Self-Synchronizing Reversible Variable Length Codes

Hiroyoshi Morita
Graduate School of Information Systems
University of Electro-Communications
Chofu Tokyo 182-8585, Japan
Email: morita@is.uec.ac.jp

Dongzhao SUN
Graduate School of Information Systems
University of Electro-Communications
Chofu Tokyo 182-8585, Japan
Email: daniel@math-sys.is.uec.ac.jp

***Abstract*— A reversible variable-length code (RVLC) is a code in which the bit stream formed by a portion of a codeword, or by the overlapped portion of two or more adjacent codewords, is not a valid codeword. A self-synchronizing RVLC allows for a sequence of codewords to be decoded either backwards or forwards without any external synchronization. In this article, we present an algorithm for constructing a self-synchronizing RVLC in which a codeword functions as a sync marker. The main idea is to replace a minimum forbidden word (MFW) of the codeword stream with a codeword. Then the MFW plays as a sync marker. Moreover, a lower bound of probability that at least one such a resynchronization codeword appears in a constant interval is obtained. A sync string of RVLC is discussed as well.**

Keywords **– reversible variable-length code (RVLC), resynchronization marker, minimum forbidden word, synchronizing sequence.**

I. Introduction

One of practical problems on noiseless variable-length source codes is synchronization slippage in the presence of channel errors. Even if there is a bit error in the sequence of codewords, a catastrophic decoding error propagation may happen. A solution to resynchronize the code is to insert periodically a *sync marker* into a sequence of codewords to be transmitted. Here, a sync marker means a string that is neither a portion of a codeword nor the overlapped portion of two or more adjacent codewords. Prefix-free codes with a sync marker have been studied extensively [1], [2], [3], [4]. Unfortunately, such a marker does not always exist for any code. But if it does, we can detect the boundary of codewords by finding it in a transmitted sequence of bits.

An alternative is to utilize a *synchronizing string*, or sync string, shortly, that appears only as a suffix of a codeword or of some adjacent codewords in a sequence of codewords. A sync string enables the decoder to continue exactly parsing the bit sequence into codewords after its occurrence, regardless of what bits preceded it. And it is known that almost all prefix-free codes have a sync string [5].

These two approaches for resynchronization described above, which have been mainly designed with prefix variable-length codes so far, have led to a growing level of interest in reversible variable-length codes as well. A reversible variable-length code (RVLC) is a code in which the bit sequence formed by a portion of a codeword, or by the overlapped portion of two or more adjacent codewords, is not a valid codeword. In other words, a codeword of an RVLC is neither prefix nor suffix of another codeword. Hence, an RVLC potentially allows for a sequence of codewords to be decoded either backwards or forwards if a sync marker or a sync string is detected in the sequence.

The algorithms for constructing an RVLC from the Huffman code have been studied in [6], [7], [8]. Especially, Lakovic and Villasenor [8] considered a formal relationship between the number of available codewords of an RVLC of length k and the structure of other codewords of length less than k, and proposed an effective algorithm to obtain a suboptimal RVLC from the Huffman code.

An RVLC is efficiently utilized to practical applications like MPEG video transmission for reducing the effect of error propagation during the transmission of the compressed video data [9]. When the decoder finds some decoding errors because of slippages, it can decode the bit sequence of codewords backwards from the point where the sync marker is found to the point where the error occurs. However, as far as the authors knows, there is no systematic approach for constructing a sync marker of an RVLC so far.

In this paper, we will present a method for constructing a self-synchronizing RVLC in which some codeword functions as a sync marker. That is, a sync marker is utilized not only for resynchronization but also for conveying information of source symbols. We call these codewords *sync codewords*. The main idea is to replace a codeword of a given RVLC with one of minimal forbidden words (MFW) [11] of a sequence of codewords in which every combination of t consecutive codewords appears where usually $t = 3$ in practical applications. To find sync codeword efficiently, a linear complexity algorithm for obtaining all MFWs of a given sequence [12] is utilized to the sequence of codewords above.

