Reducing The Error Floors Of Turbo Codes That Use Structured Interleavers

S.Velugonda Reddy¹, Ch.Ravi Kumar ²

1. P.G student, ECE Department, Sir C.R.Reddy College of Engineering, Eluru
2. Assistant professor, ECE Department, Sir C.R.Reddy College of Engineering, Eluru.

Abstract— An efficient method for lowering the error floors of turbo codes that use structured Interleavers is presented in this paper. Structured interleaver is an interleaver that follows particular property when interleaving the information bits. In this method a set of parity bits are forced to take zero bit values in p1(output of RSC1) when encoding, when decoding the received bits, we use these known zero bit parities in p1 to reduce the error floors (BER). The effectiveness of this method is shown by its ability to significantly enhance Hamming distance spectra and thereby lowers error floors by several orders of magnitude; for example, using a rate-1/3 4-state single-binary turbo code with a packet of 1504 information bits, the packet error rate at its output is improved by about two orders of magnitude.

Keywords- Error floor, Hamming distance spectrum, Structured interleavers, Turbo codes.

I. INTRODUCTION

In recent years, a number of methods for lowering the error floors of turbo codes have been proposed. Andersen proposed the use of a BCH code as an outer code. Narayanan et al. proposed the use of the List Viterbi Algorithm. Oberg et al. introduced a method that identifies positions in the systematic part that are associated with the lowest Hamming distances, and then a modified turbo-code encoder inserts zero bit values in these positions to eliminate these low weight Hamming distances, thereby lowering the error floor. S. Crozier “Forced Symbol Method” which applies repeated decoding, with one or more symbols being forced to certain values when an error is detected using an error detection method such as a CRC code. Berrou et al. introduced three-dimensional turbo codes in which the encoder is a serial concatenation of a pre-existing turbo-code encoder, and a patch that includes a rate-1/3 encoder.

II. THE PROPOSED METHOD

The proposed method consists of (1) an offline code design phase that is only performed once, in which all relevant code parameters are determined and shared with both the encoder and decoder, eliminating the need to send any additional code related side-information when transmitting; (2) a modified encoding procedure; and (3) a modified decoding procedure. In contrast to standard turbo coding where the encoding of K information bits requires an interleaver of length K, the proposed turbo coding requires an interleaver of length Z = K + L to encode K information bits, where L is the number of extra bits inserted into the K information bits to improve the Hamming distance spectrum. In contrast to Oberg’s method where all L extra bits are static (i.e., take zero bit values for all codewords), the proposed method determines L extra bits that are dynamic (i.e., vary from one codeword to another, depending on the data). The details of the method are discussed in the following subsections.

A. Design Phase

The design phase focuses on turbo codes that use interleavers with a specific repeating structure. More precisely, the design phase concentrates on dithered relative prime (DRP) interleavers, but many of the basic concepts and properties also apply to other structured interleavers, such as almost regular permutation (ARP) interleavers.

Figure 1 shows the approach used to design dithered relative prime (DRP) interleavers. The approach consists of three stages. First, the input vector, vin, is dithered (permuted locally) using a small read dither vector, r, of length R. Vector r is a permutation of indexes 0 through R-1. Next, the resulting vector, vno, is permuted using an RP interleaver to obtain good spread. Finally, the resulting vector, vno, is dithered using a small write dither vector, w, of length W, to generate the output vector vout. Vector w is a permutation of indexes 0 through W-1. The interleaver length, Z, must be a multiple of both R and W. Note that short read and write dither vectors will not destroy the good spreading properties of an RP interleaver, but will tend to lower the spread somewhat.

A DRP interleaver could be implemented using the 3-stage process shown in Figure 1, this is not the recommendation approach. The overall equivalent interleaver, I, can be written as follows.

