Comparison of Hardware Complexity between Convolutional Guided Scrambling and Galois-Field-Addition Guided Scrambling for Digital Recording

(デジタル記録における畳込み演算形GSとガロア体加算形GSのハードウェア規模の比較)

Akiomi Kunisa (member)[†]

Abstract The guided scrambling (GS) coding technique is used to control the runlength within code blocks, such as d or k , as well as to suppress the DC component. It achieves a code rate very close to the channel capacity. In a GS encoder, a selection set of channel bit streams is constructed by subjecting a source bit stream to several kinds of scramblings and to runlength limited (RLL) coding if necessary. Then one that satisfies the imposed channel constraints best is selected. Typically, the GS technique uses a convolutional operation or an addition operation in the Galois field (GF); these two schemes can generate channel bit streams that meet the channel constraints better than conventional coding techniques. The latter scheme is better in terms of symbol-error probability when both schemes are concatenated with Reed-Solomon (RS) codes. However, nothing is known about the hardware complexity of either scheme. We describe their hardware configurations using ROM tables and compare the table sizes. From this comparison, we conclude that the hardware complexity of the latter scheme is smaller under certain conditions.

Key words: digital recording, guided scrambling, runlength limited, DC-free, hardware complexity

1. Introduction

Digital modulation is among the most important techniques of signal processing for transmitting information across a band of restricted width. Practical applications include digital subscriber lines, digital magnetic recording, and digital optical recording¹⁾. This technique transforms the input data sequence into a constrained sequence which is matched to the characteristics of channel fixed by the recording media, the recording head, and so on. Such constrained codes are *runlength limited* (RLL) codes that two logical “ones” are separated by a run of at least d and at most k consecutive “zeros” or *DC-free codes* that low-frequency components near DC are eliminated^{1)–3)}. The constrained codes that have been practically realized make use of a fairly small constraint length for decoding. However, the rate of these codes is far smaller than the channel capacity.

Guided scrambling (GS)⁴⁾⁵⁾ has been proposed for the

simple design for constrained codes having extremely high efficiency. In principle, a guided scrambler converts a *data block*, which consists of a constant number of source words, into several kinds of *code block alternatives*, which consist of a constant number of codewords. At this time, an *identification data*, indicating how to scramble the data block into the code block, is multiplexed with the code block. Finally, the code block is singled out that complies with the channel constraints such as the limitation of runlength or less DC content.

Typically, the GS technique uses a convolutional operation⁴⁾⁶⁾ or an addition operation in the Galois field (GF)⁷⁾. These techniques can generate channel bit streams that meet the channel constraints better, and the GF-addition GS has lower symbol-error probability than the convolutional GS after Reed-Solomon (RS) decoding⁸⁾. However nothing is known about hardware complexity of both schemes. The hardware complexity of a GS decoder is not large, but the amount of hardware required for some GS encoders is larger than that of some general DC-free coder and/or RLL coder. Therefore the hardware complexity is one of the most important factors for evaluating the performance of GS schemes. It is useless when a GS scheme having bet-

Presented in ITE '99 Annual Conference

Received November 26, 1999; Revised March 30, 2000; Accepted May 10, 2000

[†]Hypermedia Research Center, SANYO Electric Co., Ltd.
(3-10-15, Hongo, Bunkyo-ku, Tokyo 113-8434, Japan)

ter transmission characteristics, e.g. GF-addition GS, needs much more hardware to be implemented in LSI.

In this paper, the hardware complexity of the GS schemes based on the GF-addition and convolutional operation is compared with one another. Since it is difficult to generally estimate the hardware complexity of both schemes in a fair way, it is attempted to do by means of the hardware configuration using read-only memory (ROM)⁹⁾¹⁰⁾.

This paper is organized as follows. In Section 2, we review the principle of the GS technique, and then describe the convolutional GS and GF-addition GS. In Section 3, we consider the hardware configuration using ROM to estimate the hardware complexity, and then introduce some examples of the hardware configurations of the GF-addition GS and convolutional GS. After that, we calculate the ROM sizes of both GS schemes and summarize that the hardware complexity of the GF-addition GS is smaller than that of the convolutional GS under certain conditions. In Section 4, we conclude this paper.

2. GS Techniques

Codes based on GS cannot always satisfy the imposed channel constraints, unlike the recording codes that are represented by a finite state transition diagram²⁾³⁾, but they violate the constraints with very small probability. The noiseless channel does not exist in practical systems, and this violation has little influence on reproducing a recorded data sequence correctly if the violation probability is smaller than the bit-error rate applied to a decoder¹¹⁾.

This section first provides an outline of the guided scrambling and the encoder based on it. Next, useful scrambling schemes are introduced for guiding an input data stream to meet the channel constraints.

