Run-Length Properties of a Reed-Muller RM(1,m) Code with Applications in Channels with at Most One Synchronization Error *

Lara Dolecek, Venkat Anantharam
Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA, 94720
{dolecek,ananth}@eecs.berkeley.edu

Abstract

In this paper we analyze the performance of a Reed-Muller(1,m) code in channels where at most one synchronization error, viewed as either a repetition or a deletion of a bit, occurs. We first establish several useful properties of the run-lengths of a Reed-Muller(1,m) code. Based on the transformation of a code in which a repetition operation becomes an insertion of a zero operation, we conclude that a Reed-Muller(1,m) code is immune to a sampling error that causes a single repetition of a bit. By analyzing the image of the code under this same transformation, and utilizing the previously established properties of a Reed-Muller(1,m) code, we are able to enumerate all pairs of sequences of this code that can result in the same received sequence when a sampling error causes a deletion of a bit. In addition, we propose a simple way to prune the code so that the resulting linear subcode is immune to a single deletion.

1 Introduction

In many communication systems, a binary input message $\mathbf{x}$ is encoded at the transmitter using substitution-error correcting code $C(n,k)$ into a coded sequence $\mathbf{c}$ = $C(\mathbf{x})$. The modulated version of this sequence is corrupted by additive noise and arrives at the receiver as a waveform $r(t)$,

$$r(t) = \sum_i c_i h(t - iT) + n(t), \tag{1}$$

where c_i is the ith bit of $\mathbf{c}$, $h(t)$ is the modulating pulse, and $n(t)$ is the noise introduced in the channel.

*Research supported by NSF grant ECS 0123512, Marvell Technology Group, and the California MICRO program.

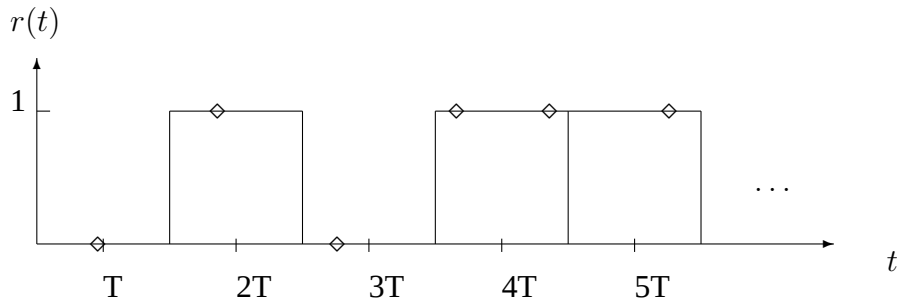

Figure 1: An example of oversampling.

Upon receiving $r(t)$, the receiver samples it at time instances $kT_s + \tau_k$. The sequence of samples is fed into the decoder which produces the most likely input message. In order to fully exploit coding gains, accurate synchronization becomes critical and many proposed methods focus on estimating and adjusting the phase offset τ_k, [6], [12]. Here, the underlying assumption is that the initial synchronization is accurate enough to ensure that, even with some frequency error, there will still be one sample per symbol. However, when the initial samples are not taken at correct places, with an incorrect frequency rate (i.e. when $T_s \neq T$), a symbol can be sampled twice or not at all. This situation can arise in many applications, for example in wireless fading channels and digital magnetic and optical recording channels, typically because of slow convergence of a decision feedback based timing recovery loop.

To illustrate the issues that arise, assume that $h(t)$ is a rectangular pulse of duration T and unit amplitude i.e. the modulation scheme is pulse-amplitude modulation (PAM). Let us also assume that we are operating in the infinite signal-to-noise (SNR) regime where the effect of $n(t)$ is negligible. As a result, $r(t)$ simply becomes

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

If the waveform is sampled at instances separated by T (e.g. samples taken in the middle of each pulse would be at $kT + T/2$), then the sampled version of $r(t)$ would be precisely $\mathbf{c}$. Now suppose that the initial frequency error causes the sampling to occur at instances $kT_s + \tau_k$.

