Good Interleavers for Concatenated Block Codes

Jan Bajcsy and Hisashi Kobayashi
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544
e-mail: bajcsy@ee.princeton.edu, hisashi@ee.princeton.edu
Fax: (609) 258-3745, Tel: (609) 258-1984

Abstract

First, a definition of a good interleaver/permutation is proposed for two block codes concatenated in parallel and in series. We prove a lower bound on their size and describe an asymptotically optimal construction of a class of such permutations.

A simple way to implement these permutations is then described. Finally, two examples are given to show how constructed interleavers outperform standard uniform block interleaver.

1 Introduction

Non-uniform interleavers are one of the key ingredients for good performance of Turbo codes [5] and, more generally, of generalized concatenated codes [3].

Designing good and relatively short interleavers is of significant interest not only from a theoretical point of view but also for applications. A good interleaver can significantly improve the overall performance of a concatenated code. On the other hand, having to use an extremely long interleaver would introduce too much of a delay and storage requirement in the actual communication system. This would consequently limit the scope of feasible applications of a given code.

Previous work on the subject of interleaver design includes the work by Ramsey [12], Andersen and Zyablov [1], and Hokfelt and Maseng [9]. The first construction technique leads to interleavers that do not inherently satisfy the goodness criterion considered in this paper. The latter two approaches design non-uniform interleavers by a computer search, where an initial uniform block interleaver is gradually modified with respect to a restriction condition.

This paper first explores the construction of good non-uniform interleavers, when two block codes are concatenated in parallel or in series. Section 2 focuses on motivating, defining and finding good permutations for two codes concatenated in parallel and in series. In Section 3 we discuss practical implementation of constructed permutations and simulation results of achieved interleaving gain. Section 4 provides concluding remarks and directions for future work.

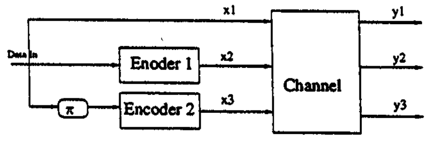

Figure 1: A generic parallel concatenated code.

2 Permutations for Two Codes in Parallel and Series

2.1 Motivating Good Interleavers

Consider a turbo code based on an binary (n, k) linear block code (not necessarily a systematic code) depicted in Figure 1. A set of parities is created by each encoder. The encoders, concatenated in parallel, are separated by a permutation π. The systematic part of the data x_1 and two sets of redundant bits x_2 and x_3 are transmitted over the channel. The overall rate of such a code is therefore $\frac{k}{k+2n}$. For simplicity of the motivation, we first consider the case where the channel is a binary erasure channel (BEC).

An iterative decoder for this overall code and channel is depicted in Figure 2. The input to the decoder is (y_1, y_2, y_3) and the component decoders will attempt to resolve the erasures in the stream y_1 to get the data x_1. Because the channel is error free, no errors will be created in this process, i.e., a bit can be either resolved with 100 % certainty or is left as an erasure. Each component decoder will use the most recent available version of resolved data x_1, denoted $\hat{x_1}$, and an appropriate set of erased parities y_2 or y_3. The component decoders are connected in an iterative loop.

In every step of the iteration, each component decoder fixes blocks that contain a small number of erasures, but has problems resolving blocks with many erasures. Consequently, isolated erasures in the stream $\hat{x_1}$ can be resolved by the decoders in most cases, but consecutive erasures or clustered erasures will remain unresolved by the decoder.

From the iterative decoder's point of view, the task of the interleaver and deinterleaver is to break apart bursts of erasures in $\hat{x_1}$ that may exist after each compo-

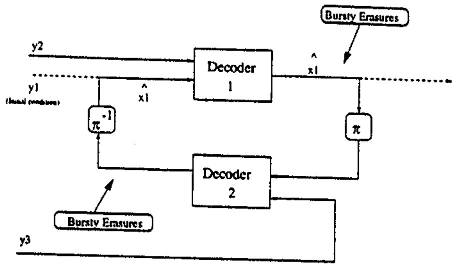

Figure 2: An iterative decoder for the code of Figure 1 transmitted over the binary erasure channel.