2.1 Outline of the GS Technique

We describe the functional principle of a GS technique⁵⁾. In a guided scrambler, first, L code block alternatives are generated from a data block, where a data block and code block are composed of predetermined numbers of user data bits and channel bits, respectively. At this time, an identification data, indicating the way of scrambling, is multiplexed with the code block. Such code block alternatives are dealt with as the selection set which consists of candidates for the code block to be transmitted next. The one that satisfies the required channel constraints best in the selection set is finally chosen as the next code block. Assuming that the iden-

Fig. 1 Straightforward implementation of a basic GS encoder. L parameters of channel constraints are calculated in parallel.

tification data consists of r bits, L is 2^r or less. The larger L is, the less weakly the channel constraints are satisfied, as the number of candidates increases.

Fig. 1 depicts a straightforward implementation of the GS encoder. First, the zero-data is augmented with each data block in the dummy-data multiplexer. Next, in order to construct candidates for the code block, a scrambling operation is performed on the augmented data block by using the data from the scrambling-data generator, and then the scrambled data is RLL-encoded if necessary. In the channel-constraints comparator, L parameters of the channel constraints of the code block alternatives are compared with each other, while L kinds of candidates are stored in each block memory. Finally, the selector chooses the code block with the parameter which complies best with the prescribed constraints.

Concrete examples of such a GS technique are the one using a convolutional operation⁴⁾ and the one having error correcting capability in the identification data by taking advantage of RS codes⁷⁾¹²⁾. These schemes are explained below in detail.

2.2 GS Based on the Convolutional Operation

This section describes the GS based on the convolutional operation⁴⁾⁶⁾. First, the guided scrambler inserts L kinds of r -bit redundant words c_i , $0 \leq i \leq L - 1$, immediately before M -bit data block $D(x)$ in order to generate L augmented data blocks $c_i x^M + D(x)$. The length of the data blocks becomes $M + r$ bits. Next, the augmented data blocks are divided by the scrambling polynomial $S(x)$ of degree N in order to obtain

quotients,

$$T_i(x) = Q_{S(x)}[(c_i x^M + D(x))x^N], \quad (1)$$

where $Q_a[b]$ stands for the quotient of a polynomial b divided by a polynomial a . x^M also signifies a delay of M time units of r bits. From among all quotients in the quotient set $\{T_0(x), \dots, T_{L-1}(x)\}$, the one whose channel-constraints are satisfied best is selected.

At the receiver's end, the code block is multiplied by $S(x)$, and then the r most significant bits which were augmented and N least significant bits are discarded to get the original data.

When the scrambling polynomial of $S(x) = x^r + 1$ is used in Eq.(1), an exclusive-OR operation in units of r bits can be introduced between the r -bit data t_{j-1} which has just been converted and the r -bit data d_j to be converted next. Such an operation is

$$t_j = \begin{cases} d_j \oplus c_i, & j = 0 \\ d_j \oplus t_{j-1}, & j = 1, \dots, M/r - 1, \end{cases} \quad (2)$$

where r is a positive number such that $M \bmod r = 0$,^{*} and all the identification data c_i , source word d_j , and converted data t_j consist of r bits. " $\oplus$ " denotes the exclusive-OR operation or addition modulo 2. By way of illustration, **Fig. 2** depicts the convolution operation. For example, when $d_j = (0, 1, 1, 0)$ and $t_{j-1} = (1, 1, 0, 0)$, $t_j = d_j \oplus t_{j-1} = (1, 0, 1, 0)$.

Given that $L = 2^r$ and $S(x) = x^r + 1$, 2^r candidates can be constituted of 2^{r-1} kinds of code blocks and their complementary code blocks in an NRZ (non-return-to-zero) notation, i.e. the code blocks having opposite disparity at every bit⁷⁾.** When suppression of the spectrum at DC is attempted, there is always a code block in the selection set that can control the value of *running digital sum* (RDS) due to $L/2$ pairs of blocks having opposite disparity.

In decoding, the original data is restored by the transformation of Eq.(2), such as an exclusive-OR operation of the adjacent codewords cut out in units of r bits, t_j and t_{j-1} .

2.3 GS Based on the GF-Addition Operation

This section elaborates on the functional principle of the GS based on the additional operation of elements in the $GF(2^{11})$. This GS technique is called *GF-addition GS*. Below, assume that the RS codes¹³⁾ consist of elements in $GF(2^m)$. Given that $m \bmod r = 0$ where r is

* $a \bmod b$ represents the remainder calculated by division of a by b .

** The code block using the identification data $c_i = i, 0 \leq i \leq L/2$, of r bits is the binary-reverted one of the code block using $c_{L-i-1} = 2^r - i - 1$, where c_i is denoted by decimal notation.

