

A New Coding Scheme for Runlength-Limited Channels

V. Pechenkin and F. R. Kschischang
Department of Electrical and Computer Engineering
University of Toronto
{vlad,frank}@comm.utoronto.ca

April 16, 2004

Abstract

A coding scheme for reliable information transmission via a runlength-limited channel is often implemented as a concatenation of an inner modulation code with an outer error-correcting code. The outer decoder has to deal with errors introduced not only by the channel but also by the inner decoder itself. We present a new almost block-decodable rate $1/2$ $(2, \infty)$ -constrained code that avoids error amplification and has very simple encoding and decoding procedures. We then estimate the redundancy of the error-correcting code required to achieve reliable transmission over the channel with random bit shifts and flips.

1 Introduction

A binary channel is said to be runlength-limited (RLL), or (d, k) -constrained, if channel sequences are required to have at least d but at most k '0's between any pair of adjacent '1's. Information transmission through an RLL channel implies two necessary steps. First, an arbitrary source word must be mapped unambiguously to a sequence that satisfies the channel constraints. Second, some error control must be implemented to protect the data from noise.

A popular approach is to utilize a concatenation of an inner modulation RLL (or constrained) code with an outer error-correcting code (ECC). The received word is first demodulated by an RLL decoder, and an ECC decoder cleans up possible errors afterwards. The errors are introduced not only by the channel but also by the inner decoder itself. In fact, a single channel error may cause a burst of demodulation errors, a phenomenon known as error propagation or error amplification.

While many good coding schemes have been designed, little has been done to exploit error-correcting capabilities of the inner code and lessen the burden on the outer code. In this paper we propose a novel $(2, \infty)$ -constrained code that has the following properties:

- encoding and decoding algorithms are very simple;
- error propagation is extremely limited;
- the degree of redundancy needed for an outer ECC can be estimated quite accurately if the channel is not too bad (quantified below).

2 The RLL Code

We construct an almost block-decodable RLL code of rate $1/2$ as follows. A source word u of length p is mapped to a codeword x of length $q = 2p$ according to Table 1. If u has '01' or '011' at the end, a dummy '0' is appended to u , x is obtained in the same manner, and the two trailing '0's of x are discarded. For example, $u = 101$ corresponds to $x = 10010$.

User Data	Constrained Bits	Phrase
0	00	A
10	1000	B
110	010000	C
111	100100	D

Table 1: Encoding table.

To concatenate two constrained words $x = (x_1, \dots, x_q)$ and $y = (y_1, \dots, y_q)$ without the constraint violation we employ a trick from [1]. Note that x_q is always 0. The 2-constraint can then be violated if and only if $x_{q-1} = y_1 = 1$. In this case we set $x_{q-1} = y_1 = 0$ and $x_q = 1$. Thus, x_q works

as a merging bit. To decode x , x_q will be used by the RLL decoder along with x_0 , the merging bit preceding x_1 .

We can compute various quantities of interest related to our coding scheme. For example, the probability q_1 that a randomly chosen bit in x is a one is $q_1 = 5/28$. Also, it is interesting to note that the distribution of runs appears to be close to maxentropic.

The RLL decoder is implemented in a sliding-window fashion. A window of length 3 slides along the word $x = (x_0, x_1, \dots, x_q)$ producing $u_i = x_{2i-2} + x_{2i-1} + x_{2i}$, $i = 1, \dots, p$, where the “plus” sign stands for logical “OR” operation. Thus, if $x_j = 1$, it “triggers” two ‘1’s at the output of the decoder if j is even and a single ‘1’ if j is odd.

3 Outer Channel Statistics

We assume that bits may be corrupted by the channel through shifts and flips. A single-position bit (or peak) shift is said to occur if ‘010’ was sent but ‘100’ (left shift) or ‘001’ (right shift) was received. A bit flip is a 1-to-0 transition (a drop-out) or a 0-to-1 transition (a drop-in).