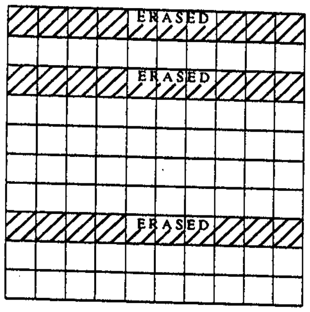

Figure 3: A 10x10 uniform block interleaver with bursty erasures from Example 1. Dashed lines denote erased bits.

nent decoder. This will consequently simplify the task of the next component decoder, because it will have to deal with less clustered noise. An analogous observation about the role of the interleaver and deinterleaver can be also made, when we deal with decision errors or soft decisions. In both cases, clusters of bad decisions are harder to remove by component decoders than isolated instances of these. For this reason, the output of the component decoders will be "bursty" in bad decisions, even if the channel is memoryless. Interleaving/deinterleaving will help to spread these, thus simplifying the task of the next decoder.

The following simple example will illustrate how a commonly used uniform block interleaver may not be good at spreading clustered erasures. We will also explain why this is the case.

Example 1 *Consider a 10×10 uniform block interleaver. Hypothetical bursty erasures after decoder 1 are shown in Figure 3. For simplicity, we assume 3 whole blocks of data are erased.*

After depicted block interleaver, there are 3 erasures in every horizontal block. This still makes the erasure appearance after the interleaver quite bursty, hence the interleaver does not achieve a very good erasure spreading. The main reason for this deficiency is that pairs of bits, coming from the same two horizontal blocks, appear inside several vertical blocks. For example, bit pairs from horizontal blocks 1 and 3 are repetitively used in vertical blocks.

The last observation in this example means, that neighboring bits at one decoder (or encoder) should be spread as much as possible after π and π^{-1} respectively, not only with respect to each other, but also with respect to their neighbors after interleaving. The next section defines neighboring bits and chooses two criteria for their good spreading.

2.2 Definitions

Let $i = 1, 2, ...$ be the indices of the data bits that are entering the interleaver.

Definition 1 *A block interleaver of size N is a permutation of N elements*

$$\pi : \{1, 2, ..., N\} \rightarrow \{1, 2, ..., N\}. \tag{1}$$

We will use the standard cyclic representation with $\pi(n)$ denoting the position of the n-th bit after the interleaver.

Note, that throughout the rest of the paper, we will use the terms interleaver and permutation as synonyms.

Definition 2 *:*

(a) Bits i and j are neighbors at the input of an (n,k) block encoder iff i and j are encoded in the same block, i.e.,

$$\left\lfloor \frac{i-1}{k} \right\rfloor = \left\lfloor \frac{j-1}{k} \right\rfloor. \tag{2}$$

(b) Bits i and j are neighbors at the output of an (n, k) block encoder iff i and j are the results of encoding within the same block, i.e.,

$$\left\lfloor \frac{i-1}{n} \right\rfloor = \left\lfloor \frac{j-1}{n} \right\rfloor. \tag{3}$$

We will usually denote either of these by $(i, j) \in PN$, using a binary relation $PN \subseteq \{1, 2, ..., N\} \times \{1, 2, ..., N\}$.

(Note, that each of the relations defined above is an equivalence relation.)

Definition 3 *An (n_1, k_1) code is concatenated in parallel with an (n_2, k_2) code, as depicted in Figure 1. Let PN_1 denote the relation of being a neighbor at the input of the first encoder. PN_2 denotes the relation of being a neighbor at the input of the second encoder, i.e., $(i, j) \in PN_2$ iff*

$$\left\lfloor \frac{\pi(i)-1}{k_2} \right\rfloor = \left\lfloor \frac{\pi(j)-1}{k_2} \right\rfloor. \tag{4}$$

A good permutation π separating these two codes satisfies the following two conditions:

1. *For $i \neq j$ if $(i, j) \in PN_1$ then $(i, j) \notin PN_2$.*
2. *There are no $i, j, k, l \in \{1, 2, ..., N\}$ (all different) such that $(i, j) \in PN_1$, $(k, l) \in PN_1$, $(i, k) \in PN_2$ and $(j, l) \in PN_2$.*

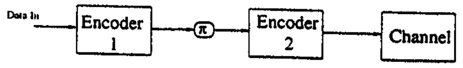

Figure 4: A generic serial concatenated code.