Fig. 2 Principle of the convolutional GS using the scrambling polynomial of $S(x) = x^r + 1$. $D(x)$ is the data block before scrambling and $T_i(x)$ is the code block after scrambling.

the length of a redundancy word, an RS-coded sequence is expressed by the two different polynomials. When RS encoding is performed, an RS codeword is represented by the polynomial

$$R(x) = \sum_{i=0}^{n-1} R_i x^i, \quad (3)$$

where $R_i \in GF(2^m)$ and x^i signifies a delay of i time units of m bits. When this sequence is utilized for scrambling, it is explained by a polynomial over $GF(2^r)$. To scramble an RS codeword, a data word of every r bits is cut out in series from the RS codeword, and so the polynomial is expressed as

$$r(y) = \sum_{i=0}^{nm/r-1} r_i y^i, \quad (4)$$

where $r_i \in GF(2^r)$ and y^i signifies a delay of i time units of r bits. The RS-coded sequence $R(x)$ is exactly equal to $r(y)$. The two polynomials must be chosen properly in consideration of the way in which the data block is processed.

Fig. 3 shows a conceptual explanation of the data scrambling using a GF-addition operation. First, dummy words of r bits are multiplexed with a source sequence $D(x)$ in order to generate a multiplexed source sequence $D'(x)$ of k symbols.^{***} As illustrated in Fig. 3, the dummy words $A(y)$ and $B(y)$ are multiplexed with the source sequence $D(x)$ which consists of $D_0(y)$ and $D_1(y)$, where $A(y)$, $B(y)$, $D_0(y)$, and $D_1(y)$ consist of elements in $GF(2^r)$. A symbol in $GF(2^m)$ is split into m/r symbols in $GF(2^r)$ since $m \bmod r = 0$, so that the dummy words of r bits may be inserted at arbitrary positions. When the sum of the bit lengths of all redundancy words within an RS codeword is m , m/r symbols in $GF(2^r)$ are brought together into a symbol in $GF(2^m)$ sequentially from the head of the multiplexed

*** The letter k is used to describe the error correction code that has different meaning from the k -constraint in the discussion of the recording code.

Fig. 3 Conceptual illustration of the data scrambling using a GF-addition operation. All blocks of $R(x)$, $R_{S_i}(x)$, and $R'_i(x)$ are RS codewords.

sequence in order to produce $D'(x)$ of k symbols over $GF(2^m)$.

Next, the parity-check sequence $P_{D'}(x)$ of $n - k$ symbols are emitted by RS-encoding the multiplexed data block. After that, L kinds of RS codewords $R'_i(x)$, $0 \leq i \leq L - 1$, are constructed by addition of $R_{S_i}(x)$ to the RS codeword $R(x)$ which is $D'(x)x^{n-k} + P_{D'}(x)$. Such addition is expressed by

$$\begin{aligned} R'_i(x) &= R(x) + R_{S_i}(x) \\ &= (D'(x)x^{n-k} + P_{D'}(x)) \\ &\quad + (S_i(x)x^{n-k} + P_{S_i}(x)). \end{aligned} \quad (5)$$

Note that $R'_i(x)$ and $R_{S_i}(x)$ are called a *scrambled RS codeword* and a *scrambling RS codeword*, respectively. Note also that $S_i(x)$ and $P_{S_i}(x)$ represent the source and parity-check sequences of the scrambling RS codeword, respectively. In Fig. 3, the data of identification of scrambling methods are embedded in the data of the scrambling RS codeword at the same positions of $A(y)$ and $B(y)$ in the source sequence, and they are added into the dummy words $A(y)$ and $B(y)$ which are normally zero elements. Consequently, the scrambled RS codeword containing the identification data is automatically produced.

Finally, the codeword that satisfies the channel constraints best is selected from among L types of scrambled RS codewords obtained through this conversion process. It is obvious that at most 2^m kinds of scrambled RS codewords can be produced by insertion of the dummy words of m bits into the data block. In this case, m is the sum of the bit lengths of dummy data $A(y)$ and $B(y)$.

The encoder of GF-addition GS is slightly different

from that of Fig. 1 as dummy words are subject to the RS encoding. The RS encoder must be inserted between the dummy-data multiplexer and the scramblers. The RS encoder is ordinarily placed before the dummy-data multiplexer, which is not shown in Fig. 1.

In the decoding process, a syndrome polynomial $Y'(x)$ is calculated, which is the remainder obtained from the division of a received polynomial by a generator polynomial $G(x)$. The received polynomial is an RS codeword polynomial $R'_i(x)$ plus a noise polynomial $E(x)$, and so this operation is expressed by

$$\begin{aligned} Y'(x) &= ((R'_i(x) + E(x)) \bmod G(x)) \\ &= (D'(x)x^{n-k} + P_{D'}(x)) \bmod G(x) \\ &\quad + (S_i(x)x^{n-k} + P_{S_i}(x)) \bmod G(x) \\ &\quad + E(x) \bmod G(x) \\ &= E(x) \bmod G(x). \end{aligned} \quad (6)$$

Eq.(6) depends on the error polynomial $E(x)$, but not on $R(x)$ nor $R_{S_i}(x)$ since the code block that the addition operation of elements in $GF(2^m)$ is executed remains an RS codeword over $GF(2^m)$ as it was before the conversion⁷⁾. Therefore it is possible to perform RS decoding to the received data first and then unscramble the RS-decoded data by subtracting the scrambling data which is regenerated from the decoded identification data. Obviously, symbol errors less than or equal to the allowable number of errors can be corrected by using Eq.(6) during the RS decoding.

