

# Using Reed–Muller $RM(1, m)$ Codes Over Channels With Synchronization and Substitution Errors

Lara Dolecek, *Student Member, IEEE*, and Venkat Anantharam, *Fellow, IEEE*

**Abstract**—We analyze the performance of a Reed–Muller  $RM(1, m)$  code over a channel that, in addition to substitution errors, permits either the repetition of a single bit or the deletion of a single bit; the latter feature is used to model synchronization errors. We first analyze the run-length structure of this code. We enumerate all pairs of codewords that can result in the same sequence after the deletion of a single bit, and propose a simple way to prune the code by dropping one information bit such that the resulting linear subcode has good post-deletion and post-repetition minimum distance. A bounded distance decoding algorithm is provided for the use of this pruned code over the channel. This algorithm has the same order of complexity as the usual fast Hadamard transform based decoder for the  $RM(1, m)$  code.

**Index Terms**—Deletion, Hadamard transform, insertion, Reed–Muller codes, synchronization.

## I. INTRODUCTION

IN a typical communication system, a binary input message  $\mathbf{x}$  is encoded at the transmitter, using a substitution-error correcting code  $C$ , into a coded sequence  $\mathbf{c} = C(\mathbf{x})$ , which we will assume is also a binary sequence. The modulated version of this sequence may be modeled as being corrupted by additive noise, so the received waveform after matched filtering can be written as

$$r(t) = \sum_i c_i h(t - iT) + n(t) \quad (1)$$

where  $c_i$  is the  $i$ th bit of  $\mathbf{c}$ ,  $h(t)$  is convolution of the modulating pulse and the matched filter, and  $n(t)$  represents the noise introduced by the channel.

The receiver samples  $r(t)$  at time instances  $\{kT_s + \tau_k\}$ , and the sequence of samples is fed into the decoder which decides on the most likely input message. Accurate synchronization of the sampling instants, i.e., that  $T_s$  be equal to  $T$  and that each  $\tau_k$  be ideal, is critical for the full utilization of the coding gain of the substitution-error correcting code. As the operating requirements under which timing recovery must be performed become more stringent, because of higher data rates and/or longer delays in the decision feedback loop that adjusts the sampling instants,

Manuscript received April 26, 2006; revised December 13, 2006. This work was supported by the National Science Foundation under Grant ECS 0123512, Marvell Semiconductor, and the California MICRO program. The material in this paper was presented in part at the 42nd Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, October 2004.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94720 USA (e-mail: dolecek@eecs.berkeley.edu; ananth@eecs.berkeley.edu).

Communicated by M. P. C. Fossorier, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2007.892776

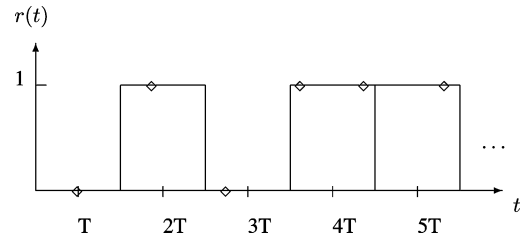


Fig. 1. An example of oversampling.

such synchronization is becoming harder to achieve. Several authors have studied the problem of accurate timing recovery. Proposed solutions include building a more sophisticated timing recovery block [16], multiple hypothesis analysis of the sampling instances [13], and for the intersymbol interference (ISI) channels in particular, a soft-output detector for both ISI and timing errors [24], and an iterative timing recovery approach that incorporates timing recovery in turbo equalization [1].

As an alternative to more complex and more expensive timing recovery schemes, we propose to shift the emphasis away from the timing recovery block and instead modify the decoding procedure and the code itself to compensate for inadequate synchronization. By analyzing the robustness of a substitution-error correction code to synchronization errors, one could use a subcode of the original code that would have good minimum distance under both substitution and sampling errors. The tradeoff would be between the incurred rate loss associated with the code modification versus the increased complexity and latency associated with the existing approaches mentioned above. The challenge of the proposed approach lies in determining the synchronization error correction capabilities of individual codes of interest, and in determining as large as possible a subcode with the desired properties.

To illustrate the issues that arise when adequate timing recovery is missing, assume (for purposes of argument) that  $h(t)$  is a rectangular pulse of duration  $T$  and unit amplitude and that we are operating in the infinite signal-to-noise ratio (SNR) regime where the effect of  $n(t)$  is negligible. Then  $r(t)$  simply becomes

$$r(t) = \sum_i c_i 1(iT \leq t < (i+1)T). \quad (2)$$

If samples were taken in the middle of each pulse, the sampled version of  $r(t)$  would be precisely  $\mathbf{c}$ . Now suppose that inadequate timing recovery causes the sampling to occur at time instants  $kT_s + \tau_k$ .

As an example, consider a sequence  $\mathbf{c} = (0, 1, 0, 1, 1, \dots)$  that results in the waveform  $r(t)$  shown in Fig. 1. The sampling points  $kT_s + \tau_k$  are marked in the figure by  $\diamond$ . In this example,

$T_s < T$  causes oversampling, and the sampled version of  $r(t)$  contains a repeated bit (here the fourth bit is sampled twice). Analogously, when  $T_s > T$ , undersampling can cause the separation between two consecutive samples to be so large that some bit is not sampled at all. Therefore, without adequate timing recovery the sampled version of  $r(t)$  results in a sequence obtained by repeating or deleting some bits in  $\mathbf{c}$ .

A codeword  $\mathbf{c}$  can in general give rise to a whole set of received sampled versions of  $r(t)$ . The possible set of such sequences depends on how good the timing recovery scheme is. When two distinct codewords  $\mathbf{c}_1$  and  $\mathbf{c}_2$  can result in the same sampled sequence, it is no longer possible to uniquely determine the coded sequence or its pre-image  $\mathbf{x}$  from the received sequence, even in the noise-free environment. We then say that the substitution-error correcting code  $C$  has an *identification problem*. We also say that the pair of codewords  $\mathbf{c}_1$  and  $\mathbf{c}_2$  has an identification problem.

More generally, two distinct codewords  $\mathbf{c}_1$  and  $\mathbf{c}_2$  could result in sampled sequences with poor Hamming distance. This would result in poor performance over a channel that permits substitution errors. In this case we say that the substitution-error correcting code  $C$  has *poor identification*. We also say that the pair of codewords  $\mathbf{c}_1$  and  $\mathbf{c}_2$  has poor identification.

In this paper, we adopt a set-theoretic model for the synchronization errors, in which a codeword gives rise to a set of possible received sampled sequences, which depends on how many bits are allowed to be repeated or deleted. In this context, our goal is to ensure that we have good identification by restricting attention to a large linear subcode for which each pair of distinct codewords has good post-synchronization error Hamming distance. Further, we would like to analyze the performance of this subcode when used over a channel that introduces both substitution and synchronization errors. In this paper we address such questions for the  $\text{RM}(1, m)$  code.

It should be mentioned that several authors have studied codes immune to insertions and deletions of bits. For example, the so-called Varshamov–Tenengolts code proposed in [23] and popularized by Levenshtein in [14] has been further studied by Ferreira *et al.*, [11], Levenshtein [15], Sloane [19], and Tenengolts [21]. Related constructions were proposed in [3], [4], [7], [12], and [22]. Even though these constructions result in codes that are immune to a given number of insertions and deletions of bits, they have a limited guarantee for other desirable properties of standard substitution-error correcting codes (such as linearity and a good minimum Hamming distance). Several other authors have proposed concatenated codes that correct synchronization errors, such as in [5], [6], and [8]. These have a significant incurred rate loss penalty. In contrast to these works, our approach is to start with known substitution-error correcting codes and propose how to modify them with only a small loss in the rate in order to continue to provide good performance under synchronization errors, which are themselves modeled as a certain number of repetitions or deletions of bits. A related problem of a code construction for frame synchronization was studied in [20] and [2].

We study  $\text{RM}(1, m)$  codes in this paper. In Section II, we prove several structural properties of the run-lengths of such a code. Using these properties, in Section III we systematically

analyze the identification problem for such codes for single deletion errors. We propose a simple way to prune an  $\text{RM}(1, m)$  code to obtain a linear subcode that does not suffer from the identification problem for a single deletion. This subcode is also shown to have good post-deletion and post-repetition minimum distance. In Section IV, we discuss how to decode the pruned code over channels in which substitution errors are present in addition to possibly the deletion of a single bit or the repetition of a single bit. We present a bounded distance decoding algorithm that is a variant of the fast Hadamard matrix based decoding which is traditionally used to decode the  $\text{RM}(1, m)$  codes. The complexity of this algorithm is of the same order as that of the traditional decoder. Finally, Section V concludes the paper and proposes future extensions of this work.

## II. RUN LENGTH PROPERTIES OF THE $\text{RM}(1, M)$ CODES

The first-order Reed–Muller codes  $\text{RM}(1, m)$  are linear  $(2^m, m + 1)$  substitution-error correcting codes [17]. They have good minimum distance, equal to  $2^{m-1}$ , simple encoding, and a relatively low complexity maximum-likelihood decoding algorithm ( $O(n \log n)$  for  $n = 2^m$ ). On the negative side, they have low rate.

From now on, let  $C(m)$  denote the  $\text{RM}(1, m)$  code. The code  $C(m)$  may be described by an  $(m + 1) \times 2^m$  generator matrix  $\mathbf{G}_m$  given by

$$\mathbf{G}_m = \begin{bmatrix} \mathbf{1} \\ \mathbf{M}_m \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \dots & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & 1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & \dots & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & \dots & 1 & 0 & 1 & 0 \end{bmatrix}$$

where  $\mathbf{1}$  denotes the binary string of length  $2^m$  with all entries equal to 1, and the  $m$  by  $2^m$  submatrix  $\mathbf{M}_m$  consists of lexicographically decreasing binary columns of length  $m$ . Observe that the  $i$ th row of  $\mathbf{G}_m$ , for  $1 < i \leq m + 1$ , consists of  $2^{i-1}$  alternating runs of ones and zeros, and that each run is of length  $2^{m-i+1}$ .

For future reference, we recall that every codeword in  $C(m + 1)$  is either the concatenation of a codeword in  $C(m)$  with itself or the concatenation of a codeword in  $C(m)$  with its bitwise complement [17, Theorem 2, p. 374]. The concatenation of two binary strings  $a$  and  $b$  will be written as  $[a | b]$ . If  $c$  is a codeword in  $C(m)$  it is straightforward to check that its bitwise complement, denoted  $\bar{c}$ , is also a codeword in  $C(m)$ . Further, its reversal, i.e., the binary string achieved by reading  $c$  from its end to its beginning, denoted  $\overleftarrow{c}$ , is also a codeword in  $C(m)$ . Since the operations of bitwise complementation and reversal commute, we may unambiguously denote the complement of the reversal of  $c$  as  $\overleftarrow{\bar{c}}$ .

The purpose of this section is to prove several properties of the run length structure of the codes  $C(m)$ . These properties will be used in the subsequent sections. They may also be of independent interest.

*Lemma 1:* The codewords in  $C(m)$  can be partitioned into  $2^{m-1} + 1$  distinct nonempty groups  $G_j^m$ , for  $0 \leq j \leq 2^{m-1}$ . Here  $G_j^m$  is comprised of those codewords in  $C(m)$  that have  $j$  runs of ones.  $G_0^m$  is comprised of exactly one codeword, namely, the all-zero codeword. This codeword will be denoted  $c_0^m(00)$ . There are four distinct codewords in each group  $G_j^m$ , for  $1 \leq j < 2^{m-1}$ . These codewords may be uniquely identified by their first and last bit. They may thus be unambiguously denoted as  $c_j^m(11)$ ,  $c_j^m(10)$ ,  $c_j^m(01)$ , and  $c_j^m(00)$ , respectively. There are three distinct codewords in the group  $G_{2^{m-1}}^m$ . These codewords may also be uniquely identified by their first and last bit and may be unambiguously denoted as  $c_{2^{m-1}}^m(11)$ ,  $c_{2^{m-1}}^m(10)$ , and  $c_{2^{m-1}}^m(01)$  respectively.

*Proof:* The proof is by induction on  $m$ . For  $m = 1$  and  $m = 2$ , the statement can be verified by inspection. Suppose the assertion holds for all  $1 \leq m \leq m_0$ .

Let us first consider the group  $G_j^{m_0}$  for  $1 \leq j < 2^{m_0-1}$ . By assumption, it contains four codewords, unambiguously denoted as  $c_j^{m_0}(11)$ ,  $c_j^{m_0}(01)$ ,  $c_j^{m_0}(10)$ , and  $c_j^{m_0}(00)$ , respectively. Out of the eight possible concatenations of each such codeword with either itself or its complement, three result in codewords in  $G_{2j-1}^{m_0+1}$  (these are  $[c_j^{m_0}(11)|c_j^{m_0}(11)]$ ,  $[c_j^{m_0}(11)|\overline{c_j^{m_0}(11)}]$ , and  $[c_j^{m_0}(01)|\overline{c_j^{m_0}(01)}]$ ), four result in codewords in  $G_{2j}^{m_0+1}$  (these are  $[c_j^{m_0}(01)|c_j^{m_0}(01)]$ ,  $[c_j^{m_0}(10)|\overline{c_j^{m_0}(10)}]$ ,  $[c_j^{m_0}(10)|c_j^{m_0}(10)]$ , and  $[c_j^{m_0}(00)|c_j^{m_0}(00)]$ ), and one results in the codeword  $[c_j^{m_0}(00)|\overline{c_j^{m_0}(00)}]$  in  $G_{2j+1}^{m_0+1}$ . By varying  $j$  from 1 to  $2^{m_0-1} - 1$ , inclusive, we thus describe three codewords in  $G_1^{m_0+1}$ , four codewords in each  $G_{j'}^{m_0+1}$  for  $2 \leq j' \leq 2^{m_0} - 2$ , and one codeword in  $G_{2^{m_0}-1}^{m_0+1}$  such that no two codewords that belong to the same group  $G_{j'}^{m_0+1}$  agree in both the first and the last bit.

Now consider the group  $G_{2^{m_0-1}}^{m_0}$ . By assumption it has three codewords unambiguously denoted as  $c_{2^{m_0-1}}^{m_0}(11)$ ,  $c_{2^{m_0-1}}^{m_0}(01)$ , and  $c_{2^{m_0-1}}^{m_0}(10)$ , respectively. There are six possibilities arising from concatenations of such a codeword with itself or its complement. Of these, three result in codewords in  $G_{2^{m_0}-1}^{m_0+1}$  (these are  $[c_{2^{m_0-1}}^{m_0}(01)|\overline{c_{2^{m_0-1}}^{m_0}(01)}]$ ,  $[c_{2^{m_0-1}}^{m_0}(11)|c_{2^{m_0-1}}^{m_0}(11)]$ , and  $[c_{2^{m_0-1}}^{m_0}(11)|\overline{c_{2^{m_0-1}}^{m_0}(11)}]$ ) and the remaining three result in the codewords of  $G_{2^{m_0}}^{m_0+1}$ . Note that none of the latter three concatenations has both outer bits equal to "0." Note that we have now described a total of four codewords in the group  $G_{2^{m_0}-1}^{m_0+1}$ , no two agree in both first and last bit, and we have also described three codewords in the the group  $G_{2^{m_0}}^{m_0+1}$  of the desired form.

The concatenation of the all-zero codeword in  $C(m_0)$  with the all-ones codeword yields the fourth codeword in  $G_1^{m_0+1}$ , and its concatenation with itself yields the only codeword in  $G_0^{m_0+1}$ .

We have therefore described  $1 + 4 \times (2^{m_0} - 1) + 3 = 2^{m_0+2}$  codewords in  $C(m_0 + 1)$ , which is precisely the cardinality of this code, and we showed that the proposed statement holds for it.  $\square$

By exploiting the result in Lemma 1, it is easy to verify the following, which may also of course be seen more directly.