We denote such a permutation $\pi_{par}(k_1, k_2)$.

The first condition guarantees that no two neighboring bits at one encoder are neighbors at the other. The second condition prevents repetition of neighboring pairs observed in Example 1.

Example 2 *An example of a good permutation for $k_1 = k_2 = 2$ is $(2,3)(4,5)$. It reorders the data as follows:*

$$[1,2,\ |\ 3,4,\ |\ 5,6] \rightarrow [1,3,\ |\ 2,5,\ |\ 4,6],$$

where the vertical lines separate bits encoded by different blocks of the error-correcting code. One can easily see that no two bits originally in the same block are in the same block after the re-ordering. It can be also checked by hand that the second goodness condition holds too.

By an analogous argument for two block codes concatenated in series (Figure 4) we can define a good permutation for two codes concatenated in series.

Definition 4 *An (n_3, k_3) code is concatenated in series with an (n_4, k_4) code, as depicted in Figure 4. Let PN_1 denote the relation of being a neighbor at the output of the first encoder, i.e., $(i,j) \in PN_1$ iff*

$$\left\lfloor \frac{i-1}{n_3} \right\rfloor = \left\lfloor \frac{j-1}{n_3} \right\rfloor. \qquad (5)$$

PN_2 denotes the relation of being a neighbor at the input of the second encoder, i.e., $(i,j) \in PN_2$ iff

$$\left\lfloor \frac{\pi(i)-1}{k_4} \right\rfloor = \left\lfloor \frac{\pi(j)-1}{k_4} \right\rfloor. \qquad (6)$$

A good permutation π separating these two codes satisfies the following two conditions:

1. *For $i \neq j$ if $(i,j) \in PN_1$ then $(i,j) \notin PN_2$.*
2. *There are no $i,j,k,l \in \{1,2,3,...,N\}$ (all different) such that $(i,j) \in PN_1$, $(k,l) \in PN_1$, $(i,k) \in PN_2$ and $(j,l) \in PN_2$.*

We denote such a permutation $\pi_{ser}(n_3, k_4)$.

Theorem 1 *Let be $n_3 = k_1 = a$ and $k_2 = k_4 = b$. A good permutation $\pi_{par}(k_1, k_2)$ is also a good permutation $\pi_{ser}(n_3, k_4)$ and vice versa.*

Proof: Follows from Definitions 3 and 4 by setting $n_3 = k_1 = a$ and $k_4 = k_2 = b$. ∎
Consequently, we will just have to look at good permutations $\pi(a,b)$, corresponding to good $\pi_{par}(a,b)$ or $\pi_{ser}(a,b)$.

Theorem 2 *$\pi(a,b)$ is good iff its' inverse is good.*

Proof: This follows from PN_1 and PN_2 being symmetric and from the symmetry of conditions 1 and 2 in Definitions 3 and 4. ∎

After making these definitions and basic observations, two questions can be asked:

- Given the two codes, what is the minimum size of N, s.t. there is a good permutation of size N?
- Given the two codes, how can we construct a good permutations of size as small as possible ?

2.3 Problem Restatement

In order to make the problem mathematically solvable, we restate it as follows. Using relation PN_1 at the first encoder, we represent bits as colored balls. First a bits are represented as a balls of color 1, the next a bits are represented as a balls of color 2; etc., and the last a bits from the (N/a)-th block are represented as a balls of color (N/a). After the permutation, we divide the data (balls) into consecutive blocks of size b that are to be encoded by encoder 2. We denote these blocks as boxes of size b.

Then the problem of creating a good permutation reduces into partitioning N colored balls (a balls of each of the (N/a) colors) into (N/b) boxes of size b, so that the following two conditions hold:

1. No two balls of the same color are in the same box.
2. No pair of colors is repeated in 2 or more different boxes.

It is easy to see that a good permutation specifies such a partition. One just has to represent bits by colored balls as described above. On the other hand, a ball partition of colored balls, satisfying the above conditions, represents a whole class of good permutations. (There are $a!$ different ways to match up balls of color c with bits from the c-th block at encoder 1.)