Reliability of the identification data acquired after RS decoding is raised due to reverting the decoding order from the encoding order⁷⁾¹²⁾. Kunisa and Itoh have reported that the symbol-error probability of the GF-additional GS is lower than that of the convolutional GS⁸⁾.

3. Hardware Complexity

In the previous section, we introduced the convolutional GS and GF-addition GS. In the literature 8), we have measured the symbol-error probabilities of both schemes, and showed the GF-addition GS outperforms the convolutional GS in this characteristic. However it would be meaningless if the hardware complexity of the GF-addition GS were much larger than that of the convolutional scheme.

Recently, an AS-MO (advanced storage magneto-optical) drive has been developed for storage of over 6 gigabytes in a CD sized Disk¹⁴⁾. This drive uses the convolutional GS of $r = 4$ as a modulation technique¹⁵⁾.

We prospect that it is not difficult to cast the encoder of GF-addition GS into a silicon chip if it does need as much hardware complexity as the convolutional scheme. In this section, the hardware complexity of the GS encoders based on the GF-addition and convolutional operation is estimated and compared with one another for finding out the possibility of hardware implementation.

3.1 Hardware Configuration Using ROM

Although it is difficult to generally estimate the hardware complexity of both schemes in a fair way, it can be done by comparing the hardware configurations using read-only memory (ROM)⁽⁹⁾⁽¹⁰⁾. The ROM shall consist of an EX-OR operator i , an RLL encoder i and a part of the scrambling-data generator that are shown in Fig. 1, where $0 \leq i \leq L-1$, for the purpose of minimizing the RDS⁽⁴⁾⁽⁵⁾ or largest runlength⁽¹¹⁾⁽¹⁶⁾. The block diagram of the encoder is transformed into the configuration shown in Fig. 4 when the ROM is put in the encoder.

In this block diagram, suppose that the RLL encoder is represented by a conventional finite-state sequential machine (FSSM)⁽²⁾ as following: The RLL encoder emits a codeword as a function of an input source word and current state. By making use of the output function h , the k th codeword c_k is

$$c_k = h(s_k, b_k), \quad (7)$$

where s_k and b_k are the k th state and k th source word in an input sequence, respectively. When a codeword emanates, a state transition occurs simultaneously. By using the state-transition function f , the next state, i.e. $(k+1)$ th state, is

$$s_{k+1} = f(s_k, c_k). \quad (8)$$

The modules other than the ROMs in Fig. 4 can be made use of in common except for position of the RS encoder whenever any GS scheme is applied to this block diagram. As described in Section 2.3, the RS encoder is placed before the dummy-data multiplexer or after that when the convolutional GS or GF-addition GS is concatenated with the RS encoder, respectively. The input data to the ROM are exactly alike for both channel-constraint parameters.

Below, assume that the ROM operates on units of information data of $l \cdot p$ bits, where p is the number of bits of a source word and l is a natural number. In other words, the ROM operates on units of the data which are an integer multiple of the source word length. This parameter l indicates the trade-off between the speed of processing of GS and the scrambling performance⁽⁷⁾.

Fig. 4 Block diagram of a GS encoder when a ROM $i, 0 \leq i \leq L-1$, for calculating the characteristic parameter is utilized. The data sequence stored in the block memory is encoded, using the scrambling method that satisfies the channel constraints best.

The larger the trade-off parameter l is, the slower the GS can be processed, but the worse the performance becomes.

3.2 ROM Configuration

Each of the ROM configurations of the convolutional GS and GF-addition GS is considered for controlling the DC content or maximum runlength in this section.

(1) Convolutional GS

Assume that an exclusive-OR operation of r bits is performed and $l \cdot p \bmod r = 0$. The ROM configurations of the convolutional GS are shown in Fig. 5. The input data to the ROM are the data of the last r bits among $l \cdot p$ -bit data which was just scrambled, the data before scrambling of $l \cdot p$ bits, and the S -bit data indicating the current state of an RLL encoder.

The output information from the ROM is the data of the last r bits among the scrambled data of $l \cdot p$ bits, and the data, indicating the next state of an RLL encoder, of S bits. These output data are used in common whenever any constraint is given since they are fed back to the input port of the ROM in order to process the next data of $l \cdot p$ bits.