Throughout this paper, we will be concerned with only an RVLC obtained by the Lakovic and Villasenor algorithm [8] while the proposed method is applicable to any RVLC as well.

This paper is organized as follows. Section 2 gives an introduction of synchronous variable-length codes and the definition of a sync codeword in an RVLC. Then, in Section 3, the definition and properties of MFWs are introduced. In Section 4, we propose a construction algorithm for synchronous RVLC and show our experiment results. And a probabilistic analysis

This work was partially presented at ISITA2006, Seoul, 2006.

on occurrence of sync codeword are discussed in Section 5. Finally, Section 6 summaries our results.

II. Synchronous Variable-Length Codes

For a finite alphabet $\mathcal{X}$, let $\mathcal{X}^n$ be the set of all the strings of length n over $\mathcal{X}$ and $\mathcal{X}^* = \bigcup_{n\geq 0} \mathcal{X}^n$, which is the set of all finite-length strings including the null string λ of length 0. A set of strings $\mathcal{C} \subset \mathcal{X}^*$ is called a *code*. An element $\boldsymbol{w} \in \mathcal{C}$ is called a *codeword*.

A code is said to be *prefix-free* or *suffix-free*, if any codeword is neither a prefix nor a suffix of another, respectively. A code that is simultaneously prefix-free and suffix-free is called *fix-free* or *biprefix* [10].

Let $\mathcal{C}^n$ be the n-ary Cartesian product of $\mathcal{C}$ and define

$$\mathcal{C}^* = \bigcup_{n\geq 0} \mathcal{C}^n.$$

like as defining $\mathcal{X}^*$ from $\mathcal{X}$. Although an element of $\mathcal{C}^*$ is the null string or a finite-length sequence $(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n)$ of n codewords in $\mathcal{C}$ for some $n \geq 1$, we identify this sequence with a string $\boldsymbol{w} = \boldsymbol{w}_1\boldsymbol{w}_2 \ldots \boldsymbol{w}_n$ over $\mathcal{X}$ that is formed by concatenating them. Due to the prefix-free or suffix-free property of $\mathcal{C}$, the string $\boldsymbol{w}$ is uniquely parsed into codewords $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$.

A *synchronous code* $\mathcal{C}$ is a code such that at least one codeword $\sigma \in \mathcal{C}$ functions as a sync marker. Whenever σ is received without errors, the decoder must automatically resynchronize regardless of the preceding synchronization slippage. The formal definition of a sync codeword is given below:

*Definition 1 (*According to [1]): A codeword σ of a prefix-free code $\mathcal{C}$ is called a sync codeword if it satisfies either of the following two conditions:

(1.1) If σ is a substring of a codeword τ of $\mathcal{C}$, then σ is a suffix of τ but not equal to any other substring of τ.

(1.2) If there is a suffix α of a codeword of $\mathcal{C}$ such that $\sigma = \alpha\beta$, then $\beta \in \mathcal{C}^*$. □

Note that a sync string [5] is essentially defined in the same way as Definition 1 except that a sync string is not necessarily a codeword.

Definition 1 can not be directly applied to an RVLC since a codeword satisfying (1.1) and (1.2) of the RVLC is not always suffix-free against other codewords. Hence we will define sync codeword of an RVLC as follows:

Definition 2: A codeword σ of an RVLC $\mathcal{C}$ is called a sync codeword if the following condition holds:

If σ is a substring of $\boldsymbol{w}$ in $\mathcal{C}^*$, that is, $\boldsymbol{w} = \alpha\sigma\beta$ for $\alpha, \beta \in \mathcal{X}^*$, then $\alpha, \beta \in \mathcal{C}^*$.

An RVLC is said to be *synchronous* if it has at least one sync codeword.

III. Minimal Forbidden Words

An MFW is a minimal word that never appears in a string of symbols, which are used in a source coding scheme, called Data Compression using Antidicitionary (DCA) which is proposed by Crochemore *et al.*[11].