2.4 A Lower Bound on the Size of Good Interleavers

Theorem 3 *The size N of every good permutation $\pi(a,b)$ has to satisfy*

$$N \geq \max\{a^2b - a^2 + a \ \ ; \ \ b^2a - b^2 + b\}. \qquad (7)$$

Proof:
We will use the restated problem with colored balls and boxes. Consequently, a balls of color 1 have to be put into different boxes. Moreover, all remaining other balls in these boxes have to be of different color. (Otherwise we get two pairs $(1,i)$ in two different boxes.) Thus there have to be at least $a(b-1)+1$ colors, implying

$$N \geq a(a(b-1)+1) = a^2b - a^2 + a. \qquad (8)$$

By Theorem 2, π^{-1} is also good, hence we can apply the same argument for it and get

$$N \geq b(b(a-1)+1) = b^2a - b^2 + b. \tag{9}$$

Since both inequalities have to hold, we conclude

$$N \geq \max\{a^2b - a^2 + a \ ; \ b^2a - b^2 + b\}. \tag{10}$$

∎

Example 3 *If $a = b = 3$, then by the previous theorem $N \geq 21$. A construction with colored balls and boxes that achieves the lower bound is shown bellow (columns denote the boxes):*

$$\begin{array}{ccccccc} 1 & 1 & 1 & 2 & 2 & 3 & 3 \\ 2 & 4 & 6 & 4 & 5 & 4 & 5 \\ 3 & 5 & 7 & 6 & 7 & 7 & 6 \end{array}. \tag{11}$$

One possible reordering of bits this construction implies is shown bellow:

$$[1,4,7,2,10,13,3,16,19,5,11,17,6,14,20,8,12,21,9,15,18] \tag{12}$$

Remark 1 *A good permutation, achieving equality in Theorem 3, does not always exist. For $a = b = k$, a solution of the colored balls and boxes problem is equivalent to existence of a square 2-$(k^2-k+1,k,1)$ design. By Bruck-Ryser-Chowla theorem [6], a necessary condition for existence of such a design is that the equation*

$$z^2 = (k-1)x^2 + (-1)^{\frac{k(k-1)}{2}}y^2 \tag{13}$$

has an integer solution $(x,y,z) \neq (0,0,0)$. For k=7, the equation does not have such a solution by infinite descent for divisibility by 3.

2.5 Constructing Good Interleavers

Theorem 4 *For all $a, b = 1,2,3,\ldots$ and $p \geq a,b$ a prime, there is a good permutation $\pi(a,b)$ of size*

$$N = abp. \tag{14}$$

Proof:
Consider the following construction with the colored balls and boxes. The first set of boxes, i.e., boxes $1,2,\ldots,p$, are filled in as follows (columns again denote the boxes):

$$\begin{array}{cccccc} 1 & 2 & 3 & \cdots & (p-1) & p \\ (p+1) & (p+2) & (p+3) & \cdots & . & 2p \\ \vdots & & & & & \vdots \\ (bp-p+1) & (bp-p+2) & . & \cdots & . & bp \end{array} \tag{15}$$

The second set of boxes, i.e., boxes $(p+1),(p+2),\ldots,2p$, are obtained by cyclically shifting to the right the first row above by 0, the second row by 1, the third row by 2, ..., and the b-th row by $(b-1)$. Thus we obtain

$$\begin{array}{cccccc} 1 & 2 & 3 & \cdots & (p-1) & p \\ 2p & (p+1) & (p+2) & \cdots & . & (2p-1) \\ \vdots & & & & & \vdots \\ (bp-b+2) & (bp-b+3) & . & bp \ 1 \ . & . & (bp-b+1) \end{array} \tag{16}$$

Boxes $(2p+1),(2p+2),\ldots,3p$ boxes are obtained from the second set of p boxes using the same cyclic shifts of rows. I.e., the first row is shifted by 0, the second row by 1, the third row by 2, ..., and the b-th row by $(b-1)$. One proceeds the same way to obtain the fourth set of p boxes from the third one, etc. Finally, the a-th set of boxes is obtained from the $(a-1)$-th.

