

DATA DE-ROTATOR AND DE-INTERLEAVER

Technical Field of the Invention

The present invention relates to an apparatus and method for de-rotating and/or de-interleaving data.

Background of the Invention

In an 8 VSB terrestrial communication system, information may be transmitted to a receiving station over the air. In an example of such a system, each two bits of data to be communicated is converted to a three-bit trellis code which is mapped as a symbol having eight possible levels so that each byte of data is represented by four symbols. The symbols are interleaved, and some of the symbols are rotated before they are transmitted in data frames.

A data frame that is used in an 8 VSBT system is disclosed in the ATSC Digital Television Standard (published on September 16, 1995) and particularly in Annex D, section 4.1 thereof. This data frame is comprised of a plurality of segments (i.e., lines) wherein each segment contains groups of data and wherein each group of data includes a plurality of data elements. For example, each segment may contain sixty-nine groups of data, each group of data may include twelve data elements, and a data element may be a symbol of data. The first segment of such a data frame contains frame synchronization information, and each subsequent segment in the data frame has an initial portion containing segment synchronization information (in the form of four segment sync symbols) and a succeeding portion containing data.

Because of the frame and segment synchronization information contained in a data frame, the data elements in certain groups of data may be rotated in order to facilitate processing of the data by the receiver. This rotation is described in Annex D, section 4.2.5 and in table 2 of the aforementioned Standard. Accordingly, the rotated and interleaved data received by a receiver must be de-rotated and de-interleaved.

The present invention is directed to an apparatus for de-rotating and/or de-interleaving data elements such as symbols.

Summary of the Invention

In accordance with one aspect of the present invention, an apparatus for processing rotated and interleaved data comprises de-rotating means and de-interleaving means. The de-rotating means de-rotates the rotated and interleaved data in order to produce de-rotated and interleaved data. The de-interleaving means de-interleaves the de-rotated and interleaved data in order to produce de-rotated and de-interleaved data.

In accordance with another aspect of the present invention, an apparatus for processing rotated and interleaved data comprises first and second storing means and first and second addressing means. The first storing means stores D elements of the rotated and interleaved data. The first addressing means addresses the first storing means in order to read de-rotated and interleaved data out of the first storing means and to write rotated and interleaved data into the first storing means. The

second storing means stores nD elements of the de-rotated and interleaved data read out of the first storing means. The second addressing means addresses the second storing means in order to read de-rotated and de-interleaved data out of the second storing means and to write de-rotated and interleaved data from the first storing means into the second storing means.

In accordance with yet another aspect of the present invention, an apparatus is arranged to de-rotate rotated data. The rotated data is rotated on the basis of D data elements. The apparatus comprises storing means and addressing means. The storing means stores D data elements of the rotated data in corresponding D storage locations. The addressing means addresses the storing means in order to read de-rotated data out of the D storage locations and to write rotated data into the D storage locations.

Brief Description of the Drawings

These and other features and advantages of the present invention will become more apparent from a detailed consideration of the invention when taken in conjunction with the drawings in which:

Figure 1 illustrates an example of data in row major order and is useful in explaining interleaving of data to be de-interleaved by the present invention;

Figure 2 illustrates an example of data and is useful in explaining rotation of data to be de-rotated by the present invention;

Figure 3 illustrates a de-rotating and de-interleaving apparatus according to the present invention;

Figure 4 is a timing diagram for the demodulator 30 of Figure 3;

Figure 5 is a timing diagram for the controller 34 of Figure 3;

Figure 6 illustrates the address generator 36 of Figure 3 in additional detail;

Figure 7 illustrates an example of data which is useful in explaining the operation of the address generator 36 illustrated in Figure 6;

Figure 8 illustrates one embodiment of the address generator 40 of Figure 3 in additional detail;

Figure 9 represents the memory locations of the second memory means 38;

Figure 10 illustrates the address sequences produced by the address generator 40 of Figure 3;

Figures 11A and 11B are timing diagrams for the address generator 40 of Figure 3;

Figure 12 illustrates a first embodiment of the seed generator 82 of Figure 8 in additional detail;

Figure 13 illustrates a second embodiment of the seed generator 82 of Figure 8 in additional detail;

Figure 14 illustrates the multiply by twelve and modulo forty-seven operator 98 of Figure 13 in additional detail;

Figure 15 illustrates the modulo forty-seven correction logic 106 of Figure 14 in additional detail; and,

Figure 16 illustrates another embodiment of the address generator 40 of Figure 3 in additional detail.

Detailed Description

Figure 1 illustrates a plurality of bytes of data elements which have resulted from scanning part of a scene. The first byte of data contains the data symbols A_1 , A_2 , A_3 , and A_4 . Each of these data symbols may represent, for example, two bits of data. The first byte of data symbols, therefore, represents eight bits of data. The second byte of data contains the data symbols $B_1 - B_4$, the third byte of data contains the data symbols $C_1 - C_4$, . . . and the twelfth byte of data contains the data symbols $L_1 - L_4$.