The dictionary $\mathcal{D}(\boldsymbol{x})$ of $\boldsymbol{x}$ in $\mathcal{X}^*$ is defined as the set of all the substrings of $\boldsymbol{x}$, that is,

$$\mathcal{D}(\boldsymbol{x}) = \{x_l x_{l+1} \cdots x_m | 1 \leq l \leq m \leq \ell(\boldsymbol{x})\} \cup \{\lambda\}.$$

where $\ell(\boldsymbol{x})$ is the length of $\boldsymbol{x}$.

Definition 3: A string $\boldsymbol{v} = v_1 v_2 \cdots v_m$ such that

i) $\boldsymbol{v} \in \mathcal{X}^* \setminus \mathcal{D}(\boldsymbol{x})$

ii) $v_1 v_2 \cdots v_{m-1} \in \mathcal{D}(\boldsymbol{x})$ and $v_2 v_3 \cdots v_m \in \mathcal{D}(\boldsymbol{x})$

is called a minimal forbidden word (MFW) of $\boldsymbol{x}$.
The antidictionary of $\boldsymbol{x}$, denoted by $\mathcal{AD}(\boldsymbol{x})$, is defined as the set of all the MFWs of $\boldsymbol{x}$.

There is a linear complexity algorithm to build $\mathcal{AD}(\boldsymbol{x})$ for a string $\boldsymbol{x}$ over a finite alphabet [12]. By means of this algorithm, we can obtain $\mathcal{AD}(\boldsymbol{x})$ in linear time proportional to length of $\boldsymbol{x} \in \mathcal{X}^*$. Once we get $\mathcal{AD}(\boldsymbol{x})$, it is easy to access any MFW through a tree representation of $\mathcal{AD}(\boldsymbol{x})$ [13].

According to Definition 2, to construct a synchronous RVLC, first we have to find an MFW of a certain sequence of codewords in which all possible arrangements of t codewords appear for a given t.

Suppose that $\mathcal{C}$ consists of m codewords and each codeword is labelled by an integer from 1 to m. Let $\mathcal{C}[t]$ be a sequence of codewords $\boldsymbol{w}_{\ell_1}\boldsymbol{w}_{\ell_2} \ldots \boldsymbol{w}_{\ell_N}$ in $\mathcal{C}$ where $N = m^t + t - 1$ and ℓ_i stands for the label of the ith codeword in the sequence. Moreover, we assume that all the substrings $\boldsymbol{w}_{\ell_i}\boldsymbol{w}_{\ell_{i+1}} \cdots \boldsymbol{w}_{\ell_{i+t-1}}$ consisting of t codewords in $\mathcal{C}[t]$ are distinct for $i = 1, \ldots, N - t + 1$, that is,

$$\#\left\{\boldsymbol{w}_{\ell_i}\boldsymbol{w}_{\ell_{i+1}} \cdots \boldsymbol{w}_{\ell_{i+t-1}} \,|\, i = 1, \ldots, N - t + 1\right\} = m^t$$

where $\#\mathcal{A}$ means the cardinality of the set $\mathcal{A}$.

In other words, a sequence $\boldsymbol{s}[t]$ of the indices of codewords in $\mathcal{C}[t]$ is one of the Hamiltonian paths on the m-ary de Bruijn graph of order t [14] where m is equal to $\#\mathcal{C}$. For example, for $m = 2$ and $t = 3$ where $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$ where the index means the label of each codeword, we have

$$\mathcal{C}[3] = \boldsymbol{c}_1\boldsymbol{c}_1\boldsymbol{c}_1\boldsymbol{c}_2\boldsymbol{c}_2\boldsymbol{c}_2\boldsymbol{c}_1\boldsymbol{c}_2\boldsymbol{c}_1\boldsymbol{c}_1.$$