Let \( \lfloor x \rfloor \) denote the floor(x) function and again let \( [x]_{\text{mod} m} \) denote x modulo-m arithmetic. With these definitions, the equations for the various DRP interleaver vectors shown in Figure 1 can be expressed as follows:

\[
V_a (i) = V_{in} (I_a (i)) \quad V_b (i) = V_a (I_b (i)),
\]

\[
V_{out} (i) = V_b (I_c (i)) \quad i = 0 \ldots Z - 1
\]

where

\[
I_a (i) = \lfloor i / R \rfloor + r (\lfloor i / R \rfloor) \quad i = 0 \ldots Z - 1
\]

\[
I_b (i) = \lfloor s + i p \rfloor \quad i = 0 \ldots Z - 1
\]
Thus, the input vector can be interleaved using
\[ v_{\text{out}}(i) = v_{\text{in}}(1(i)) , \quad i = 0 \ldots Z-1 \quad (5) \]
where the interleaver is completely defined by
\[ I(i) = \lfloor i \mod \pi(i) \rfloor , \quad i = 0 \ldots Z-1 \quad (6) \]
All the indexes of \( I \) can be computed using equations (2), (3), (4), and (6).

A DRP interleaver can be stored by just storing \( r, w, s \) and \( p \). This represents a significant reduction in storage, as compared to storing all \( Z \) indexes, but further simplifications and reductions are possible. Let \( M \) be the least common multiple (LCM) of \( R \) and \( W \). It can be shown that
\[ I(i) + M\pi(i) \mod Z = 0 \ldots Z-1 \quad (7) \]
It follows that the interleaver indexes can be computed recursively by cycling through \( M \) index increments. That is, \( I(i) = \lfloor I(i-1) + M\pi(i-1) \rfloor \mod Z , \quad i = 0 \ldots Z-1 \quad (8) \)
where \( I(0) \) and the \( M \) index increments in vector \( P \) are defined by (6) and \( P(i) = \lfloor I(i) - I(i-1) \rfloor \mod Z , \quad i = 0 \ldots M-1 \quad (9) \)
Thus, all the indexes of \( I \) can be computed using the simple recursion in (8), and the interleaver can be stored by just storing \( P \). \((0)\) is arbitrary.) Further, equation (8) is simple enough to accommodate “on-the-fly” index generation, saving even more memory. In particular, this method works well with the circular buffer feature provided by most modern digital signal processors.

A few important properties are now explained further. Dual tail-biting is assumed for convenience. A rotational (modulo-\( Z \)) shift in \( v_{\text{in}} \) or \( v_{\text{out}} \) does not affect the spread or distance properties of the TC. However, a rotational shift in \( v_{\text{a}} \) or \( v_{\text{b}} \) can affect the spread and distance properties. It can be shown that any shift in \( v_{\text{a}} \) or \( v_{\text{b}} \) is equivalent to shifting \( v_{\text{a}} \) and/or \( v_{\text{b}} \) and using a different value of \( s \) in the RP interleaver. Thus, the \( s \) parameter is sufficient for testing different shifts when searching for good interleavers. Consider the special case where \( R \) and \( W \) are relative primes. In this case we have \( M = R \times W \). Thus, a small amount of dither (small values for \( R \) and \( W \)) can still force a large number of index increments, \( M \). This is undesirable since \( M \) is also the resolution of the interleaver bank (i.e. \( Z \) must be a multiple of \( M \)). There is also no benefit derived from trying different \( s \) values since all relative shifts between dither vectors \( r \) and \( W \) will occur for every value of \( s \). At the other extreme we have the special case where \( M = R = W \). This case offers the largest amount of dither for the smallest number of index increments, \( M \), and the finest interleaver bank resolution. In this case, different results can be achieved for all shifts \( s = 0 \ldots M-1 \), and thus all of the different shift values are worth considering. This second case is more convenient and has generally been found to give better distance results. For example, \( M \) values of 8, 16 and 32 have generally been found to be good choices for interleaver lengths on the order of 500, 2000 and 8000, respectively. The results presented later are for \( M = 16 \).

DRP interleavers have the following important structural property:
\[ \pi(i + Mz) = [\pi(i) + Mp]z , \quad i = 0 \ldots Z-1 \quad (10) \]
where \([z]_x = x \mod Z \). Let vector \( v \) denote the first \( M \) indices of \( \pi \) (i.e., \( v(0) = \pi(0) \), \( v(M - 1) = \pi(M - 1) \)). It follows from equation (1) that the entire DRP interleaver \( \pi \) is completely determined by vector \( v \) and the parameters \( M, p \) and \( Z \) (or \( D \)).

