Application of Symbol Avoidance in Reed-Solomon Codes to Improve their Synchronization

Thokozani Shongwe
Department of Electrical
and Electronic Engineering Science,
University of Johannesburg,
P.O. Box 524, Auckland Park, 2006,
Johannesburg, South Africa
Email: tshongwe@uj.ac.za

A. J. Han Vinck
University of Duisburg-Essen,
Institute for Experimental Mathematics,
Ellernstr. 29, 45326 Essen, Germany
E-mail: vinck@iem.uni-due.de

Hendrik C. Ferreira
Department of Electrical
and Electronic Engineering Science,
University of Johannesburg,
P.O. Box 524, Auckland Park, 2006,
Johannesburg, South Africa
Email: hcferreira@uj.ac.za

***Abstract*—In our previous work we introduced a method for avoiding/excluding some symbols in Reed-Solomon (RS) codes, called *symbol avoidance*. In this paper, we apply the symbol avoidance method to make synchronization of RS encoded data more effective. We avoid symbols in a RS code and then perform conventional frame synchronization on RS encoded data by appending sync-words on the data. The symbols in the RS code are avoided according to the sync-word used, such that the sync-word has very low probability of being found in the RS codewords, where it was not inserted. Therefore, for different sync-words, different symbols need to be avoided in the RS code. The goal here is to reduce the probability of mistaking data for the sync-word in the RS encoded framed data. Hence, the probability of successful synchronization is improved. Not only does our symbol avoidance code improve probability of successful synchronization, it also reduces the overall amount of redundancy required when the channel is very noisy.**

***Index Terms*—Reed-Solomon codes, Frame Synchronization, Sync-words.**

I. Introduction

The conventional frame synchronization technique, employing sync-words to delimit the beginning or end of a frame, is widely deployed in digital systems and has been shown to be a more practical solution to synchronization compared to synchronizable block code techniques like comma-free codes [1] [2] [3] [4], prefix codes [5] [6] [7] [8], comma codes [9] [10]. In conventional frame synchronization, performance is optimized in two ways. Firstly, to design good sync-words with good aperiodic auto-correlation functions (ACFs), that is the sync-word gives a high value only when there is a perfect match. Secondly, to reduce the probability of data being mistaken for the sync-word, by using longer sync-words. However, there is a problem with using long sync-words in channels with high probability of error. Long sync-words are more susceptible to channel errors compared to short ones. There is therefore a need to find ways of frame synchronization that can reap the benefits of reduced probability of mistaking data for a sync-word without significantly increasing the lengths of the sync-words.

In our previous work [11], we developed a method for avoiding particular symbols in a given RS code, such that the new code does not have any of the symbols we avoided. We called the method for avoiding symbols in a RS code, *Symbol Avoidance*. Symbol avoidance falls under a small field of coding theory called *codes with constraints*, as discussed in [12, Chapter 7]. In this paper, we use the symbol avoidance method to avoid symbols in a RS code in order to improve synchronization when RS encoding is used in transmission. This work can be seen as a continuation of our previous work in [13], where we only focused on reducing the probability of sync-word false acquisition in Reed-Solomon (RS) codes. In this work we will again use RS codes together with well known sync-words, modifying the RS codes by avoiding some symbols such that the probability of mistaking data for the sync-words is reduced. We will give results on the probability of synchronization (correctly locating the sync-word) for RS codes. We will also show that with our synchronization method, the overall performance of synchronization will be improved without significantly increasing the lengths of the sync-words.

Our method of synchronization of RS codes using symbol avoidance and sync-words actually borrows from the ideas of synchronizable block codes (prefix codes and comma codes) and frame synchronization with sync-words. Prefix codes and comma codes require that the sync-word (prefix or comma) should not appear in the codewords or in overlaps of codewords, which implies that the probability of mistaking data for a sync-word is zero in the absence of noise. In our synchronization method, we relax this requirement imposed on prefix and comma codes, that is the zero probability of mistaking data for a sync-word. We do this by using symbol avoidance to reduce the probability of mistaking data for a sync-word such that it approaches zero in the absence of noise. In some cases, the symbol avoidance method achieves the zero probability of mistaking data for the sync-word, for individual codewords (not overlapping codewords). In addition to reducing the probability of mistaking data for a sync-word, applying symbol avoidance also reduces the overall required redundancy when the channel is very noisy.

II. Preliminaries

This section serves to facilitate our discussion of synchronization of RS encoded data, with symbol avoidance applied and using sync-words, by providing the necessary "tools" to aid the discussion.

A. Aperiodic Auto-correlation of Sequences

Since the pioneering work by Barker [14] in 1953 on frame synchronization, most sequences that were used as sync-words were designed based on aperiodic auto-correlation, and some of these sequences are Barker [14], Turyn [15], Willard [16] and Maury and Styles [17] sequences. These sequences have "good" aperiodic Auto-Correlation Functions (ACFs) which makes them "optimal" sync-words. By good aperiodic ACF here we mean that the sequence has an aperiodic auto-correlation function that has a high *main lobe* that is, only when correlated with zero-shifted version of itself, and minimal *side lobes* with all other shifted versions of itself. Also, the sequence is considered optimal if the side lobes of its aperiodic ACF satisfy a particular required minimal value.

While considering the aperiodic ACF as the criterion for designing/searching for optimal sync-words is a good guideline, there are other ways of looking at the problem of optimizing synchronization. One such criterion was presented by Scholtz [18], which is consideration of the probability of false acquisition on data (probability of mistaking data for the sync-word), P_{FAD} and probability of true acquisition on the sync-word, P_{TAM}. Scholtz [18] went further to derive the lower bound on probability of acquisition, P_A as a function of P_{FAD} and P_{TAM}. The relationship between P_A, P_{FAD} and P_{TAM} is given by

$$P_{\text{A}} = (1 - P_{\text{FAD}})P_{\text{TAM}}.$$

B. Binary Symmetric Channel Frame Sync Algorithm

We give a *Binary Symmetric Channel (BSC) Frame Sync Algorithm* found in [18]. In the binary symmetric channel, an error results when one bit turns into another. We denote the channel bit error probability by P_e. Let S be the sync-word, of length N and, H be error tolerance. Now, let the received framed data consisting of N sync-word bits and D data bits, be $\mathcal{R}$. $\mathcal{R}$ is of length $N + D$; each bit position in $\mathcal{R}$ is indexed by t. Then the frame synchronization algorithm is given as follows.

BSC Frame Sync Algorithm

Input: $\mathcal{R}$, S, H, D
(a) set $t = 1$
(b) for $t = 1$ to $D - N + 1$
do a Hamming distance comparison
between S and $\mathcal{R}_t\mathcal{R}_{t+1}\ldots\mathcal{R}_{t+N-1}$
if $d_H(S, \mathcal{R}_t\mathcal{R}_{t+1}\ldots\mathcal{R}_{t+N-1}) \leq H$,
go to (c)
if $d_H(S, \mathcal{R}_t\mathcal{R}_{t+1}\ldots\mathcal{R}_{t+N-1}) > H$,
$t = t + 1$, go to (b)
(c) if $t = D - N + 1$ then output:
frame synchronization successful
else output:
frame synchronization fails

In summary, the BSC Frame Sync Algorithm given above is simply a "sliding window" method and Hamming distance comparison.

In practice, several frames are usually tested before a synchronization decision is taken. However, in this paper we are interested in the performance comparison between conventional RS codes and RS codes with symbol avoidance. This performance comparison can be captured with single-frame testing. Having stated that, the algorithm can be extended to test multiple frames.