For a given t, there is a possibility that an MFW of $\mathcal{C}[t]$ is used as a sync codeword in an RVLC. Suppose that an MFW $\boldsymbol{x}$ of $\mathcal{C}[t]$ has no correlation with itself. That is, no prefix $\boldsymbol{p}$ of $\boldsymbol{x}$ coincides with its suffix that has the same length as $\boldsymbol{p}$. Moreover, if $\mathcal{C} \bigcup \{\boldsymbol{x}\}$ is an RVLC, then the condition of Definition 2 holds for $\sigma = \boldsymbol{x}$. Even if $\boldsymbol{x}$ has a correlation with itself, it is still possible that $\boldsymbol{x}$ is a sync codeword of $\mathcal{C} \bigcup \{\boldsymbol{x}\}$. We take into account this condition in the next section.

IV. Construction of Synchronous RVLC

In order to construct a synchronous RVLC, we first build an RVLC by means of the LV algorithm [8], then replace one codeword with a sync codeword.

Algorithm MakeSyncRVLC

input: an RVLC $\mathcal{C}$.

output: a synchronous RVLC $\mathcal{C}^+$.

Step 1. Pick up a codeword $\boldsymbol{w} \in \mathcal{C}$ to be replaced and set $\mathcal{C}^\bigstar = \mathcal{C} \setminus \{\boldsymbol{w}\}$.

Step 2. Given t, generate $\mathcal{C}^\bigstar[t]$.

Step 3. Construct $\mathcal{AD}(\mathcal{C}^\bigstar[t])$ of the string $\mathcal{C}^\bigstar[t]$.

Step 4. /$*$ check the RVLC condition $*$/
Create the largest subset $\mathcal{S}$ of $\mathcal{AD}(\mathcal{C}^\bigstar[t])$ such that none of codewords in $\mathcal{C}$ is a prefix or a suffix of $\boldsymbol{x} \in \mathcal{S}$.

Step 5. /$*$ check the sync codeword condition $*$/
Let $\mathcal{T}$ be the empty set. For each $\boldsymbol{x} \in \mathcal{S}$, if $\boldsymbol{x}$ satisfies the condition of Definition 2, then put it into the set $\mathcal{T}$.

Step 6. Pick up the shortest $\boldsymbol{z}$ from $\mathcal{T}$, and then $\mathcal{C}^+ = \mathcal{C}^\bigstar \cup \{\boldsymbol{z}\}$.

Step 7. Stop. □

Here we give some remarks on MakeSyncRVLC above.

Remark 1: In our observation on the RVLCs constructed by the LV algorithm in **Step 1**, the zero-run $0\ldots0$ or then one-run $1\ldots1$ among the shortest codewords are often selected as $\boldsymbol{w}$ where the LV algorithm always produces a zero-run or one-run as one of the shortest codewords.

Remark 2: In **Step 2**, usually $t = 3$ can be used in practical applications. A program to construct $\boldsymbol{s}[3]$ is found in [16].

Remark 3: A linear complexity algorithm [12] is utilized to construct the antidictionary $\mathcal{AD}(\mathcal{C}^\bigstar[t])$ in **Step 3**.

Remark 4: According to Definition 3, an MFW $\boldsymbol{x}$ of $\mathcal{C}^\bigstar[t]$ is neither a prefix nor a suffix of any codeword of $\mathcal{C}^\bigstar$. However, it is possible that a codeword of $\mathcal{C}^\bigstar$ is either a prefix or a suffix of $\boldsymbol{x}$ since any substrings of $\boldsymbol{x}$ except itself appear in $\mathcal{C}^\bigstar[t]$. **Step 4** eliminate the MFWs in $\mathcal{AD}(\mathcal{C}^\bigstar[t])$ satisfying this condition.

Remark 5: **Step 5** is the most time consuming part of this algorithm. First, we find all pairs of strings α and β such that $\boldsymbol{x} = \alpha\gamma\beta$ where α is a suffix of either $\boldsymbol{x}$ or a codeword of $\mathcal{C}^\bigstar$ and β is a prefix of either $\boldsymbol{x}$ or a codeword of $\mathcal{C}^\bigstar$. Then, we can check if γ is the null string or completely parsed into codewords of $\mathcal{C}^\bigstar$. If this condition never holds for any pair of α and β described above, then $\boldsymbol{x}$ satisfies the condition of Definition 2.