A symbol interleaver in a transmitter may be configured to interleave the data symbols of Figure 1, although any other arrangement of data symbols or data elements may be interleaved. Accordingly, the symbol interleaver converts the data symbols illustrated in Figure 1 from row major ordering to column major ordering so that the data symbols are transmitted as column major ordered data elements. Consequently, instead of the transmitter transmitting the data symbol A_2 after the data symbol A_1 , the data symbol A_3 after the data symbol A_2 , the data symbol A_4 after the data symbol A_3 , the data symbol B_1 after the data symbol A_4 , and so on, the transmitter (ignoring other processing which may change the ordering of the data symbols) transmits in sequence the data symbol A_1 , the data symbol B_1 , . . . the data symbol L_1 , the data symbol A_2 , Thus, each column of data symbols as shown in Figure 1 represents a group of twelve data elements such that the first column is transmitted first, the second column is transmitted second, and so on.

Figure 2 shows one group of data symbols in a segment 20 of a data frame. As shown in Figure 2, the group of data symbols in this group of data symbols in the segment 20 are not rotated. However, other groups of data symbols in the segment 20 or in other segments may be rotated. For example, the data symbols in a group of data symbols in a segment 22 have been rotated by eight, and the data symbols in a group of data symbols in a segment 24 have been rotated by four (although in the embodiment of the present invention described hereinafter, it is assumed that the transmitter does not rotate data by four). Thus, the data symbols shown in Figures 1 and 2 are then transmitted as interleaved and rotated data.

Figure 3 illustrates an apparatus 28 for de-rotating and de-interleaving interleaved and rotated data. The apparatus 28 includes a demodulator 30 (such as an ATSC 8 VSB demodulator). The demodulator 30 receives an ATSC standard 8 VSB terrestrial mode signal and produces the following four output signals: a frame sync signal, which occurs at the end of the frame sync segment; a segment sync signal, which occurs coincident with the first symbol clock of each data segment; a symbol clock signal; and, interleaved and rotated data. The interleaved and rotated data are supplied to a first memory 32. There are 312 segment sync signals produced for every frame, but there is no segment sync created for the frame sync segment. There are 828 symbol clocks produced for each data segment, and each symbol clock is coincident with a corresponding data symbol. However, there are no symbol clocks created during

either the time corresponding to the frame sync segment or the time corresponding to the four segment sync symbols of a segment sync. Therefore, symbol clocks only exist corresponding to data symbols. Figure 4 is a timing diagram illustrating the outputs of the demodulator 30.

The frame sync signal, the segment sync signal, and the symbol clock signal are supplied to a controller 34, and the controller 34 produces the following five output signals: a seg count signal; a group count signal; two reset signals, RESET 1 and RESET 2; and, a byte clock signal. The seg count signal is the output of a divide by four counter which is reset to zero by the coincidence of the frame sync and seg sync signals. Thereafter, it is incremented by each seg sync signal which causes the seg count signal to cycle through the counts 0, 1, 2, 3 0, 1, 2, 3 Because there are 312 segment sync signals per frame, and because 312 is divisible by four, the value of seg count corresponding to the last segment of the frame will be 3. Therefore, the seg count signal cycles through its four values 78 times ($312/4$) per frame.

The group count signal is the output of a counter that counts groups of twelve symbol clocks. The group count signal is incremented by one when twelve symbol clocks have occurred. It is reset to zero by the coincidence of the seg sync signal and the symbol clock signal. Because there are 828 symbols per segment, and because $828/12 = 69$, the group count signal will cycle from 0 to 68 during each data segment. Thus, the group count signal output cycles through the counts 0, 1, . . . 68, 0,

1 . . . 68, 0 The RESET 1 signal is derived from the frame sync signal. The RESET 2 signal is a twelve symbol clock delayed version of the RESET 1 signal. The byte clock signal is generated every fourth symbol clock. Figure 5 is a timing diagram illustrating the outputs of the controller 34.

The segment count signal, the group count signal, the symbol clock signal, and the RESET 1 signal are supplied to a first address generator 36 which supplies addresses to the first memory 32. Accordingly, in response to the segment count signal, the group count signal, the symbol clock signal, and the RESET 1 signal from the controller 34, the address generator 36 reads out of the first memory 32 a group of de-rotated and interleaved data and writes into the first memory 32 a group of rotated and interleaved data to replace the group of data which was read out.

In the above example where a group of data includes twelve data symbols and where each data symbol represents two bits of information, the first memory 32 is a twelve symbol memory. Thus, in response to the segment count signal, the group count signal, the symbol clock signal, and the RESET 1 signal, the address generator 36 reads out of the first memory 32 twelve de-rotated and interleaved data symbols and writes into the first memory 32 twelve rotated and interleaved data symbols to replace the twelve data symbols which were read out. More specifically, in response to a first symbol clock pulse, the address generator 36 reads out of a first storage

location of the first memory 32 a first data symbol and writes into the first storage location of the first memory 32 a data symbol to replace the first data symbol which was read out. In response to a second symbol clock pulse, the address generator 36 increments its output address by one, reads out of a second storage location of the first memory 32 a second data symbol and writes into the second storage location of the first memory 32 a data symbol to replace the second data symbol which was read out, and so on until all twelve data symbols in a group have been read out and replaced. The addresses that are supplied by the first address generator 36 to the first memory 32 are arranged to de-rotate the rotated data symbols stored in the first memory 32. The de-rotation of the rotated data stored in the first memory 32 is explained in more detail in connection with Figure 6.