In the simulations, the probability of acquisition will be the fraction of the number of successful frame synchronization events, as indicated in step (c) of the BSC Frame Sync Algorithm.

C. Symbol Avoidance in Reed-Solomon Codes

In our previous work [11], we presented a method for avoiding symbols in Reed-Solomon codes, which we called symbol avoidance. In this subsection, we briefly describe the symbol avoidance method.

Let us define a linear RS code as $(n,\ k,\ d)W$ over GF(q), where n is the length, k is the dimension, d is the minimum Hamming distance and q is the size of the field which is a power of a prime. From the linear RS code $(n,\ k,\ d)W$ we produce a new code $(n,\ k',\ d')W'$, of length n, dimension k' and minimum Hamming distance d', over an alphabet of size q'. We call the operation by which W' is obtained from W, symbol avoidance. This operation is given in simplified form as

$$(n,\ k,\ d)W \rightarrow \left\{ \begin{matrix} Symbol \\ Avoidance \\ Operation \end{matrix} \right\} \rightarrow (n,\ k',\ d')W',$$

where $d' \geq d$, $q' < q$, $k' < k$, and $(n,\ k',\ d')W'$ may be non-linear. $q' = q - |A|$, where A is a set of elements/symbols to be avoided in $(n,\ k',\ d')W'$. Next, we explain the symbol avoidance operation in detail.

The conventional systematic generator matrix of the RS code $(n,\ k,\ d)W$, $G = [I_k|P_{n-k}]$ with the symbols taken from a Galois field GF(q), is decomposed into two parts. The

first part of G, which is composed of k' rows of G, is denoted $G^{k'}$. The second part of G, which is composed of r rows of G such that $k = k' + r$, is denoted G^r.

$$G = \begin{bmatrix} I_{k'} & 0_r^{k'} & | \, P_{n-k}^{k'} \\ 0_{k'}^r & I_r & | \, P_{n-k}^r \end{bmatrix},$$

where $G^{k'} = [I_{k'} \; 0_r^{k'} | P_{n-k}^{k'}]$, $G^r = [0_{k'}^r \; I_r | P_{n-k}^r]$ and

$$P_{n-k}^{k'} = \begin{bmatrix} P_{11}^{k'} & P_{21}^{k'} & \dots & P_{1(n-k)}^{k'} \\ \vdots & \vdots & & \vdots \\ P_{k'1}^{k'} & P_{k'2}^{k'} & \dots & P_{k'(n-k)}^{k'} \end{bmatrix}, \tag{1}$$

$$P_{n-k}^{r} = \begin{bmatrix} P_{11}^{r} & P_{21}^{r} & \dots & P_{1(n-k)}^{r} \\ \vdots & \vdots & & \vdots \\ P_{r1}^{r} & P_{r2}^{r} & \dots & P_{r(n-k)}^{r} \end{bmatrix}. \tag{2}$$

$G^{k'}$ is used to encode a k'-tuple ($M = m_1 m_2 \dots m_{k'}$, where $m_i \in$ GF(q)), and this results in a codeword $C = MG^{k'}$. G^r encodes an r-tuple ($V = v_1 v_2 \dots v_r$, where $v_i \in$ GF(q)), resulting in what we shall call a *control vector* $R = VG^r$. The difference between C and R will be in their usage, otherwise they are both results of RS encoding. The control vectors (collection of the vectors R) are used to control the presence/absence of a particular symbol(s) in each codeword C, as will be demonstrated shortly. For a q-ary linear block code with a systematic generator matrix, *undesired symbols* in the codeword, due to the identity part, can be avoided by simply not including those symbols in the message to be encoded. However, for the parity part of the generator, a different method to avoid undesired symbols needs to be applied. It is therefore the main task of this section to show that we can avoid undesired symbols in a Reed-Solomon code while maintaining or improving its minimum Hamming distance even though the new code may be non-linear. Using control vectors, we control which symbols to avoid in the parity part of the RS code. C is the RS codeword and R is a control vector to be used on C, in case C has an undesired symbol. By undesired symbols we are referring to the symbols we do not want occurring in the code.

We now focus on the parity parts of C and R. We denote by $P_1^C, P_2^C, \dots, P_{n-k}^C$ and $P_1^R, P_2^R, \dots, P_{n-k}^R$ the parity symbols for C and R, respectively. Using (1), each parity symbol of a codeword, P_i^C, becomes $P_i^C = m_1 P_{1i}^{k'} + \dots + m_{k'} P_{k'i}^{k'}$, for $1 \le i \le n-k$. Using (2), each parity symbol of a control vector, P_j^R, becomes $P_j^R = v_1 P_{1j}^r + \dots + v_r P_{rj}^r$, for $1 \le j \le n-k$. Let us define a set $A = \{a_1, a_2, \dots, a_{|A|}\}$, which is a set of symbols taken from GF(q), where $|.|$ here indicates the cardinality of a set. The symbols in set A are the symbols we want to avoid in all codewords that resulted from the encoding by $G^{k'}$. If any $P_i^C \in A$ then we ought to eliminate/avoid that P_i^C using a corresponding P_j^R, where $j = i$. This is done by adding the corresponding parity parts of C and R such that the resulting parity symbols in the new codeword, P_i ($= P_i^C + P_i^R$), do not have any of the symbols in A. This procedure can be expressed mathematically as $P_i \neq a_x$, or $P_i^C + P_i^R \neq a_x$, for $1 \le x \le |A|$ and $1 \le i \le n-k$. Depending on their suitability in a sentence, the words eliminate and avoid will be used interchangeably since they convey the same message.

The next example illustrates the symbol avoidance procedure.

Note: when describing RS codes, we will be using integer symbols to represent elements of GF(q), because the integer symbols make it easier to follow operations on the RS codes and also aid presentation. See [11] for the relationship between elements of GF(q) and integer symbols.

Example 1: For this example, we take an ($n = 7$, $k = 3$) RS code over GF(2^3) with the generator G in (3). The field GF(2^3) is generated by a primitive polynomial, $p(I) = I^3 + I + 1$.

$$G = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 6 & 1 & 6 & 7 \\ 0 & 1 & 0 & 4 & 1 & 5 & 5 \\ 0 & 0 & 1 & 3 & 1 & 2 & 3 \end{array}\right]. \tag{3}$$

Then by choosing $r = 1$, we have $k' = 2$,

$$G^{k'} = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 6 & 1 & 6 & 7 \\ 0 & 1 & 0 & 4 & 1 & 5 & 5 \end{array}\right] \tag{4}$$

and

$$G^r = \left[\begin{array}{ccc|cccc} 0 & 0 & 1 & 3 & 1 & 2 & 3 \end{array}\right]. \tag{5}$$

Let 7 be the symbol to avoid in codewords encoded by $G^{k'}$ in (4), hence $A = \{7\}$. The G^r in (5) gives the following control vectors we can use when we encounter a codeword with symbol 7,

$$\left[\begin{array}{ccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 1 & 2 & 3 \\ 0 & 0 & 2 & 6 & 2 & 4 & 6 \\ 0 & 0 & 3 & 5 & 3 & 6 & 5 \\ 0 & 0 & 4 & 7 & 4 & 3 & 7 \\ 0 & 0 & 5 & 4 & 5 & 1 & 4 \\ 0 & 0 & 6 & 1 & 6 & 7 & 1 \\ 0 & 0 & 7 & 2 & 7 & 5 & 2 \end{array}\right]. \tag{6}$$