As an example, consider the sequence $\mathbf{c}$ = (0,1,0,1,1,. . .) that results in the waveform $r(t)$ shown in Figure 1. The sampling points $kT_s + \tau_k$ are marked in the figure by $\diamond$. In this example, $T_s < T$ causes oversampling. As a result, the sampled version of $r(t)$ contains a repeated bit (the fourth bit is sampled twice). If $T_s > T$, undersampling occurs, and the separation between two consecutive samples can become so large that some bit is not sampled at all. Therefore, without the initial synchronization, a frequency error causes the sampled version of $r(t)$ to be the sequence obtained by repeating or deleting some bits in $\mathbf{c}$.

Assuming that the exact sampling instances are not known, a coded sequence $\mathbf{c}$ can give rise to a whole set of received sampled versions of $r(t)$. When two distinct sequences $\mathbf{c_1}$ and $\mathbf{c_2}$ 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. We then say that the code

$C(n,k)$ suffers from an *identification problem*.

We adopt a set-theoretic model in which a coded sequence gives rise to a set of possible received sampled sequences depending on whether a bit was repeated or deleted. We say that the identification problem occurs whenever two distinct codewords have intersecting sets. In this context, our goal is to determine necessary and sufficient conditions for the identification problem to not occur. Furthermore, we wish to determine the largest linear subcode for which the identification problem is eliminated. In the remainder of the paper, we will address these questions for a Reed-Muller(1,m) code.

It is worth mentioning that several authors have studied codes immune to a deletion or an insertion of a bit. For example, the so-called Varshamov-Tenengolts code proposed in [16] and popularized by Levenshtein in [9] has been further studied by Ferreira et al., [5], Levenshtein [10], Sloane [13], and Tenengolts [14]. Related constructions were proposed in [1], [2], [3], [7] and [15]. Even though these constructions assure that the code is immune to a deletion or an insertion of a bit, they do not guarantee any other desirable properties of standard substitution error correcting codes (such as linearity and a good minimum Hamming distance) and are therefore not known to be used in practice.

To address this issue, we instead consider a Reed-Muller(1,m) code known to have good substitution error correcting properties. We analyze the identification problem for this code, and propose a pruning technique to eliminate this problem. The motivation for this approach is that, with a small loss in the overall rate, and a small additional overhead needed to extract the appropriate subcode (e.g. this processing could precede the encoding block and would need to be done only once), a subcode of a code already used to protect the data from additive noise would also be immune to synchronization errors.

In order to determine the codewords that cause the identification problem, the code must be analyzed in a systematic manner. For this purpose, we express the codewords in terms of their runs from which a series of structural properties of a Reed-Muller(1,m) code follow. These useful properties are presented in Section 2, and are used in Section 3 for the analysis of the identification problem under one synchronization error.

2 Structural Properties of RM(1,m)

First order Reed-Muller codes (RM(1,m)) are an instance of linear substitution error-correcting codes. They are known for their good minimum distance, simple encoding, and relatively easy maximum likelihood decoding. On the negative side, they have low rate.

This code is described by a $k \times n$ generator matrix $\mathbf{G_m}$ where $k = m + 1$ and $n = 2^m$. Let $\mathbf{g_0}(\mathbf{m})$ denote the all-ones vector of length 2^m, and let $\mathbf{g_1}(\mathbf{m})$ be an $m \times 2^m$ matrix whose columns are binary m-tuples in the decreasing order.

The generator matrix of the RM(1,m) code is then

$$\mathbf{G_m} = \begin{bmatrix} \mathbf{g_0(m)} \\ \mathbf{g_1(m)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \dots & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & 1 & \dots & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & \dots & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & \dots & 1 & 0 & 1 & 0 \end{bmatrix}. \tag{3}$$

Observe that the first row of $\mathbf{G_m}$ consists of all ones, and the ith row of $\mathbf{G_m}$, for $1 < i \leq m+1$ consists of 2^{i-1} alternating runs of ones and zeros, where each run is of size 2^{m-i+1}, and the leftmost run in each row is a run of ones. Let $C(m)$ denote the RM(1,m) code. Note that every codeword in $C(m+1)$ is either the concatenation of a codeword in $C(m)$ with itself or with its bitwise complement.