*Lemma 2:* For each  $1 \leq k \leq 2^m$ , in  $C(m)$  there are exactly two codewords which have a total of  $k$  runs, and they are bitwise complements of each other.

*Proof:* The complementary codewords  $c_{j-1}^m(00)$  and  $c_j^m(11)$  each have  $2j - 1$  runs. Letting  $j$  run from 1 through  $2^{m-1}$  gives  $2^{m-1}$  such complementary pairs of codewords. The complementary codewords  $c_j^m(10)$  and  $c_j^m(01)$  each have  $2j$  runs. Letting  $j$  run from 1 to  $2^{m-1}$  gives another  $2^{m-1}$  such complementary pairs of codewords. This completes the proof.  $\square$

*Lemma 3:* Consider a codeword  $\mathbf{c}$  in  $C(m)$ . Either  $\mathbf{c}$  has all its runs of the same length, which is a power of 2, or the runs in  $\mathbf{c}$  are of two different lengths, and these two lengths are consecutive powers of 2. In addition, if there are runs of two different lengths in  $\mathbf{c}$ , the outer runs (i.e., the leftmost run and the rightmost run) in  $\mathbf{c}$  are of the smaller length.

*Proof:* The proof is by induction on  $m$ . It is straightforward to check the truth of the statement for  $m = 1$  and  $m = 2$ . Suppose now that the given statement is true for all  $1 \leq m \leq m_0$ . For a codeword  $\mathbf{c}$  in  $C(m_0)$ , let  $[\mathbf{c}|\mathbf{c}]$  and  $[\mathbf{c}|\overline{\mathbf{c}}]$  denote the codewords in  $C(m_0 + 1)$  that are the concatenation of  $\mathbf{c}$  with itself and the concatenation of  $\mathbf{c}$  with its complement, respectively.

Suppose first that  $\mathbf{c}$  has all its runs of the same length, equal to  $2^s$  for some  $s \geq 0$ . If  $\mathbf{c}$  has the same starting and ending bits then in the concatenation  $[\mathbf{c}|\overline{\mathbf{c}}]$  all runs have the same length  $2^s$ , so the statement of the lemma holds. In the concatenation  $[\mathbf{c}|\mathbf{c}]$  all runs except the run at the point of concatenation (if there are any such runs) have length  $2^s$  and the run at the point of concatenation has length  $2^{s+1}$ . The proposed statement continues to be true both in the case in which there are some runs other than the one at point of concatenation and in the case when there are no such runs. If  $\mathbf{c}$  starts and ends with different bits, we may repeat the previous argument *mutatis mutandis*.

Now suppose that  $\mathbf{c}$  has runs of different lengths, which are two consecutive powers of 2, say  $2^s$  and  $2^{s+1}$ . By assumption, the outer runs are of length  $2^s$  each, and there is at least one run of length  $2^{s+1}$ . As before, if  $\mathbf{c}$  starts and ends in the same bit, the concatenation  $[\mathbf{c}|\overline{\mathbf{c}}]$  will have all its runs of lengths either  $2^s$  or  $2^{s+1}$ . Further, the outer runs in  $[\mathbf{c}|\overline{\mathbf{c}}]$  have the same length as the ones in  $\mathbf{c}$ , i.e., they are of length  $2^s$  each, so the statement of the lemma is valid. In the concatenation  $[\mathbf{c}|\mathbf{c}]$ , the last run in the left copy of  $\mathbf{c}$  and the first run in the right copy of  $\mathbf{c}$  are merged together, and all other runs are unchanged in length. By assumption, the outer runs in  $\mathbf{c}$  have length  $2^s$  each, so their merger results in a run in  $[\mathbf{c}|\mathbf{c}]$  of length  $2 \times 2^s = 2^{s+1}$ . Thus, all runs in  $[\mathbf{c}|\mathbf{c}]$  have length either  $2^s$  or  $2^{s+1}$ . Since the outer runs in  $[\mathbf{c}|\mathbf{c}]$  are of the same length as the outer runs in  $\mathbf{c}$ , they have length  $2^s$ , as required. For  $\mathbf{c}$  starting and ending in different bits, we repeat this argument *mutatis mutandis*.

Since each codeword in  $C(m_0 + 1)$  can be written as a concatenation of a codeword in  $C(m_0)$  either with itself or with its complement, the proof of the Lemma is complete.  $\square$

For the analysis in subsequent sections we also need to record some properties of the runs of runs in the codewords of  $\text{RM}(1, m)$ .

*Definition 1:* For a codeword  $\mathbf{c} \in C(m)$  let  $\mathbf{d} = d(\mathbf{c})$  be the string whose entries are the lengths of consecutive runs in  $\mathbf{c}$ , read from left to right. Let  $\mathcal{D}_m = \{\mathbf{d} \mid \mathbf{d} = d(\mathbf{c}), \mathbf{c} \in C(m)\}$ , so that  $\mathcal{D}_m$  represents the collection of all possible sequences of run lengths associated with the codewords of  $C(m)$ .  $\square$

As an example, consider a codeword  $\mathbf{c} = "10010110,"$  where  $\mathbf{c} \in C(3)$ . Then, the associated  $\mathbf{d} = d(\mathbf{c})$  is  $\mathbf{d} = "121121."$

We now state several results about such sequences of run lengths, which we will prove together.

*Lemma 4 [mirror-symmetry]:*  $\forall \mathbf{c} \in C(m)$ , the string  $\mathbf{d} = d(\mathbf{c})$  possesses the mirror-symmetry property, i.e., the entry in position  $p$  in  $\mathbf{d}$ , denoted by  $\mathbf{d}(p)$ , is the same as the entry in position  $l - p + 1$ , denoted by  $\mathbf{d}(l - p + 1)$ , where  $l$  represents the length of string  $\mathbf{d}$ .

*Lemma 5:* If all entries in  $\mathbf{d} = d(\mathbf{c})$  are either 1 or 2, with at least one entry being 1 and one being 2, then the following holds.

- 1) The leftmost entry equal to 2 must be in position  $2^p$ , for some  $p \geq 1$ .
- 2) Each run of 2's in  $\mathbf{d}$  is of length  $2^q - 1$ , for some  $q \geq 1$ .
- 3) Each inner run of 1's (where the inner run denotes a run with neighboring runs on each side) in  $\mathbf{d}$  is of length  $2^r - 2$ , for some  $r \geq 1$ .

*Proof:* We prove these statements by induction. We first directly verify them for small values of  $m$ . The codewords in  $C(1)$  are "00," "11," "01," and "10," so  $\mathcal{D}_1 = \{ "2," "11" \}$ . The truth of the statements can be directly verified in this case. The codewords in  $C(2)$  are "0000," "1111," "1100," "0011," "0110," "1001," "1010," and "0101," so  $\mathcal{D}_2 = \{ "4," "22," "121," "1111" \}$ , and again the proposed statements can be verified. Similarly, the set associated with  $C(3)$  is

$$\mathcal{D}_3 = \{ "8," "44," "242," "2222," "12221," "121121," "1112111," "11111111" \}$$

and the statements hold. In particular, Lemmas 5.1 and 5.2 are applicable for the strings "12221," "121121," and "1112111," and Lemma 5.3 is applicable for the string "121121."

Suppose now that the proposed Lemmas hold for all elements of  $\mathcal{D}_m$  for  $1 \leq m \leq m_0$ . For a codeword  $\mathbf{c}$  in  $C(m_0)$  let  $\mathbf{c}' = [e|\mathbf{c}]$  and  $\mathbf{c}'' = [e|\bar{\mathbf{c}}]$ , and let  $\mathbf{d} = d(\mathbf{c})$ ,  $\mathbf{d}' = d(\mathbf{c}')$ , and  $\mathbf{d}'' = d(\mathbf{c}'')$ .

First consider the case when the outermost bits in  $\mathbf{c}$  are complements of each other. Then, in constructing  $\mathbf{c}'$  from  $\mathbf{c}$ , no runs are altered and the statements in Lemmas 4 and 5 which by assumption hold for  $\mathbf{d}$ , continue to hold for  $\mathbf{d}' = [d|\mathbf{d}]$ . In particular, if  $\mathbf{d}$  has length  $l_0$ ,  $\mathbf{d}'$  has length  $2l_0$ . The entry  $\mathbf{d}'(p)$ , for  $1 \leq p \leq l_0$  is the same as  $\mathbf{d}(l_0 - p + 1)$ , by assumption, which is the same as  $\mathbf{d}(l_0 - p + 1 + l_0) = \mathbf{d}(2l_0 - p + 1)$ . Thus, the mirror-symmetry property is preserved. The leftmost entry equal to 2 in  $\mathbf{d}'$ , if there is one, is in the same position as the leftmost entry equal to 2 in  $\mathbf{d}$  and Lemma 5.1 holds trivially. If  $\mathbf{d}'$  has only entries equal to 1 or 2, and has at least one entry of each kind, the outermost runs in  $\mathbf{c}'$  and therefore in  $\mathbf{c}$  must be 1-bit runs by Lemma 3. As an easy consequence, Lemma 5.2 continues to hold for  $\mathbf{c}'$ . By assumption, the leftmost 2 in  $\mathbf{c}$  is in

position  $2^p$  for some  $p$ , so that the leftmost run of 1's in  $\mathbf{d}$  is of length  $2^p - 1$ . The rightmost run of 1's in  $\mathbf{d}$  is also  $2^p - 1$  by the mirror symmetry assumption. At the point of concatenation of  $\mathbf{c}$  with itself, two sequences of 1-bit runs each of length  $2^p - 1$  are concatenated, and as a result, an inner run of 1's in  $\mathbf{d}'$  of length  $2(2^p - 1) = 2^{p+1} - 2$  is created. All other runs in  $\mathbf{d}'$  are of the same length as the runs in  $\mathbf{d}$ , and Lemma 5.3 follows.

We now focus on  $\mathbf{c}''$  and its  $\mathbf{d}''$ . All runs in  $\mathbf{d}''$  remain the same as in  $\mathbf{d}' = [d|\mathbf{d}]$ , except that the two innermost entries (which are the same by the mirror-symmetry property of  $\mathbf{d}$ ) are replaced by a single entry of their sum. For  $\mathbf{d}$  of length  $l_0$ ,  $\mathbf{d}''$  has length  $2l_0 - 1$ . The entry  $\mathbf{d}''(p)$  for  $1 < p \leq l_0 - 1$  is the same as  $\mathbf{d}''(l_0 - p + 1)$ , which is also the same as  $\mathbf{d}''(l_0 - p + 1 + l_0 - 1) = \mathbf{d}''((2l_0 - 1) - p + 1)$ . For  $p = 1$ , the entry in the first position in  $\mathbf{d}''$  is the same as both the first and the last entry in  $\mathbf{d}$ , which is itself equal to the last entry in  $\mathbf{d}''$ . Therefore, the mirror-symmetry property (Lemma 4) continues to hold for  $\mathbf{d}''$ .

If  $\mathbf{d}$  has at least one entry equal to 2, its leftmost 2 is in the same position as the leftmost 2 in  $\mathbf{d}'$ , and Lemma 5.1 remains to hold. If  $\mathbf{d}$  has all entries equal to 1, then the length of  $\mathbf{d}$  is  $2^{m_0}$  and  $\mathbf{d}''$  has a single 2 in the middle position, which is then a power of 2, and both Lemmas 5.1 and 5.2 hold.

By Lemma 3, if  $\mathbf{c}$  has both 1- and 2-bit runs, the outermost runs must be 1-bit runs. If the outermost 1-bit runs in  $\mathbf{c}$  are neighbored by another 1-bit runs, the innermost run of 2's in  $\mathbf{d}''$  is then of length 1. If the outermost 1-bit runs in  $\mathbf{c}'$  are neighbored by a sequence of consecutive 2-bit runs, which each by assumption and the symmetry property of  $\mathbf{c}$  must contain  $2^{q_0} - 1$  consecutive 2-bit runs, then the innermost run of 2's (at the point of concatenation in  $\mathbf{c}''$ ) in  $\mathbf{d}''$  is of length  $2(2^{q_0} - 1) + 1 = 2^{q_0+1} - 1$ . Since all other runs in  $\mathbf{c}''$  remain unaltered we can conclude that Lemma 5.2 holds as well. Finally, Lemma 5.3 continues to hold trivially since all inner runs of 1's in  $\mathbf{d}''$  already existed as inner runs of 1's in two copies of  $\mathbf{d}$ .

If the outermost bits in  $\mathbf{c}$  are the same, we can mimic the above proof by simply exchanging  $\mathbf{c}'$  and  $\mathbf{c}''$ . As discussed before, since each codeword in  $C(m_0 + 1)$  is either a concatenation of a codeword in  $C(m_0)$  with itself or with its complement, we can conclude that Lemmas 4 and 5 continue to hold for  $C(m_0 + 1)$ .  $\square$

Another useful observation is given in the following:

*Lemma 6:* If  $\mathbf{d}_a = d(\mathbf{c}_a)$  and  $\mathbf{d}_b = d(\mathbf{c}_b)$ , for  $\mathbf{c}_a, \mathbf{c}_b \in C(m)$  ( $\mathbf{d}_a, \mathbf{d}_b \in \mathcal{D}_m$ ) and  $m > 2$ , are such that they have  $2k + 1$  and  $2k$  entries, respectively, and all their entries are 1 or 2, then in the first leftmost position in which they differ, call it  $p$ , the entry is 1 in  $\mathbf{d}_a$  and is 2 in  $\mathbf{d}_b$ , and  $p < k$ .

*Proof:* Let  $s$  be the largest power of 2 that divides  $2k$ . By assumption  $s \geq 1$ . By Lemma 2, there exists a codeword in  $C(m - s)$ , call it  $\mathbf{c}_b^*$ , that has  $r_1 = 2k/2^s$  runs and has the same leftmost bit as  $\mathbf{c}_b$ . In particular, if  $2k$  is itself a power of 2,  $\mathbf{c}_b^*$  has a single run of length  $2^m/2k$ . By the existence of  $\mathbf{c}_a$  in  $C(m)$  with  $2k + 1$  runs,  $2k$  is strictly less than  $2^m$ , and thus,  $m - s \geq 1$ . Consider a codeword in  $C(m - s)$  that has  $r_1 + 1$  runs, and the same leftmost bit as  $\mathbf{c}_a$ , and call it  $\mathbf{c}_a^*$ . Since  $r_1$  is odd,  $r_1 + 1 \leq 2^{m-s}$  and  $\mathbf{c}_a^*$  exists by Lemma 2.

Let  $\mathbf{c}_e$  be a codeword in  $C(m - s - 1)$  that has  $(r_1 + 1)/2$  runs and the same leftmost bit as  $\mathbf{c}_a$  (since  $m - s \geq 1$ , the code

$C(m-s-1)$  and its codeword  $\mathbf{c}_e$  exist). If  $\mathbf{c}_e$  starts and ends in the same bit, which corresponds to odd  $(r_1+1)/2$ , we consider the codewords  $\mathbf{c}'_e = [\mathbf{c}_e | \overline{\mathbf{c}_e}]$  and  $\mathbf{c}''_e = [\mathbf{c}_e | \mathbf{c}_e]$  in  $C(m-s)$ , and associate  $\mathbf{d}'_e = d(\mathbf{c}'_e)$  and  $\mathbf{d}''_e = d(\mathbf{c}''_e)$  to them. Note that  $|\mathbf{d}'_e| = |\mathbf{d}''_e| + 1$ , where  $|\mathbf{d}'_e|$  indicates the length of string  $\mathbf{d}'_e$ . Moreover, the middle entry (in position  $(r_1+1)/2$ ) in  $\mathbf{d}''_e$  is the sum of two innermost entries in  $\mathbf{d}'_e$  (which span positions  $(r_1+1)/2$  and  $(r_1+1)/2+1$ , and are equal to each other by Lemma 4), and all other entries in these two strings are the same.