Remark 6: Due to the suboptimality of the LV algorithm, the obtained sync codeword $\boldsymbol{z}$ tends to be longer than $\boldsymbol{w}$ selected at **Step 1**.

Remark 7: In our observation on the RVLCs constructed by the LV algorithm, a string of all zeros $0\ldots0$ or of all ones $1\ldots1$ among the shortest codewords is often chosen as $\boldsymbol{w}$ since the LV algorithm always produces those strings of the shortest codewords. □

As an example, the results of MakeSyncRVLC for an $i.i.d.$ source over English alphabet [8] are shown in Table I, where $p(a)$ denotes the probability of symbol a and the strings with boldfaced type indicates sync codewords in the RVLC obtained. The sync codeword 00000011 is found by MakeSyncRVLC with eliminating the codeword 000 which has the highest probability. And as a by-product, four codewords of the input RVLC that are indicated by 'bonus' in Table I become sync codewords.

We can see that a sync codeword of the obtained synchronous RVLC occurs with probability 0.03454 in spite of increasing the average codeword length by 0.20387 bits. In other words, for the $i.i.d.$ source in Table I, a sync codeword appears once in $\dfrac{1}{0.03454} \approx 29$ codewords on the average.

TABLE I
THE RVLCS IN [8] AND OBTAINED BY MakeSyncRVLC

a	$P(a)$	RVLC in [8]	proposed RVLC	sync status
E	0.14878	000	001	
T	0.09351	001	0100	
A	0.08833	0100	0101	
O	0.07245	0101	0110	
R	0.06872	0110	1010	
N	0.06498	1010	1011	
H	0.05831	1011	1100	
I	0.05644	1100	1101	
S	0.05537	1101	01110	
D	0.04376	01110	01111	
L	0.04124	01111	10010	
U	0.02762	10010	10011	
P	0.02575	10011	11110	
F	0.02455	11110	11111	
M	0.02361	11111	100010	
C	0.02081	100010	100011	
W	0.01868	100011	1000010	
G	0.01521	1000010	1000011	
Y	0.01521	1000011	1110111	
B	0.01267	1110111	**00000011**	found
V	0.01160	10000010	**10000010**	bonus
K	0.00867	10000011	**10000011**	bonus
X	0.00146	11100111	11100111	
J	0.00080	100000010	**100000010**	bonus
Q	0.00080	1000000010	**1000000010**	bonus
Z	0.00053	1110000111	1110000111	
Average Length		4.25145	4.45529	
Resync Prob.		0.00000	0.03454	

V. A LOWER BOUND OF PROBABILITY OF MARKERS

Suppose that the transmitted codewords are pooled into a buffer of size B in a FIFO manner at the receiver side. If there is a sync codeword in the buffer, we can decode data backward up to the position where an error occurred.

Let $\mathcal{M}$ be the set of all sync codewords in an RVLC $\mathcal{C}$ and $\mathcal{N} = \mathcal{C} \setminus \mathcal{M}$. Suppose that X be the position of the first sync

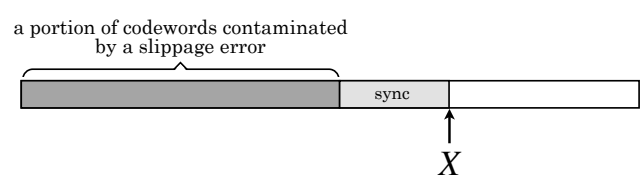

Fig. 1. The first sync codeword is located at X in the buffer.

codeword appears in the buffer (see Figure 1). We would like to evaluate the probability $\Pr\{X \le B\}$.