The last control vector, [0 0 7 | 2 7 5 2], is included for completeness, otherwise it cannot be used to eliminate symbol 7 since it has 7 in its information part. To illustrate the elimination of symbol 7, we pick one codeword with the symbol 7 (in the parity part) from the list of codewords produced by $G^{k'}$, and that codeword is $C = [0\ 3\ 0 \mid 7\ 3\ 4\ 4]$. From C, we have $P_1^C = 7$, $P_2^C = 3$, $P_3^C = 4$ and $P_4^C = 4$. Taking the control vector to use as $R = [0\ 0\ 1 \mid 3\ 1\ 2\ 3]$ from (6), we have $P_1^R = 3$, $P_2^R = 1$, $P_3^R = 2$ and $P_4^R = 3$. Adding the corresponding parity symbols, we have $P_1 = P_1^C + P_1^R = 4$, $P_2 = P_2^C + P_2^R = 2$, $P_3 = P_3^C + P_3^R = 6$ and $P_4 = P_4^C + P_4^R = 7$. The new codeword has $P_1 = 4$, $P_2 = 2$, $P_3 = 6$ and $P_4 = 7$, which still contains symbol 7, hence $R = [0\ 0\ 1 \mid 3\ 1\ 2\ 3]$ is not a suitable control vector. We repeat this procedure with every control vector in (6) until we find one that is suitable, and in this case it is $R = [0\ 0\ 3 \mid 5\ 3\ 6\ 5]$, which gives $P_1 = 2$, $P_2 = 0$, $P_3 = 2$ and $P_4 = 1$. It can also be verified that $R = [0\ 0\ 5 \mid 4\ 5\ 1\ 4]$, $[0\ 0\ 6 \mid 1\ 6\ 7\ 1]$ or $[0\ 0\ 2 \mid 6\ 2\ 4\ 6]$ is a suitable control vector.

We just showed in Example 1 that the key operation of the symbol avoidance procedure is about adding two codewords of the original RS code W to form a new codeword without the symbols in A. This new codeword then belongs to W' together with all other codewords without the symbols in A. Since W is a linear code, then it always holds that $W' \subseteq W$. According to the definition of linear codes, W' can be non-linear, but its minimum Hamming distance can be the same as that of W or even greater.

The symbol avoidance procedure steps can simply be presented by the following algorithm.

Symbol Avoidance Algorithm

Input: G, r, A
(a) split G into $G^{k'}$ and G^r, and form codebooks $C^{k'}$ and C^r, respectively
(b) set $i = 0$, $j = 1$
(c)
  (i) if j is less or equal to $|C^{k'}|$ and i is equal to $|C^r|$, go to (e)
  (ii) if j is greater than $|C^{k'}|$, go to (d)
  (iii) test if there are undesired symbols in the codeword, C_j
    if YES then $i = i + 1$, $C_j = C_j + R_i$, go to (c)
    if NO then $j = j + 1$, go to (c)
(d) output: $(n, k', d')W'$
(e) output: Symbol avoidance impossible.

In the algorithm, C_j are codewords in codebook $C^{k'}$, and R_i are control vectors in a collection of control vectors, C^r. The algorithm works with any size of the generator matrix, G as long as there is enough memory to hold the codewords of codebooks $C^{k'}$ and C^r.

It should be noted that our symbol avoidance method is mainly applicable to short RS codes. Short RS codes are discussed in [19] and [20], where RS codes of lengths 15 and 31 are considered. In [21], RS codes of length 32 and 64 are considered for OFDM systems.

Decoding with symbol avoidance

We now know that the code (n, k', d')W' over GF(q') is obtained by the symbol avoidance operation from the linear RS code (n, k, d)W over GF(q). The decoding algorithm of the RS code W is known and well defined. The code W' is easily decoded using the decoding algorithm of the code W, but with some "minor added operations" which enhance the performance of W'. To describe these minor added operations, let $D \in W'$, and $\tilde{D}$ be a received noise corrupted codeword version of D. We assume additive noise which changes one symbol into another within the GF(q). Note that $\tilde{D}$ can have undesired symbols, i.e. symbols that were originally avoided, due to noise corruption in the channel.

Decoding steps of $\tilde{D}$:

1) If the received codeword $\tilde{D}$ has undesired symbols, the undesired symbols are marked as erasures and then minimum distance decoding is performed.
2) If the minimum distance decoding results in a codeword not in W', an error is detected and the codeword is decoded to the nearest codeword in W'.

The code W' is highly likely to have better performance than the original RS code W because of one or a combination of the following:

- $d' \geq d$.
- In some cases, the weight enumerator of W' is improved compared to that of W. This is because some codewords of W are excluded in W'.

When the distance properties are not important, as is the case in the rest of the paper, we will refer to the RS codes as $(2^m - 1, k)$ RS code (or (n, k) RS code), where m is the number of bits per symbol, k is the dimension of the code and $n = 2^m - 1$ is the codeword length.

D. System Model

The system model for communication is shown in Figure 1.

At the transmitter, binary data is encoded using conventional Reed-Solomon codes, however, the generator matrix of the conventional RS code is split into two parts as described in the symbol avoidance algorithm and illustrated in Example 1. If a codeword in the encoded data contains undesired symbols, one of the control vectors is used to avoid those symbols. The symbols of a codeword (without the undesired symbols) are then converted to their binary equivalent, and a binary sync-word is appended to each codeword which is now in binary format. Before transmission, digital modulation is applied to the bits (codeword and sync-word bits). We assume that the transmitted data experiences additive white Gaussian noise (AWGN).

At the receiver, firstly, digital demodulation is performed to obtain the transmitted bits. Then the BSC frame sync algorithm is applied to search for the sync-word. The sync-word, if found is removed and the remaining bits (assumed to be RS codeword bits) are then converted to their equivalent RS symbols. The final step is decoding the noise corrupted RS codeword.

III. Probability of a Sync-word in a RS Code with Symbol Avoidance

When symbol avoidance is performed on the RS code, the symbol probability is no longer the same for all symbols. The probability of each symbol, in the RS code with symbol avoidance, can still be calculated by going through the entire codebook. However, even after getting the probability of each symbol, the symbols no longer behave as if they occur independently like in the case when there is no symbol avoidance done in the RS code. Therefore, it is quite difficult, if not impossible, to analytically find the probability of a sync-word in the RS code with symbol avoidance.

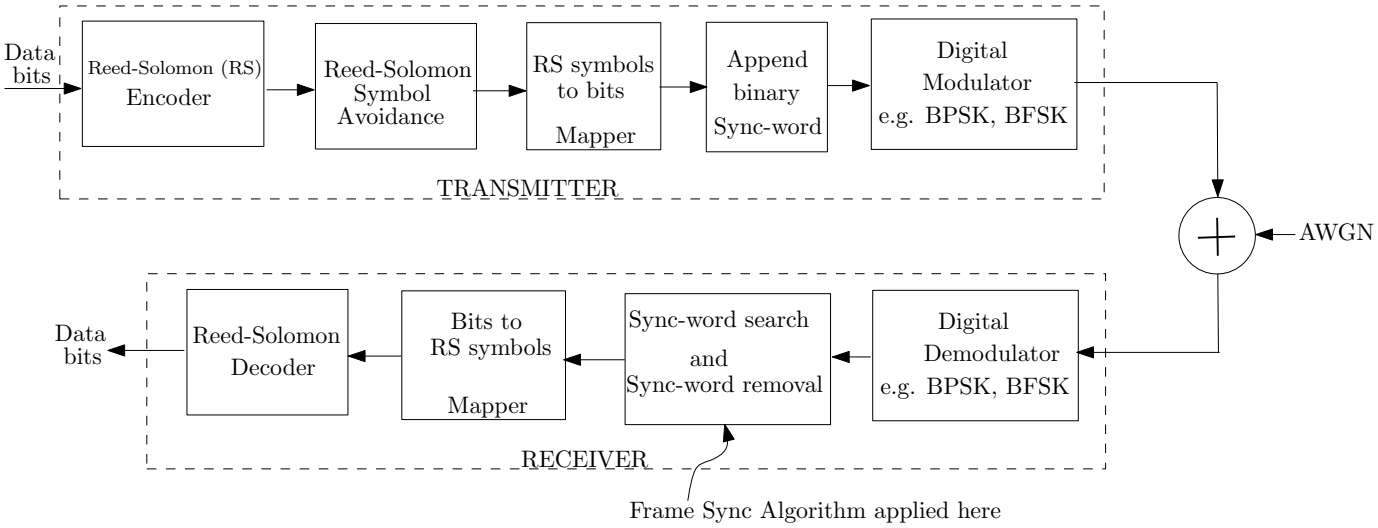


Fig. 1. System model for communication with Reed-Solomon symbol avoidance.