Definition 2.1 (of G_j^m) *The set of all codewords of $C(m)$ can be partitioned into groups, such that the codewords belonging to the same group have the same number of runs of ones. Let G_j^m denote the group in which all codewords have exactly j runs of ones.*

Definition 2.2 (of $c_j^m(10)$, $c_j^m(01)$, $c_j^m(11)$ and $c_j^m(00)$) *Let $c_j^m(10)$ denote codewords in $C(m)$ that have j runs of ones, start with a '1' and end with a '0'. Similarly, let $c_j^m(11)$, $c_j^m(00)$, and $c_j^m(01)$ denote codewords in $C(m)$ that have j runs of ones, and start with '1','0', and '0', respectively, and end with '1','0', and '1', respectively.*

We are now ready to prove several structural properties of the RM(1,m) code.

Lemma 1 *For $C(m)$, $m \geq 1$, there are $2^{m-1}+1$ distinct non-empty groups G_j^m, for $0 \leq j \leq 2^{m-1}$. There are 4 distinct codewords in each group G_j^m, for $0 < j < 2^{m-1}$, and there are 3 distinct codewords in the group $G_{2^{m-1}}^m$. There is one codeword in the group G_0^m, namely the all-zero codeword. No two codewords in each group agree in both the first and last bit. In $G_{2^{m-1}}^m$ no codeword both starts and ends in a '0'.*

Proof : The proof is by induction on m. For $m = 1$ the statement can be verified by inspection. Suppose this assertion holds for an arbitrary $m = m_0$. Then all codewords in the same group $G_j^{m_0}$ have different starting and ending bits, so we use $c_j^{m_0}(11)$, $c_j^{m_0}(01)$, $c_j^{m_0}(10)$, and $c_j^{m_0}(00)$ as a shorthand for these codewords. Out of eight possible concatenations of codewords in $G_j^{m_0}$ for $1 \leq j < 2^{m_0-1}$, 3 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 $[c_j^{m_0}(01)|\overline{c_j^{m_0}(01)}]$), 4 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 1 results in the codeword $[c_j^{m_0}(00)|\overline{c_j^{m_0}(00)}]$ in $G_{2j+1}^{m_0+1}$ (here the overline denotes complementation and | sign denotes concatenation). By varying j from 1 to $2^{m_0-1}-1$, inclusive, we

can describe 3 codewords in $G_1^{m_0+1}$, 4 codewords in each $G_{j'}^{m_0+1}$ for $2 \leq j' \leq 2^{m_0} - 2$ and 1 codeword in $G_{2^{m_0}-1}^{m_0+1}$ such that no two codewords that belong to the same group agree in both the first and the last bit. While preserving the condition that the codewords in $C(m_0+1)$ in the same group have distinct outermost bits, the remaining 8 codewords in $C(m_0+1)$ are obtained by concatenating the three codewords in $G_{2^{m_0-1}}^{m_0}$, neither of which has both the first and the last bit equal to '0', each with itself and with its complement (thereby completing the groups $G_{2^{m_0}-1}^{m_0+1}$ and $G_{2^{m_0}}^{m_0+1}$), and by concatenating the all-zero codeword $c_0^{m_0}(00)$ with itself and with $c_1^{m_0}(11)$ (completing the groups $G_0^{m_0+1}$ and $G_1^{m_0+1}$). ■

Now we know that $c_j^m(xy)$ for $x, y \in \{0, 1\}$ represents at most one codeword, so from now on we will use $c_j^m(xy)$ as a shorthand for the codeword in $C(m)$, if it exists, that has j runs of ones, starts with x and ends with y for $x, y \in \{0, 1\}$.