A lower bound of this problem can be derived by using the generating function method [15]. Let $\mathcal{G}$ be the set of all the strings that consist of a sequence over $\mathcal{N}$ followed by a sync codeword in $\mathcal{M}$. In other words, $\mathcal{G}$ can be written as

$$\mathcal{G} = \bigcup_{n \ge 0} \mathcal{N}^n \cdot \mathcal{M}. \tag{1}$$

where $\boldsymbol{u} \in \mathcal{N}^n \cdot \mathcal{M}$ means that there exist $\boldsymbol{v} \in \mathcal{N}^n$ and $\boldsymbol{w} \in \mathcal{M}$ such that $\boldsymbol{u} = \boldsymbol{v}\boldsymbol{w}$.

For a codeword $\boldsymbol{x}$ for source symbol a in $\mathcal{C}$, let $P(\boldsymbol{x})$ denote the probability of the occurence of a.

The equation on sets in (1) can be translated into the equation of the probability generating function

$$G(z) = \frac{M(z)}{1 - N(z)} \tag{2}$$

where

$$N(z) = \sum_{\boldsymbol{v} \in \mathcal{N}} p(\boldsymbol{v}) \cdot z^{\ell(\boldsymbol{v})} \tag{3}$$

$$M(z) = \sum_{\boldsymbol{w} \in \mathcal{M}} p(\boldsymbol{w}) \cdot z^{\ell(\boldsymbol{w})} \tag{4}$$

$$G(z) = \sum_{n \ge 0} \Pr\{X = n\} z^n. \tag{5}$$

A simple calculation using the Chernoff bound gives the following lower bound.

Theorem 1 ([16]): The probability of sync codeword appears in buffer of size θ satisfies

$$\Pr\{X \le \theta\} \ge \sup_{0 < s < \varphi} \left(1 - e^{-s\theta} \cdot G(e^s)\right),$$

where $\varphi \in (0, \infty)$ is a real root of $N(e^s) = 1$.

Proof: Omitted. □

For example, we calculated the lower bound of Theorem 1 for the source of English letters in Table I. Figure 2 shows the curve of the lower bound for $\theta \le 2000$. We also did computer simulation to estimate the probability using a long random sequence produced by a random number generator [17]. The results of the estimation are also illustrated in Figure 2. At $\theta = 2000$, the estimated probability is .99969 while the value of the lower bound is .99319. The difference is very slight for $\theta \ge 1000$.

VI. Sync strings of an RVLC

Up to here, we have mainly discussed on sync codewords of an RVLC where a single codeword plays a sync marker. However, there exists another type of a sync marker which composes of two or more non-sync codewords. As an example, Table II shows composite sync markers consisting of two non-sync codewords $\boldsymbol{v}$ and $\boldsymbol{w}$ of the RVLC in Table I. Once we have such a composite sync marker of an RVLC, some of its substrings can be still utilized as a sync string to resynchronize the code.

Definition 4: A string $\boldsymbol{u} \in \mathcal{X}^*$ is called a *sync string* of an RVLC $\mathcal{C}$ if there exists only a pair of a suffix $\boldsymbol{x}$ of a codeword $\boldsymbol{v} \in \mathcal{C}$ and a prefix $\boldsymbol{y}$ of a codeword $\boldsymbol{w} \in \mathcal{C}$ such that $\boldsymbol{u} = \boldsymbol{x}\boldsymbol{y}$. □

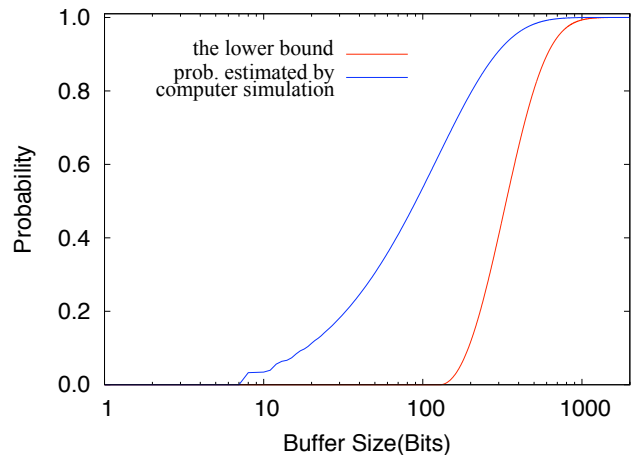

Fig. 2. Plots of the estimated probability (above) and the lower bound (below)