In this section, we present numerical results for the probabilities of sync-words in the codewords of the RS codes with symbol avoidance (W'), as well as the corresponding codes W for comparison purposes. We find the probability of the sync-word, of a given number of bits, occurring in a $(2^m-1,k)$ RS code, given symbols to avoid. These probabilities are then used to indicate which symbols are best to avoid in order to reduce the probability of false acquisition on data, P_{FAD}, for a given sync-word. We use the P_{FAD} of the RS code without symbol avoidance (W) as the benchmark for comparison against the P_{FAD} of W'. All results presented here are for binary sync-words by Maury and Styles [17]. Since the binary sync-words can be very long, for ease of presentation we will write them in their octal representation.

An exhaustive search through all the codewords of the RS code is performed for the sync-word *individually*, in the absence of noise. The codewords are the data, but the sync-word itself is not included at the end/beginning of each codeword because we are only interested in the probability of false acquisition on data. Searching the codewords individually for the sync-word means that codewords are treated independently, such that there is no overlap of bits from different codewords.

To better understand the results in Tables I–IV we have to first discuss the results in Tables A.1 and A.2, in the Appendix. Tables A.1 and A.2 in the Appendix show results for a $(2^3-1,3)$ RS code, for various binary sync-words with avoided symbols. For each subtable, we show the probabilities of a sync-word occurring in the RS code, under the column "Probability of False Acquisition", and the corresponding avoided symbol(s), under the column "Avoided Symbols". The rows are arranged in ascending order according to the "Probability of False Acquisition". The results were found by avoiding the symbol(s) in the second column and then searching for the sync-word to find its probability of being in the RS code. The limitation to the length of sync-words tested in the results was when all avoided symbols give the lower bound on the probability of false acquisition, $P_{\text{FAD}}=0$. Therefore all tables not included in the results are those where all the avoided symbols give $P_{\text{FAD}}=0$.

TABLE I
COMPARISON OF THE BEST P_{FAD} FOR W' AND P_{FAD} FOR W, FOR BINARY SYNC-WORDS OF LENGTHS $7-11$, WRITTEN IN THEIR OCTAL FORMAT. A $(2^3-1,3)$ RS CODE.

Binary Sync-words	Length, N	P_{FAD} for W	P_{FAD} for W'
130	7	7.81E-03	1.85E-03
270	8	3.91E-03	0
560	9	1.95E-03	0
1560	10	1.95E-03	0
2670	11	1.95E-03	0

In Table A.1 only one symbol is avoided for the corresponding RS codes, where $r=1$. We went further to avoid two symbols, where possible for $r=1$, in an attempt to achieve $P_{\text{FAD}}=0$ at shorter sync-word lengths. The results for two avoided symbols are shown in Table A.2.

Looking at Table A.1 and Table A.2 we can see that some avoided symbols begin to give the desired result of $P_{\text{FAD}}=0$ for sync-words of lengths 8. This means that avoiding one symbol is good enough to give us the desired result.

A major problem with finding these tables (Tables A.1 and A.2) is the amount of computer processing power needed to generate all the possibilities of avoided symbols, especially when avoiding more than one symbol. It is also not that practical to present all combinations of avoided symbols as this will result in very long tables. For example, for $m=6$, if $|A|=2$ (two symbols avoided) the table will have ${}^{2^m}C_{|A|}=2016$ rows. For most of our synchronization purposes, we only need to choose from the best performing avoided symbols (the ones at the top of the table). If the avoided symbols have the same P_{FAD}, any can be chosen. To reduce the lengths of very long tables, only a few of the avoided symbols giving the best performance are tabulated and presented. Now, when $|A|=2$

we use the avoided symbols giving the best performance from the results of $|A| = 1$ to form the tables for $|A| = 2$ as follows. From simulation results we found that it is enough to consider the top two symbols for the $|A| = 1$ results to make the table for $|A| = 2$, for the same RS code. Therefore, the shaded symbols in the subtables for $|A| = 2$ in Table A.2 are the top two symbols in the corresponding subtables for $|A| = 1$ in Table A.1.

TABLE II
COMPARISON OF THE BEST P_{FAD} FOR W' AND P_{FAD} FOR W, FOR BINARY SYNC-WORDS OF LENGTHS $7-11$, WRITTEN IN THEIR OCTAL FORMAT. A $(2^4-1,3)$ RS CODE.

Binary Sync-words	Length, N	P_{FAD} for W	P_{FAD} for W'
130	7	7.81E-03	4.35E-03
270	8	3.91E-03	2.12E-03
560	9	1.95E-03	1.96E-04
1560	10	9.77E-04	0
2670	11	4.88E-04	0

Table I gives a summary of the results of the P_{FAD} for the $(2^3-1,3)$ RS code with symbol avoidance (summary of Tables A.1 and A.2), compared with the P_{FAD} for the $(2^3-1,3)$ RS code without symbol avoidance. Table I gives an indication of the amount of improvement on the P_{FAD} in the code W' from the code W.

For other RS codes with symbol avoidance (W'), instead of showing the complete tables (for example, Tables A.1 and A.2), we only show the comparison of the best P_{FAD} for codes W' and the P_{FAD} for corresponding codes W, as shown in Table I. Such results are shown in Tables II, III and IV, for $(2^4-1,3)$ RS code, $(2^3-1,5)$ RS code and $(2^4-1,5)$ RS code, respectively.

The results in Tables II, III and IV also show the amount of improvement on the P_{FAD} in the code W' from the code W. For some cases, the improvement is so small that it is not worth the effort of employing symbol avoidance and the loss in efficiency (reduced code rate) incurred from W to W'.

IV. PROBABILITY OF ACQUISITION: RS CODE VS RS CODE WITH SYMBOL AVOIDANCE

In this section we use the sync-word in its bits representation form and have the RS code symbols represented in bit-format as well. The probability of acquisition is simulated using a BSC frame synchronization algorithm as described in Section II-B, with bit error probability, P_e. We will present performance comparison results for $(2^m-1,k)$ RS codes, where $3 \leq m \leq 6$ and $k = 3$.

The results in this section will confirm the notion that sync-words of short length, compared to those of longer lengths, have the problem of higher P_{FAD}, while having the benefits of higher P_{TAM} in the presence of noise. Longer sync-words bring the benefit of reducing P_{FAD}, however, they are more susceptible to noise corruption. The other obvious fact is that longer sync-words reduce transmission efficiency (increase redundancy).