If  $\mathbf{c}_e$  starts and ends in complementary bits, which happens for even  $(r_1+1)/2$ , instead let  $\mathbf{c}'_e = [\mathbf{c}_e | \mathbf{c}_e]$  and  $\mathbf{c}''_e = [\mathbf{c}_e | \overline{\mathbf{c}_e}]$ , and associate  $\mathbf{d}'_e = d(\mathbf{c}'_e)$  and  $\mathbf{d}''_e = d(\mathbf{c}''_e)$  with them. Observe that  $|\mathbf{d}'_e| = |\mathbf{d}''_e| + 1$  as well as that  $\mathbf{d}''_e$  is the same as  $\mathbf{d}'_e$  except for the two innermost entries in  $\mathbf{d}'_e$ , which are replaced by their sum to yield the middle entry of  $\mathbf{d}''_e$ . By the uniqueness of a codeword in  $C(m-s)$  having  $|\mathbf{d}'_e|$  runs and starting with a particular bit (that being the leftmost bit of  $\mathbf{c}_a$ ), established in Lemma 2, we conclude that  $\mathbf{c}_a^* = \mathbf{c}'_e$ , and similarly  $\mathbf{c}_b^* = \mathbf{c}''_e$ .

Therefore, the first leftmost position in which  $\mathbf{d}_b^* = d(\mathbf{c}_b^*)$  (same as  $\mathbf{d}'_e$ ) and  $\mathbf{d}_a^* = d(\mathbf{c}_a^*)$  (same as  $\mathbf{d}''_e$ ) differ is their  $(r_1+1)/2$ th position, such that the entry in that position in  $\mathbf{d}_b^*$  is twice its counterpart in  $\mathbf{d}_a^*$ . By assumption on the entries of  $\mathbf{d}_a$  and  $\mathbf{d}_b$  being at most 2, it further follows that the entry is 1 in  $\mathbf{d}_a^*$  and 2 in  $\mathbf{d}_b^*$ .

By constructing a sequence of codewords  $\{\mathbf{c}_{b,i}\}$ , for  $1 \leq i \leq s+1$ , starting from  $\mathbf{c}_{b,1} = \mathbf{c}_b^*$ , and where  $\mathbf{c}_{b,i} \in C(m-s-1+i)$  is the result of concatenation of  $\mathbf{c}_{b,i-1}$  either with itself or with its complement (former if the outermost bits in  $\mathbf{c}_{b,i-1}$  are different and latter if they are the same), we arrive at  $\mathbf{c}_b$ . In particular, the associated  $\mathbf{d}_{b,i} = d(\mathbf{c}_{b,i})$  have length  $2^{i-1}r_1$ , and for the last term in the sequence  $\mathbf{d}_{b,s+1}$  is of length  $2^s r_1 = 2k$ , which is precisely the length of  $d(\mathbf{c}_b)$ .

Similarly, we construct a sequence of codewords  $\{\mathbf{c}_{a,i}\}$ , for  $1 \leq i \leq s+1$ , starting from  $\mathbf{c}_{a,1} = \mathbf{c}_a^*$ . Now  $\mathbf{c}_{a,i} \in C(m-s-1+i)$  is the result of concatenation of  $\mathbf{c}_{a,i-1}$  with itself if the outermost bits in  $\mathbf{c}_{a,i-1}$  are the same, otherwise it is the result of concatenation of  $\mathbf{c}_{a,i-1}$  with its complement. The associated  $\mathbf{d}_{a,i} = d(\mathbf{c}_{a,i})$  have length  $2^{i-1}r_1 + 1$ , so that the last term in the sequence has  $2^s r_1 + 1 = 2k + 1$  runs, which is precisely the length of  $\mathbf{d}_a = d(\mathbf{c}_a)$ . Thus, in starting from  $\mathbf{c}_a^*$ , by a series of concatenations in which the runs at the point of concatenation are always merged, we arrive at  $\mathbf{c}_a$ . Since the first leftmost entry in which  $\mathbf{d}_b^*$  and  $\mathbf{d}_a^*$  differ are in their  $(r_1+1)/2$ th leftmost positions, the first position in which  $\mathbf{d}_b$  and  $\mathbf{d}_a$  differ are still in their  $(r_1+1)/2$ th leftmost positions. Since  $s$  is at least 1,  $(r_1+1)/2 \leq (k+1)/2 < k$ , for  $k > 1$ . If  $k = 1$ ,  $\mathbf{d}_a$  is “ $2^{m-1}2^{m-1}$ ” and  $\mathbf{d}_b$  is “ $2^{m-2}2^{m-1}2^{m-2}$ .” For  $m > 2$ ,  $2^{m-2} > 1$ , which exceeds the requirement on the entries of  $\mathbf{d}_b$  being at most 2.  $\square$

### III. THE IDENTIFICATION PROBLEM FOR RM(1, m) CODES

#### A. Model

We recall the discussion of synchronization errors from Section I. We adopt the following model in the infinite SNR limit. Suppose  $C$  is an  $(n, k)$  linear block code. A codeword  $\mathbf{c} \in C$  is modulated using pulse-amplitude modulation (PAM), and the received waveform  $r(t)$  is sampled noise free. Let  $\mathbf{r}$  be the sampled version of  $r(t)$  of length  $l$  bits. We assume that the location

of the first and the last bit of  $\mathbf{r}$  in the received string of data is known, so that the codewords can be analyzed in isolation. Then, from  $l$  we would know the difference between the number of repetitions and the number of deletions that occurred over the channel. For instance, if the channel model permits one repetition, then if  $l = n$  we know that the sampled version of  $r(t)$  equals  $\mathbf{c}$ , while if  $l = n+1$  the sampled version of  $r(t)$  is  $\mathbf{c}$  but with one bit repeated. Similarly, if the channel model permits one deletion, then if  $l = n$  we know that the sampled version of  $r(t)$  equals  $\mathbf{c}$ , while if  $l = n-1$  the sampled version of  $r(t)$  is  $\mathbf{c}$  with one bit deleted. These are the two channel models that we consider in this paper. Note that in these examples the location of the repeated (respectively, deleted) bit is not known.

In general, in the infinite SNR limit, a channel with synchronization errors could be modeled as introducing a certain number of repetitions and deletions in the transmitted codeword. Assuming, as above, that the location of the first and the last bit in the received string of data is known, codewords could be analyzed in isolation, and we would learn the difference  $l - n$  between the number of repetitions and the number of deletions that occurred over the channel. However, we would not know the location of the repetitions and/or the deletions. This more general kind of model is not analyzed here.

This paper is concerned with use of RM(1, m) codes over channels permitting substitution and synchronization errors under the two kinds of synchronization error models discussed in the first paragraph: the single repetition model and the single deletion model. In this section, we analyze the identification problem for codewords of the RM(1, m) codes over channels permitting a single deletion. Before doing so, we first deal with the much simpler case of channels permitting only (an arbitrary number of) repetition errors.

#### B. The Case of Repetition Errors

We have the following simple result:

*Theorem 1:* In  $C(m)$ , no two codewords can result in the same string when they experience repetitions.

*Proof:* For the case of one, or any number of repetitions, two codewords in  $C(m)$  resulting in the same string must have the same number of runs, and the same sequence of runs. By Lemma 2, there are exactly two codewords with the same number of runs. However, these two codewords are also complements of each other and therefore cannot have the same sequence of runs. We can conclude that  $C(m)$  is immune to repetition errors.  $\square$

It should be noted, nevertheless, that even single repetitions can result in pairs of codewords of the RM(1, m) code having poor identification. For instance, the codeword  $c_{2^{m-1}}(01)$  and its complement  $c_{2^{m-1}}(10)$  have a post-repetition Hamming distance of 2.

#### C. The Case of a Single Deletion

The analysis of the identification problem for RM(1, m) codes over channels permitting a single deletion is considerably more interesting, see Theorem 2. Before proceeding to the main theorem, we first make a couple of simple remarks.

*Remark 3.1 [Complementarity]:* Consider two distinct codewords  $\mathbf{c}_a$  and  $\mathbf{c}_b$  in  $C(m)$ . If  $\mathbf{c}_a$  and  $\mathbf{c}_b$  can give rise to the same string after experiencing one deletion each, the same is true for their bitwise complements  $\overline{\mathbf{c}_a}$  and  $\overline{\mathbf{c}_b}$ .

*Remark 3.2 [Reversibility]:* Consider two distinct codewords  $\mathbf{c}_a$  and  $\mathbf{c}_b$  in  $C(m)$ , If  $\mathbf{c}_a$  and  $\mathbf{c}_b$  can give rise to the same string after experiencing one deletion each the same is true for their reversals  $\overleftarrow{\mathbf{c}_a}$  and  $\overleftarrow{\mathbf{c}_b}$ .

Here is a description of the pairs of codewords in  $\text{RM}(1, m)$  which suffer from the identification problem over channels with a single deletion, for small values of  $m$ :

*Remark 3.3:* For  $m = 0, 1, 2$  we can show by inspection the following.

$m = 0$ : The only codewords are “0” and “1” and they can both result in an empty string.

$m = 1$ : The codewords are “00,” “11,” “01,” and “10.” The codewords “00,” “01,” and “10” can all result in “0,” and the codewords “11,” “10,” and “01” can all result in “1.”

$m = 2$ : The codewords are “0000,” “1100,” “0011,” “0110,” “1111,” “1010,” “0101,” and “1001.” The codeword “0011” and any one of “0110,” “0101,” and “1001” can result in the same string. Similarly, the codeword “1100” and any one of “1001,” “1010,” and “0110” can result in the same string. The same is true for “0110,” and any one of “1010” and “0101” as well as for “1001” and any one of “0101” and “1010.” Also, “1010” and “0101” can result in the same string.  $\square$

We may now complete the analysis of the identification problem for  $\text{RM}(1, m)$  codes over channels permitting a single deletion:

*Theorem 2:* Let  $j = 2^{m-1}$  and  $k = 2^{m-2}$ . For  $m \geq 3$ , there is a total of 11 pairs of distinct codewords in  $C(m)$  that result in the same string when each experiences a deletion. These are

- |                                     |   |         |
|-------------------------------------|---|---------|
| 1. $c_j^m(10)$ and $c_j^m(01)$      | } | Group 1 |
| 2. $c_j^m(10)$ and $c_j^m(11)$      | } | Group 2 |
| 3. $c_j^m(10)$ and $c_{j-1}^m(00)$  |   |         |
| 4. $c_j^m(01)$ and $c_j^m(11)$      |   |         |
| 5. $c_j^m(01)$ and $c_{j-1}^m(00)$  |   |         |
| 6. $c_k^m(01)$ and $c_k^m(00)$      | } | Group 3 |
| 7. $c_k^m(01)$ and $c_{k+1}^m(11)$  |   |         |
| 8. $c_k^m(10)$ and $c_k^m(00)$      |   |         |
| 9. $c_k^m(10)$ and $c_{k+1}^m(11)$  |   |         |
| 10. $c_j^m(01)$ and $c_{j-1}^m(01)$ | } | Group 4 |
| 11. $c_j^m(10)$ and $c_{j-1}^m(10)$ |   |         |

*Proof:* Observe that we have already established this result for  $m = 2$  in the previous remark. In the rest of the proof, we will assume that  $m \geq 3$ .

Note that it is sufficient to assume that the deletion occurs at the end of a run, since the string resulting from a deletion of a bit in some codeword is the same irrespective of where the deleted bit was located within the run it belonged to.

Suppose  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are distinct codewords in  $C(m)$  which result in the same string when each experiences one deletion. Let  $\mathbf{d}_a = d(\mathbf{c}_a)$  and  $\mathbf{d}_b = d(\mathbf{c}_b)$  be as defined in Definition

1. We first observe that during a deletion, the total number of runs in the codeword stays the same, decreases by one, or by two. Suppose a codeword  $\mathbf{c}_a$  experiences a deletion in a run of length at least 2. Then the length of  $\mathbf{d}_a$  remains unchanged. If  $\mathbf{c}_a$  experiences a deletion in a run of length 1, the neighboring runs (if any) will merge and the total number of runs will decrease. In particular, if this deleted run of length 1 is an outermost run, the length of  $\mathbf{d}_a$  decreases by 1. If this deleted run of length 1 is located somewhere else in  $\mathbf{c}_a$ , the length of  $\mathbf{d}_a$  decreases by 2. It is therefore sufficient to consider the cases when the lengths of  $\mathbf{d}_a$  and  $\mathbf{d}_b$  differ by 0, 1, and 2. Without loss of generality, assume that  $|\mathbf{d}_a| \geq |\mathbf{d}_b|$ . We treat the cases  $|\mathbf{d}_a| = |\mathbf{d}_b|$ ,  $|\mathbf{d}_a| = |\mathbf{d}_b| + 1$ , and  $|\mathbf{d}_a| = |\mathbf{d}_b| + 2$  separately.

*Case 1:  $|\mathbf{d}_a| = |\mathbf{d}_b|$ .*

By Lemma 2, it must be that  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are complements of each other, and consequently,  $\mathbf{d}_a = \mathbf{d}_b$ . Either both  $\mathbf{c}_a$  and  $\mathbf{c}_b$  experience deletions in runs of length at least 2 each, or both experience deletions in different outermost runs of length 1 each or in inner runs of length 1 each.

Since  $\mathbf{c}_a$  and  $\mathbf{c}_b$  differ in their leftmost bits, a deletion must occur in the leftmost bits in either  $\mathbf{c}_a$  or  $\mathbf{c}_b$ . Without loss of generality, we can assume that the leftmost bit in  $\mathbf{c}_a$  is deleted. If this bit belonged to a run of length at least 2,  $\mathbf{c}_b$  itself would start with a run of length at least 2, but then it would be impossible to obtain the same string from  $\mathbf{c}_a$  and  $\mathbf{c}_b$  when each experiences exactly one deletion. Therefore, the leftmost run in  $\mathbf{c}_a$  is a run of length 1, and by Lemma 3, all runs in  $\mathbf{c}_a$  (and  $\mathbf{c}_b$ ) must be of length 1 or 2. Since  $\mathbf{d}_a$  decreases by 1, the same must be true for  $\mathbf{d}_b$ , so that  $\mathbf{c}_b$  experiences a deletion in its outermost bit, which then must be its rightmost bit. Then  $\mathbf{c}_a(p) = \mathbf{c}_b(p-1)$  for  $1 < p \leq 2^m$  (here and in the remainder  $\mathbf{c}_a(p)$  denotes the bit in the  $p$ th leftmost position of  $\mathbf{c}_a$ ), and by using the fact that  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are complements of each other, it follows that  $\mathbf{c}_a$  and  $\mathbf{c}_b$  consist of alternating bits. Thus,  $\mathbf{c}_a$  is either  $c_j^m(10)$  or  $c_j^m(01)$  for  $j = 2^{m-1}$ , and  $\mathbf{c}_b$  is its complement. This codeword pair is listed under 1 and is labeled Group 1.

*Case 2:  $|\mathbf{d}_a| = |\mathbf{d}_b| + 1$ .*

Suppose a deletion occurs in position  $p_a$  in  $\mathbf{c}_a$ , and in position  $p_b$  in  $\mathbf{c}_b$  (we assume that the deletion occurs at the end of a run), where we index the bits in the codewords with 1 through  $2^m$ , from left to right. It must be that either: a)  $\mathbf{c}_a$  experiences a deletion in an outermost run of length 1, while  $\mathbf{c}_b$  experiences a deletion in a run of length at least 2, or b)  $\mathbf{c}_a$  experiences a deletion in an inner run of length 1 and  $\mathbf{c}_b$  experiences a deletion in an outermost run of length 1.