Accordingly, the data supplied at the output of the first memory 32 is de-rotated but is still interleaved. A second memory 38 and a second address generator 40 are provided in order to de-interleave the de-rotated, but still interleaved, data. The symbol clock signal from the demodulator 30 and the RESET 2 signal from the controller 34 are provided to the address generator 40. In response to the symbol clock signal, the second address generator 40 generates addresses for the second memory 38 such that each address causes one of the data symbols stored in a corresponding storage location of the second memory 38 to be read out and causes a data symbol from the first memory 32 to be written into that storage location of the second

memory 38 to replace the data symbol just read out, and so on. Based upon the addresses generated by the second address generator 40, the de-rotated but interleaved data stored in the second memory 38 is read out as de-rotated and de-interleaved data. This de-rotated and de-interleaved data is supplied to a series-to-parallel converter 42 which receives each two bit data symbol from the second memory 38, and converts the bits of four data symbols to a byte of eight parallel bits in response to the byte clock from the controller 34.

The first address generator 36 of Figure 3 is shown in more detail in Figure 6. The first address generator 36 includes an address counter 50, an offset generator 52, and a modulo twelve adder 54. The address counter 50 counts twelve symbol clocks and then resets. The output of the address counter 50, therefore, varies from zero to eleven and then resets to zero. The offset generator 52 responds to the group count signal and the segment count signal from the controller 34 in order to generate an offset (which may alternatively be referred to as a seed.) The modulo twelve adder 54 adds the output count from the address counter 50 and the offset from the offset generator 52 in order to generate an address which is supplied to the first memory 32. The offset generator 52 may be simply a set of standard logic elements which are configured according to the equations shown in Figure 6. Alternatively, the offset generator 52 may include a read only memory and an address decoder which decodes the group count and segment count signals in order to access certain

memory locations in the read only memory for the offsets to be supplied to the modulo twelve adder 54.

Figure 6 provides an example of the offsets which may be generated by the offset generator 52. These offsets are based upon the following rotation protocol (see the aforementioned ATSC Digital Television Standard) implemented by a transmitter: (i) the data symbols in the first segment of each set of four segments are not rotated; (ii) the data symbols in groups 0, 1, and 2 of the second segment in each set of four segments are rotated by eight and the other groups of data symbols in this segment are not rotated; (iii) the data symbols in groups 0 and 1 of the third segment in each set of four segments are rotated by eight and the data symbols in the other groups of data in this segment are not rotated; and (iv) the data symbols in group 0 of the fourth segment in each set of four segments are rotated by eight and the data symbols in the other groups of data in this segment are not rotated. Thus, the rotation protocol is repeated every four segments, and only certain groups of data in certain segments are rotated.

Accordingly, because the first segment in each set of four segments (which corresponds to a segment count of zero) is not rotated, the offset generator 52 provides an output offset of zero for the first segment in each set of four segments. This output offset of zero is added by the modulo twelve adder 54 to twelve output counts from the address counter 50 in order to generate twelve addresses to read out twelve data symbols from the first memory 32 and

write in twelve new data symbols to the first memory 32.

When the segment count is one and the group count is zero (i.e., the first group of twelve data symbols which is in the second segment in each set of four segments and which was rotated by eight), the offset generator 52 provides an output offset of zero. This output offset of zero is added by the modulo twelve adder 54 to twelve output counts from the address counter 50 in order to generate twelve addresses to read out twelve data symbols from the first memory 32. When the segment count is one and the group count is one (i.e., the second group of twelve data symbols which is in the second segment in each set of four segments and which was rotated by eight), the offset generator 52 provides an output offset of eight. This output offset of eight is added in modulo twelve fashion by the modulo twelve adder 54 to twelve output counts from the address counter 50 in order to generate twelve addresses to read out twelve data symbols from the first memory 32. When the segment count is one and the group count is two (i.e., the third group of twelve data symbols which is in the second segment in each set of four segments and which was rotated by eight), the offset generator provides an output offset of four which is added in modulo twelve fashion by the modulo twelve adder 54 to twelve output counts from the address counter 50 in order to generate twelve addresses to read out twelve data symbols from the first memory 32. When the segment count is one and the group count is three through sixty-eight, the offset generator provides an output offset of zero

which is added in modulo twelve fashion by the modulo twelve adder 54 to sets of twelve output counts from the address counter 50 in order to generate corresponding sets of twelve addresses to read out corresponding sets of twelve data symbols from the first memory 32.

When the segment count is two and the group count is zero (i.e., the first group of twelve data symbols in the third segment in each set of four segments), the offset generator 52 provides an output offset of zero which is added by the modulo twelve adder 54 to twelve output counts from the address counter 50 in order to generate twelve addresses to read out twelve data symbols from the first memory 32. When the segment count is two and the group count is one (i.e., the second group of twelve data symbols in the second segment in each set of four segments), the offset generator 52 provides an output offset of eight which is added in modulo twelve fashion by the modulo twelve adder 54 to twelve output counts from the address counter 50 in order to generate twelve addresses to read out twelve data symbols from the first memory 32. When the segment count is two and the group count is two through sixty-eight, the offset generator provides an output offset of four which is added in modulo twelve fashion by the modulo twelve adder 54 to sets of twelve output counts from the address counter 50 in order to generate corresponding sets of twelve addresses to read out corresponding sets of twelve data symbols from the first memory 32.