TABLE III
COMPARISON OF THE BEST P_{FAD} FOR W' AND P_{FAD} FOR W, FOR BINARY SYNC-WORDS OF LENGTHS $7-16$, WRITTEN IN THEIR OCTAL FORMAT. A $(2^3-1,5)$ RS CODE.

Binary Sync-words	Length, N	P_{FAD} for W	P_{FAD} for W'
130	7	7.81E-03	3.40E-03
270	8	3.91E-03	1.10E-03
560	9	1.95E-03	0
1560	10	9.77E-04	0
2670	11	4.88E-04	0
6540	12	2.44E-04	0
16540	13	1.22E-04	0
34640	14	6.10E-05	0
73120	15	3.05E-05	0
165620	16	3.05E-05	0

TABLE IV
COMPARISON OF THE BEST P_{FAD} FOR W' AND P_{FAD} FOR W, FOR BINARY SYNC-WORDS OF LENGTHS $7-20$, WRITTEN IN THEIR OCTAL FORMAT. A $(2^4-1,5)$ RS CODE.

Binary Sync-words	Length, N	P_{FAD} for W	P_{FAD} for W'
130	7	7.81E-03	6.27E-03
270	8	3.91E-03	3.23E-03
560	9	1.95E-03	1.44E-03
1560	10	9.77E-04	7.36E-04
2670	11	4.88E-04	2.53E-04
6540	12	2.44E-04	1.21E-04
16540	13	1.22E-04	6.13E-05
34640	14	6.10E-05	2.90E-05
73120	15	3.05E-05	2.23E-05
165620	16	1.53E-05	9.61E-06
363240	17	7.63E-06	4.49E-06
746500	18	3.81E-06	4.59E-07
1746240	19	1.91E-06	4.70E-07
3557040	20	9.53E-07	0

We have shown that by employing symbol avoidance we can reduce P_{FAD} while keeping the sync-words short to benefit from their higher P_{TAM}, hence increasing the probability of acquisition P_A. One issue that immediately comes up with symbol avoidance in the RS code is the reduction in transmission efficiency of the RS code. However, this does not become an issue anymore when comparing the required lengths of the sync-words, for the RS codes with and without symbol avoidance, in light of the P_A performance. The RS code without symbol avoidance requires longer sync-words to achieve the same P_A as its equivalent RS code with symbol avoidance. In most cases, for higher probabilities of bit errors, the RS code without symbol avoidance always delivers inferior performance compared to the RS code with symbol avoidance.

We develop the following expressions for calculating and comparing the efficiencies of the RS codes with and without symbol avoidance, in terms of redundant bits. The redundancy, in bits, of a $(2^m-1,k)$ RS code is given by $R = m(2^m-1-k)$. We denote by R_X the redundancy of the $(2^m-1,k)$ RS code including an X-bit sync-word, such that $R_X = R + X$,

$$R_X = m(2^m-1) - mk + X. \tag{7}$$

For the same $(2^m-1,k)$ RS code with symbol avoidance, we denote the redundancy including a Y-bit sync-word by R_Y, such that

$$R_Y = m(2^m-1) - \lfloor (k-r)\log_2(2^m-|A|)\rfloor + Y. \quad (8)$$

Remember from Section II-C that r is the number of rows no longer used to encode information in the generator matrix of the RS code, and $|A|$ is the number of symbols to be avoided from the set of 2^m symbols.

Example 2: To demonstrate the redundancy comparison between RS codes with and without symbol avoidance, we use the results from Figure 2. A $(2^m-1,k)$ RS code with $m=3$, $k=3$, $r=1$ is used as an example. For the RS code with symbol avoidance, we use the sync-word $S=\{10111000\}$, of length $Y=8$ with $A=\{6\}$. At $P_e=3\times10^{-4}$, for the RS code without symbol avoidance, we need a sync-word of length $X\geq 12$ to approximately match the P_A performance of the RS code with symbol avoidance. Using $X=12$ and $Y=8$, the redundancies for both codes are then

$$\begin{aligned} R_X &= m(2^m-1)-mk+X \\ &= 3(2^3-1)-3\times3+12 \\ &= 24 \text{ bits}, \end{aligned}$$

and

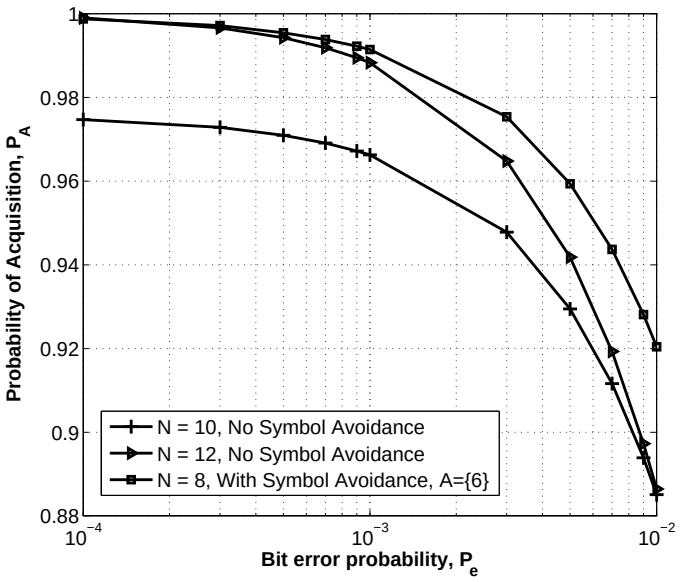
$$\begin{aligned} R_Y &= m(2^m-1) - \lfloor (k-r)\log_2(2^m-|A|)\rfloor + Y \\ &= 3(2^3-1) - \lfloor (3-1)\log_2(2^3-1)\rfloor + 8 \\ &= 24 \text{ bits}, \end{aligned}$$

which results in $R_Y = R_X$.

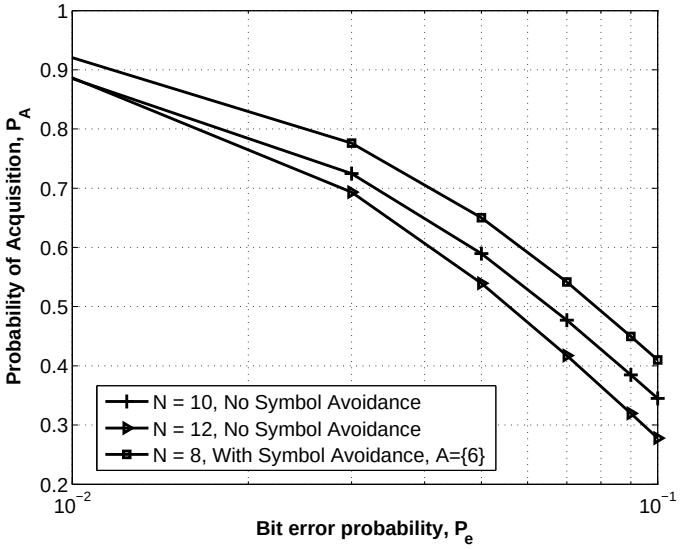
The channel bit error probability $P_e = 3\times10^{-4}$ was specifically chosen to demonstrate the case where the P_A performances of the RS code with and without symbol avoidance are roughly the same. Otherwise, for $P_e > 3\times10^{-4}$, the RS code without symbol avoidance never reaches the performance of the RS code with symbol avoidance, for $R_X = R_Y$.