In addition to these output data, the ROM for DC suppression outputs the disparity corresponding to codewords of $l \cdot q$ bits, that is called *codeword digital sum* (CDS), and the polarity corresponding to the codewords. The polarity indicates a sign of a waveform at the end of codewords starting at the minus level.* The CDS consists of $\lceil \log_2(l \cdot q + 1) \rceil$ bits, and the polarity consists of 1 bit, where $\lceil A \rceil$ denotes the smallest integer larger than or equal to A . As a result, the capacity of the ROM for suppressing the DC is

* Note that the polarity here does not mean the polarity of a CDS value.

Fig. 5 ROM configurations of the convolutional GS. a) and b) are used for suppressing the DC and for controlling the maximum runlength, respectively.

$$C_m = ([\log_2(l \cdot q + 1)] + 1 + r + S) \times 2^{l \cdot p + r + S} \text{ [bits]}, \quad (9)$$

as shown in Fig. 5 a).

We now show the way to calculate the RDS by using the data of CDS and polarity. An example of calculation is illustrated in **Fig. 6**. Fig. 6 a) shows an RLL codeword having a CDS value of “+1” and a polarity of “0”,* that is to be connected to the code sequence. As shown in Fig. 6, the two situation is considered when the CDS and polarity are renewed. The first case is that the polarity at the end of the previous codeword is minus. In this case, the CDS of the current codeword is added to the previous RDS to have the new RDS. In the second case where the polarity is plus, the renewed RDS is obtained by the subtraction of the CDS from the previous RDS. In this way, a GS-encoder always keeps the polarity at the end of the codeword. Therefore the ROM must output the data of the polarity of a codeword. The polarity of a recored sequence is renewed by an exclusive-OR operation between the previous polarity of the recorded sequence and the polarity of the current codeword.

The ROM for controlling the maximum runlength is shown in Fig. 5 b). It is required to output the maximum runlength rl_c within a codeword of $l \cdot q$ bits. For measuring the runlength that occurred in the border of adjacent $l \cdot q$ -bit codewords, it is also required to give out the leading runlength rl_l at the beginning of the codeword and the trailing runlength rl_t at the end of the codeword, where $rl_c, rl_l, rl_t \in \{0, 1, \dots, l \cdot q\}$. As a result, the capacity of the ROM for controlling the runlength is

$$C_m = (3 \cdot [\log_2(l \cdot q + 1)] + r + S) \times 2^{l \cdot p + r + S} \text{ [bits]}. \quad (10)$$

* For convenience, we allot the minus level to a bit “0” and the plus level to a bit “1”, respectively.

Fig. 6 Illustrative examples of calculating a RDS and polarity. The upper diagram a) indicates a current codeword to be connected to the code sequence. The lower diagram b) shows that values of RDS and polarity are renewed after the connection of this codeword. In the second case, the NRZI waveform of the current codeword is reverted since the polarity is plus at the end of the previous codeword.

(2) GF-Addition GS

Fig. 7 shows the ROM configurations of the GF-additional GS, where it is supposed that the parity-check sequence within a data block consists of K bits. The input information to the ROM is the data before scrambling of $l \cdot p$ bits, S -bit data indicating the current state of an RLL encoder, and address of parity data of $[\log_2(K/(l \cdot p) + 1)]$ bits, where $K \bmod l \cdot p = 0$. This address is defined as shown in **Fig. 8**. The data of every $l \cdot p$ bits that is cut out of the K parity bits is numbered from 1 to $K/(l \cdot p)$ sequentially. The address “0” is always assigned to the data of $l \cdot p$ bits among the information sequence of an RS codeword since the same data pattern of $l \cdot p$ bits is repeated in that region.

The ROM outputs the state information of S bits irrespective of channel-constraint parameters. The output data, which depend on the characteristic parameter, are the same as those of the ROM of the convolutional GS shown in Fig. 5. Therefore the ROM for DC suppression has a capacity of

$$C_m = ([\log_2(l \cdot q + 1)] + 1 + S) \times 2^{l \cdot p + [\log_2(K/(l \cdot p) + 1)] + S} \text{ [bits]} \quad (11)$$

and the ROM for lowering the maximum runlength has

Fig. 7 ROM configurations of the GF-addition GS. a) and b) are used for suppressing the DC and for control of the maximum runlength, respectively.

Fig. 8 Address of parity data of $l \cdot p$ bits for the ROM of the GF-addition GS. In the source symbols of an RS codeword, the same data of $l \cdot p$ bits is repeated, so only one address of "0" is mapped to that.

a capacity of

$$C_m = (3 \cdot \lceil \log_2(l \cdot q + 1) \rceil + S) \times 2^{l \cdot p + \lceil \log_2(K/(l \cdot p) + 1) \rceil + S} \text{ [bits]}. \quad (12)$$

3.3 Working Examples

In this section, the two examples of ROMs of Fig. 5 and Fig. 7 are explained, including how they operate.