The repetitive structure of the interleaver \( \pi \) in steps of \( M \) results in repetitive distance properties [9]. That is, if circular (tail-biting) encoding [10] is used, then the distances measured and associated with a specific test position \( j \) will also be measured at positions \([i + k \cdot p]z \mod Z \), where \( k = 1, 2, \ldots, D-1 \). Other good termination techniques tend to disrupt the repetitive structure somewhat, but typically result in similar distance properties with only slightly reduced multiplicities.

The design phase is the core of the proposed method as it deals with the improvement of the Hamming distance spectrum, and thereby lowers the error floors of turbo codes. The improvement of the Hamming distance spectrum is achieved through the forcing of \( L \) parity bits at positions \( j_1, j_2, \ldots, j_L \) in \( P1 \) to take zero bit values, where \( P1 \) is the parity stream generated by the \( Z \) uninterleaved data bits. The proper size, \( L \), and the proper locations \( j_1, j_2, \ldots, j_L \) are determined through a search within the \( Z \) parity bits of \( P1 \). The search of \( j_1, j_2, \ldots, j_L \) in a random way has the following drawbacks: (1) the search complexity is very high, or even prohibitive, because the cardinality of the search space is \( Z^L \), and (2)
randomness of a limited practical search makes it difficult to eliminate completely low-weight code words or even reduce their numbers because, as discussed above, distances repeat in a structured, not random way. A better approach that overcomes these drawbacks is to construct the L positions \( j_1, j_2, \ldots, j_l \) from a small number of positions \( j_1, j_2, \ldots, j_l \) that repeat in steps of \( M \), where \( l = L/D \). That is, \( j_1, j_2, \ldots, j_l = (j_1, j_2, \ldots) \); \( (j_1 + M, j_2 + M, \ldots, j_l + M); \ldots; (j_1 + (D - 1)M, j_2 + (D - 1)M, \ldots, j_l + (D - 1)M) \). Recall that \( D = Z/M \) is the number of blocks of size \( M \) in \( p1 \). Note that with this approach, the cardinality of the search space is reduced enormously from \( \binom{L}{2} \) to \( \binom{M}{2} \).

The remainder of this subsection proposes a practical approach to determine the proper size, \( L \), and the proper positions \( j_1, j_2, \ldots, j_l \), using distance measurement or estimation methods such as Garello’s method [11] and the double impulse method [12]. Since the proposed method requires the employment of an interleaver of length \( Z = K + L \), an increase in decoding complexity occurs when compared to standard turbo decoding, which uses an interleaver of length \( K \). Thus, the choice of \( L \) depends on the amount of acceptable increase in decoding complexity. The steps of the proposed approach are:

1) Determine \( D = Z/M \) based on the selected structured interleaver.
2) Select a value for \( L \) that represents an acceptable increase in decoding complexity.
3) Identify a stopping criterion such as an acceptable error rate at a given SNR operating point, which can be approximated by a truncated union bound using the first few terms of a distance spectrum.
4) Select an acceptable processing limit for the given stopping criterion.
5) Choose a small set of positions \( j_1, j_2, \ldots, j_l \), where \( l = L/D \).
6) Construct \( j_1, j_i, \ldots, j_{l} \) using \( j_1, j_2, \ldots, j_{l} \). That is, \( j_1, j_2, \ldots, j_{l} = (j_1, j_2, \ldots); (j_1 + M, j_2 + M, \ldots, j_{l} + M); \ldots; (j_1 + (D - 1)M, j_2 + (D - 1)M, \ldots, j_{l} + (D - 1)M) \).
7) Force the \( L \) parity bits, located at positions \( j_1, j_2, \ldots, j_{l} \) in \( P1 \), to take zero bit values.
8) Apply a distance measurement method to determine if the stopping criterion is satisfied. If yes, then stop.
9) If the processing limit is not exceeded then go back to step 5; otherwise go back to step 3.