Next, we present graphical results, in Figures 2–5, showing the performance comparison of RS codes with and without symbol avoidance. The simulation results in the figures take into account the number of data symbols used as well. N_C is the number of codewords, hence the number of data bits is $D = N_C(2^m-1)m$. The sync-word length is denoted by N. For all simulations we set $N_C = 1$, which means that a sync-word of length N is appended to every codeword to form a frame. We use $N_C = 1$ because we make use of the tabulated results in Section III, where the probability of false acquisition was per codeword. $N_C > 1$ can also be used if necessary. Sync-words of different lengths are used to indicate the amount of redundancy needed for the RS codes without symbol avoidance to achieve performance close to that of RS codes with symbol avoidance. The largest value of N for the RS code without symbol avoidance was chosen such that the amount of redundancy is the same as that of the RS code with symbol avoidance, $R_X = R_Y$.

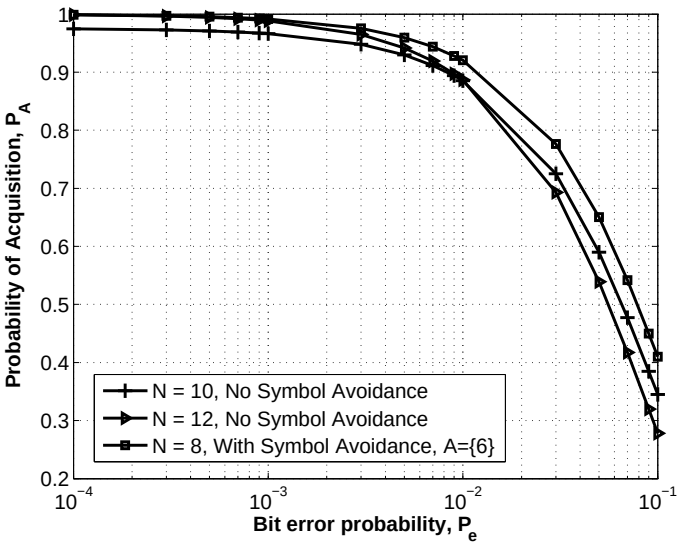
For the simulation results in Figure 2, we use $N=8$ and $A=\{6\}$ for the RS code with symbol avoidance because that



(a) Channel bit error probability, $10^{-4} \leq P_e \leq 10^{-2}$.

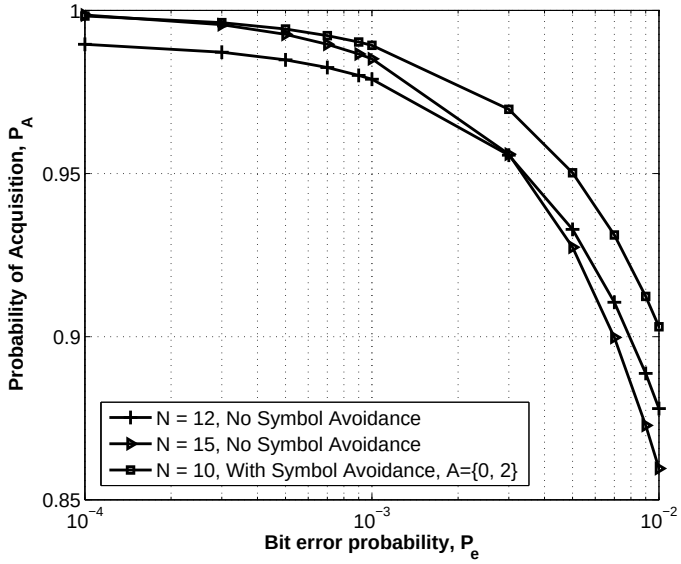


(b) Channel bit error probability, $10^{-2} \leq P_e \leq 10^{-1}$.

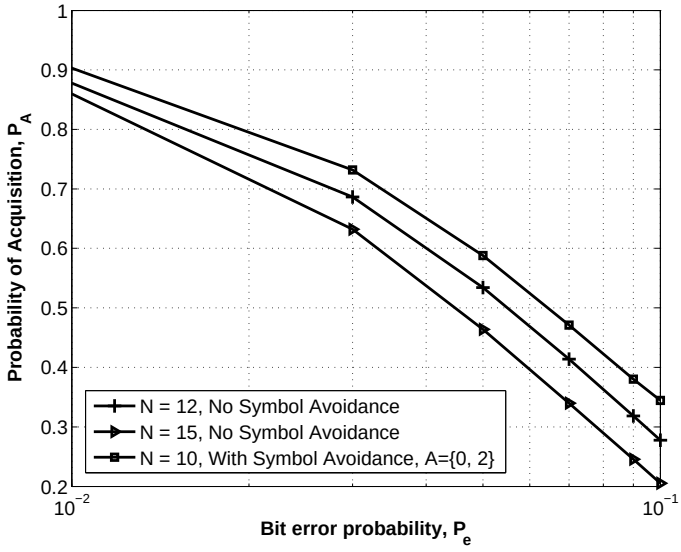


(c) Channel bit error probability, $10^{-4} \leq P_e \leq 10^{-1}$.

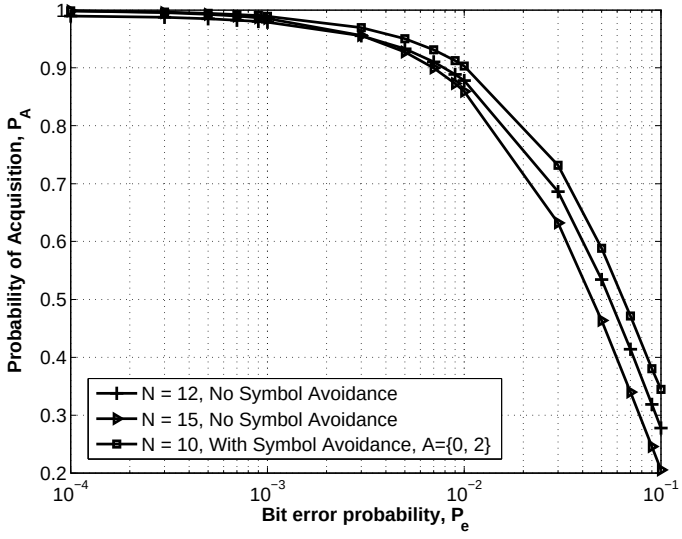
Fig. 2. Probability of acquisition comparison of binary sync-words with and without symbol avoidance applied on a $(2^3-1,3)$ RS code.



(a) Channel bit error probability, $10^{-4} \leq P_e \leq 10^{-2}$.



(b) Channel bit error probability, $10^{-2} \leq P_e \leq 10^{-1}$.



(c) Channel bit error probability, $10^{-4} \leq P_e \leq 10^{-1}$.

Fig. 3. Probability of acquisition comparison of binary sync-words with and without symbol avoidance applied on a $(2^4 - 1, 3)$ RS code.