TABLE II

Composite Sync Markers of the RVLC in Table I

string	composite sync	$\boldsymbol{v}$	$\boldsymbol{w}$
VE	1000010001	1000010	001
VZ	10000101110000111	1000010	1110000111
ZZ	11100001111110000111	1110000111	1110000111

From Definition 4, a sync string $\boldsymbol{u}$ can be partitioned into two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ that satisfy one of the following conditions:

Type 1. $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{N}$.

Type 2. $\boldsymbol{x}$ is a suffix of $\boldsymbol{v} \in \mathcal{C}$, and $\boldsymbol{y} \in \mathcal{N}$.

Type 3. $\boldsymbol{x} \in \mathcal{N}$ and $\boldsymbol{y}$ is a prefix of $\boldsymbol{w} \in \mathcal{C}$.

Type 4. $\boldsymbol{x}$ is a suffix of $\boldsymbol{v} \in \mathcal{C}$ and $\boldsymbol{y}$ is a prefix of $\boldsymbol{w} \in \mathcal{C}$.

Here $\mathcal{N}$ is the set of non-sync codewords of $\mathcal{C}$.

These types are illustrated in Figure 3. For example, a composite sync codeword of an RVLC is of type 1. A sync string of either of a prefix-free code and a suffix-free code is categorized into type 2 and type 3, respectively. Type 4 is the

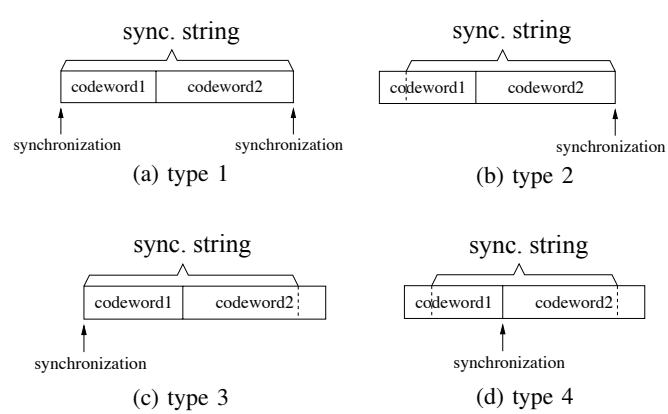

Fig. 3. Examples of sync strings constructed by two codewords

TABLE III

SYNC STRINGS OF THE PROPOSED RVLC IN TABLE I

type 2	type 3	type 4
000010001	1000010001	000010001
0000101110000111	100001011100001	00001011100001
00001111110000111	111000011110000001	000011111100001

most general condition which includes other types.

Theorem 2: If a substring $\boldsymbol{z}$ of the concatenation of two codewords $\boldsymbol{v}, \boldsymbol{w}$ of an RVLC $\mathcal{C}$, including $\boldsymbol{vw}$, contains a sync string $\boldsymbol{u}$ as a substring, then $\boldsymbol{z}$ is a sync string as well.

Proof: Omitted. □

Corollary 1: If $\boldsymbol{vw}$ is not a sync string, then no substring of $\boldsymbol{vw}$ is a sync string. □

Corollary 1 is useful in finding sync strings among codeword pairs of an RVLC. If a pair of codewords is not a sync string, then we know that none of its substring is a sync string.

Table III illustrated an example of sync strings of the RVLC obtained from the English alphabet source given in Table I. All of the sync strings in Table III have been obtained based on the composite sync markers in Table II.