Recall that the design phase is done offline and only once per code. Furthermore, all positions \( j_1, j_2, \ldots, j_{l} \) are known to both the encoder and decoder, eliminating the need to send any additional code-related side-information when transmitting.

### B. Modified Encoding Procedure

Each time when encoding, \( L \) bits (called “insertion bits”) are inserted into the \( K \) information bits to force \( L \) parity bits at positions \( j_1, j_2, \ldots, j_{l} \) in \( P1 \) to take zero bit values. Remember that \( P1 \) is the parity stream generated by the first constituent encoder which operates on the \( Z \) uninterleaved data bits. Thus, these \( L \) zero bit parities are easily forced through the inclusion of \( L \) insertion bits into the \( K \) uninterleaved information bits. The value of each of these \( L \) insertion bits is easily determined “on the fly” by simply checking the current state of the first constituent encoder. The inclusion of the \( L \) insertion bits into the original \( K \) information bits requires an interleaver of length \( K + L \), and yields a change in the code rate. Note that these \( L \) insertion bits, when interleaved, will also generate \( L \) additional unknown parity bits in the parity stream generated by the second constituent encoder. A simple and effective way to maintain the same code rate is to puncture (remove) all \( L \) insertion bits and their corresponding \( 2L \) parity bits from both parity streams. Note that the complexity of the modified encoding process remains very low relative to the complexity of the decoding process, and is thus inconsequential.

### C. Modified Decoding Procedure

The main goal of the modified decoding procedure is the exploitation of the \( L \) known parity bits, previously forced to take zero bit values during the modified encoding procedure. There are a number of ways to instruct the decoder to exploit these \( L \) zero bit parities. A simple way is to insert large log-likelihood ratio (LLR) values, corresponding to these \( L \) zero bit parities, into the received codeword. Another way is to reject all bit values that differ from the \( L \) zero bit parities; for example, in a trellis decoder, specific parity bit values can be rejected by removing their corresponding branches in the trellis, or by adjusting their corresponding branch metric values. While there are several ways to instruct a decoder to exploit these \( L \) zero bit parities, the former method of inserting large LLR values in the received codeword is generally preferred because it requires no changes to the basic structure of the existing decoder. The decoder is also informed about the other unknown punctured bits by simply inserting zero-valued LLRs into the appropriate locations in the received codeword. With this approach, the complexity of the decoding process remains proportional to the number of states per stage and the number of stages in the constituent code trellises. Thus, the complexity of the modified decoding process is simply increased by a factor of \( (K+L)/K \), relative to standard turbo decoding; for example, using \( L = K/2 \) increases the decoding complexity by 50%, whereas doubling the number of states per stage increases the complexity by 100%.

### III. HAMMING DISTANCE SPECTRAL ANALYSIS AND ERROR RATE PERFORMANCE RESULTS

The method presented above was applied to a rate-1/3 4-state turbo code that uses two identical RSC constituent codes in parallel. The RSC code uses feedback and feed forward polynomials of 7 and 5 in octal, respectively, and both trellises start in state zero and are terminated in state zero.
using the dual-termination method in [7]. The \( L \) zero bit parities are entirely located in the first parity part \( P_1 \) (i.e., the parity part generated by the uninterleaved data).

For evaluation purposes, a DRP interleaver of length 2192 bits, with dither vector length \( M = 16 \), was used, where \( M \) is also the number of repeating index increments required to implement the DRP interleaver. A search using repeating insertion patterns of length 16 was conducted to determine the size of \( L \) and the locations of the \( L \) parity bits that need to be forced to take zero bit values. The pattern length of 16 was intentionally chosen to match, and take full advantage of, the repetitive structure of the DRP interleaver with \( M = 16 \). A good solution was found with \( L = 5 \), where \( j_1, j_2, \ldots, j_5 = \{0, 3, 4, 8, 11\} \). This results in \( L = 5 \times \frac{2192}{50} = 685 \); however, one of these 685 positions is ignored because it is within the positions reserved for dual-termination. The remaining 684 zero bit parities are located at positions \( j_1, j_2, \ldots, j_{684} = \{0, 3, 4, 8, 11\} \); \{16, 19, 20, 24, 27, \ldots\}). To keep the code size and rate unchanged, all \( L \) insertion bits, as well as all \( 2L \) parity bits from both parity parts \( (P_1 \) and \( P_2) \) that are directly caused by the \( L \) insertion bits, are punctured; \( p_2 \) is the parity part generated by the interleaved data.