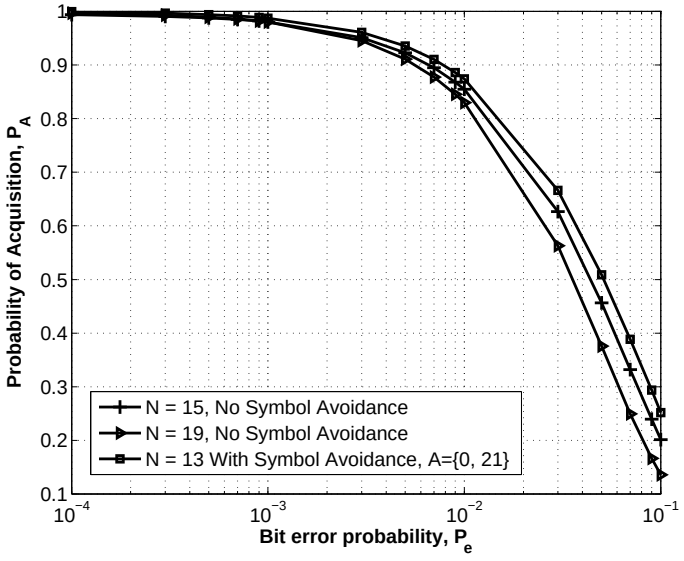


Fig. 4. Probability of acquisition comparison of binary sync-words with and without symbol avoidance applied on a $(2^5 - 1, 3)$ RS code. Channel bit error probability, $10^{-4} \leq P_e \leq 10^{-1}$.

gives $P_{\text{FAD}} = 0$ as indicated in Table A.1 (b). In Figure 2, looking at the performance of the RS code without symbol avoidance, one can see that the longer sync-word ($N = 12$) is better for low P_e than the shorter sync-word ($N = 10$). The $N = 10$ sync-word begins to give a better performance for $P_e > 10^{-2}$. The $N = 8$ sync-word for the RS code with symbol avoidance, where $A = \{6\}$, gives the best performance for all the values of P_e in the graph.

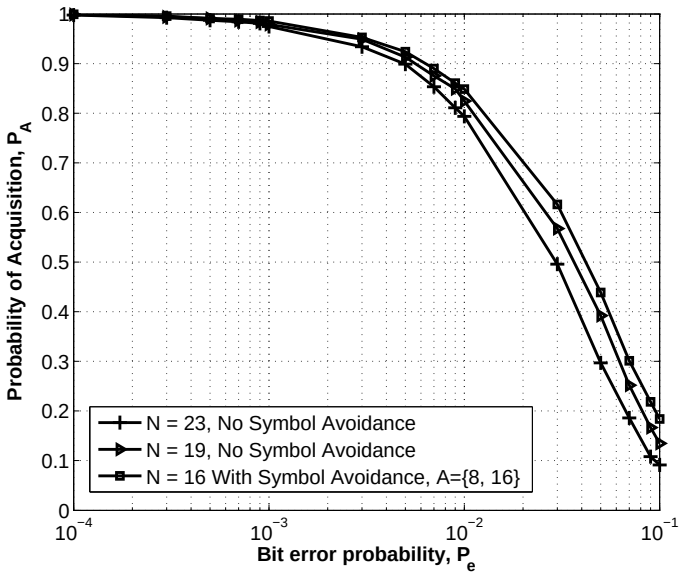


Fig. 5. Probability of acquisition comparison of binary sync-words with and without symbol avoidance applied on a $(2^6 - 1, 3)$ RS code. Channel bit error probability, $10^{-4} \leq P_e \leq 10^{-1}$.

For the simulation results in Figure 3, we use $N = 10$ and $A = \{0, 2\}$ for the RS code with symbol avoidance because

that gives $P_{\text{FAD}} = 0$. In Figure 3, looking at the performance of the RS code without symbol avoidance, one can see that the longer sync-word ($N = 15$) is better for low P_e than the shorter sync-word ($N = 12$). The $N = 12$ sync-word begins to give a better performance for $P_e > 3 \times 10^{-3}$. The $N = 10$ sync-word for the RS code with symbol avoidance, where $A = \{0, 2\}$, gives the best performance for all the values of P_e in the graph.

The performance trend observed in Figures 2 and 3 is also seen in Figures 4 and 5.

It is clear from Figures 2–5 that the RS code with symbol avoidance outperforms the RS code without symbol avoidance. Also, for most sync-word lengths, the RS code with symbol avoidance will perform better than the RS code without symbol avoidance at high channel bit (symbol) error probabilities. In each of the Figures 2–5, (7) and (8) can be used to verify that the longest sync-word used for the RS code without symbol avoidance results in the same amount of redundancy as the RS code with symbol avoidance. Increasing the length of the sync-word, for the RS code without symbol avoidance, beyond what we have in Figures 2–5 may result in a slightly better performance over the RS code with symbol avoidance. However, as the sync-words get longer, the performance continues to degrade for low P_e values, and can never match that of the RS code with symbol avoidance. Hence very long sync-words are not only inefficient (in terms of redundancy), but also result in poor performance at low P_e values. Therefore, the advantages of using RS codes with symbol avoidance (W') over the conventional RS codes (W) are: the codes W' require less redundancy than the codes W; in terms of probability of synchronization, the codes W' always outperform the codes W.

Concerning the amount of redundancy (efficiency), we need to view the efficiency of the codes in light of their probability of acquisition performance. This makes it difficult to exactly give a general number on the amount of redundancy for the codes because that is dependent on the required probability of acquisition. Despite this, we can safely state that the codes W' require less redundancy than the codes W. This is because when the sync-words for the codes W are made longer to try and match the good performance of the codes W', the performance of the codes W degrades even further at low P_e.

V. Conclusion

For Reed-Solomon codes with symbol avoidance (W'), we gave simulation results of the probability of false acquisition in a codeword. We also gave a performance comparison, in terms of probability of acquisition, between codes W and W'. It was shown that the performance of Reed-Solomon codes with symbol avoidance is good for all channel bit error probabilities. The performance of Reed-Solomon codes with symbol avoidance proved to be superior to that of Reed-Solomon codes without symbol avoidance for higher channel bit error probabilities. The Reed-Solomon codes with symbol avoidance require less redundancy to deliver the superior performance over the conventional Reed-Solomon codes without symbol avoidance.

References

[1] S. W. Golomb, B. Gordon, and L. R. Welch, "Comma-free codes," *Canadian Journal of Mathematics*, vol. 10, no. 2, pp. 202–209, 1958.

[2] B. H. Jiggs, "Recent results in comma-free codes," *Canadian Journal of Mathematics*, vol. 15, pp. 178–187, 1963.

[3] W. L. Eastman, "On the construction of comma-free codes," *IEEE Transactions on Information Theory*, vol. 11, no. 2, pp. 263–267, Apr. 1965.

[4] W. B. Kendall and I. S. Reed, "Path invariant comma-free codes," *IEEE Transactions on Information Theory*, vol. 8, no. 6, pp. 350–355, Oct. 1962.

[5] E. N. Gilbert, "Synchronization of binary messages," *IEEE Transactions on Information Theory*, vol. 6, no. 4, pp. 470–477, Sept. 1960.

[6] L. J. Guibas and A. M. Odlyzko, "Maximal prefix-synchronized codes," *SIAM Journal of Applied Mathematics*, vol. 35, no. 2, pp. 401–418, Sept. 1978.

[7] A. J. van Wijngaarden and H. Morita, "Extended prefix synchronization codes," in *Proceedings of the 1995 IEEE International Symposium on Information Theory*, Whistler, BC, Canada, 1995, p. 465.

[8] H. Morita, A. J. van Wijngaarden, and A. J. H. Vinck, "On the construction of maximal prefix-synchronized codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 2158–2166, Nov. 1996.

[9] W. B. Kendall, "Optimum synchronizing words for fixed word-length code dictionaries," in *Space Program Summary*. Pasadena, California: Jet Propulsion Laboratory, California Institute of Technology 4, 1964, p. 37.

[10] J. J. Stiffler, "Instantaneously synchronizable block code dictionaries," in *Space Program Summary*. Pasadena, California: Jet Propulsion Laboratory, California Institute of Technology 4, 1965, p. 268.

[11] T. Shongwe and A. J. H. Vinck, "Reed-solomon code symbol avoidance," *SAIEE Africa Research Journal*, vol. 105, no. 1, pp. 13–19, Mar. 2014.

[12] A. J. H. Vinck, *Coding Concepts and Reed-Solomon Codes*. Institute for Experimental Mathematics, Essen, Germany, 2013.