When the segment count is three and the group count is zero (i.e., the first group of twelve data symbols

in the fourth segment in each set of four segments), the offset generator 52 provides an output offset of four which is added in modulo twelve fashion by the modulo twelve adder 54 to twelve output counts from the address counter 50 in order to generate twelve addresses to read out twelve data symbols from the first memory 32. When the segment count is three and the group count is one through sixty-eight, the offset generator provides an output offset of zero which is added by the modulo twelve adder 54 to sets of twelve output counts from the address counter 50 in order to generate corresponding sets of twelve addresses to read out corresponding sets of twelve data symbols from the first memory 32.

Figure 7 illustrates an example of how the address generator 36 operates in cooperation with the first memory 32 in order to de-rotate the rotated and interleaved data from the demodulator 30. When the data in the last group of segment zero has been stored in the first memory 32, this data is stored in the storage locations of the first memory 32 in the data order shown by row 60. This data is unrotated. Row 62 contains the first group of data of segment one which is currently being received by the demodulator 30. This data is rotated by eight, as indicated by the row 62. Because the segment count is one and the group count is zero at this time, the offset which is applied by the offset generator 52 to the modulo twelve adder 54 is zero. Accordingly, the addresses provided by the modulo twelve adder 54 are in order of the output counts supplied by the address counter 50.

Thus, the modulo twelve adder 54 supplies an address of zero to the memory 32 in order to read out the data symbol A of row 60 from storage location zero of the first memory 32 and in order to write the data symbol E of row 62 into storage location zero of the first memory 32. The modulo twelve adder 54 then supplies an address of one to the memory 32 in order to read out the data symbol B of row 60 from storage location one of the first memory 32 and in order to write the data symbol F of row 62 into storage location one of the first memory 32. The addresses supplied by the modulo twelve adder 54 increase in order up to eleven so that the data symbols of row 60 are read out of the first memory 32 and are replaced by the corresponding data symbols of row 62. Consequently, row 64 indicates the contents of the first memory 32 when row 62 has been written into the first memory 32.

At this point, the segment count is one and the group count increments to one, and the next group of data symbols, as shown in row 66, is received by the demodulator 30. As this data is received, the offset generator 52 generates an offset of eight, which is supplied to the modulo twelve adder 54. Accordingly, the modulo twelve adder 54 adds the offset of eight to the first output count of zero from the address counter 50 to generate an address of eight. In response to an address of eight, the data symbol A stored at the storage location eight of the first memory 32 (see row 64) is read out and is replaced by the first received data symbol of group one, i.e., data symbol E of row 66. Similarly, the data symbol B is read out and

is replaced by the data symbol F when the address is nine (offset of eight plus a count of one), the data symbol C is read out and is replaced by the data symbol G when the address is ten (offset of eight plus a count of two), the data symbol D is read out and is replaced by the data symbol H when the address is eleven (offset of eight plus a count of three), the data symbol E is read out and is replaced by the data symbol I when the address is zero (offset of eight plus a count of four which produces an address of zero following the modulo twelve operation), and so on such that row 68 of Figure 7 indicates the data symbols now stored in the storage locations zero through eleven of the first memory 32.

Row 70 indicates the next group of data which is received, i.e., when the segment count is one and the group count is two. When the segment count is one and the group count is two, the offset generator 52 generates an offset of four. This offset of four is added by the module twelve adder 54 to the output counts supplied by the address counter 50 in order to generate the addresses for the first memory 32. As the data in row 70 is received, the modulo twelve adder 54 adds the offset of four to the first output count of zero from the address counter 50 to generate an address of four. In response to an address of four, the data symbol A stored at the storage location four of the first memory 32 (see row 68) is read out and is replaced by the first received data symbol of group two, i.e., data symbol E of row 70. Similarly, the data symbol B is read out and is replaced by the data symbol F when the address

is five (offset of four plus a count of one), the data symbol C is read out and is replaced by the data symbol G when the address is six (offset of four plus a count of two), the data symbol D is read out and is replaced by the data symbol H when the address is seven (offset of four plus a count of three), the data symbol E is read out and is replaced by the data symbol I when the address is eight (offset of four plus a count of four), and so on such that row 72 of Figure 7 indicates the data symbols now stored in the storage locations zero through eleven of the first memory 32. It is noted that when the output count of the address counter 50 reaches eight, the modulo twelve adder 54 provides an address of zero (a count of eight plus an offset of four followed by a modulo twelve operation) to the first memory 32.

Row 74 indicates that the segment one, group three data is unrotated. Thus, with an offset of zero, the contents of the first memory 32 as shown by row 72 is replaced by the received data symbols shown in row 74 so that the contents of the memory 32 is as shown in row 72 after the segment one, group three data are fully stored in the first memory 32.

As data is read out of the first memory 32, the data is de-rotated, but still interleaved. This de-rotated but interleaved data is processed by the second memory 38 and the second address generator 40 in order to de-interleave the de-rotated but interleaved data. Because the data symbols in each byte of data are interleaved with corresponding data symbols in eleven other bytes of data,

and because there are four symbols per byte, the second memory 38 is a 48 symbol memory.