*Subcase 2-1:  $|\mathbf{d}_a|$  is even.* We view  $\mathbf{c}_a$  as the result of concatenation applied to the same codeword  $\mathbf{c}' \in C(m-1)$ , whereby  $\mathbf{c}_a = [\mathbf{c}' | \mathbf{c}']$  if  $\mathbf{c}'$  has opposite outermost bits, and  $\mathbf{c}_a = [\mathbf{c}' | \overline{\mathbf{c}'}]$  if the outermost bits in  $\mathbf{c}'$  are the same.

In either case a) or b) there exists at least one entry in  $\mathbf{d}_a$  equal to 1. Then, by Lemma 3, the outermost runs in  $\mathbf{c}_a$  and  $\mathbf{c}'$  are all of length 1. By mirror-symmetry (Lemma 4) we can express  $\mathbf{d}_a$  and  $\mathbf{d}_b$  as  $\mathbf{d}_a = [A11A^R]$  and  $\mathbf{d}_b = [A2A^R]$ , where  $A = [A_1A_2 \dots A_l]$  is a substring of  $\mathbf{d}_a$ ,  $A^R$  is its reverse, and  $A_1 = 1$ .

For the situation described in a), by the reversibility property, we may as well assume that the leftmost bit in  $\mathbf{c}_a$  is deleted.

Then the entry in position  $p$  in  $\mathbf{d}_b$  must correspond to the entry in position  $p+1$  in  $\mathbf{d}_a$ , in the sense that  $\mathbf{d}_a(p+1) = \mathbf{d}_b(p) \forall p$  except for exactly one, call it  $p^*$ , for which  $\mathbf{d}_a(p^* + 1) = \mathbf{d}_b(p^*) + 1$ . In particular, if this entry in  $\mathbf{d}_b$  is bigger than 2, by Lemma 3, it would have to be at least 4, further implying the existence of a run in  $\mathbf{c}_a$  of length at least 3, which is impossible by Lemma 3 and the fact that there is at least one run of length 1 in  $\mathbf{c}_a$ .

Therefore,  $\mathbf{d}_b(p^*) = 2$  and  $\mathbf{d}_a(p^* + 1) = 1$ . Since  $\mathbf{d}_b(l + 1) = 2$  and  $\mathbf{d}_a(l + 2) = 1$  by construction, it follows that  $p^* = l + 1$ . Furthermore,  $A_2 = A_1, A_3 = A_2, \dots, A_l = A_{l-1}$ , so that  $\mathbf{d}_a$  consists of all 1's and  $\mathbf{d}_b$  has all 1's except for its innermost entry which is 2. Consequently,  $\mathbf{c}_a$  is either  $c_j^m(10)$  or  $c_j^m(01)$ , and  $\mathbf{c}_b$  is either  $c_j^m(11)$  or  $c_{j-1}^m(00)$  for  $j = 2^{m-1}$ . One can check that all four pairs of candidate codewords suffer from the identification problem. This is the set of pairs listed under Group 2. This group of codeword pairs is closed under complementation and reversal.

Now, for the situation described in b), by the reversibility property, we may as well assume that the rightmost bit in  $\mathbf{c}_b$  is deleted.

The first leftmost entries where  $\mathbf{d}_a$  and  $\mathbf{d}_b$  differ are their  $(l + 1)$ th entries so the deletion in  $\mathbf{c}_a$  must be in its  $(l + 2)$ th run, which then disappears altogether. Moreover, both  $(l + 1)$ th and  $(l + 3)$ th runs in  $\mathbf{c}_a$  must be of length 1 each because the  $(l + 1)$ th run of  $\mathbf{c}_b$  is of length 2. Therefore  $A^R(1) = A_l = 1$ .

The entry in position  $l + 2$  in  $\mathbf{d}_b$  (which is  $A^R(1)$ ) must be the same as the entry in position  $l + 4$  in  $\mathbf{d}_a$ , which is  $A^R(2) = A_{l-1}$ . The entry in position  $l + 3$  in  $\mathbf{d}_b$ , which is itself  $A^R(2)$ , is the same as the entry in  $\mathbf{d}_a$  in position  $l + 5$ , which is  $A^R(3)$ .

By continuing forward until the end of  $A^R$ , we conclude that  $A^R$  consists of all 1's, thereby making  $\mathbf{d}_a$  be all 1's as well, and  $\mathbf{d}_b$  be all 1's except for 2 in the middle. These two  $\mathbf{d}_a$  and  $\mathbf{d}_b$  have already been encountered in the situation described in a), and yield the codeword pairs listed under Group 2.

*Subcase 2-2:  $|\mathbf{d}_a|$  is odd.* In either case a) or b)  $\mathbf{d}_a$  has at least one entry equal to 1, so all its entries are either 1 or 2 by Lemma 3. If  $\mathbf{d}_b$  had an entry larger than 3, by Lemma 3 case b) would not be even possible. For case a) it would require an existence of a run in  $\mathbf{c}_a$  of length at least 3, which is also impossible by the same lemma. Since all entries in  $\mathbf{d}_a$  and  $\mathbf{d}_b$  are then precisely 1 or 2, we can use their mirror symmetry and apply Lemma 6 to conclude that  $\mathbf{d}_a$  and  $\mathbf{d}_b$  have the following formats:  $\mathbf{d}_a = [A1B1A^R]$  and  $\mathbf{d}_b = [A2C2A^R]$ , where  $|B| = |C| + 1$  and  $A$  and  $C$  are possibly empty.

Let  $|A| = p - 1$ . Further, note that  $|C|$  is even.

For the situation described in a) we may as well assume, by the reversibility property, that the rightmost bit in  $\mathbf{c}_a$  is deleted, and that it belonged to a 1-bit run. Then the deletion in  $\mathbf{c}_b$  must be in its  $p$ th leftmost run (of length 2).

Since  $A^R(p-1) = 1$  in  $\mathbf{d}_a$ , by mirror symmetry,  $A(1) = 1$  (or by Lemma 3). Since the rightmost entry in  $\mathbf{d}_b$  is the same as the second rightmost entry in  $\mathbf{d}_a$ , it further follows that  $A^R(p-2) = 1$ , which in turn implies that  $A(2) = 1$ , and so on, until the end of  $A$ , thereby requiring that  $A$  consists of all 1's. Similarly, the entry in  $\mathbf{d}_b$  in position  $|\mathbf{d}_b| - (p - 1)$ , which is 2 by assumption, is the same as the entry in  $\mathbf{d}_a$  in position  $|\mathbf{d}_a| - p$ , which is itself the last entry in  $B$ . Thus,  $B$  ends in 2 and by mirror symmetry it also starts with 2. This in turn implies that  $C$  starts and

ends with 2, which then implies that the next to the last entry in  $B$  is also 2. By continuing on until all entries in  $B$  and  $C$  have been encountered we can conclude that  $B$  and  $C$  consist only of 2's. Then  $\mathbf{d}_a = "1.12.21.1"$  and  $\mathbf{d}_b = "1.12.21.1"$  (if  $A$  nonempty) or  $\mathbf{d}_b = "2.2"$  (if  $A$  empty), where "1.1" ("2.2") indicates a nonempty run of 1's (2's). For  $|\mathbf{d}_b|$  even, the run of 2's in  $\mathbf{d}_b$  would have to have even length (since the neighboring "1.1" runs are of the same length by the mirror-symmetry property) which is impossible by Lemma 5.2. Thus,  $\mathbf{d}_b = "2.2,"$   $A$  is empty, and then  $\mathbf{d}_a = "12.21."$  Consequently,  $\mathbf{c}_b$  itself is either  $c_k^m(01)$  or  $c_k^m(10)$  for  $k = 2^{m-2}$ , and  $\mathbf{c}_a$  is either  $c_k^m(00)$  or  $c_{k+1}^m(11)$ . It can be checked that all four codeword pairs suffer from the identification problem. These are the pairs listed in Group 3. This group of codeword pairs is also closed under complementation and reversal.

For b) we may as well assume, by the reversibility property, that the rightmost bit in  $\mathbf{c}_b$  is deleted, so that  $\mathbf{d}_b$  ends in a 1. Note that this implies that  $A^R$  (and  $A$ ) cannot be empty, and therefore  $p > 1$ . Then the first leftmost entry in which  $\mathbf{d}_a$  and  $\mathbf{d}_b$  differ is compensated for by the deletion in a 1-bit run in  $\mathbf{c}_a$ . Since all runs in  $\mathbf{c}_b$  are of length at most 2, the deleted run in  $\mathbf{c}_a$  must be bordered by two 1-bit runs. Therefore, the  $(p + 1)$ th run (of length 1) in  $\mathbf{c}_a$  is deleted, and both  $(p)$ th and  $(p + 2)$ th run in  $\mathbf{c}_a$  are also of length 1. Furthermore, the entry in position  $t$  for  $p + 1 \leq t \leq |\mathbf{d}_b| - 1$  in  $\mathbf{d}_b$  is the same as the entry in position  $t + 2$  in  $\mathbf{d}_a$ .

In particular, the entry in  $\mathbf{d}_b$  in position  $|\mathbf{d}_b| - p + 1$ , which is 2, is the same as the entry in position  $|\mathbf{d}_a| - p + 2$  in  $\mathbf{d}_a$ , which is  $A^R(1)$ . By mirror symmetry entries in positions  $p - 1$  in both  $\mathbf{d}_a$  and  $\mathbf{d}_b$  are equal to 2. Then the entry in  $\mathbf{d}_b$  in position  $|\mathbf{d}_b| - p + 2$  is also 2, as is the entry in  $\mathbf{d}_a$  in position  $|\mathbf{d}_a| - p + 3$ . By continuing onwards until  $t = |\mathbf{d}_b| - 1$ , and by using the mirror symmetry, we conclude that  $A$  (and  $A^R$ ) consists of all 2's, which is in contradiction with the earlier requirement that the deletion in  $\mathbf{c}_b$  occurs in its outermost run of length 1.

*Case 3:  $|\mathbf{d}_a| = |\mathbf{d}_b| + 2$ .* We now consider the remaining case where the deletion in  $\mathbf{c}_a$  occurs in an inner run of length 1 and in  $\mathbf{c}_b$  in a run of length at least 2. This deletion in a 1-bit run of  $\mathbf{c}_a$  causes its neighboring runs to merge. By Lemma 3, these runs are of length 1 or 2 each. If they were both of length 2 each, there would exist an inner run of 1's in  $\mathbf{d}_a$  of length 1, which is impossible by Lemma 5.3. If one neighboring run was of length 1 and the other of length 2, the merging would require an existence of a 3-bit run in the postdeletion  $\mathbf{c}_b$ . By Lemma 3, the deletion in  $\mathbf{c}_b$  would then have to be in a 4-bit run, and by the same lemma, the outermost runs in  $\mathbf{c}_b$  would be of length at least 2. These would have to correspond to the outermost runs in  $\mathbf{c}_a$ , which are themselves of length 1 each. Therefore, the deletion in  $\mathbf{c}_a$  must occur in an inner 1-bit run neighbored by two 1-bit runs, and all entries in both  $\mathbf{d}_a$  and  $\mathbf{d}_b$  can be only 1 or 2.

Consider  $\mathbf{c}_c \in C(m)$  which has  $|\mathbf{d}_b| + 1$  runs. For  $|\mathbf{d}_a|$  even, we can think of  $\mathbf{c}_a$  as being the result of concatenating a codeword  $\mathbf{c}_d \in C(m - 1)$  with itself if  $|\mathbf{d}_a|/2$  is even, and with its complement if  $|\mathbf{d}_a|/2$  is odd, such that  $\mathbf{c}_d$  and  $\mathbf{c}_a$  have the same leftmost bits (the existence of such codeword in  $C(m - 1)$  follows from Lemma 2). Furthermore, in the former case, we can view  $\mathbf{c}_c$  as the result of concatenating  $\mathbf{c}_d$  with its complement, and in the latter case as the result of concatenating  $\mathbf{c}_d$  with it-

self. Then  $\mathbf{d}_a = [\mathbf{d}_a | \mathbf{d}_a]$ , and  $\mathbf{d}_c = [\mathbf{d}_a(1, l-1) | (\mathbf{d}_a(l) + \mathbf{d}_a(1)) | \mathbf{d}_a(2, l)]$ , where  $\mathbf{d}_a = d(\mathbf{c}_a)$  and  $l = |\mathbf{d}_a|$ . The leftmost entry in which  $\mathbf{d}_a$  and  $\mathbf{d}_c$  differ is their  $(|\mathbf{d}_c| + 1)/2$ th leftmost entry. By mirror symmetry of  $\mathbf{d}_a$ , this entry in  $\mathbf{d}_c$  is twice its counterpart in  $\mathbf{d}_a$ . Since all entries in  $\mathbf{d}_a$  are 1 or 2, and its outermost entries are 1, it follows that all entries in  $\mathbf{d}_c$  are also at most 2. Then the first leftmost entry in which  $\mathbf{d}_c$  and  $\mathbf{d}_b$  differ is say in position  $p$ , for  $p < |\mathbf{d}_b|/2$  and  $\mathbf{d}_c(p) = 1$  and  $\mathbf{d}_b(p) = 2$ , by Lemma 6. Since  $|\mathbf{d}_b| < |\mathbf{d}_c| + 1$ , the first leftmost entries in which  $\mathbf{d}_a$  and  $\mathbf{d}_b$  differ is in the  $p$ th position, where  $p < |\mathbf{d}_b|/2$ .

A similar argument holds for  $|\mathbf{d}_a|$  odd when the first leftmost entry in which  $\mathbf{d}_a$  and  $\mathbf{d}_c$  differ is then in some position  $p$ , for  $p < |\mathbf{d}_c|/2$ , and the first leftmost entry in which  $\mathbf{d}_c$  and  $\mathbf{d}_b$  differ is in their  $(|\mathbf{d}_b| + 1)/2$ th entry. Then the first leftmost entry in which  $\mathbf{d}_a$  and  $\mathbf{d}_b$  differ is still in position  $p$ .

As a result and by mirror symmetry, we can then express  $\mathbf{d}_a$  and  $\mathbf{d}_b$  as  $\mathbf{d}_a = [A1B1A^R]$  and  $\mathbf{d}_b = [A2C2A^R]$ , where  $|B| = |C| + 2$ ,  $|A| = p - 1$ , and  $A$  and  $C$  are possibly empty.

By the reversibility property, we can assume that the leftmost error is a deletion in  $\mathbf{c}_a$ , which then must be in the  $(p + 1)$ th run in  $\mathbf{c}_a$  (of length 1), neighbored by 1-bit runs on each side, such that the substring “111” starts at position  $p$  in  $\mathbf{d}_a$  and the substring “2” in  $\mathbf{d}_b$  is at position  $p$ .

From  $t = p + 1$  onwards, the entry in position  $t$  in  $\mathbf{d}_b$  must be the same as the entry in position  $t + 2$  in  $\mathbf{d}_a$ , except for one pair of entries. In this exception, the entry is 2 in  $\mathbf{d}_b$  and 1 in  $\mathbf{d}_a$ . By mirror symmetry, the entry in  $\mathbf{d}_b$  in position  $|\mathbf{d}_b| - p + 1$  is 2 and the entry in  $\mathbf{d}_a$  in position  $|\mathbf{d}_a| - p + 1 = |\mathbf{d}_b| - p + 1 + 2$  is 1.