[13] T. Shongwe, A. J. H. Vinck, and H. C. Ferreira, "Reducing the probability of sync-word false acquisition with reed-solomon codes," in *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Vancouver, Canada, Aug. 27–29, 2013, pp. 159–164.

[14] R. H. Barker, "Group synchronizing of binary digital systems," in *Communication Theory*. W. Jackson. Ed. New York: Academic-Butterworth, 1953.

[15] R. Turyn, "Sequences with small correlation," in *Error Correcting Codes*. H. B. Mann, Ed. New York: Wiley, 1968, pp. 195–228.

[16] M. W. Willard, "Optimum code patterns for PCM synchronization," in *Proceedings of the 1962 National Telemetering Conference*, 1962.

[17] J. L. Maury and F. Styles, "Development of optimum frame synchronization codes for Goddard space flight center PCM telemetry standards," in *Proceedings of the 1965 National Telemetering Conference*, 1965.

[18] R. A. Scholtz, "Frame synchronization techniques," *IEEE Transactions on Communications*, vol. 28, no. 8, pp. 1782–1789, Aug. 1980.

[19] J. Jiang and K. R. Narayanan, "Iterative soft decoding of reed-solomon codes," *IEEE Communications Letters*, vol. 8, no. 4, pp. 244–246, 2004.

[20] B. VUCETIC, V. PONAMPALAM, V. Jelena, *et al.*, "Low complexity soft decision decoding algorithms for reed-solomon codes," *IEICE transactions on communications*, vol. 84, no. 3, pp. 392–399, 2001.

[21] V. N. Papilaya, T. Shongwe, A. J. H. Vinck, and H. C. Ferreira, "Selected subcarriers QPSK-OFDM transmission schemes to combat frequency disturbances," in *Proceedings of the 2012 IEEE International Symposium on Power Line Communications and its Applications*, Beijing, China, Mar. 27–30, 2012, pp. 200–205.

APPENDIX

TABLE A.1
RESULTS OF THE P_{FAD} FOR CORRESPONDING AVOIDED SYMBOLS, FOR BINARY SYNC-WORDS OF LENGTHS $7-11$ WRITTEN IN THEIR OCTAL FORMAT. A $(2^3-1,3)$ RS CODE WAS USED FOR THE SYMBOL AVOIDANCE, $|A|=1$ AND $r=1$.

(a) Binary sync-word 130, length 7

Probability of False Acquisition	Avoided Symbols
6.80E-03	0
6.80E-03	2
6.80E-03	4
8.16E-03	3
8.16E-03	5
8.16E-03	7
9.52E-03	1
1.09E-02	6

(b) Binary sync-word 270, length 8

Probability of False Acquisition	Avoided Symbols
0	6
1.46E-03	3
1.46E-03	5
2.92E-03	1
2.92E-03	7
4.37E-03	4
5.83E-03	0
5.83E-03	2

(c) Binary sync-word 560, length 9

Probability of False Acquisition	Avoided Symbols
0	3
0	6
1.57E-03	5
3.14E-03	0
3.14E-03	1
3.14E-03	4
3.14E-03	7
6.28E-03	2

(d) Binary sync-word 1560, length 10

Probability of False Acquisition	Avoided Symbols
0	3
0	6
1.70E-03	1
1.70E-03	4
1.70E-03	5
1.70E-03	7
3.40E-03	0
5.10E-03	2

(e) Binary sync-word 2670, length 11

Probability of False Acquisition	Avoided Symbols
0	3
0	5
0	6
1.86E-03	1
1.86E-03	4
1.86E-03	7
3.71E-03	0
5.57E-03	2

TABLE A.2
RESULTS OF THE P_{FAD} FOR CORRESPONDING AVOIDED SYMBOLS, FOR BINARY SYNC-WORDS OF LENGTHS $7-11$ WRITTEN IN THEIR OCTAL FORMAT. A $(2^3-1,3)$ RS CODE WAS USED FOR THE SYMBOL AVOIDANCE, $|A|=2$ AND $r=1$.

(a) Binary sync-word 130, length 7

Probability of False Acquisition	Avoided Symbols	
1.85E-03	0	5
3.70E-03	0	1
3.70E-03	0	2
3.70E-03	0	3
3.70E-03	0	4
3.70E-03	0	6
3.70E-03	1	3
3.70E-03	1	4
3.70E-03	3	6
3.70E-03	4	6
5.56E-03	1	7
5.56E-03	2	4
5.56E-03	3	5
7.41E-03	1	5
7.41E-03	1	6
7.41E-03	2	3
7.41E-03	2	5
7.41E-03	3	4
7.41E-03	4	5
7.41E-03	4	7
9.26E-03	1	2
9.26E-03	2	6
9.26E-03	2	7
9.26E-03	3	7
9.26E-03	5	6
9.26E-03	6	7
1.11E-02	0	7
1.11E-02	5	7

(b) Binary sync-word 270, length 8

Probability of False Acquisition	Avoided Symbols	
0	1	3
0	1	5
0	1	6
0	2	3
0	2	6
0	3	5
0	3	6
0	4	5
0	4	6
0	5	6
0	6	7
1.98E-03	0	1
1.98E-03	0	6
1.98E-03	3	4
1.98E-03	5	7
3.97E-03	0	3
3.97E-03	0	4
3.97E-03	2	5
3.97E-03	3	7
5.95E-03	0	2
5.95E-03	0	5
5.95E-03	1	2
5.95E-03	1	4
7.94E-03	0	7
7.94E-03	2	4
7.94E-03	4	7
9.92E-03	1	7
1.19E-02	2	7

(c) Binary sync-word 560, length 9

Probability of False Acquisition	Avoided Symbols	
0	0	1
0	0	3
0	0	4
0	0	6
0	1	3
0	1	5
0	1	6
0	2	3
0	2	6
0	3	4
0	3	5
0	3	6
0	3	7
0	4	5
0	4	6
0	5	6
0	6	7
2.14E-03	5	7
4.27E-03	0	7
4.27E-03	2	5
6.41E-03	0	2
6.41E-03	0	5
6.41E-03	1	2
6.41E-03	1	4
6.41E-03	4	7
8.55E-03	1	7
8.55E-03	2	4
1.28E-02	2	7

(d) Binary sync-word 1560, length 10

Probability of False Acquisition	Avoided Symbols	
0	0	1
0	0	3
0	0	4
0	0	6
0	1	3
0	1	5
0	1	6
0	2	3
0	2	6
0	3	4
0	3	5
0	3	6
0	3	7
0	4	5
0	4	6
0	5	6
0	6	7
2.31E-03	5	7
4.63E-03	0	7
4.63E-03	1	2
4.63E-03	1	4
4.63E-03	2	5
4.63E-03	4	7
6.94E-03	0	2
6.94E-03	0	5
6.94E-03	1	7
6.94E-03	2	4
1.16E-02	2	7

(e) Binary sync-word 2670, length 11

Probability of False Acquisition	Avoided Symbols	
0	0	1
0	0	3
0	0	4
0	0	6
0	1	3
0	1	5
0	1	6
0	2	3
0	2	6
0	3	4
0	3	5
0	3	6
0	3	7
0	4	5
0	4	6
0	5	6
0	6	7
2.53E-03	5	7
5.05E-03	0	7
5.05E-03	1	2
5.05E-03	1	4
5.05E-03	2	5
5.05E-03	4	7
7.58E-03	0	2
7.58E-03	0	5
7.58E-03	1	7
7.58E-03	2	4
1.26E-02	2	7