A first embodiment of the address generator 40 is illustrated in Figure 8. This address generator includes a divide by forty-eight counter 80 which provides any output on each count of forty-eight from the symbol clock. The carry output from the divide by forty eight counter 80 is supplied to a seed generator 82, and the seed generator 82 supplies a seed to a first input of an adder 84. The output of the adder 84 is supplied to a modulo forty-seven operator 86 which performs a modulo forty-seven operation on the output of the adder 84. The output of the modulo forty-seven operator 86 is connected to a latch 88 which latches the output of the modulo forty-seven operator 86 under control of the symbol clock. The output of the latch 88 provides an address to the second memory 38 and is also fed back to a second input of the adder 84.

The second memory 38 must have sufficient storage locations to store forty-eight symbols. Although the second memory 38 is a linear memory array, with storage locations addressed 0 to 47, it can be thought of as a rectangular array with $R = 12$ rows and $C = 4$ columns. Therefore, $N = R \times C = 48$. Figure 9 illustrates the addresses of the second memory 38 viewed in this manner. If x_n is a sequence of input symbols (where $n=0 \dots 47$), and if $M_i(n)$ represents a mapping of the input symbols x_n to the memory locations in the second memory 38, $M_i(n)$ is the address in the second memory 38 where a corresponding symbol x_n is to be stored. (The mapping

$M_1(n)$ has a subscript i because there are a number of mappings, i.e. a number of address sequences supplied by the second address generator 40). If the initial mapping sequence $M_0(n)$ is chosen such that $M_0(n) = n$ for $n = 0, 1, 2, \dots, (N-1)$, each input symbol x_n in the first block of forty-eight symbols is stored in memory location n . This can be visualized by overlaying Figure 1 on top of Figure 9.

In order to deinterleave this block of data, it must be read out of the second memory 38 in a different order than the order in which it was written into the second memory 38. So, as the next block of forty-eight symbols is received a symbol at a time, the symbols of the first block are read out from the second memory 38 and are replaced by the corresponding symbols just received from the second block. To deinterleave the first block, $M_1(n)$ must = 0, 12, 24, 36, 1, 1335, 47.

Deinterleaving the first block of symbols according to these addresses causes the second block to be written into the second memory 38 in a different order than the first block was written. Therefore, to deinterleave the second block (while simultaneously writing the third block to the second memory 38), a new addressing sequence $M_2(n)$ must be generated that is different than $M_1(n)$. This process continues through a series of addressing sequences so as to continuously receive and deinterleave blocks of symbols.

If the first mapping sequence is $M_0(n)=0, 1, 2, 3, 4, \dots, 47$, then $M_1(n)$ can be written as

$$M_1(n) = M_0((n \cdot R) \bmod (N-1)) \text{ for } n = 0, \dots, N-1. \quad (1)$$

This expression arises because the interleave depth is R. Adjacent horizontal symbols are spaced by R items in the mapping function. The modulo portion properly controls the end of line wrap around and causes the proper offset as the index scans through the mapping function.

Similarly, mapping any sequence may be generated from the previous mapping according to the following equation:

$$M_i(n) = M_{i-1}((n \cdot R) \bmod (N-1)) \text{ for } n = 0, 1, 2, \dots, N-1. \quad (2)$$

Equivalently, using induction, this mapping function may be generalized to express the nth entry in the ith sequence using the initial (i = 0) sequence as follows:

$$M_i(n) = M_0((n \cdot R^i) \bmod (N-1)) \text{ for } n = 0, 1, 2, \dots, N-1. \quad (3)$$

If it is assumed that the initial sequence is $M_0(n) = n$, equation (3) may be simplified as follows:

$$M_i(n) = (n \cdot R^i) \bmod (N-1) \text{ for } n = 0, 1, 2, \dots, N-1. \quad (4)$$

Equation (4) leads to an expression for calculating the nth entry in any sequence from the nth entry in the preceding sequence as follows:

$$M_i(n) = (M_{i-1}(n) \cdot R) \bmod (N-1) \text{ for } n = 0, 1, 2, \dots, N-1. \quad (5)$$

Equation (4) describes all of the addressing sequences that are necessary to deinterleave the input data from one block to the next. There are a finite number of

different addressing sequences that equation (4) produces before returning to the initial sequence. This finite number of sequences depends upon the values of R and C (or equivalently upon the values of R and N). Accordingly, there are L unique mappings, where L is the lowest number such that:

$$(R^L) \bmod (N-1) = 1 \text{ for } L \text{ not } 0. \quad (6)$$

Finally, an expression relating the nth entry in any sequence i to the (n-1)th entry in that same sequence can be stated as follows:

$$\begin{aligned} M_i(n) &= 0 \text{ for } n = 0 \\ M_i(n) &= (M_i(n-1) + R') \bmod (N-1) \text{ for } n = 1, 2, \dots, N-1. \end{aligned} \quad (7)$$

If R is 12 (as in the example shown in Figure 1), a value of 23 for L satisfies equation (6). That is, equation (4) produces 23 unique sequences which are repeated beginning with L = 24. Accordingly, for the data array shown in Figure 1, there are 23 sequences which are required in order to de-interleave data that is interleaved as described above in connection with Figure 1. Figure 10 shows the twenty-three sequences and their corresponding seeds.