We now re-express  $\mathbf{d}_a$  as  $[A111D1A^R]$  and  $\mathbf{d}_b$  as  $[A2D2A^R]$ , such that  $B = 11D$ . In particular,  $D$  is nonempty as otherwise  $\mathbf{d}_b$  would have a run of 2's of even length which by Lemma 5.2 would imply that  $\mathbf{d}_b$  consists of all 2's. As a consequence,  $\mathbf{d}_a$  would have an inner run of 1's of length 4, which is impossible by Lemma 5.3.

We suppose that  $|D| = l, l > 0$ . By mirror symmetry of  $\mathbf{d}_a$ ,  $D(l) = D(l - 1) = 1$ , and then by mirror symmetry of  $\mathbf{d}_b$ ,  $D(1) = D(2) = 1$  as well. By mirror symmetry of  $\mathbf{d}_a$ ,  $D(l - 2) = D(l - 3) = 1$ . By continuing on with matching up the appropriate entries in  $\mathbf{d}_a$  and  $\mathbf{d}_b$ , and by utilizing mirror symmetry we conclude that  $D$  consists of all 1's. Then,  $\mathbf{d}_a = [A1.1A^R]$  and  $\mathbf{d}_b = [A21.12A^R]$ , and by Lemma 5.3,  $|\mathbf{d}_b|$  is even, as is then  $|\mathbf{d}_a|$ .

Consider  $\mathbf{d}'_b = \mathbf{d}_b(1, |\mathbf{d}_b|/2)$ , and  $\mathbf{d}'_a = \mathbf{d}_a(1, |\mathbf{d}_a|/2)$ . Since  $|\mathbf{d}_a|$  and  $|\mathbf{d}_b|$  are even, there exist codewords  $\mathbf{c}'_a, \mathbf{c}'_b \in C(m - 1)$  for which  $\mathbf{d}'_a = d(\mathbf{c}'_a)$  and  $\mathbf{d}'_b = d(\mathbf{c}'_b)$ . Then  $\mathbf{d}'_a = [A1.1]$  and  $\mathbf{d}'_b = [A21.1]$ . If 2 following  $A$  in  $\mathbf{d}'_b$  is not in its innermost position, then it would have a mirror image in  $A$  in  $\mathbf{d}'_b$  (it cannot have a mirror image in the run of 1's) but such 2 in  $A$  in  $\mathbf{d}'_a$  would not have 2 as its mirror image. Thus,  $|A| = |\mathbf{d}'_b|/2 - 1$  and  $A$  has all 1's. Then  $\mathbf{d}_a$  itself has all 1's, and  $\mathbf{d}_b$  is “1.121.121.1,” so that  $\mathbf{c}_a$  is  $c_j^m(10)$  or  $c_j^m(01)$ , and  $\mathbf{c}_b$  is  $c_{j-1}^m(10)$  or  $c_{j-1}^m(01)$ , for  $j = 2^{m-1}$ . By the current assumption on the deletion locations, it follows that  $\mathbf{c}_a$  and  $\mathbf{c}_b$  must have the same leftmost bit. The resulting two pairs of codewords are listed in Group 4. By reversibility and complementarity these are the only such pairs.  $\square$

*Remark 3.4:* It is well known that a code capable of correcting a deletion is also capable of correcting an insertion [14]. More-

over, the codeword pairs that cause the identification problem under a single insertion are the same as the codeword pairs that cause the identification problem under a single deletion, and thus Theorem 2 also gives the identification error causing codeword pairs under a single insertion.  $\square$

Having identified all pairs of codewords in  $\text{RM}(1, m)$  that have an identification problem, our next goal is to construct a linear subcode that has good identification under single deletion errors. It turns out this can be done with the loss of only one information bit, and furthermore, this subcode also has good identification for single repetition errors.

#### D. Pruning of the Code

Let us first recall that the  $i$ th row of  $\mathbf{G}_m$ , for  $1 < i \leq m + 1$  consists of  $2^{i-1}$  alternating runs of ones and zeros, and that each run is of length  $2^{m-i+1}$  (see Section II). Observe that the  $i$ th row is then precisely  $c_{2^{i-2}}^m(10)$ . In particular, the last two rows of  $\mathbf{G}_m$  are  $c_{2^{m-2}}^m(10)$  for  $i = m$  and  $c_{2^{m-1}}^m(10)$  for  $i = m + 1$ .

We write  $\mathbf{c} \in C(m)$  as  $\mathbf{xG}_m$ , where  $\mathbf{x}$  is a  $(m + 1)$ -dimensional message vector so that  $c_{2^{m-1}}^m(10) = [0, 0, \dots, 0, 1]\mathbf{G}_m$  and  $c_{2^{m-1}}^m(01) = [1, 0, \dots, 0, 1]\mathbf{G}_m$ . Similarly,  $c_{2^{m-2}}^m(10)$  is  $[0, 0, \dots, 0, 1, 0]\mathbf{G}_m$  and  $c_{2^{m-2}}^m(01)$  is  $[1, 0, \dots, 0, 1, 0]\mathbf{G}_m$ .

Observe that  $c_{2^{m-1}}^m(10)$  appears in pairs 1 through 3, and the pair 11 in Theorem 2. Its complement, the codeword  $c_{2^{m-1}}^m(01)$  appears in pair 1, 4, 5, and 10. For both these codewords, there is a nonzero component in the last, i.e.,  $(m + 1)$ th position in the corresponding message vectors. Note that  $c_{2^{m-2}}^m(10)$  appears in pairs 8 and 9 and that its complement  $c_{2^{m-2}}^m(01)$  appears in pairs 6 and 7. Furthermore, the sum of the last two entries in the message vectors corresponding to these two codewords is 1.

We may now try to find as large as possible a linear subcode of  $C(m)$ , in which no two codewords cause the identification problem under one deletion. The generator matrix  $\hat{\mathbf{G}}$  of this subcode can have at most  $m$  rows. Consider a matrix consisting of the top  $m - 1$  rows of  $\mathbf{G}_m$ , followed by a binary sum of the last two rows of  $\mathbf{G}_m$ . Now,  $\hat{\mathbf{G}}$  has  $m$  rows and no linear combinations of its rows give rise to codewords causing the identification problem.

Therefore, if instead of using  $C(m)$  of rate  $\frac{m+1}{2^m}$  we use its linear subcode  $\hat{C}(m)$  of rate  $\frac{m}{2^m}$ , generated by the top  $m - 1$  rows of  $\mathbf{G}_m$  and the binary sum of the last two rows of  $\mathbf{G}_m$ , we are able to eliminate the identification problem under a single deletion while preserving the linearity of the code and suffering a very small loss in the overall rate.

*Remark 3.5:* Since a code is immune to a single insertion if and only if it is immune to a single deletion [14], it immediately follows that in the subcode  $\hat{C}(m)$  no two codewords cause the identification problem under a single insertion.  $\square$

In the next section, we will see that the subcode we have constructed is not just immune to single deletions; it also has good identification under the single deletion model and under the single repetition model.

In principle, one can utilize the run-length structure of the  $\text{RM}(1, m)$  code to determine large subcodes immune to any number of deletions, or even to combinations of repetitions and deletions. Such analysis quickly becomes very compli-

cated. The first author has carried out a detailed analysis of the identification problem for the RM(1,  $m$ ) codes under the infinite SNR channel model which permits *both* one repetition and one deletion [9]. Some additional structural properties of the codewords in RM(1,  $m$ ) codes that may be of independent interest are also contained in [9].

#### IV. DECODING THE MODIFIED RM(1, $m$ ) CODE OVER A CHANNEL WITH SYNCHRONIZATION AND SUBSTITUTION ERRORS

In the previous section, we described how to extract a linear subcode of the RM(1,  $m$ ) code that is immune to a single deletion. We now consider the behavior of such a subcode over channels in which, in addition to substitution errors, synchronization errors can occur as well. We consider two kinds of channel models for synchronization errors: channels where the deletion of a single bit can occur, and channels where the repetition of a single bit can occur. As in Subsection III-A, we assume in each case that the receiver learns from the sampled output whether a deletion (respectively, a repetition) has occurred or not.

In this section, we first determine the minimum distance between the sets of strings obtained by applying a deletion of a single bit to codewords of the modified RM(1,  $m$ ) code. We then compute the minimum distance between the sets of strings obtained by applying the repetition of a single bit to codewords of the modified RM(1,  $m$ ) code. Finally, in each case, we propose a bounded distance decoding algorithm for up to half the corresponding minimum distance over a channel where, in addition to substitution errors, the synchronization error can occur as well. The complexity of the decoding algorithm is of the same order as that of the usual fast Hadamard transform based decoding for RM(1,  $m$ ) codes.

##### A. Minimum Distance

In this subsection, we first determine the minimum Hamming distance between the elements of sets associated with distinct codewords of the modified code that result from the deletion of a single bit. Let  $\hat{C}(m)$  denote the code whose generator matrix consists of the top  $m-1$  rows of  $\mathbf{G}_m$  and the binary sum of the last two rows of  $\mathbf{G}_m$ . The code  $\hat{C}(m)$  is immune to one deletion by construction.

We first make the following observation:

*Remark 4.1:* For  $m \geq 2$ , a codeword  $\mathbf{c}$  of  $C(m)$  belongs to  $\hat{C}(m)$  if and only if all its quadruplets starting at position  $i$  for  $i \equiv 1 \pmod{4}$  are all “1111” or “0000” or all are “0110” or “1001.”

In the remainder, we will call quadruplets of  $\mathbf{c}$  starting at position  $i$  for  $i \equiv 1 \pmod{4}$  *constituent* quadruplets.

For  $\mathbf{c} \in \hat{C}(m)$ , let  $S_d(\mathbf{c})$  denote the set of strings obtained by applying the deletion of a single bit to  $\mathbf{c}$ .

*Lemma 7:* For  $\mathbf{c}_a, \mathbf{c}_b$  distinct codewords in  $\hat{C}(m)$ , let  $D(\mathbf{c}_a, \mathbf{c}_b)$  be the smallest Hamming distance between  $\mathbf{s}_a$  and  $\mathbf{s}_b$  where  $\mathbf{s}_a$  ranges over all elements in the set  $S_d(\mathbf{c}_a)$  and  $\mathbf{s}_b$  ranges over all elements in the set  $S_d(\mathbf{c}_b)$ . Let

$$D_{\min}^m = \min_{\mathbf{c}_a, \mathbf{c}_b \in \hat{C}(m), \mathbf{c}_a \neq \mathbf{c}_b} D(\mathbf{c}_a, \mathbf{c}_b).$$

Then for  $m > 2$ ,  $D_{\min}^m = 2^{m-3}$ . Further, for  $m \geq 3$ ,  $D(\mathbf{c}_a, \mathbf{c}_b) = 2^{m-3}$  only for  $\mathbf{c}_a = c_j^m(01)$  or  $c_j^m(10)$  for  $j = 3 \times 2^{m-3}$ , and  $\mathbf{c}_b$  either  $\mathbf{c}_a + c_1^m(10)$ , or  $\mathbf{c}_a + c_1^m(01)$ , or *vice versa*, and in addition for  $m \geq 4$ ,  $\mathbf{c}_b$  is also  $\mathbf{c}_a + c_1^m(00)$ , or *vice versa*.

*Proof:* Suppose that  $\mathbf{c}_a$  experiences a deletion in position  $p_1$  and  $\mathbf{c}_b$  experiences a deletion in position  $p_2$ . Without loss of generality we can assume that  $p_1 < p_2$ . Let  $p'_1 = \lfloor (p_1 - 1)/4 \rfloor 4 + 1$  and let  $p'_2 = \lfloor (p_2 - 1)/4 \rfloor 4 + 4$ , so that  $p'_1$  denotes the first position of the constituent quadruplet  $p_1$  belongs to, and  $p'_2$  denotes the last position of the constituent quadruplet  $p_2$  belongs to. We also let  $l_1$  be the Hamming distance between the strings  $\mathbf{c}_a(1, p'_1 - 1)$  and  $\mathbf{c}_b(1, p'_1 - 1)$ ,  $l_2$  be the Hamming distance between the strings  $\mathbf{c}_a(p'_1, p'_2)$  and  $\mathbf{c}_b(p'_1, p'_2)$ , and  $l_3$  be the Hamming distance between the strings  $\mathbf{c}_a(p'_2 + 1, 2^m)$  and  $\mathbf{c}_b(p'_2 + 1, 2^m)$ , where the notation  $\mathbf{c}_i(p, q)$  indicates the substring of the codeword  $\mathbf{c}_i$  starting at position  $p$  and ending at position  $q$ . In addition, let  $n_c = (p'_2 - p'_1 + 1)/4$  be the total number of quadruplets spanned by positions  $p'_1$  and  $p'_2$ . By the standard properties of a Reed–Muller (1,  $m$ ) code,  $l_1 + l_2 + l_3$  is either  $2^{m-1}$  or  $2^m$ . Let  $\tilde{\mathbf{c}}_a = [\mathbf{c}_a(1, p_1 - 1) | \mathbf{c}_a(p_1 + 1, 2^m)]$ , and  $\tilde{\mathbf{c}}_b = [\mathbf{c}_b(1, p_2 - 1) | \mathbf{c}_b(p_2 + 1, 2^m)]$ . Then the Hamming distance  $d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b)$  between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  is

$$\begin{aligned} d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b) &= d_H(\tilde{\mathbf{c}}_a(1, p'_1 - 1), \tilde{\mathbf{c}}_b(1, p'_1 - 1)) \\ &\quad + d_H(\tilde{\mathbf{c}}_a(p'_1, p'_2 - 1), \tilde{\mathbf{c}}_b(p'_1, p'_2 - 1)) \\ &\quad + d_H(\tilde{\mathbf{c}}_a(p'_2, 2^m - 1), \tilde{\mathbf{c}}_b(p'_2, 2^m - 1)). \end{aligned}$$

Observe that the first term in the sum is simply  $l_1$  and that the last term is  $l_3$ . We let  $\tilde{l}_2$  denote the middle term,  $d_H(\tilde{\mathbf{c}}_a(p'_1, p'_2 - 1), \tilde{\mathbf{c}}_b(p'_1, p'_2 - 1))$ , and we establish the relationship between  $\tilde{l}_2$  and  $l_2$  for all choices of  $\mathbf{c}_a$  and  $\mathbf{c}_b$ , from which the bound on the overall distance will follow.

1) Let us first consider the case when the constituent quadruplets in  $\mathbf{c}_a$  are “0110” and “1001” and in  $\mathbf{c}_b$  are “0000” and “1111,” or *vice versa*. In this case, the Hamming distance between  $\mathbf{c}_a$  and  $\mathbf{c}_b$  is  $2^{m-1}$ , and the constituent quadruplet pairs starting at the same positions in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  each contribute 2 to the overall Hamming distance. Therefore,  $l_2 = 2n_c$ .

If  $p'_2 - p'_1 = 3$ , then the deletions occur in the same quadruplet,  $l_2$  is 2 to begin with, and the Hamming distance between  $\tilde{\mathbf{c}}_a(p'_1, p'_2 - 1)$  and  $\tilde{\mathbf{c}}_b(p'_1, p'_2 - 1)$  is at least 1, which can be verified by checking all cases. Hence, the Hamming distance between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  is at least  $2^{m-1} - 1$ , which is strictly greater than  $2^{m-3}$ .