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

Proof : Let us collect the codewords across different groups G_j^m as follows. The complement codewords $c_{j-1}^m(00)$ and $c_j^m(11)$ both have $2j - 1$ runs, and complements $c_j^m(10)$ and $c_j^m(01)$ both have $2j$ runs. Letting j vary from 1 to 2^{m-1} and recalling that except for G_0^m which contains only $c_0^m(00)$, and $G_{2^{m-1}}^m$ which contains only $c_{2^{m-1}}^m(11)$, $c_{2^{m-1}}^m(10)$ and $c_{2^{m-1}}^m(01)$, all other G_j^m contain all four of $c_j^m(10)$, $c_j^m(01)$, $c_j^m(00)$, and $c_j^m(11)$, we arrive at the proposed result. ■

Lemma 3 *Consider a codeword c in $C(m)$. The runs in c are of at most two different sizes, and these two sizes are consecutive powers of 2. In addition, if there are runs of two different sizes in c, the outer runs (i.e. the leftmost run and the rightmost run) in c are of the smaller size.*

Proof : (Outline) The proof is by induction on m. For small values of m the statement can be verified directly. Suppose now that the given statement is true for an arbitrary $m = m_0$. For a codeword c in $C(m_0)$ let c' and c'' denote the codewords in $C(m_0+1)$ that are the concatenation of c with itself, and with its complement, respectively. By separately analyzing the cases when c has one and two different run-sizes, and distinguishing the sub-cases when c has the same outermost bits, and when these two bits are different, the statement follows. For a complete proof please see [4]. ■

Lemma 4 *With the exception of the all-ones codeword, all codewords belonging to the group G_j^m for $2^{p-1} < j \leq 2^p$ for some p, $0 \leq p \leq m-1$ have all runs of ones either of length 2^{m-p-1} or of length 2^{m-p}. Moreover, $(j - 2^{p-1}) \times 2$ runs out of these j runs have length 2^{m-p-1}, and the remaining $2^p - j$ runs have length 2^{m-p}.*

In particular for j a complete power of 2 the Lemma says the following: All codewords belonging to the group G_j^m for $j = 2^p$ for some p, $0 < p \leq m - 1$ have all runs of ones of length

2^{m-p-1} each. If $j = 2^p$ and $p = 0$ all but one codeword, namely the all-ones codeword, have the single run of ones of length 2^{m-p-1}.

Proof : (Outline) To prove the statement we use induction on m. For small values of m, the proposed statement can be verified directly. Suppose now that the assertion holds for some $m = m_0$. We analyze the concatenations of the codewords that belong to the same group $G_j^{m_0}$, $0 \leq j \leq 2^{m_0-1}$ each with itself and with its complement. By Lemma 1 there are at most 8 such concatenations, namely, $c_j^m(11)$ with itself and with $c_{j-1}^m(00)$, $c_j^m(00)$ with itself and with $c_{j+1}^m(11)$, $c_j^m(10)$ with itself and with $c_j^m(01)$, and $c_j^m(01)$ with itself and with $c_j^m(10)$. We treat the cases $j = 0$, $j = 1$ and $j > 1$ separately. For the case $j > 1$ we rely on the observation proved in Lemma 3. In particular, in analyzing the concatenations of $c_j^m(11)$ with itself and of $c_j^m(00)$ with its complement, we distinguish the subcases $j = 2^p$ and $2^{p-1} < j < 2^p$. Similarly, for the concatenation of $c_j^m(11)$ with its complement we analyze the subcases $j - 1 = 2^{p-1}$ and $j - 1 > 2^{p-1}$ separately. By combining the results for each concatenation, the lemma follows. For a detailed proof please see [4]. ■

We now address the identification problem for this code, while making use of the structural properties established so far.

3 RM(1,m) in Channels With Synchronization Errors

Before proceeding with the analysis of the RM(1,m) code under at most one repetition or deletion, let us first introduce a useful general transformation in which we express the number of runs of the original codeword in terms of the weight of a string in the transformed domain. Suppose G is a $k \times n$ generator matrix of some code $C(n, k)$.