(1) Runlength Control by Convolutional GS

In this example, the ROM is used for the convolutional GS of $r = 4$ for the purpose of maximum runlength control on the condition that the minimum runlength d is 0. In this case, an RLL encoder is not used, so $p = 1$, $q = 1$, and $S = 0$. Assume that $l = 4$, the number of input bits to the ROM is 8. The output data consists of the convolutional result of 4 bits and 3 types of runlengths of $3 \cdot \lceil \log_2(4 \cdot 1 + 1) \rceil$ bits. In total, the number of output bits from the ROM is 13. Therefore the capacity of this ROM is $2^8 \times 13$ bits.

Next, data processing in the ROM is explained. As an instance of this processing, assume that the input data word of 4 bits is "0001", and the scrambled data which was just converted and output from the ROM is "1000". An exclusive-OR operation is performed between "0001" and "1000", and "1001" is obtained. In this case, rl_c , rl_t , and rl_r are 2, 0, and 0, respectively. These values exist in the runlength set $\{0, 1, 2, 3, 4\}$, and the bits corresponding to these values are output from

the ROM.*

(2) DC Suppression by Concatenation of GF-Addition GS with (1, 7) RLL Coding

In the next example, the ROM is used in the concatenation of the guided scrambler based on a GF-addition operation with a (1, 7) RLL encoder for the purpose of DC suppression. In Table 1, the look-up table of a (1, 7) RLL code is shown, which has been designed by Weather and Wolf¹⁷⁾. It has a data word of 2 bits and four states of "a" to "d", so that $p = 2$ and $S = 2$, respectively. In an error correction code, it is assumed that the numbers of source bits and parity bits are 560 and 80, respectively. Also assuming that $l = 2$, the address of the parity data consists of the following number of bits: $\lceil \log_2(80/(2 \cdot 2) + 1) \rceil = 5$. After all, the number of input bits is 11. As for the output data, the numbers of bits representing CDS, polarity and state are $\lceil \log_2(2 \cdot 3 + 1) \rceil$, 1 and 2, respectively. Therefore the capacity of this ROM is $2^{11} \times 6$ bits.

Next, the way to operate in the ROM is made apparent. In this example of a ROM, it is assumed that the sequence in the information part of a scrambling RS codeword is $\{0101\}^{140}$, i.e. the data "0101" is repeated 140 times.** First, we describe the case in which the value of RDS is calculated in the information part. Assume that the current state is "a" and the input data of 4 bits is "1000". Since the address of parity data is 0, "0101" is selected as the data of the scrambling RS codewords. An exclusive-OR operation is performed between "1000" and "0101", and "1101" is obtained. In the time when the scrambled data are RLL-encoded by the conversion rule of Table 1, the state changes from "a" to "b" as the data word of the first 2 bits is "11", and then changes again from "b" to "c" as the data word of the last 2 bits is "01". At the same time, the code-word "001010" is obtained, and so the values of CDS and polarity are "-2" and "0", respectively. This CDS of the codeword of 6 bits is the value in the CDS set $\{-6, -4, -2, 0, +2, +4, +6\}$. Therefore the 3-bit data corresponding to "-2" is given out from the ROM.

When the RDS is calculated in the parity sequence of the RS codeword, the operation in a ROM is equivalent to that of the information sequence. The scrambling

* As an example of the bit allocation, the runlengths of "0", "1", "2", "3", and "4" are assigned to bits of "000", "001", "010", "011", and "100", respectively. In the channel-constraints comparator shown in Fig. 4, these bits are mapped inversely to runlengths and then the maximum runlength within a data block is calculated.

** L kinds of ROMs have different data patterns. One of these ROMs has the data of $\{0101\}^{140}$.

Table 1 Look-up table of a (1, 7) RLL code of rate 2/3, which has four states.

Data word State	00		01		10		11	
	Code word	Next state	Code word	Next state	Code word	Next state	Code word	Next state
<i>a</i>	000	<i>d</i>	000	<i>c</i>	001	<i>a</i>	001	<i>b</i>
<i>b</i>	010	<i>d</i>	010	<i>c</i>	010	<i>a</i>	000	<i>a</i>
<i>c</i>	100	<i>d</i>	100	<i>c</i>	101	<i>a</i>	101	<i>b</i>
<i>d</i>	010	<i>d</i>	010	<i>c</i>	010	<i>a</i>	000	<i>a</i>

data is automatically decided from the address of the parity as there is a one-to-one correspondence between the scrambling data words and the addresses. After that, the operation is executed in the same way that the information sequence is done.