Now suppose that  $p'_2 - p'_1 > 3$ . Then  $n_c > 1$ . After the deletions, the Hamming distance between  $\tilde{\mathbf{c}}_a(p'_1 + 4i, p'_1 + 3 + 4i)$  and  $\tilde{\mathbf{c}}_b(p'_1 + 4i, p'_1 + 3 + 4i)$ , for  $1 \leq i \leq n_c - 2$  is at least 1, as is the distance between the substrings  $\tilde{\mathbf{c}}_a(p'_2 - 3, p'_2 - 1)$  and  $\tilde{\mathbf{c}}_b(p'_2 - 3, p'_2 - 1)$ , and between the substrings  $\tilde{\mathbf{c}}_a(p'_1, p'_1 + 3)$  and  $\tilde{\mathbf{c}}_b(p'_1, p'_1 + 3)$  (which again can be verified by checking all cases). Then,  $\tilde{l}_2 \geq (n_c - 2) \times 1 + 1 \times 1 + 1 \times 1 = l_2/2$ .

Since the Hamming distance between  $\tilde{\mathbf{c}}_a(p'_1, p'_2 - 1)$  and  $\tilde{\mathbf{c}}_b(p'_1, p'_2 - 1)$  is at least  $l_2/2$ , the Hamming distance between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  is then at least  $l_1 + l_2/2 + l_3$ , which is lower-bounded by  $2^{m-2}$ , and thus strictly greater than  $2^{m-3}$ .

2) Suppose now that the constituent quadruplets are “0000” and “1111” in both  $\mathbf{c}_a$  and  $\mathbf{c}_b$ . The Hamming distance between

$\mathbf{c}_a$  and  $\mathbf{c}_b$  is either  $2^{m-1}$  or  $2^m$ , and the constituent quadruplet pairs starting at the same positions in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  each contribute either 0 or 4 to the overall Hamming distance. In the segment spanning positions  $p'_1$  and  $p'_2$  in  $\mathbf{c}_a$  and  $\mathbf{c}_b$ ,  $l_2/4$  of the constituent quadruplet pairs each contribute 4 to the overall Hamming distance between  $\mathbf{c}_a$  and  $\mathbf{c}_b$ .

If  $p'_2 - p'_1 = 3$ , the deletions occur in the same quadruplet, and  $l_2$  is either 0 or 4. Then  $d_H(\tilde{\mathbf{c}}_a(p'_1, p'_1+2), \tilde{\mathbf{c}}_b(p'_1, p'_1+2))$  is either 0 (if  $l_2 = 0$ ) or 3 (if  $l_2 = 4$ ). The overall distance between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  is thus at least  $2^{m-1} - 1$ , which is bigger than  $2^{m-3}$  for all  $m \geq 3$ .

If  $p'_2 - p'_1 > 3$ , we consider constituent quadruplets contained within positions  $p'_1 + 4$  and  $p'_2 - 4$  in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  that start at the same positions and we denote the set of their starting positions by  $Tot$  (the set  $Tot$  is nonempty as long as  $p'_1$  and  $p'_2$  belong to nonadjacent quadruplets). Let  $Com$  be the subset of  $Tot$  whose elements index complementary quadruplets in  $\mathbf{c}_a$  and  $\mathbf{c}_b$ . Then the Hamming distance between the quadruplets  $\mathbf{c}_a(i+1, i+4)$  and  $\mathbf{c}_b(i, i+3)$ , and, consequently, between  $\tilde{\mathbf{c}}_a(i, i+3)$  and  $\tilde{\mathbf{c}}_b(i, i+3)$  for  $i \in Com$  is at least 3. In addition, if  $p'_2$  belongs to the constituent quadruplets of  $\mathbf{c}_a$  and  $\mathbf{c}_b$  that are complements of each other, the distance between  $\tilde{\mathbf{c}}_a(p'_2-3, p'_2-1)$  and  $\tilde{\mathbf{c}}_b(p'_2-3, p'_2-1)$  is at least 3. Similarly, if  $p'_1$  belongs to the constituent quadruplets of  $\mathbf{c}_a$  and  $\mathbf{c}_b$  that are complements of each other, the distance between  $\tilde{\mathbf{c}}_a(p'_1, p'_1+3)$  and  $\tilde{\mathbf{c}}_b(p'_1, p'_1+3)$  is also at least 3. Therefore, the Hamming distance between  $\tilde{\mathbf{c}}_a(p'_1, p'_2-1)$  and  $\tilde{\mathbf{c}}_b(p'_1, p'_2-1)$  is at least  $3 \times l_2/4$ , thereby making the overall distance between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  be at least  $l_1 + 3l_2/4 + l_3$ , which is again strictly greater than  $2^{m-3}$ .

3) Finally, consider  $\mathbf{c}_a$  and  $\mathbf{c}_b$  with constituent quadruplets "0110" and "1001." Again, the Hamming distance between  $\mathbf{c}_a$  and  $\mathbf{c}_b$  is either  $2^{m-1}$  or  $2^m$ , and the constituent quadruplet pairs starting at the same positions in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  each contribute either 0 or 4 to it.

For the case when  $p'_2 - p'_1 = 3$ ,  $l_2$  is either 0 or 4, so that the Hamming distance between  $\tilde{\mathbf{c}}_a(p'_1, p'_2-1)$  and  $\tilde{\mathbf{c}}_b(p'_1, p'_2-1)$  is either at least 0 for  $l_2 = 0$  or at least 1 for  $l_2 = 4$ . In the former case, the Hamming distance between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  is at least  $2^{m-1}$ , and in latter case it is at least  $2^{m-1} - 3$ . In particular, for  $m \geq 4$ ,  $2^{m-1} - 3$  is strictly bigger than  $2^{m-3}$ . For  $m = 3$ ,  $2^{m-1} - 3 = 2^{m-3}$ . Then  $\mathbf{c}_a$  and  $\mathbf{c}_b$  would have to be complements of each other in the quadruplets experiencing deletions, and would have to be the same in their other quadruplet. For  $\mathbf{c}_a + \mathbf{c}_b$  being either "00001111" or "11110000,"  $\mathbf{c}_a$  is then either  $c_3^3(10)$  or  $c_3^3(01)$  and  $\mathbf{c}_b$  is either  $\mathbf{c}_a + c_1^3(10)$ , or  $\mathbf{c}_a + c_1^3(01)$ , or *vice versa*. Observe that these are precisely the codeword pairs listed at the beginning of the proof for  $j = 3$  and  $m = 3$ .

If  $p'_2 - p'_1 > 3$ , we again consider constituent quadruplets contained within positions  $p'_1 + 4$  and  $p'_2 - 4$  in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  that start at the same positions and we denote the set of their starting positions by  $Tot$  (the set  $Tot$  is nonempty as long as  $p'_1$  and  $p'_2$  belong to nonadjacent quadruplets). Let  $Com$  be the subset of  $Tot$  whose elements index complement quadruplets in  $\mathbf{c}_a$  and  $\mathbf{c}_b$ , and let  $Sam = Tot - Com$ .

Then the Hamming distance between  $\mathbf{c}_a(i+1, i+4)$  and  $\mathbf{c}_b(i, i+3)$  for  $i \in Com$  is either 1 or 2, and we denote their total number by  $s_1^1$  and  $s_2^1$ , respectively, such that  $s_1^1 + s_2^1 = |Com|$ . The Hamming distance between  $\mathbf{c}_a(i+1, i+4)$  and  $\mathbf{c}_b(i, i+3)$

for  $i \in Sam$  is either 2 or 3, and we similarly denote their total number by  $s_2^0$  and  $s_3^0$ , respectively, where  $s_2^0 + s_3^0 = |Sam|$ . In addition, if  $p'_2$  belongs to the constituent quadruplets of  $\mathbf{c}_a$  and  $\mathbf{c}_b$  that are complements of each other, the distance between  $\tilde{\mathbf{c}}_a(p'_2-3, p'_2-1)$  and  $\tilde{\mathbf{c}}_b(p'_2-3, p'_2-1)$  is either 1, 2, or 3, which we denote by  $t_2^1$ , and is either 0, 1, or 2 if those two quadruplets in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are the same, in which case we denote it by  $t_2^0$ . Let  $J_2 = 1$  if these two quadruplets are complements and let  $J_2 = 0$  otherwise. Finally, the distance between  $\tilde{\mathbf{c}}_a(p'_1, p'_1+3)$  and  $\tilde{\mathbf{c}}_b(p'_1, p'_1+3)$  is 0, 1, 2, or 3 if the corresponding quadruplets in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are the same, when is denoted by  $t_1^0$ , and is 1, 2, 3, or 4 if these quadruplets are complements, when is denoted by  $t_1^1$ . Let  $J_1 = 1$  for complement quadruplets and  $J_1 = 0$  for the same.

The overall Hamming distance between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  is then

$$l_1 + J_1 t_1^1 + (1 - J_1) t_1^0 + s_1^1 + 2s_2^1 + 2s_2^0 + 3s_3^0 + J_2 t_2^1 + (1 - J_2) t_2^0 + l_3.$$

Observe that  $s_1^1 + s_2^1 + J_1 + J_2 = l_2/4$ .

Since  $t_1^1 \geq 1$  and  $t_2^1 \geq 1$  we have

$$\begin{aligned} d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b) &\geq l_1 + J_1 + s_1^1 + s_2^1 + J_2 + l_3 \\ &= l_1 + \frac{l_2}{4} + l_3 \\ &\geq \frac{1}{4} d_H(\mathbf{c}_a, \mathbf{c}_b) \geq 2^{m-3}. \end{aligned}$$

Equality holds in this sequence of inequalities if and only if  $d_H(\mathbf{c}_a, \mathbf{c}_b) = 2^{m-1}$  (i.e.,  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are not complements of each other),  $l_1 = 0$ ,  $l_3 = 0$ ,  $s_2^0 = 0$ ,  $s_3^0 = 0$ ,  $s_2^1 = 0$ , and one of the following four cases holds:

- (a)  $(J_1, J_2) = (1, 1)$  and  $(t_1^1, t_2^1) = (1, 1)$
- (b)  $(J_1, J_2) = (0, 1)$  and  $(t_1^0, t_2^1) = (0, 1)$
- (c)  $(J_1, J_2) = (1, 0)$  and  $(t_1^1, t_2^0) = (1, 0)$
- (d)  $(J_1, J_2) = (0, 0)$  and  $(t_1^0, t_2^0) = (0, 0)$ .

Since  $l_1 = 0$  and  $l_3 = 0$ , all constituent quadruplets in  $\mathbf{c}_a(1, p'_1-1)$  and  $\mathbf{c}_b(1, p'_1-1)$  as well as in  $\mathbf{c}_a(p'_2+1, 2^m)$  and  $\mathbf{c}_b(p'_2+1, 2^m)$  are pairwise the same. Since  $s_2^0 = s_3^0 = 0$ , the constituent quadruplets spanning positions  $p'_1+4$  through  $p'_2-4$  in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are pairwise complements of each other. Moreover, since  $s_2^1 = 0$  they are actually alternating "1001" and "0110" in  $\mathbf{c}_a$  and are alternating "0110," or "1001" in  $\mathbf{c}_b$ , or *vice versa*. The quadruplets in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  to which  $p'_1$  ( $p'_2$ ) belongs are the same if  $J_1 = 0$  ( $J_2 = 0$ ), and otherwise they are complements. Therefore, for all four cases,  $\mathbf{c}_a + \mathbf{c}_b$  is of the type "0.01.10.0," with possibly one run of zeros empty (but not both as then  $\mathbf{c}_a$  and  $\mathbf{c}_b$  would be complements), and is such that it belongs to  $\hat{C}(m)$ . Specifically,  $\mathbf{c}_a + \mathbf{c}_b$  is either  $c_1^m(10)$ ,  $c_1^m(01)$ , or  $c_1^m(00)$ , and by  $p'_2 - p'_1 > 3$ ,  $m$  is at least 3. Since  $c_1^m(00) \notin \hat{C}(m)$  for  $m = 3$ , no new pairs can result from this analysis in this case, so we may assume from now on that  $m \geq 4$ .

Let  $p_l$  and  $p_r$  be the positions of the leftmost and the rightmost 1 in  $\mathbf{c}_a + \mathbf{c}_b$ . Then  $p'_1$  is either  $p_l$  or  $p_l - 4$ , depending on the value of  $J_1$  and on the format of  $\mathbf{c}_a + \mathbf{c}_b$ , and likewise  $p'_2$  is either  $p_r$  or  $p_r + 4$ , depending on the value of  $J_2$  and  $\mathbf{c}_a + \mathbf{c}_b$ . In particular,

for  $\mathbf{c}_a + \mathbf{c}_b$  equal to  $c_1^m(10)$ ,  $J_1$  must be 1 and  $p'_1 = p_l$ , and for  $\mathbf{c}_a + \mathbf{c}_b$  equal to  $c_1^m(01)$ ,  $J_2$  must be 1 and  $p'_2 = p_r$ .

For  $m > 5$ , since there are at least  $1/2(2^{m-2}) - 2$  contiguous alternating “0110” and “1001” (or *vice versa*) spanning positions  $p'_1 + 4$  and  $p'_2 - 4$  in  $\mathbf{c}_a$ , and since  $(p_l, p_r)$  is either  $(1, 2^{m-1})$  or  $(2^{m-1} + 1, 2^m)$  or  $(2^{m-2} + 1, 3 \times 2^{m-2})$ , by the concatenation principle it follows that all quadruplets spanning positions  $p_l$  and  $p_r$  in  $\mathbf{c}_a$  are alternating “0110” and “1001,” or *vice versa*. It then follows that under the set of constraints ( $l_1 = 0, l_3 = 0, s_2^0 = 0, s_3^0 = 0, s_2^1 = 0$ ),  $\mathbf{c}_a$  can only be  $c_j^m(01)$  or  $c_j^m(10)$  for  $j = 3 \times 2^{m-3}$ , and  $\mathbf{c}_b$  is then  $\mathbf{c}_a + c_1^m(10)$ ,  $\mathbf{c}_a + c_1^m(01)$ , or  $\mathbf{c}_a + c_1^m(00)$ , or *vice versa*. It remains to determine whether these candidate codeword pairs satisfy one of the (a) through (d) cases.

From the structure of the candidate codeword pairs, it follows for example that all six codeword pairs achieve  $D_{\min}^m$  for  $(J_1, J_2, t_1^1, t_2^1) = (1, 1, 1, 1)$ , with deletions in positions  $(p_1, p_2) = (2^{m-1} + 1, 2^m)$  for  $\mathbf{c}_a + \mathbf{c}_b = c_1^m(01)$ , in positions  $(p_1, p_2) = (1, 2^{m-1})$  for  $\mathbf{c}_a + \mathbf{c}_b = c_1^m(10)$ , and in positions  $(p_1, p_2) = (2^{m-2} + 1, 3 \times 2^{m-2})$  for  $\mathbf{c}_a + \mathbf{c}_b = c_1^m(00)$ , such that both individual values of  $\mathbf{c}_a$  and  $\mathbf{c}_b$  per pair are possible.

For  $m = 4$  and  $m = 5$ , the pairs of codewords  $\{\mathbf{c}_a, \mathbf{c}_b\}$  achieving the proposed  $D_{\min}^m$  can be identified directly, and they have the same format as codewords achieving  $D_{\min}^m$  for  $m > 5$ . This concludes the proof of the lemma.  $\square$

We next determine the minimum Hamming distance between the elements of sets associated with distinct codewords of the modified code that result from the repetition of a single bit.

For  $\mathbf{c} \in \hat{C}(m)$ , let  $S_r(\mathbf{c})$  denote the set of strings obtained by applying the repetition of a single bit to  $\mathbf{c}$ . Recall that  $S_d(\mathbf{c})$  denotes the set of strings obtained by applying the deletion of a single bit to  $\mathbf{c}$ .