Let $\tilde{G} = GT_n$ and $\tilde{C}$ be the code generated by $\tilde{G}$, where T_n is a $n \times (n-1)$ matrix satisfying

$$T_n(i,j) = \begin{cases} 1, & \text{if } i = j \\ 1, & \text{if } i = j+1 \\ 0, & \text{else.} \end{cases} \tag{4}$$

Remark 3.1 *If $\mathbf{c} = \mathbf{m}G \in C$ has r runs, then $\tilde{\mathbf{c}} = \mathbf{m}\tilde{G}$ has weight $r-1$, and vice versa. Both $\mathbf{c}$ and its complement $\overline{\mathbf{c}}$ (if it exists) result in the same $\tilde{\mathbf{c}}$.*

In essence, a repetition in $\mathbf{c} \in C$ corresponds to an insertion of a zero in its counterpart $\tilde{\mathbf{c}} \in \tilde{C}$. Binary codes immune to insertions of the same kind of bit were studied in [8].

Since no new runs are created when a codeword experiences a repetition, for two codewords $\mathbf{c_a}$ and $\mathbf{c_b}$ in C to result in the same sequence when they both experience such synchronization error, it is necessary that their counterparts $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ in $\tilde{C}$ have the same weight.

However, during a deletion, the total number of runs in the codeword can stay the same or it can decrease by one or by two. Suppose a codeword $\mathbf{c_a}$ experiences a deletion in a run

of size at least 2. Then, $\tilde{\mathbf{c}}_\mathbf{a}$ will experience a deletion of a zero, so its weight will remain unchanged. If $\mathbf{c}_\mathbf{a}$ experiences a deletion in a run of size 1, the neighboring runs (if any) will merge and the total number of runs will decrease. In particular, if this deleted run of size 1 is an outermost run, the total number of runs decreases by 1, and $\tilde{\mathbf{c}}_\mathbf{a}$ experiences a deletion of an outermost '1'. If this deleted run of size 1 is located somewhere else in $\mathbf{c}_\mathbf{a}$, the total number of runs decreases by 2, and a string '11' in $\tilde{\mathbf{c}}_\mathbf{a}$ is replaced by a '0'. Therefore, for two codewords $\mathbf{c}_\mathbf{a}$ and $\mathbf{c}_\mathbf{b}$ to result in the same sequence when they both experience one deletion, it is necessary that their counterparts $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ in $\tilde{C}$ differ in weight by at most 2.

In particular, when the original code C is a Reed-Muller(1,m) code, the following symmetry property holds for $\tilde{C}$.

Lemma 5 *Suppose $\mathbf{G_m}$ is a generator matrix of $C(m)$. Let $\tilde{C}(m)$ be the code generated by $\mathbf{G_m}T_{2^m}$, where T_{2^m} is given by Eq.(4). Then, in $\tilde{C}(m)$, all codewords are mirror-symmetric, i.e. $\forall\, \tilde{\mathbf{c}} = (\tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_{2^m-1}) \in \tilde{C}(m)$, $\tilde{c}_i = \tilde{c}_{2^m-i}$ for $1 \leq i \leq 2^m - 1$.*

$\mathbf{Proof}$: The proof is by induction on m. By inspection we can conclude that the statement is true for $m = 0, 1$ and 2. Suppose the statement is true for $m = m_0$. We wish to show that it is also true for $m = m_0+1$. Recall from the previous section that a string is a codeword in $C(m_0+1)$ if and only if it is obtained by concatenating a codeword in $C(m_0)$ either with itself or with its complement. Consider a codeword $\mathbf{c_0} \in C(m_0)$. Let $\tilde{\mathbf{c}}_\mathbf{0} = \mathbf{c_0}T_{2^{m_0}} = \overline{\mathbf{c_0}}T_{2^{m_0}} \in \tilde{C}(m_0)$ (the overline denotes the complement codeword). By assumption, $\tilde{\mathbf{c}}_\mathbf{0}$ is mirror-symmetric. Consider two codewords $\mathbf{c_{00}}$= $[\mathbf{c_0}|\mathbf{c_0}]$ and $\mathbf{c_{01}}$=$[\mathbf{c_0}|\overline{\mathbf{c_0}}]$ that belong to $C(m_0+1)$ (vertical bars denote concatenation). Now, $\mathbf{c_{00}}T_{2^{m_0+1}} = [\tilde{\mathbf{c}}_\mathbf{0}|x|\tilde{\mathbf{c}}_\mathbf{0}]$, and $\mathbf{c_{01}}T_{2^{m_0+1}} = [\tilde{\mathbf{c}}_\mathbf{0}|\overline{x}|\tilde{\mathbf{c}}_\mathbf{0}]$, where x is 0 or 1 depending on whether the last run and the first run in $\mathbf{c_0}$ are of the same type. In either way, in both $\mathbf{c_{00}}T_{2^{m_0+1}}$ and $\mathbf{c_{01}}T_{2^{m_0+1}}$, by assumption, the bit in position p is the same as the bit in position $2^{m_0} - p$ for $1 \leq p \leq 2^{m_0} - 1$, which is in turn equal to the bit in position $2^{m_0} - p + 2^{m_0} = 2^{m_0+1} - p$. ■