Our channel is the cascade of a peak-shift channel (PSC), followed by an independent binary asymmetric channel (BAC). The PSC does not affect ‘0’s directly but shifts each ‘1’ by a single position with probability, say, p_s , the left and the right shifts being equally likely. Each shift occurs independently of the others. We only consider single-position shifts to ensure that no two ‘1’s occupy the same bit slot at the output of the PSC. The BAC introduces both drop-outs with probability p_{do} and drop-ins with probability p_{di} .

Let u be an unconstrained sequence at the output of the ECC and x be a corresponding channel word. Suppose that a noisy sequence y is received and the RLL decoder maps y to \hat{u} . We want to know how u and \hat{u} are related probabilistically. That is, we want to know the crossover probabilities $a = p(\hat{u}_i = 1 | u_i = 0)$ and $b = p(\hat{u}_i = 0 | u_i = 1)$, where $i = 1, \dots, p$. These values effectively give us the ECC redundancy needed. If the channel is not too noisy, an accurate estimate of a and b can be obtained following the technique described in [2]. To simplify the argument we let $q \rightarrow \infty$ and neglect merging effects.

The key idea is to assume that x is corrupted by a single channel error of any type and compute the corresponding expected number of demodulation errors. The expected number of 0-to-1 transitions, N_{01} , and the expected number of 1-to-0 transitions, N_{10} , are evaluated separately. Next,

we compute N_{ch} , the average number of channel errors of each type expected to occur when a random x is sent. The crossovers are then calculated as $a = N_{ch}N_{01}/p$ and $b = N_{ch}N_{10}/p$. Finally, we approximate the resultant probability of error as a sum of independent contributions from different types of errors.

The computation of N_{01} and N_{10} is straightforward but relatively lengthy. We will outline the procedure at the end of this section but let us present the main results for the crossovers a and b first assuming that N_{01} and N_{10} are known.

Single bit shift. $N_{01} = 37/80$ and $N_{10} = 2/5$. Now, to compute N_{ch} we notice that a constrained sequence of q bits contains qq_1 ‘1’s on average so the average number of shifts is $N_{ch} = qq_1p_s$. After the substitution of $q_1 = 5/28$ and $q = 2p$ the crossover probabilities become $a = (37/112)p_s$ and $b = (2/7)p_s$.

Single drop-out. $N_{01} = 0$, $N_{10} = 7/5$, and $N_{ch} = qq_1p_{do}$. Thus, $a = 0$ and $b = p_{do}/2$.

Single drop-in. $N_{01} = 21/23$, $N_{10} = 0$, and $N_{ch} = q(1 - q_1)p_{di}$. Hence, $a = (3/2)p_{di}$ and $b = 0$.

Combining all the contributions we obtain

$$a = \frac{37}{112}p_s + 3p_{di}, \quad b = \frac{2}{7}p_s + p_{do}. \quad (1)$$

The total probability of error p_e is thus

$$p_e = \frac{a + b}{2} = \frac{69}{224}p_s + \frac{3}{2}p_{di} + \frac{1}{2}p_{do}. \quad (2)$$

If $p_{do} = p_{di} = p_f$ then $a > b$ unless the channel is noiseless. In this case the outer channel is binary asymmetric. If $p_s \gg p_f$, the degree of asymmetry is $a/b \approx 37/32$. If, on the other hand, $p_f \gg p_s$, $a/b \approx 3$.

X	$Q(X)$	$p_1(X)$	$p_0(X)$	$p_{11}(X)$	$p_{00}(X)$
A	1/2	–	8/23	–	4/23
B	1/4	2/5	6/23	2/5	2/23
C	1/8	1/5	5/23	1/5	1/23
D	1/8	2/5	4/23	1/5	1/23

Table 2: Statistical properties of channel sequences.

Let us now briefly describe how to compute N_{01} and N_{10} . We will need Table 2, an extended version of Table 1. We say that an input string from Table 1 gives rise to a channel *phrase* $X \in \{A, B, C, D\}$. A channel sequence is thus a concatenation of phrases. Table 2 summarizes the following statistical properties of our channel sequences.

- $Q(X)$ is the probability that a randomly picked phrase is of type X .

- $p_1(X)/p_0(X)$ is the probability that a randomly picked ‘1’/‘0’ belongs to a phrase of type X .
- $p_{11}(X)/p_{00}(X)$ is the probability that a randomly picked ‘1’/‘0’ comes from a phrase of type X and corresponds to a ‘1’/‘0’ at a certain position.