*Lemma 8:* For  $\mathbf{c}_a, \mathbf{c}_b$  distinct codewords in  $\hat{C}(m)$ , let  $R(\mathbf{c}_a, \mathbf{c}_b)$  be the smallest Hamming distance between  $\mathbf{t}_a$  and  $\mathbf{t}_b$  where  $\mathbf{t}_a$  ranges over all elements in the set  $S_r(\mathbf{c}_a)$  and  $\mathbf{t}_b$  ranges over all elements in the set  $S_r(\mathbf{c}_b)$ . Let

$$R_{\min}^m = \min_{\mathbf{c}_a, \mathbf{c}_b \in \hat{C}(m), \mathbf{c}_a \neq \mathbf{c}_b} R(\mathbf{c}_a, \mathbf{c}_b).$$

Then for  $m > 2$ ,  $R_{\min}^m = 2^{m-3} + 1$ . Further, for  $m \geq 3$ ,  $R(\mathbf{c}_a, \mathbf{c}_b) = 2^{m-3} + 1$  only for  $\mathbf{c}_a = c_j^m(01)$  or  $c_j^m(10)$  for  $j = 3 \times 2^{m-3}$ , and  $\mathbf{c}_b$  either  $\mathbf{c}_a + c_1^m(10)$ , or  $\mathbf{c}_a + c_1^m(01)$ , or *vice versa*, and in addition for  $m \geq 4$ ,  $\mathbf{c}_b$  is also  $\mathbf{c}_a + c_1^m(00)$ , or *vice versa*.

*Proof:* We first observe that  $0 \leq R(\mathbf{c}_a, \mathbf{c}_b) - D(\mathbf{c}_a, \mathbf{c}_b) \leq 2$ , where  $D(\mathbf{c}_a, \mathbf{c}_b)$  is as defined in Lemma 7. To see this, consider  $\mathbf{s}_a$  obtained by deleting a bit in  $\mathbf{c}_a$  in position  $p_a$ , and  $\mathbf{s}_b$  obtained by deleting a bit in  $\mathbf{c}_b$  in position  $p_b$ . For  $p_a < p_b$ ,  $d_H(\mathbf{s}_a, \mathbf{s}_b)$  is

$$\begin{aligned} d_H(\mathbf{s}_a, \mathbf{s}_b) &= d_H(\mathbf{c}_a(1, p_a - 1), \mathbf{c}_b(1, p_a - 1)) \\ &\quad + d_H(\mathbf{c}_a(p_a + 1, p_b), \mathbf{c}_b(p_a, p_b - 1)) \\ &\quad + d_H(\mathbf{c}_a(p_b + 1, n), \mathbf{c}_b(p_b + 1, n)). \end{aligned}$$

For  $\mathbf{t}_a \in S_r(\mathbf{c}_a)$  and  $\mathbf{t}_b \in S_r(\mathbf{c}_b)$  such that the bit in position  $p_b$  ( $p_a$ ) is the bit that gets repeated in  $\mathbf{t}_a$  ( $\mathbf{t}_b$ ), write  $d_H(\mathbf{t}_a, \mathbf{t}_b)$  as

$$\begin{aligned} d_H(\mathbf{t}_a, \mathbf{t}_b) &= d_H(\mathbf{c}_a(1, p_a - 1), \mathbf{c}_b(1, p_a - 1)) + d_1 \\ &\quad + d_H(\mathbf{c}_a(p_a + 1, p_b), \mathbf{c}_b(p_a, p_b - 1)) + d_2 \end{aligned}$$

$$+ d_H(\mathbf{c}_a(p_b + 1, n), \mathbf{c}_b(p_b + 1, n))$$

where  $d_1 = \mathbf{c}_a(p_a) + \mathbf{c}_b(p_a)$  and  $d_2 = \mathbf{c}_a(p_b) + \mathbf{c}_b(p_b)$ . Therefore,  $0 \leq d_H(\mathbf{t}_a, \mathbf{t}_b) - d_H(\mathbf{s}_a, \mathbf{s}_b) = d_1 + d_2 \leq 2$ . A similar argument gives the same inequality for  $p_a > p_b$ . Taking the minimum over all  $(p_a, p_b)$ , the claim of this paragraph follows.

By Lemma 7,  $D_{\min}^m = 2^{m-3}$ , so that  $R_{\min}^m$  is at most  $2^{m-3} + 2$ . We use the nomenclature introduced in Lemma 7 to determine the codewords  $\mathbf{c}_a, \mathbf{c}_b$  for which  $D(\mathbf{c}_a, \mathbf{c}_b)$  yields the proposed bound on  $R(\mathbf{c}_a, \mathbf{c}_b)$ , i.e., the codewords for which  $D(\mathbf{c}_a, \mathbf{c}_b)$  is at most  $2^{m-3} + 1$ .

1) Let us first consider the case when the constituent quadruplets in  $\mathbf{c}_a$  are “0110” and “1001” and in  $\mathbf{c}_b$  are “0000” and “1111,” or *vice versa*. From the proof of Lemma 7 it follows that  $D(\mathbf{c}_a, \mathbf{c}_b)$  is at least  $2^{m-2}$ , as is then  $R(\mathbf{c}_a, \mathbf{c}_b)$ . The proposed lower bound can only be met for  $m = 3$ . By checking all cases, for  $m = 3$ , it follows that  $R(\mathbf{c}_a, \mathbf{c}_b)$  is at least 4, thus exceeding the proposed lower bound.

2) Suppose now that the constituent quadruplets are “0000” and “1111” in both  $\mathbf{c}_a$  and  $\mathbf{c}_b$ . From the proof of Lemma 7 it follows that  $D(\mathbf{c}_a, \mathbf{c}_b)$  is at least  $3 \times 2^{m-3}$ , and thus  $R(\mathbf{c}_a, \mathbf{c}_b)$  is strictly greater than the proposed lower bound.

3) Finally, consider  $\mathbf{c}_a$  and  $\mathbf{c}_b$  with constituent quadruplets “0110” and “1001.”

We assume that  $p'_1, p'_2, \tilde{c}_a, \tilde{c}_b, l_1, l_2, l_3$  as well as  $t_1^0, t_1^1, t_2^0, t_2^1, s_1^1, s_2^1, s_2^0$ , and  $s_3^0$  are as defined in the proof of Lemma 7.

In the notation of Lemma 7, if  $p'_2 - p'_1 = 3$ ,  $D(\mathbf{c}_a, \mathbf{c}_b)$  is at least  $2^{m-1} - 3$ , thus exceeding  $2^{m-3} + 1$  for  $m \geq 4$ . For  $m = 3$ , there are four codewords in  $\hat{C}(m)$  having “0110” and “1001” as constituent quadruplets. It follows by direct checking that  $R(\mathbf{c}_a, \mathbf{c}_b) = 2^{m-3} + 1 = 2$  only for  $\mathbf{c}_a = “01101001”$  or “10010110” and  $\mathbf{c}_b = \mathbf{c}_a + c_1^3(01)$ , or  $\mathbf{c}_b = \mathbf{c}_a + c_1^3(10)$ , or *vice versa*. In the remainder, we will assume  $m \geq 4$ .

For  $p'_2 - p'_1 > 3$ , in the notation of Lemma 7, the overall Hamming distance between  $\tilde{\mathbf{c}}_a$  and  $\tilde{\mathbf{c}}_b$  is

$$\begin{aligned} d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b) &= l_1 + J_1 t_1^1 + (1 - J_1) t_1^0 + s_1^1 + 2s_2^1 \\ &\quad + 2s_2^0 + 3s_3^0 + J_2 t_2^1 + (1 - J_2) t_2^0 + l_3 \quad (3) \end{aligned}$$

where  $s_1^1 + s_2^1 + J_1 + J_2 = l_2/4$ .

As established in Lemma 7, for  $d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b)$  to equal  $2^{m-3}$  for  $m \geq 4$  it is necessary that  $\mathbf{c}_a$  is  $c_j^m(01)$  or  $c_j^m(10)$  for  $j = 3 \times 2^{m-3}$ , and  $\mathbf{c}_b$  is either  $\mathbf{c}_a + c_1^m(10)$ ,  $\mathbf{c}_a + c_1^m(01)$ , or  $\mathbf{c}_a + c_1^m(00)$ , or *vice versa*. By direct checking, it follows that  $R(\mathbf{c}_a, \mathbf{c}_b)$  is precisely  $2^{m-3} + 1$  for all six codeword pairs (since the deletions in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  yielding  $D_{\min}^m$  are such that one of them belongs to a run of size 2 and the other belongs to a run of size 1).

It remains to determine  $\mathbf{c}_a, \mathbf{c}_b$  for which  $d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b)$  equals  $2^{m-3} + 1$ , and such that both deletions occur in runs of size bigger than 1. Using the expression in (3) it follows that  $d_H(\mathbf{c}_a, \mathbf{c}_b) = 2^{m-3}$ ,  $l_1 = 0, l_3 = 0, s_2^0 = 0, s_3^0 = 0$  (use  $(\Delta)$  as a shorthand for this set of conditions and one of the following holds:

- (a)  $(J_1, J_2) = (0, 1)$  and  $(t_1^0, t_2^1, s_2^1) = (0, 2, 0)$
- (b)  $(J_1, J_2) = (0, 1)$  and  $(t_1^0, t_2^1, s_2^1) = (1, 1, 0)$
- (c)  $(J_1, J_2) = (1, 0)$  and  $(t_1^1, t_2^0, s_2^1) = (2, 0, 0)$
- (d)  $(J_1, J_2) = (1, 0)$  and  $(t_1^1, t_2^0, s_2^1) = (1, 1, 0)$

- (e)  $(J_1, J_2) = (0, 0)$  and  $(t_1^0, t_2^0, s_2^1) = (1, 0, 0)$
- (f)  $(J_1, J_2) = (0, 0)$  and  $(t_1^0, t_2^0, s_2^1) = (0, 1, 0)$
- (g)  $(J_1, J_2) = (1, 1)$  and  $(t_1^1, t_2^1, s_2^1) = (2, 1, 0)$
- (h)  $(J_1, J_2) = (1, 1)$  and  $(t_1^1, t_2^1, s_2^1) = (1, 2, 0)$
- (i)  $(J_1, J_2) = (0, 1)$  and  $(t_1^0, t_2^1, s_2^1) = (0, 1, 1)$
- (j)  $(J_1, J_2) = (1, 0)$  and  $(t_1^1, t_2^0, s_2^1) = (1, 0, 1)$
- (k)  $(J_1, J_2) = (0, 0)$  and  $(t_1^0, t_2^0, s_2^1) = (0, 0, 1)$
- (l)  $(J_1, J_2) = (1, 1)$  and  $(t_1^1, t_2^1, s_2^1) = (1, 1, 1)$ .

Since  $l_1 = 0$  and  $l_3 = 0$ , all constituent quadruplets in  $\mathbf{c}_a(1, p'_1 - 1)$  and  $\mathbf{c}_b(1, p'_1 - 1)$  as well as in  $\mathbf{c}_a(p'_2 + 1, 2^m)$  and  $\mathbf{c}_b(p'_2 + 1, 2^m)$  are pairwise the same. Since  $s_2^0 = s_3^0 = 0$ , the constituent quadruplets spanning positions  $p'_1 + 4$  through  $p'_2 - 4$  in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are pairwise complements of each other. Therefore, for all cases,  $\mathbf{c}_a + \mathbf{c}_b$  is of the type “0.01.10.0,” with possibly one run of zeros empty (but not both as then  $\mathbf{c}_a$  and  $\mathbf{c}_b$  would be complements), and is such that it belongs to  $\hat{C}(m)$ .

First observe that for cases (a) through (h), the common constraint  $s_2^1 = 0$ , along with the constraint set  $(\Delta)$  is the same as the set of constraints on the same parameters established in the proof of Lemma 7. As given in the proof of Lemma 7 under the set of constraints  $(l_1 = 0, l_3 = 0, s_2^0 = 0, s_3^0 = 0, s_2^1 = 0)$ ,  $\mathbf{c}_a$  can only be  $c_j^m(01)$  or  $c_j^m(10)$  for  $j = 3 \times 2^{m-3}$ , and  $\mathbf{c}_b$  can only then be  $\mathbf{c}_a + c_1^m(10)$ ,  $\mathbf{c}_a + c_1^m(01)$ , or  $\mathbf{c}_a + c_1^m(00)$ , or *vice versa*. Observe that these codeword pairs are already established in the earlier case when  $d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b) = 2^{m-3}$  was analyzed (though it can also be verified that these candidate codeword pairs satisfy at least one of the (a) through (h) cases, and with deletions in appropriate runs of size 2 result in strings with Hamming distance  $2^{m-3} + 1$ ).

The remaining cases (i) through (l) all share the same constraint that  $s_2^1 = 1$ , which implies that all constituent quadruplets spanning positions  $p'_1 + 8$  through  $p'_2 - 4$  in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are the complements of their left neighboring quadruplets, except for one constituent quadruplet which is the same as its left neighboring quadruplet.

Let  $p_l$  and  $p_r$  be the positions of the leftmost and the rightmost 1 in  $\mathbf{c}_a + \mathbf{c}_b$ , so that  $(p_l, p_r)$  is either  $(1, 2^{m-1})$  or  $(2^{m-1} + 1, 2^m)$  or  $(2^{m-2} + 1, 3 \times 2^{m-2})$ . Depending on the values of  $J_1$  and  $J_2$  and the structure of  $\mathbf{c}_a + \mathbf{c}_b$ ,  $p'_1$  is either  $p_l - 4$  or  $p_l$  and  $p'_2$  is either  $p_r + 4$  or  $p_r$ , so that all constituent quadruplets spanning positions  $p_l + 8$  through  $p_r - 4$  in  $\mathbf{c}_a$  and  $\mathbf{c}_b$  are the complements of their left neighboring quadruplets with the exception of one constituent quadruplet which is the same as its left neighboring quadruplet.

For  $m > 5$ , by the concatenation principle it follows that this singular constituent quadruplet must be the one starting at the position  $2^{m-1} + 1$  so that  $\mathbf{c}_a + \mathbf{c}_b$  is  $c_1^m(00)$ . Moreover, the constituent quadruplets spanning positions  $p_l = 2^{m-2} + 1$  and  $2^{m-1}$  are then alternating “0110” and “1001” (or *vice versa*), followed by alternating “1001” and “0110” (or *vice versa*) that span positions  $2^{m-1} + 1$  and  $p_r = 3 \times 2^{m-2}$ . As a result, it follows that  $\mathbf{c}_a$  is  $c_j^m(11)$  or  $c_{j-1}^m(00)$  for  $j = 3 \times 2^{m-3} - 1$ , and  $\mathbf{c}_b$  is either  $\mathbf{c}_a + c_1^m(10)$ ,  $\mathbf{c}_a + c_1^m(01)$ , or  $\mathbf{c}_a + c_1^m(00)$ , or *vice versa*. However, in all cases, when  $d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b) = 2^{m-3} + 1$ , not both deletion errors can be in runs of size bigger than 1 (which

can be verified by direct checking of all possible constraint sets as given by (i) through (l)), and therefore,  $R(\mathbf{c}_a, \mathbf{c}_b)$  is strictly greater than  $2^{m-3} + 1$ .

For  $m = 4, 5$  it can be checked directly that the only codeword pairs achieving  $d_H(\tilde{\mathbf{c}}_a, \tilde{\mathbf{c}}_b) = 2^{m-3} + 1$  under the current constraints are the same as for  $m > 5$ . Again, not both deletions can occur in runs of size 2, and thus  $R(\mathbf{c}_a, \mathbf{c}_b)$  is again strictly greater than  $2^{m-3} + 1$ .  $\square$

## B. Decoding Algorithm

In this subsection, we first propose a bounded distance decoding scheme for  $\hat{C}(m)$  which corrects one deletion and up to  $2^{m-4} - 1$  substitution errors. We outline the algorithm and discuss its correctness and complexity.