We now state the main results regarding the existence of the identification problem in a Reed-Muller(1,m) code under one synchronization error.

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

$\mathbf{Proof}$: For the case of one repetition (or any number of repetitions for that matter) 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. ■

Theorem 2 *When codewords in $C(m)$ experience a deletion the following ones can result in the same string and thereby cause the identification problem:*

$m = 0$ *The only codewords are '0' and '1' and they can both result in an empty string.*

$m = 1$ *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 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.*

$m \geq 3$ *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: $c_j^m(10)$ and $c_j^m(01)$, $c_j^m(10)$ and $c_j^m(11)$, $c_j^m(10)$ and $c_{j-1}^m(00)$, $c_j^m(01)$ and $c_j^m(11)$, $c_j^m(01)$ and $c_{j-1}^m(00)$, $c_j^m(01)$ and $c_{j-1}^m(01)$, $c_j^m(10)$ and $c_{j-1}^m(10)$, $c_k^m(01)$ and $c_k^m(00)$, $c_k^m(01)$ and $c_{k+1}^m(11)$, $c_k^m(10)$ and $c_k^m(00)$, $c_k^m(10)$ and $c_{k+1}^m(11)$, where $j = 2^{m-1}$ and $k = 2^{m-2}$.*

Proof : (Outline) For values $m = 0, 1, 2,$ the statement can be verified directly. Suppose $\mathbf{c_a}$ and $\mathbf{c_b}$ are distinct codewords in $C(m)$ for $m \geq 3$, which result in the same string when each experiences one deletion. Let $\tilde{\mathbf{c}}_\mathbf{a} = \mathbf{c_a} T_{2^m}$ and $\tilde{\mathbf{c}}_\mathbf{b} = \mathbf{c_b} T_{2^m}$, where T_{2^m} is given by Eq.(4). Without loss of generality assume that Weight($\tilde{\mathbf{c}}_\mathbf{a}$) $\geq$ Weight($\tilde{\mathbf{c}}_\mathbf{b}$). We treat the cases Weight($\tilde{\mathbf{c}}_\mathbf{a}$)=Weight($\tilde{\mathbf{c}}_\mathbf{b}$), Weight($\tilde{\mathbf{c}}_\mathbf{a}$)=Weight($\tilde{\mathbf{c}}_\mathbf{b}$)+1, and Weight($\tilde{\mathbf{c}}_\mathbf{a}$)=Weight($\tilde{\mathbf{c}}_\mathbf{b}$)+2 separately. In the first case, it can be shown by using Lemmas 2 and 4 that $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ must be equal. It is further necessary that both $\mathbf{c_a}$ and $\mathbf{c_b}$ experience deletions in the outermost runs of size 1. By combining weight properties of a Reed-Muller code ([11]) and the results of Lemmas 3 and 4 we conclude that the only choice for $\mathbf{c_a}$ and $\mathbf{c_b}$ is $c_{2^{m-1}}^m(10)$ and $c_{2^{m-1}}^m(01)$.

