length and is divided into $(2m+1)$ byte-blocks, and

wherein byte-length Y of said ECC block rows is $2^n \times 2m < Y < 2^n \times (2m+1)$, and each of said ECC block rows is stored by the block of 2^n byte-length alternately in said byte-blocks of said first and second banks in an interleave mode.

(8) The error correction method according to claim 7, wherein a syndrome corresponding to each of said ECC block rows is stored in $[2^n \times (2m+1) - Y]$ bytes of empty area of a bank byte-block where the $(2m+1)$ block in the corresponding ECC block row is stored.

(9) The error correction method according to any one of claims 6 through 8, wherein said ECC block includes a column error correction code for each column,

wherein said decoder performs error correction processing for ECC block columns in said buffer memory if necessary after the error correction processing of ECC block rows.

(10) A data memory with error correction function, comprising:

a storage medium storing sector data including data and error correction codes in a product code format;

a formatter that formats sector data read from said storage medium and generates an ECC block including data arranged in rows and columns, row error correction codes for the respective rows, and column error correction codes for the respective columns;

a syndrome generator that sequentially receives the ECC block rows from said formatter and generates a syndrome based on each of the respective ECC block rows;

a first buffer memory receiving the ECC block rows and associated syndromes from said syndrome generator;

a second buffer memory receiving said syndrome from said first buffer memory; and

a decoder that receives said syndromes from said second buffer memory and performs error correction processing for the ECC block rows stored in said first buffer memory.

(12) The data memory according to claim 11, wherein said first buffer memory has first and second banks,

wherein each row of said banks has $2^n \times (2m+1)$ (here, n and m are positive integers) of byte length and is divided into $(2m+1)$ byte-blocks,

wherein byte-length Y of said ECC block rows is $2^n \times 2m < Y < 2^n \times (2m+1)$, and each of said ECC block rows is stored by the block of 2^n byte-length alternately in said byte blocks of the first and second banks in an interleave mode.

(13) The data memory according to claim 12, wherein said syndrome corresponding to each of said ECC block rows is stored in $[2^n \times (2m+1) - Y]$ bytes of empty area of a bank byte-block where the $(2m+1)$ block in the corresponding ECC block row is stored.

(14) The data memory according to claim 13, wherein said Y is 182 bytes long, n is 6, m is 1, and said syndrome is 10 bytes long.

(15) The data memory according to any one of claim 10 through 14, wherein said first buffer memory sequentially receives the ECC block rows from said formatter and stores 1 ECC block of data,

wherein said decoder performs error correction processing for ECC block columns stored in said first buffer memory if necessary, after the error correction processing of ECC block rows.