The algorithm itself is as follows. Consider a combination YXZ of phrases X , Y , and Z . Select a bit in X and corrupt it by a shift or a flip. Decode the resultant sequence and determine the number of demodulation errors committed. Average over all possible X , Y , and Z .

In fact, due to a short decoding window for our scheme we need to consider combinations of at most two phrases. Indeed, it can be seen that

- a shift in X never affects the decoding of Z ;
- a drop-in in X never affects the decoding of Y ;
- a drop-out in X never affects the decoding of either Y or Z .

Still, the tables are too bulky to fit into this paper. Therefore, we illustrate the technique by two simple examples.

In the first example let us compute the average number of demodulation errors introduced by a drop-out. It turns out that a drop-out at an even position causes two errors, while a drop-out at an odd position induces just one error, all errors being 1-to-0 transitions. Let E and O be the events that a randomly picked ‘1’ is at an even or odd position, respectively. From Table 2 we have $p(E) = p_{11}(C) + p_{11}(D) = 2/5$ and $p(O) = p_{11}(B) + p_{11}(D) = 3/5$. The expected number of errors is then $N_{10} = 2p(E) + p(O) = 7/5$.

In the second example we determine the probability of the event $S = \{\text{a bit shift produces no demodulation errors}\}$. A detailed analysis of all pairs YX reveals that this occurs in the following cases:

- $X = B, Y = D$, left shift;
- $X = D$, right shift of the leading ‘1’;
- $X = D, Y = D$, left shift of the leading ‘1’.

The probability of S is thus

$$p(S) = \frac{Q(D)[p_{11}(B) + p_{11}(D)] + p_{11}(D)}{2} = \frac{11}{80}.$$

Equations (1) are approximate. There will be a small correction due to possible cancellation of neighboring errors. Still, we found that (1) is in a good agreement with simulation results for values of p_e up to 0.1. For noisier channels errors occur

more frequently and tend to cancel each other more often so the true probability of error is actually less than predicted by (2). This is demonstrated in Figure 1 for $p_{di} = p_{do}$ and $p_s = 2p_{do}$. Channels with $p_e > 0.1$ are unlikely to be practically interesting, however.

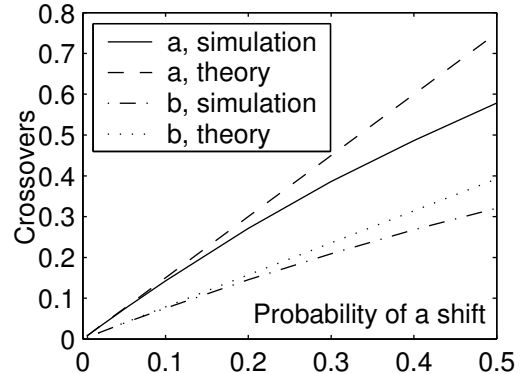


Figure 1. The crossover probabilities a and b versus p_s : a comparison between the theoretical estimate and simulation results.

4 Discussion

The new coding scheme is mainly designed for channels where bit shifts dominate. Our analysis reveals that any bit shift introduced by the channel can cause at most one demodulation error, a characteristic property of a Gray code. Although we considered single-position bit shifts only, possible multiple-position shifts do not present a severe problem. For example, a two-position shift may result in at most two demodulation errors. The error propagation is extremely limited simply because the decoding window is just three bits wide.

5 Conclusion

In this paper we proposed a constrained code for $(2, \infty)$ RLL channels designed to mitigate the influence of bit shifts and flips introduced by the channel on the outer ECC. The code was found to be very resilient to channel perturbations thanks to a short decoding window and a Gray-like nature.

References

- [1] G. Khachatryan and K. A. S. Immink, “Construction of simple runlength-limited codes,” *Electronics Letters*, vol. 35(2), p. 140, 1999.
- [2] D. G. Howe and H. M. Hilden, “Shift error propagation in 2, 7 modulation code,” *IEEE J. Select. Areas Comm.*, vol. 10(1), pp. 223–232, 1992.