3.4 Comparison of ROM Sizes

We have calculated ROM sizes for controlling the maximum runlength by using Eqs.(10) and (12). **Fig. 9** illustrates the ROM sizes when the number of parity bits K is varied in the case that $l \cdot p$ equals 2, 4, or 8,* where 4-bit exclusive-OR operations are used**. On the condition that both GS schemes make use of the same values of $l \cdot p$ and r one another, this comparison leads us to a fair judgement because both schemes can control the parameters of channel constraint to almost the same extent⁷⁾⁽¹¹⁾. The runlength control is performed as described in Section 3.3 (1). It is apparent from Fig. 9 that complexity of convolutional GS is independent of K and that of GF-addition GS varies in steps. The ROM size of the GF-addition GS is smaller than that of the convolutional GS when K is smaller than about 30, 60, and 130 with $l \cdot p$ of 2, 4, and 8, respectively. The ROM size of the GF-addition GS is slightly larger than that of the convolutional GS when K is in the region of 30 to 60, 60 to 130, and 130 to 250 with $l \cdot p$ of 2, 4, and 8, respectively. In these regions, it is presumably feasible to implement the hardware of the GF-addition GS with present-day LSI technology like the convolutional GS can be implemented.

We have also calculated ROM sizes for suppressing the DC content by using Eqs.(9) and (11). **Fig. 10** illustrates the ROM sizes when the number of parity bits

Fig. 9 Comparison of ROM sizes for controlling the maximum runlength when the number of parity bits K is varied. Exclusive-OR operations of 4 bits are used in both schemes.

Fig. 10 Comparison of ROM sizes for minimizing the DC content when the number of parity bits K is varied. Exclusive-OR operations of 4 bits are used in both schemes.

K is varied in the case that $l \cdot p$ equals 2, 4, and 8, where 4-bit exclusive-OR operations are used. The DC suppression is performed as described in Section 3.3 (2). It is evident from comparison of the plotted graphs that relative ROM sizes of the GF-addition GS to the convolutional GS shown in Fig. 10 are equal to those shown in Fig. 9. The larger $l \cdot p$ is, the larger the number of parity bits is, under the condition of equal-sized ROMs, for both schemes.

Next we have computed the ROM sizes of both

* The simulation in which $l \cdot p$ is over 8 is not performed since the processing under this condition presumably makes a channel sequence contain more DC than the content desired in a practical recording system.

** The decision to use 4-bit conversions has been made on the basis of the literature 7). The conversion using a smaller number of bits is not enough for optimizing the parameter of the channel constraints. On the other hand, the conversion using a larger number of bits requires to perform a huge amount of operations as the exponentially increasing number of candidates for the code block.

Fig. 11 Comparison of ROM sizes when the trade-off parameter l ranges from 1 to 8. In the case of $S = 0$, the runlength control is performed. In the case of $S = 2$, the DC suppression for a (1, 7) RLL code is performed.

schemes when the number of parity bits K is fixed at 80 and the trade-off parameter l is varied. In recent recording systems such as CD (Compact Disc) and DVD (Digital Versatile Disk), RS codes over $GF(2^8)$ are used and the number of parity bits ranges from 30 to 80. For the CD format, a *cross-interleaved Reed-Solomon code* (CIRC) with the (32, 28) outer code and (28, 24) inner code is used¹³⁾. For the DVD format, a *Reed-Solomon product code* (RS-PC) with the (208, 192) outer code and (182, 172) inner code is used¹⁸⁾. In playback systems, the decoding of an inner RS code can be performed first after the RLL decoding. Therefore we chose a parity data of 80 (10×8) bits like the inner RS code of the DVD format. **Fig. 11** illustrates the ROM sizes of the convolutional GS and GF-addition GS when the trade-off parameter l is varied. In the case of $S = 0$, the runlength control is performed as described in Section 3.3 (1). In the case of $S = 2$, the suppression of the DC content within the (1, 7) RLL-coded sequences is performed as described in Section 3.3 (2). In each case, the ROM size of the convolutional GS is smaller than that of the GF-addition GS when $l \cdot p$ is less than about 5, and vice versa.^{***}

^{***} In the case of $S = 2$, the ROM sizes of both schemes are almost equal each other when $l \cdot p$ equals about 5 because p equals 2 and the point of intersection between the straight line and dotted line in Fig. 11 is about 2.5. As a result, the ROM sizes of both cases are equal when $l \cdot p$ is close to 5.

4. Conclusion

First we introduced the principle of GS, and described the convolutional and GF-additional techniques. Next we considered the hardware configuration using ROM to calculate the hardware complexity, and then compared the ROM sizes of the convolutional GS and GF-addition GS. From this comparison, it was found that the ROM size of the GF-addition GS is smaller than that of the convolutional GS when the number of parity bits is not too large. The comparison also indicated that the ROM size of the GF-addition GS is smaller than that of the convolutional GS when the duration of sampling on the parameter of the channel constraints is not too short. Under these conditions, we expect that the hardware implementation of the GF-addition GS is feasible as that of convolutional GS is realized in present-day LSI technology.