When Weight($\tilde{\mathbf{c}}_\mathbf{a}$)=Weight($\tilde{\mathbf{c}}_\mathbf{b}$)+1, we consider the cases of even and odd weight of $\tilde{\mathbf{c}}_\mathbf{a}$ separately. In the former case, by using Lemmas 2, 4 and the mirror symmetry of $\tilde{C}(m)$ (Lemma 5), it can be shown that $\tilde{\mathbf{c}}_\mathbf{a}$ must be an all-ones string whereas $\tilde{\mathbf{c}}_\mathbf{b}$ has all but the middle bit equal to 1. It can be easily verified that all four choices for the pair $\mathbf{c_a}$ and $\mathbf{c_b}$ are possible, namely, $c_{2^{m-1}}^m(10)$ and $c_{2^{m-1}}^m(11)$, or $c_{2^{m-1}}^m(10)$ and $c_{2^{m-1}-1}^m(00)$, or $c_{2^{m-1}}^m(01)$ and $c_{2^{m-1}}^m(11)$, or $c_{2^{m-1}}^m(01)$ and $c_{2^{m-1}-1}^m(00)$. In the case when the weight of $\tilde{\mathbf{c}}_\mathbf{a}$ is odd, it can be similarly shown that the strings $\tilde{\mathbf{c}}_\mathbf{a}$ and $\tilde{\mathbf{c}}_\mathbf{b}$ consist of alternating bits (former starting with a '1', latter with a '0'). Again, all four choices for $\mathbf{c_b}$ and $\mathbf{c_a}$, which are $c_{2^{m-2}}^m(01)$ and $c_{2^{m-2}}^m(00)$, $c_{2^{m-2}}^m(01)$ and $c_{2^{m-2}+1}^m(11)$, $c_{2^{m-2}}^m(10)$ and $c_{2^{m-2}}^m(00)$, and $c_{2^{m-2}}^m(10)$ and $c_{2^{m-2}+1}^m(11)$, are possible.

Finally, for Weight($\tilde{\mathbf{c}}_\mathbf{a}$)=Weight($\tilde{\mathbf{c}}_\mathbf{b}$)+2, by using the structural properties of $C(m)$ established in Lemmas 3, 4 and 5, one can show that $\tilde{\mathbf{c}}_\mathbf{a}$ is the all-ones string whereas $\tilde{\mathbf{c}}_\mathbf{b}$ has all ones except in positions 2^{m-2} and $2^m - 2^{m-2}$. By inspection it can be shown that the only choices for $\mathbf{c_a}$ and $\mathbf{c_b}$ are $c_{2^{m-1}}^m(10)$ and $c_{2^{m-1}-1}^m(10)$, or $c_{2^{m-1}}^m(01)$ and $c_{2^{m-1}-1}^m(01)$. For a detailed proof please see [4]. ■

We now wish to find the largest linear subcode of $C(m)$, in which no two codewords cause the identification problem under one deletion. From the previous theorem, we know that

the generator matrix of such subcode has 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 $\mathbf{G_m}$. This matrix has m rows and it can be shown that no linear combinations of its rows give rise to codewords causing the identification problem. We are therefore able to eliminate the identification problem while preserving the linearity of the code and suffering a very small loss in the overall rate.

4 Conclusion and Future Work

In this paper we studied the performance of a Reed-Muller(1,m) code, as an instance of a substitution error-correcting code, in channels in which a sampling error causes a repetition or a deletion of a bit. This error can occur in the absence of adequate timing recovery in a variety of applications, for example in magnetic recording and in wireless transmission. The advantage of analyzing a good substitution error correcting code under varying sampling rate over designing a brand new code immune to synchronization errors is that the resulting subcode would both provide protection against additive noise and would be robust to sampling errors. The additional processing needed to extract the subcode and the rate loss incurred from using the subcode should be made reasonably small.

When a bit is repeated or deleted, two distinct codewords (even in the absence of noise) could give rise to the same string, which is known as the identification problem. Having established several useful structural properties of a Reed-Muller(1,m) code, we are able to conclude that no two codewords of this code cause the identification problem when they experience a repetition. Moreover, we are able to determine all pairs of codewords that can yield the same string when they experience a deletion. Based on the structure of the codewords causing the identification problem, we showed how one can modify a Reed-Muller(1,m) code so that the resulting linear subcode is immune to a single deletion error, while preserving good Hamming distance properties of the original code.