VII. CONCLUSION

We proposed an algorithm for constructing a synchronous RVLC and applied it to an RVLC constructed by the LV algorithm. First, we find MFWs of a specific sequence of codewords, in which any combinations of t codewords appear for a certain value of t. Then, one of these MFWs takes the place of a codeword of the original RVLC. This MFW functions as a sync codeword as well as a codeword in a usual sense. Also, by using the probabilistic generating functions of sequence of codewords, we evaluated the probability of occurrence of sync codeword.

We also considered how to define the sync strings of an RVLC. Unlike to a prefix-free code, there are four types of sync strings in an RVLC. Our definition of the sync string is based on a composite sync marker which consists of two non-sync codewords. To characterize a sync string constructed by more than two codewords is one of our future works.

ACKNOWLEDGMENT

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Gran-in-Aid for Scientific Research (B), 17360178, 2005-6.

REFERENCES

[1] T. J. Ferguson and J. H. Rabinowitz, "Self-synchronizing Huffman codes," *IEEE Transactions on Information Theory*, vol.30, No.4, pp.687–693, July 1984.

[2] B. L. Montgomery and J. Abrahams, "Synchronization of binary source codes," *IEEE Transactions on Information Theory*, vol.32, No.6, pp.849–854, November 1986.

[3] R. M. Capocelli, L. Gargano, and U. Vaccaro, "On the characterization of statistically synchronizable variable-length codes", *IEEE Transactions on Information Theory*, vol.34, No.4, pp.817–825, July 1988.

[4] R. M. Capocelli, A. A. De Santis, L. Gargano, and U. Vaccaro, "On the construction of statistically synchronizable codes," *IEEE Transactions on Information Theory*, vol.38, No.2, pp.407–414, March 1992.

[5] C. F. Freiling, D. S. Jungreis, F. Theberge, and K. Zeger, "Almost all complete binary prefix codes have a self-synchronizing string," *IEEE Trans. Inform. Theory*, vol.49, no.9, pp.2219-2225, Sep. 2003.

[6] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Trans. on Commun.*, vol. 43, pp.158–162, 1995.

[7] C. W. Tsai and J. L. Wu, "On constructing the Huffman-code based reversible variable length codes," *IEEE Transactions on Communications*, vol.49, pp.1506–1509, September 2001.

[8] Ksenija Lakovic and John Villasenor, "An algorithm for construction of efficient fix-free codes," *IEEE Commun. Letters*, vol.7, No.8, pp.391–393, August 2003.

[9] J. Wen and J. D. Villasenor, Reversible variable length codes for efficient and robust image and video coding," *Proc. the 1998 IEEE Data Compression Conference*, pp. 471–480, 1998.

[10] C. Ye and R. W. Yeung, "Some basic properties of fix-free codes," *IEEE Trans. Inform. Theory*, vol.47, pp.72–87, January 2001.

[11] M. Crochemore, F. Mignosi, A. Restivo, and S. Salemi, "Data compression using antidictionaries," *Proceedings of the IEEE*, vol.88, No.11, pp.1756–1768, 2000.

[12] T. Ota and H. Morita, "On the construction of an antidictionary of a binary string with linear complexity," *Proc. of IEEE Intr'l Symp. of Inform. Theory,* Seattle, pp. 2343–2347, 2006.

[13] H. Morita and T. Ota, "A tight upper bound on the size of the antidictionary of a binary string," *2005 International Conference on the Analysis of Algorithms(AofA'05), Discrete Mathematics and Theoretical Computer Science Proceedings*, AD, pp.393–398, 2005.

[14] N. G. de Bruijn, "A combinatorial problem," *Proc. Nederl. Akad. Wetensch.*, 49, pp.758–764, 1946.

[15] R. Sedgewick and P. Flajolet, *An introduction to the analysis of algorithms,* Addison-Wesley, November, 1995.

[16] Dongzhao Sun, *On multiple smoothed transmission of MPEG4 video streams with error resilience*, PhD dissertation submitted to the Univ. of Electro-communications, Tokyo, Japan, Dec. 2006.

[17] `http://www.toshiba.co.jp/efort/product/rmaster/` (in Japanese)