Acknowledgment

The author is grateful to Hisashi Osawa for providing valuable remarks which improved the early version of this paper. He also wishes to thank Naoyuki Takagi and anonymous reviewers whose remarks contributed greatly to the quality of this paper.

[References]

- 1) J. W. M. Bergmans, *Digital Baseband Transmission and Recording*, Kluwer Academic Publishers, 1996.
- 2) K. A. S. Immink, *Codes for Mass Data Storage Systems*, Shannon Foundation Publishers, 1999.
- 3) K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for Digital Recorders," *IEEE Trans. Inform. Theory*, vol. IT-44, no. 6, pp. 2260-2299, Oct. 1998.
- 4) I. J. Fair, W. D. Gover, W. A. Krzymien, and R. I. MacDonald, "Guided Scrambling: A New Line Coding Technique for High Bit Rate Fiber Optic Transmission Systems," *IEEE Trans. Commun.*, vol. COM-39, no. 2, pp. 289-297, Feb. 1991.
- 5) A. Kunisa, S. Takahashi, and N. Itoh, "A Study of DC Component Suppressing Method for Digital Signal Recording," *ITE Japan*, vol. 49, no. 10, pp. 1361-1364, Oct. 1995.
- 6) A. Kunisa and N. Itoh, "DC-Component Suppressing Method Based on Convolution Operation for Digital Recording," *Proc. Tokai-Section Joint Conference of the Six Institutes of Electrical and Related Engineers*, 515, Oct. 1996.
- 7) A. Kunisa, "Guided scrambling Based on Galois Field Addition Operation and Assessment of its Performance," *ITE Japan*, vol. 52, no. 10, pp. 1485-1493, Oct. 1998.
- 8) A. Kunisa and N. Itoh, "On Symbol Error Probability of DC Component Suppressing Systems," *IEICE Trans. Fundamentals*, vol. E81-A, no. 10, pp. 2174-2179, Oct. 1998.
- 9) A. Kunisa, "Comparison of Hardware Sizes between Convolutional GS and GF-Addition GS," *ITE Annual conference*, Aug. 1999.
- 10) A. Kunisa, "A Proposal on Simple Hardware Constitution of Guided Scrambling," *Proc. IEICE General Conference*, C-7-6, Mar. 1999.
- 11) A. Kunisa, "Runlength Control Based on Guided Scrambling for Digital Magnetic Recording," *IEICE Trans. Electronics*, vol. E82-C, no. 12, pp. 2209-2217, Dec. 1999.
- 12) A. Kunisa, S. Takahashi, and N. Itoh, "Digital Modulation Method for Recordable Digital Video Disc," *IEEE Trans. Consum. Electron.*, vol. CE-42, no. 3, pp. 820-825, Aug. 1996.

- 13) S. B. Wicker, and V. K. Bhargava, *Reed-Solomon Codes and their Applications*, IEEE Press, 1994.
- 14) S. Sumi, A. Takahashi, and T. Watanabe, "Advanced Storage Magneto-optical Disk (AS-MO) System," *Proc. Magneto-Optical Recording Int. Symposium '99*, pp. 173-176, 1999.
- 15) S. Maeda, H. Fuji, T. Okumura, H. Sato, Y. Murakami, and A. Takahashi, "Format of Advanced Storage - Magneto Optical," *Sharp Tech. J.*, no. 72, pp. 46-50, Dec. 1998.
- 16) K. A. S. Immink, "Weakly Constrained Codes," *Electronics Letters*, vol. 33, no. 23, pp. 1943-1944, Sept. 1997.
- 17) A. D. Weathers and J. K. Wolf, "A New rate 2/3 Sliding Block Code for the (1, 7) Runlength Constraint with the Minimal Number of Encoder States," *IEEE Trans. Inform. Theory*, vol. IT-37, no. 3, pp. 908-913, May 1991.
- 18) K. A. S. Immink, "The Digital Versatile Disc (DVD) : System Requirements and Channel Coding," *SMPTE Journal*, pp. 483-489, Aug. 1996.

Akiomi Kunisa was born in Hiroshima, on July 24, in 1969. He received the B.E. and M.E. degrees from Hiroshima University, Hiroshima, in 1992 and in 1994, respectively, and the Dr.Eng. degree from Ehime University, Matsuyama, in 1999. Since 1994, he has been with SANYO Electric Co., Ltd. and engaged in research and development related to digital signal processing. He is currently a visiting researcher at Columbia University, New York, USA. His research interests include channel coding techniques for recording systems, picture coding, and watermarking. He was a recipient of the Young Engineer Award from SITA of Japan in 1997. He is a member of IEEE, IEICE of Japan and SITA of Japan.