Equations (7) are the basis for the address generator 40 in order to generate the required repeating address sequences. Also required is a seed generator which embodies equation (6) in order to generate the seed for each address sequence.

The seeds, which are generated by the seed

generator 82, are given by equation (6), where L varies from 0 to 22 in the example given in the present patent application. These seeds consist of the following: 1, 12, 3, 36, 9, 14, 27, 42, 34, 32, 8, 2, 24, 6, 25, 18, 28, 7, 37, 21, 17, 16, and 4. On each forty-eighth symbol clock pulse, the seed generator 82 supplies a corresponding one of these seeds to the adder 84 which adds the seed to the output of the latch 88 on each symbol clock pulse. The modulo forty-seven operator 86 performs a modulo forty-seven operation (if >47, subtract 47) on the output of the adder 84 and supplies the result to the latch 88. The latch 88 latches this result and supplies this latched result as an address to the second memory 38. In this manner, a unique sequence of forty-eight addresses is produced for each seed generated by the seed generator 82.

The address generator 40 is driven by RESET 2 and the symbol clock because the processing that occurs in the first memory 32 takes exactly 12 symbol clocks. Because the symbols passing through the first memory 32 are delayed by twelve symbol clocks, RESET 2 lines up correctly with the first symbol of a frame as it is output from the first memory 32. The timing diagrams of Figures 11A and 11B assume recent acquisition of the 8 VSB signal, so that the initial value of the counter 80, the carry output of the counter 80, the seed generator 82, and the latch 88 are unknown. RESET 2 and the first symbol clock cause the counter 80 to initialize to zero which in turn assures that the carry output is low. RESET 2 also causes the seed generator 82 to output the first seed (1). RESET 2 and the

first symbol clock also cause the output of the latch 88 to synchronously clear to zero. Subsequent symbol clocks cause the counter 80 to increment.

When the counter 80 reaches a count = 47, it outputs a carry signal which in turn causes the seed generator 82 to output the next sequential seed (12). This carry signal also causes the latch 88 to synchronously clear at the next symbol clock. Accordingly, the carry signals from the counter 80 cause the seed generator 82 to cycle through its sequence of 23 seeds repeatedly. Each seed thus has a duration of 48 symbols clocks. The full sequence of 23 seeds, therefore, encompasses $23 \times 48 = 1104$ symbols. Because there are 258336 data symbols in a frame, the sequence of 23 seeds occurs exactly 234 times in each frame ($234 \times 1104 = 258336$). Therefore, the beginning of a new frame (indicated in the timing diagram of Figures 11A and 11B by RESET 2 pulse #2) always lines up with the first seed (1). This alignment is an advantage because it allows use of the first RESET 2 signal (derived from the frame sync signal as previously explained) after 8 VSB signal acquisition to initialize the seed generator 82 to an output of 1 without concern for the effect of subsequent RESET 2 signals (one per frame) because they will always coincide with a point in time when the seed generator 82 should have output of 1.

The sequence of seeds, once initiated after signal acquisition, should not be broken. This is necessary for correct continuous deinterleaver operation. The timing diagram also shows the latch 88 output

corresponding to the required $M_1(n)$. Portions of $M_0(n)$, $M_1(n)$, and $M_2(n)$ are illustrated, occurring after RESET 2 pulse #1. The sequences are assumed to have cycled through all the $M_1(n)$ many times until at the end of the first frame $M_{22}(n)$ is shown to occur just before reset 2 pulse #2. The second frame begins with $M_0(n)$ as expected.

The de-interleaving operation is similar to the de-rotating operation which was explained with respect to Figures 6 and 7. That is, when a data symbol is read out from a storage location in the second memory 38, as determined by the second address generator 40, that data symbol is replaced by the next data symbol which is received from the first memory 32. Accordingly, when the symbol clock causes the divide by 48 counter 80 to reach a count of 47, the counter 80 will output a carry signal which is supplied to the seed generator 82 and to the latch 88 (via the OR gate 89). This carry output causes the seed generator 82 to supply a new seed to the adder 84. This carry output also clears the latch 88 (at the next symbol clock) so that its output is zero. The latch output is also supplied back to the second input of the adder 84.

For each of the 23 sequences shown in Figure 10, the adder 84 adds the zero output of the latch 88 (which represents the first address in each sequence) to the seed supplied by the seed generator 82. This output of the adder 84 is controlled by a modulo forty-seven operation and the result is supplied to the input of the latch 88. The next symbol clock causes the latch 88 to supply this result of the modulo forty-seven operator 86 to the second

memory 38. This output of the latch 88 is again fed back to the adder 84 to be added to the seed from the seed generator 82. Thus, the seed from the seed generator 82 is recursively added to itself modulo forty-seven to produce a corresponding sequence of addresses which is supplied to the second memory 38.

Each such sequence of addresses is given by equation (7). In each rotation of the twenty-three sequences, the first sequence is always in order from zero to 47. Also, the seed for this first sequence as supplied by the seed generator 82 is one, as determined by equation (6).

Figure 12 illustrates a first embodiment of the seed generator 82 in more detail. The seed generator 82 includes a multiplier 92, a modulo forty-seven operator 94, and a latch 96. The multiplier 92 multiplies the output of the latch 96 by twelve, the modulo forty-seven operator 94 performs a modulo forty-seven operation on the output of the adder 92, and the latch 96 latches the output of the modulo forty-seven operator 94.