A common technique for decoding a codeword in a Reed–Muller  $(1, m)$  code that has experienced a certain number of substitution errors involves computing a fast Hadamard transform of the received string, [17, Sec. 4, Ch. 14]. Specifically, the received string  $\mathbf{s}$  (of length  $n$ ) is multiplied by a Hadamard matrix  $H_n$  to form  $\mathbf{s}H_n$ . The computation is done efficiently by starting with the binary string  $\mathbf{s}$  of length  $n = 2^m$  and carrying out  $m$  stages, each of which involves  $n = 2^m$  additions of integers, to return the integer valued string  $\mathbf{s}H_n$  of length  $n$ . Subsequently, one needs to find the coordinate in this integer string of maximum absolute value. The complexity of the overall algorithm is therefore normally quoted as  $O(n \log n)$ .

In our situation, let  $\mathbf{c} \in \hat{C}(m)$  for  $m \geq 5$  be the transmitted codeword. Let  $\mathbf{s}$  be the received string obtained from  $\mathbf{c}$  by one deletion and at most  $2^{m-4} - 1$  substitution errors. Thus, the received string  $\mathbf{s}$  is of length  $n - 1$ . The objective is to recover  $\mathbf{c}$  from  $\mathbf{s}$ . In principle, one could construct strings of length  $n$  by inserting either 0 or 1 at each position in  $\mathbf{s}$  and compare each resulting string with candidate codewords from  $\hat{C}(m)$ , which would be equivalent to performing  $2n$  standard decoding operations. The complexity of such an algorithm would be  $O(n^2 \log n)$ . However, it is possible to do much better.

For any codeword  $\tilde{\mathbf{c}} \in \hat{C}(m)$ , write  $\tilde{\mathbf{c}} = [\tilde{\mathbf{c}}^L | \tilde{\mathbf{c}}^R]$ , where  $\tilde{\mathbf{c}}^L$  and  $\tilde{\mathbf{c}}^R$  are each of length  $2^{m-1}$ . In particular, the transmitted codeword  $\mathbf{c}$  is written as  $\mathbf{c} = [\mathbf{c}^L | \mathbf{c}^R]$ . From the received string  $\mathbf{s}$  we create  $\mathbf{s}^L = [s(1) \dots s(2^{m-1})]$  and  $\mathbf{s}^R = [s(2^{m-1}) \dots s(2^m - 1)]$ . Each of these strings is of length  $2^{m-1}$ .

If the location of the deletion is in the second half of the codeword, then  $\mathbf{s}^L$  is obtained from  $\mathbf{c}^L$  by at most  $2^{m-4} - 1$  substitution errors. Further, for every  $\tilde{\mathbf{c}} \in \hat{C}(m)$  other than  $\mathbf{c}$  and  $\mathbf{c} + c_1^m(01)$  we have

$$\begin{aligned} d_H(\mathbf{s}^L, \tilde{\mathbf{c}}^L) &\geq d_H(\mathbf{c}^L, \tilde{\mathbf{c}}^L) - d_H(\mathbf{s}^L, \mathbf{c}^L) \\ &\geq 2^{m-2} - (2^{m-4} - 1) \\ &> 2^{m-4}. \end{aligned}$$

If one uses the fast Hadamard transform to compute  $[\mathbf{s}^L | \mathbf{0}]H_n$ , the coordinate with maximum absolute value will then correspond to either the pair comprised of  $\mathbf{c}$  and its bitwise complement or the pair comprised of  $\mathbf{c} + c_1^m(01)$  and its bitwise complement. Further, there will be at most two competing locations for the maximum absolute value.

Similarly, if the location of the deletion is in the first half of the codeword, then  $\mathbf{s}^R$  is obtained from  $\mathbf{c}^R$  by at most  $2^{m-4} - 1$  substitution errors, so by using the fast Hadamard transform to compute  $[\mathbf{0} | \mathbf{s}^R]H_n$ , the coordinate with maximum absolute value will correspond to either the pair comprised of  $\mathbf{c}$  and its bitwise complement or the pair comprised of  $\mathbf{c} + \mathbf{c}_1^m(10)$  and its bitwise complement. Again, there will be at most two competing locations for the maximum absolute value.

Thus, in  $O(n \log n)$  operations we will be presented with at most eight candidates for the transmitted codeword. We may now go to the naive step of considering all the  $2n$  strings of length  $n$  got by inserting either 0 or 1 at each position in  $\mathbf{s}$  and compare each resulting string with each of these eight candidate codewords. In  $O(n)$  operations we will arrive at the true codeword.

There are some obvious inefficiencies in the algorithm just described. For instance, it is not really necessary to compare the received string with the columns of  $H_n$  that correspond to strings in  $C(m)$  that are not in  $\hat{C}(m)$ . An analysis of this inefficiency could save a constant factor. The second stage could also undoubtedly be improved, but this is less interesting because the overall complexity is dominated by the first stage. Since using the first stage as described has the significant practical advantage that the existing hardware which is used to decode when there is no deletion can also be used when there is a deletion, we have preferred to describe the overall algorithm as above.

Finally, along similar lines, we propose a bounded distance decoding scheme for  $\hat{C}(m)$  for  $m \geq 4$  which corrects one repetition and up to  $2^{m-4}$  substitution errors and discuss its correctness and complexity. Let  $\mathbf{s}$  be the received string obtained from  $\mathbf{c}$  by one repetition and at most  $2^{m-4}$  substitution errors. Thus, the received string  $\mathbf{s}$  is of length  $n + 1$ . As before, for any codeword  $\tilde{\mathbf{c}} \in \hat{C}(m)$ , write  $\tilde{\mathbf{c}} = [\tilde{\mathbf{c}}^L | \tilde{\mathbf{c}}^R]$ , where  $\tilde{\mathbf{c}}^L$  and  $\tilde{\mathbf{c}}^R$  are each of length  $2^{m-1}$ . The transmitted codeword  $\mathbf{c}$  is written as  $\mathbf{c} = [\mathbf{c}^L | \mathbf{c}^R]$ . From the received string  $\mathbf{s}$  we create  $\mathbf{s}^L = [s(1) \dots s(2^{m-1})]$  and  $\mathbf{s}^R = [s(2^{m-1}) \dots s(2^m - 1)]$ . Each of these strings is of length  $2^{m-1}$ .

If the location of the repetition is in the second half of the codeword, then we get  $\mathbf{s}^L$  from  $\mathbf{c}^L$  by at most  $2^{m-4}$  substitution errors. Further, for every  $\tilde{\mathbf{c}} \in \hat{C}(m)$  other than  $\mathbf{c}$  and  $\mathbf{c} + \mathbf{c}_1^m(01)$  we have

$$\begin{aligned} d_H(\mathbf{s}^L, \tilde{\mathbf{c}}^L) &\geq d_H(\mathbf{c}^L, \tilde{\mathbf{c}}^L) - d_H(\mathbf{s}^L, \mathbf{c}^L) \\ &\geq 2^{m-2} - 2^{m-4} \\ &> 2^{m-4}. \end{aligned}$$

If one uses the fast Hadamard transform to compute  $[\mathbf{s}^L | \mathbf{0}]H_n$ , the coordinate with maximum absolute value will then correspond to either the pair comprised of  $\mathbf{c}$  and its bitwise complement or the pair comprised of  $\mathbf{c} + \mathbf{c}_1^m(01)$  and its bitwise complement. Further, there will be at most two competing locations for the maximum absolute value.

Similarly, if the location of the repetition is in the first half of the codeword, then we get  $\mathbf{s}^R$  from  $\mathbf{c}^R$  by at most  $2^{m-4}$  substitution errors, so by using the fast Hadamard transform to compute  $[\mathbf{0} | \mathbf{s}^R]H_n$ , the coordinate with maximum absolute value will correspond to either the pair comprised of  $\mathbf{c}$  and its bitwise complement or the pair comprised of  $\mathbf{c} + \mathbf{c}_1^m(10)$  and its bitwise

complement. Again, at most two locations will compete for the maximum absolute value.

Thus, in  $O(n \log n)$  operations, there will be at most eight candidates for the transmitted codeword. We may now again follow the naive way and consider all the  $2n$  strings of length  $n$  obtained by inserting either 0 or 1 at each position in  $\mathbf{s}$  and compare each resulting string with each of these eight candidate codewords. Using this approach, in  $O(n)$  operations the true codeword will follow.

## V. CONCLUSION AND FUTURE WORK

In this paper, we studied the performance of a Reed–Muller  $RM(1, m)$  code, as an instance of a substitution-error correcting code, over channels in which, in addition to substitution errors, a sampling error can cause synchronization errors. Specifically, we studied the cases where the synchronization error results in the deletion of a single bit and where it results in the repetition of a single bit. The model we worked with is aimed at handling the kinds of errors that can occur in a variety of applications, such as magnetic recording and wireless transmission, in the absence of adequate timing recovery. Our approach to handling synchronization errors is to start with a good substitution-error correcting code, to analyze which codeword pairs cause the identification problem, and then find a linear subcode of as high a rate as possible that would both provide protection against substitution errors and be robust to the synchronization errors. The rate loss incurred from using the subcode and the increase in the complexity of the decoding algorithm should of course be reasonably small for such an approach to work.

Another contribution of this paper is to develop several structural properties of the  $RM(1, m)$  codes, which were motivated by this point of view. These structural properties may be of interest in their own right.

In general, we provided an analysis that is combinatorially much tighter than might be needed for our immediate concerns. These combinatorial results may also be of independent interest. Specifically, we enumerated all pairs of codewords of the  $RM(1, m)$  codes that suffer from an identification problem over a channel allowing for the deletion of a single bit. We introduced a pruned linear subcode of the  $RM(1, m)$  code, with the loss of one information bit, which does not suffer from the identification problem under the deletion of a single bit. Given a pair of codewords in the pruned code, the appropriate notion of distance between them over a channel permitting synchronization errors is the minimum Hamming distance between any pair of strings which are derived, respectively, from each codeword after the application of such synchronization error. We gave a combinatorially tight analysis of the minimum distance of the pruned code for this notion of distance for both the case of the deletion of a single bit and the case of the repetition of a single bit. Specifically, we explicitly identified all pairs of codewords of the pruned code for which the post-synchronization error Hamming distance equals the corresponding post-synchronization minimum distance of the pruned code.

Finally, we provided a bounded distance decoding algorithm, suitable for the use of the pruned code over a channel where in addition to possibly one deletion error (respectively, one repetition error), substitution errors can occur as well. The complexity

of this algorithm is of the same order as that of the usual fast Hadamard transform based decoding for the  $RM(1, m)$  code. What is more, the proposed algorithm can in fact be essentially run on the same hardware platform as in the case without synchronization errors.

There are of course many codes that are superior to  $RM(1, m)$  codes in several respects (for instance, having higher rates). Future work would involve studying the behavior under our synchronization error model of other families of codes with good substitution-error correcting properties. The analysis should also be broadened to include more general models in which several repetitions and deletions are simultaneously allowed. As in this paper, the aim of such an analysis would be to find pruned versions of such codes, with low rate loss and only moderate increase in decoding complexity, which would not only have good substitution error-correcting capabilities but would also provide protection against the sampling errors of interest. Additional work in progress of ours along these lines has been reported in [10].

#### ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for constructive comments that improved the presentation of this paper.

#### REFERENCES

- [1] J. Barry, A. Kavčić, S. McLaughlin, A. Nayak, and W. Zeng, "Iterative timing recovery," *IEEE Signal Processing Mag.*, vol. 21, no. 1, pp. 89–102, Jan. 2004.
- [2] R. C. Bose and J. G. Caldwell, "Synchronizable error-correcting codes," *Inf. Contr.*, vol. 10, pp. 616–630, 1967.
- [3] P. A. H. Bours, "Construction of fixed-length insertion/deletion correcting runlength-limited code," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 1841–1856, Nov. 1994.
- [4] L. Calabi and W. E. Hartnett, "A family of codes for the correction of substitution and synchronization errors," *IEEE Trans. Inf. Theory*, vol. IT-15, no. 1, pp. 102–106, Jan. 1969.
- [5] G. Chen, M. Mitzenmacher, C. Ng, and N. Varnica, "Concatenated codes for deletion channels," in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, Japan, Jun./Jul. 2003, p. 218.
- [6] W. A. Clarke and H. C. Ferreira, "A new linear, quasicyclic multiple insertion/deletion correcting code," in *2003 IEEE Pacific Rim Conf. Communications Computers and Signal Processing*, Victoria, BC, Canada, Aug. 2003, pp. 899–902.
- [7] W. A. Clarke, A. S. J. Helberg, and H. C. Ferreira, "Constrained codes for the correction of synchronization and additive errors," in *Proc. 1993 IEEE South African Symp. Communications and Signal Processing. COMSIG*. New York: IEEE, 1993, pp. 58–62.
- [8] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.
- [9] L. Dolecek, "On Structural Properties of Reed-Muller  $RM(1, m)$  Codes and Their Use in Channels with Synchronization Errors," Elec. Eng. Comp. Sci. Dept., Univ. Calif. Berkeley, Tech. Rep., Apr. 2006 [Online]. Available: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-43.pdf>
- [10] L. Dolecek and V. Anantharam, "On Array-Based LDPC Codes in Channels With Varying Sampling Rate," Elec. Eng. Comp. Sci. Dept., Univ. Calif. Berkeley, Tech. Rep., Jan. 2006 [Online]. Available: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-7.pdf>
- [11] H. C. Ferreira, W. A. Clarke, A. S. J. Helberg, K. A. S. Abdel-Gaffar, and A. J. H. Vinck, "Insertion/deletion correction with spectral nulls," *IEEE Trans. Inf. Theory*, vol. 43, no. 2, pp. 722–732, Mar. 1997.
- [12] T. Kløve, "Codes correcting a single insertion/deletion of a zero or a single peak-shift," *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 279–283, Jan. 1995.
- [13] P. Kovintavewat, J. R. Barry, M. F. Erden, and E. Kurtas, "Per-survivor iterative timing recovery for coded partial response channels," in *Proc. IEEE Int. Conf. Communications*, Dallas, TX, Nov./Dec. 2004, pp. 2604–2608.
- [14] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys—Dokl.*, vol. 10, no. 8, pp. 707–710, Feb. 1966.
- [15] ———, "On perfect codes in deletion and insertion metric," *Discr. Math. Appl.*, vol. 2, no. 3, pp. 241–258, 1992.
- [16] J. Liu, H. Song, and B. V. K. V. Kumar, "Symbol timing recovery for low-SNR partial response recording channels," in *Proc. GLOBECOM'02—IEEE Global Telecommunications Conf.*, Taipei, Taiwan, Nov. 2002, pp. 1129–1136.
- [17] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
- [18] A. R. Nayak, J. Barry, and S. W. McLaughlin, "Joint timing recovery and turbo equalization for coded partial response channels," *IEEE Trans. Magn.*, vol. 38, no. 5, pp. 2295–97, Sep. 2002.
- [19] N. J. A. Sloane, "On Single Deletion Correcting Codes" [Online]. Available: <http://www.att.research.com/~njas> 2000
- [20] J. J. Stiffler, "Comma free error correcting codes," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 1, pp. 107–112, Jan. 1965.
- [21] G. Tenengolts, "Nonbinary codes correcting single deletion or insertion," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 5, pp. 766–769, Sep. 1984.
- [22] J. D. Ullman, "Near optimal single synchronization error correcting code," *IEEE Trans. Inf. Theory*, vol. IT-12, no. 4, pp. 418–424, Oct. 1966.
- [23] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors," *Avtomat. Telemekh.*, vol. 26, no. 2, pp. 288–292, 1965.
- [24] W. Zhang and A. Kavčić, "Optimal soft output detector for channels with intersymbol interference and timing errors," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 2555–557, Sep. 2003.