Although the structure of a Reed-Muller(1,m) code lent itself nicely to the analysis of the identification problem under one synchronization error, there are many other codes that have superior properties (e.g. higher rate and larger minimum distance). Future work would involve studying the behavior of other families of codes with good substitution error-correcting properties. The analysis should also be broadened to include a more general model in which several repetitions and deletions are allowed. In doing so, one would arrive at a (pruned) version of the code which would not only have good substitution error-correcting capabilities, but would also provide protection against a variety of sampling errors.

References

[1] P. A. H. Bours. ”Construction of fixed-length insertion/deletion correcting runlength-limited codes”. *IEEE Trans. Inform. Theory*, 40(6):1841–56, Nov. 1994.

[2] L. Calabi and W. E. Hartnett. ”A family of codes for the correction of substitution and synchronization errors”. *IEEE Trans. Inform. Theory*, 15(1):102–06, Jan. 1969.

[3] W. A. Clarke, A. S. J. Helberg, and H. C. Ferreira. ”Constrained codes for the correction of synchronization and additive errors”. In *Proceedings of the 1993 IEEE South African Symposium on Communications and Signal Processing. COMSIG*, New York, NY, USA, 1993.

[4] L. Dolecek. ”Run-Length Properties of A Reed-Muller RM(1,m) Code with Applications in Channels With at Most One Synchronization Error”. Master’s thesis, U.C. Berkeley, 2004. Available online at http://www.eecs.berkeley.edu/˜dolecek/Masters.pdf.

[5] H. C. Ferreira, W. A. Clarke, A. S. J. Helberg, K. A. S. Abdel-Ghaffar, and A. J. Han Vinck. ”Insertion/deletion correction with spectral nulls”. *IEEE Trans. Inform. Theory*, 43(2):722–32, Mar. 1997.

[6] X. Jin and A. Kavcic. ”Cycle-slip-detector-aided iterative timing recovery”. *IEEE Trans. On Magnetics*, 38(5):2292–94, Sept. 2002.

[7] T. Kløve. ”Codes correcting a single insertion/deletion of a zero or a single peak-shift”. *IEEE Trans. Inform. Theory*, 41(1):279–83, Jan. 1995.

[8] V. I. Levenshtein. ”Binary codes capable of correcting spurious insertions and deletions of ones”. *Problems of Information Transmission*, 1(1):8–17, Jan. 1965.

[9] V. I. Levenshtein. ”Binary codes capable of correcting deletions, insertions and reversals”. *Sov. Phys-.-Dokl.*, 10(8):707–10, Feb. 1966.

[10] V. I. Levenshtein. ”On perfect codes in deletion and insertion metric”. *Discrete Math. Appl.*, 2(3):241–58, 1992.

[11] F. J. MacWilliams and N. J. A. Sloane. *”The Theory Of Error-Correcting Codes”*. North Holland Publishing Company, Amsterdam, Holland, 1977.

[12] A. R. Nayak, J. R. Barry, and S. W. McLaughlin. ”Joint timing recovery and turbo equalization for coded partial response channels”. *IEEE Trans. On Magnetics*, 38(5):2295–97, Sept. 2002.

[13] N. J. A. Sloane. ”On single-deletion-correcting codes”. 2000. Available online at http://www.research.att.com/˜njas/doc/dijen.pdf.

[14] G. M. Tenengolts. ”Nonbinary codes correcting single deletion or insertion”. *IEEE Trans. Inform. Theory*, 30(5):766–69, Sept. 1984.

[15] J. D. Ullman. ”Near-optimal, single-synchronization-error-correcting code”. *IEEE Trans. Inform. Theory*, 12(4):418–24, Oct. 1966.

[16] R. R. Varshamov and G.M. Tenengolts. ”Codes which correct single asymmetric errors”. *Avtomatika i Telemekhanika*, 26(2):288–92, 1965.