When RESET 2 is produced by the controller 34, the latch 96 is reset so that its output is one. This one is supplied as the seed for the adder 84. This one is also multiplied by twelve by the multiplier 92 and the result is operated on by the modulo forty-seven operator 94. The output of the modulo forty-seven operator 94 is supplied to the latch 96 which latches this output through upon receipt of the next carry output from the divide by forty-eight counter 80. Upon receipt of this carry output, the output

of twelve from the latch 96 is fed back to the multiplier 92 which multiplies this twelve by twelve to produce an output of 144 which is operated on by the modulo forty-seven operator 94. The output of the modulo forty-seven operator 94 (three) is supplied to the latch 96 which latches this output through upon receipt of the next carry output from the divide by forty-eight counter 80. Upon receipt of this carry output, the output of three from the latch 96 is fed back to the multiplier 92 which multiplies this three by twelve to produce an output of thirty-six which is operated on by the modulo forty-seven operator 94. The output of the modulo forty-seven operator 94 (i.e., thirty-six) is supplied to the latch 96 which latches this output through upon receipt of the next carry output from the divide by forty-eight counter 80. Upon receipt of this carry output, the output of thirty-six from the latch 96 is fed back to the multiplier 92 which multiplies this thirty-six by twelve to produce an output of 432, and so on. Accordingly, the seed generator 82 produces the twenty-three seeds described above.

Figure 13 illustrates a second embodiment of the seed generator 82 in more detail. This seed generator 82 includes a multiply by twelve and modulo forty-seven operator 98 and a latch 100. The multiply by twelve and modulo forty-seven operator 98, as shown in more detail in Figure 14, includes an adder 102 which adds the four most significant bits from the output of the latch 100, after the two least significant bits have been discarded at 104, to a correction factor from a modulo forty-seven correction

logic 106 which is shown in more detail in Figure 15.

When RESET 2 is produced by the controller 34, the latch 100 is reset so that its output is one. This one is supplied as the seed for the adder 84. This one is also multiplied by twelve and the result is processed by a modulo forty-seven operation by the multiply by twelve and modulo forty-seven operator 98. The output of the multiply by twelve and modulo forty-seven operator 98 is supplied to the latch 100 which latches this output through upon receipt of the next carry output from the divide by forty-eight counter 80. The output of twelve from the latch is fed back to the multiplier 100 which multiplies this twelve by twelve and the result (144) is processed by a modulo forty-seven operation by the multiply by twelve and modulo forty-seven operator 98 to produce an output of three. The output of three from the multiply by twelve and modulo forty-seven operator 98 is supplied to the latch 104 which latches this output through upon receipt of the next carry output from the divide by forty-eight counter 80, and so on. Accordingly, the seed generator 82 produces the twenty-three seeds described above.

The multiply by twelve and modulo forty-seven operator 98 is shown in more detail in Figure 14. When the output of the latch 100 is one (i.e., a digital value of 000001), the two least significant bits (01) are fed to the modulo forty-seven correction logic 106 which provides an output of twelve. The output of twelve is added by the adder 102 to the four most significant bits (0000) of the output from the latch 100 and the result of twelve is

supplied to the latch 102. The latch 102 latches the twelve to its output upon receipt of the next carry output from the divide by forty-eight counter 80. When the output of the latch 100 is twelve (i.e., a digital value of 001100), the two least significant bits (00) are fed to the modulo forty-seven correction logic 106 which provides an output of zero. This output of zero is added by the adder 102 to the four most significant bits (0011) of the output from the latch 100 and the result of three is supplied to the latch 102. The latch 100 latches the three to its output upon receipt of the next carry output from the divide by forty-eight counter 80, and so on.

The modulo forty-seven correction logic 106 is shown in more detail in Figure 15 and includes logic 108. The two least significant outputs, OUT 0 and OUT 1, of the logic 108 are always zero. The four most significant outputs, OUT 2 - OUT 5, of the logic 108 are controlled by the two least significant bits in the output of the latch 100, IN 0 and IN 1.

A second embodiment of the second address generator 40 based upon equation (4) is illustrated in Figure 16. This second embodiment includes a counter 112 which counts symbol clock pulses and supplies an output on each count to a multiplier 114. The carry output of the counter 112 is supplied to a seed generator 116 which may be a state machine, or a RAM, or similar to the seed generator 82. The seed generator 116 is toggled through twenty-three states by the carry output from the counter 112 in order to supply a corresponding seed, as determined

by equation (6), upon each toggle. The multiplier 114 multiplies the output count from the counter 112 by the seed provided by the seed generator 116 and supplies the result to a modulo forty-seven operator 118. The output of the modulo forty-seven operator is supplied to the input of a latch 120. The output of the latch 120 is reset to zero by the carry output from the counter 112. Thus, at the beginning of each address sequence, the output of the latch 120 is equal to zero which is always the first address of each address sequence. At the end of each address sequence (i.e., the forty-eighth count), the seed generator 116 is toggled in order to supply a new seed to the multiplier 114.

Certain modifications of the present invention have been discussed above. Other modifications will occur to those practicing in the art of the present invention. For example, although the present invention has been specifically described in terms of two bit data symbols, it should be understood that the present invention applies to other data elements. Also, the present invention may be used in conjunction with different data frame arrangements and different rotation and/or interleaving protocols.