Binary antipodal signaling was used with an Additive White Gaussian Noise (AWGN) channel model. The signal-to-noise ratio (SNR) is measured in terms of energy per information bit, \( E_b \), over the single-sided noise power spectral density, \( N_0 \). Enhanced max-log-APP decoding with scaled extrinsic information and 16 full iterations were used. The decoder was instructed to exploit the \( L \) zero bit parities by placing a large LLR value at each position \( j \) in \( P_1 \). The decoder was also informed about the other unknown punctured bits by simply inserting zero-valued LLRs into the appropriate locations in the received codeword.

Comparison results are presented for two DRP interleavers with different lengths. Both DRP interleavers were designed using a dither vector length of \( M = 16 \). The first interleaver is used with standard turbo coding and operates on 1500 information bits and 4 termination bits (i.e., interleaver length is 1504 bits). The second interleaver is used with the proposed turbo coding, and operates on 1504 information bits, 4 termination bits, and 684 insertion bits (i.e., interleaver length is 2192 bits).

In Figure 2, the graph shows Packet Error Rate (PER) performance with the standard turbo code (using an extrinsic information scale factor (SF) of 0.85). At 2dB of SNR the PER is nearly \( 10^{-6} \). In Figure 3, the graph shows Bit Error Rate (BER) with the standard turbo code (using an extrinsic information scale factor (SF) of 0.85). At 2dB of SNR the BER is nearly \( 10^{-7} \).

The figures 2 and 3 shows Packet Error Rate (PER) and Bit error Rate (BER) using standard turbo coding technique respectively.

In Figure 4, the graph shows Packet Error Rate (PER) performance with the standard turbo code (using an extrinsic information scale factor (SF) of 0.8). At 2dB of SNR the PER is nearly \( 10^{-6} \). In Figure 5, the graph shows Bit Error Rate (BER) with the standard turbo code (using an extrinsic information scale factor (SF) of 0.8). At 2dB of SNR the BER is nearly \( 10^{-7} \).

The figures 4 and 5 shows Packet Error Rate (PER) and Bit error Rate (BER) using proposed turbo coding technique respectively. The two SF values were selected to provide the best waterfall performance for each method. The proposed method yields a PER of nearly \( 10^{-6} \) and BER of \( 10^{-7} \) at the moderate SNR value of 2 dB, while the standard turbo code
achieves a PER of $10^{-4}$ and a BER of $6 \times 10^{-5}$ at the same SNR value. This translates into a reduction in error rates by about two orders of magnitude at the moderate SNR value of 2 dB. When compared to the standard turbo code, the proposed method yields an increase in decoding complexity of about 684/1504 $\approx$ 45%.

The PER and BER performance results of nearly $10^{-6}$ and $10^{-7}$, respectively, obtained for the proposed method at the SNR value of 2 dB for rate-1/3 8-state turbo code approach with a well-designed DRP interleaver. However, compared to a conventional 4-state turbo code, the proposed method is only about 45% more complex, due to the longer interleaver, whereas the 8-state approach is about 100% more complex, due to doubling the number of states.

IV. CONCLUSION

An efficient method for reducing turbo code error floors was presented. This method involves the forcing of a set of parity bits to take zero bit values when encoding, and the exploitation of these zero bit parities when decoding. When applied to binary antipodal signalling over an AWGN channel, 4-state, rate 1/3, single-binary turbo codes with 1504 information bits per packet, PER and BER values of about $10^{-5}$ and $10^{-6}$ were achieved at the moderate SNR value of 2 dB, respectively. This is a drop in PER and BER by about two orders of magnitude in the error floor region.

REFERENCES

codes.