Accordingly, the description of the present invention is to be construed as illustrative only and is for the purpose of teaching those skilled in the art the best mode of carrying out the invention. The details may be varied substantially without departing from the spirit of the invention, and the exclusive use of all modifications which are within the scope of the appended

claims is reserved.



CLAIMS

1. An apparatus for processing rotated and interleaved data characterized by de-rotating means for de-rotating the rotated and interleaved data in order to produce de-rotated and interleaved data, and de-interleaving means for de-interleaving the de-rotated and interleaved data in order to produce de-rotated and de-interleaved data.

2. The apparatus of claim 1 characterized in that the de-rotating means includes first storing means for storing D elements of the rotated and interleaved data, and first addressing means for addressing the first storing means in order to read de-rotated and interleaved data out of the first storing means and to write rotated and interleaved data into the first storing means, and wherein the de-interleaving means includes second storing means for storing mD elements of the de-rotated and interleaved data read out of the first storing means, and second addressing means for addressing the second storing means in order to read de-rotated and de-interleaved data out of the second storing means and to write de-rotated and interleaved data from the first storing means into the second storing means.

3. The apparatus of claim 1 or 2, characterized in that the rotated and interleaved data are comprised of S segments of data elements, wherein each segment of data elements is comprised of G groups of data elements, wherein each group of data elements is comprised of D data elements, and wherein the first addressing means

This Sheet Has Been Replaced



REPLACED SHEETS		SG
☐	Description : paged to paged	
☒	Claim(s) : paged 31 to paged 35	
☐	Drawing(s) : paged to paged	
☐	Abstract : paged to paged	



comprises offset generating means for generating offsets dependent upon the segment and group of data elements and an extent of rotation of the D data elements in a group of data elements.

4. The apparatus of claim 3 characterized in that the offset generating means generates an offset of zero for a first of the S segments, wherein the offset generating means generates an offset of zero for a first group of a second of the S segments, wherein the offset generating means generates an offset of eight for a second group of the second of the S segments, wherein the offset generating means generates an offset of four for a third group of the second of the S segments, wherein the offset generating means generates an offset of zero for any remaining groups of the second of the S segments, wherein the offset generating means generates an offset of zero for a first group of a third of the S segments, wherein the offset generating means generates an offset of eight for a second group of the third of the S segments, wherein the offset generating means generates an offset of four for any remaining groups of the third of the S segments, wherein the offset generating means generates an offset of four for a first group of a fourth of the S segments, and wherein the offset generating means generates an offset of zero for any remaining groups of the fourth of the S segments.

5. The apparatus of claim 1 characterized in that the de-rotating means includes storing means for storing the rotated and interleaved data, and addressing means





for addressing the storing means in order to read de-rotated and interleaved data out of the storing means and to write rotated and interleaved data into the storing means.

6. The apparatus of claim 5 characterized in that the rotated and interleaved data are comprised of data elements, wherein the rotated and interleaved data are rotated on the basis of D data elements, and wherein the storing means stores the rotated and interleaved data D data elements at a time.

7. The apparatus of claim 5 characterized in that the rotated and interleaved data are comprised of groups of D data elements, and wherein the storing means stores the rotated and interleaved data D data elements at a time.

8. The apparatus of claim 5 characterized in that the rotated and interleaved data are comprised of S segments of data elements, wherein each segment of data elements is comprised of G groups of data elements, wherein each group of data elements is comprised of D data elements, and wherein the storing means stores the rotated and interleaved data a group of data elements at a time.

9. The apparatus of claim 1 characterized in that the deinterleaving means includes storing means for storing the de-rotated and interleaved data, and addressing means for addressing the storing means in order to read de-rotated and de-interleaved data out of the storing means and to write de-rotated and interleaved





data into the storing means.

10. The apparatus of claim 9 characterized in that the rotated and interleaved data are comprised of data elements, wherein the rotated and interleaved data is interleaved on the basis of D data elements, wherein the storing means stores the de-rotated and interleaved data mD data elements at a time, and wherein m is an integer.

11. The apparatus of claim 10 characterized in that the storing means comprises first storing means, wherein the addressing means comprises first addressing means, and wherein the de-rotating means includes second storing means for storing the rotated and interleaved data, and second addressing means for addressing the first storing means in order to read de-rotated and interleaved data out of the storing means and to write rotated and interleaved data into the second storing means.

12. The apparatus of claim 11 characterized in that the rotated and interleaved data are comprised of data elements, wherein the rotated and interleaved data are rotated on the basis of D data elements, and wherein the second storing means stores the rotated and interleaved data D data elements at a time.

13. The apparatus of claim 12 characterized in that the rotated and interleaved data are comprised of groups of D data elements, and wherein the second storing means stores the rotated and interleaved data D data elements at a time.

14. The apparatus of claim 13 characterized in that the rotated and interleaved data are comprised of S

This Sheet Has Been Replaced





segments of data elements, wherein each segment of data elements is comprised of G groups of data elements, wherein each group of data elements is comprised of D data elements, and wherein the second storing means stores the rotated and interleaved data a group of data elements at a time.

This Sheet Has Been Replaced