By this construction, each color appears in a fixed row and only once in every consecutive set of p boxes. There will be a balls of each color used, each box (column) contains b colored balls and no box (column) can contain two balls of the same color.

We will show that the second condition for having a good permutation holds by contradiction. Assume there is a pair (c_1, c_2) of colors that appears in two boxes from our construction and this happens in a column of the $A-th$ set of boxes and in a column of the $B-th$ set of boxes, $1 \leq A < B \leq a \leq p$. We may also assume that color c_1 is in row i, color c_2 in row j of the $A-th$ set of boxes, $1 \leq i < j \leq b \leq p$. Then by our construction the following has to hold:

$$(i-1)(B-A) \equiv (j-1)(B-A) \quad (mod\ p) \tag{17}$$
$$(j-i)(B-A) \equiv 0 \quad (mod\ p). \tag{18}$$

Since $0 < (j-i) < p$, $0 < (B-A) < p$ and p is a prime, the last congruence leads to a contradiction. Consequently, our construction describes a good permutation of size $N = abp$. ∎

Example 4 *This simple example shows how a good permutation $\pi(3,4)$ is constructed, e.g., for a serial concatenation of a $(3,2)$ code followed by a $(6,4)$ code. The used prime is $p = 5$. In the "colors and boxes" notation we get*

$$\begin{array}{ccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 1 & 2 & 3 & 4 & 5 & 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 & 10 & 6 & 7 & 8 & 9 & 9 & 10 & 6 & 7 & 8 \\ 11 & 12 & 13 & 14 & 15 & 14 & 15 & 11 & 12 & 13 & 12 & 13 & 14 & 15 & 11 \\ 16 & 17 & 18 & 19 & 20 & 18 & 19 & 20 & 16 & 17 & 20 & 16 & 17 & 18 & 19. \end{array} \tag{19}$$

Corollary 1 *Described construction is asymptotically optimal with respect to the lower bound from Theorem 3, i.e,*

$$\lim_{a,b\to\infty} \frac{abp}{\max\{a^2b - a^2 + a \ ; \ b^2a - b^2 + b\}} = 1 \tag{20}$$

Proof: By the Prime number theorem and Bertrand's postulate [13], there is a prime p, $p \in [k; k(1+\epsilon(k))]$, where $\epsilon(k) \leq 1$ and $\epsilon(k) \to 0$ as $k \to \infty$. Consequently the corollary follows. ∎

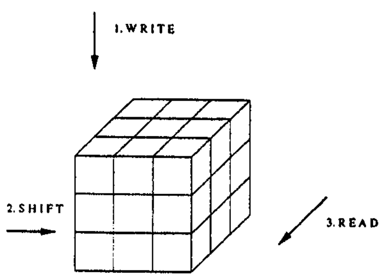

Figure 5: A schematic representation of implementing constructed good interleaver for $a = b = p = 3$.

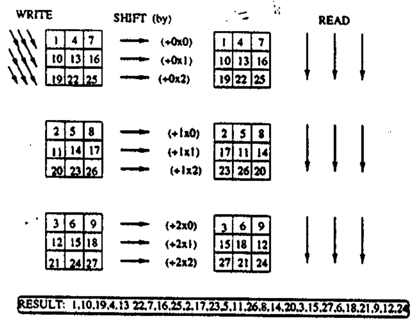
WRITE
SHIFT (by)
READ
(+0x0)
(+0x1)
(+0x2)
(+1x0)
(+1x1)
(+1x2)
(+2x0)
(+2x1)
(+2x2)
RESULT: 1,10,19,4,13 22,7,16,25,2,17,23,5,11,26,8,14,20,3,15,27,6,18,21,9,12,24

Figure 6: A floor-by-floor and step-by-step depiction of the constructed good interleaver for $a = b = p = 3$. It shows how the data is written in, shifted and read out from the 3-D array.

3 Applying Good Permutations

3.1 Implementing Good Interleavers

This section describes a simple way to implement constructed good permutations from the previous section. The implementation is done in a similar way as the uniform block interleavers are usually implemented. For the latter, the data is written into a two-dimensional (2-D) array row-wise and read out column-wise.

The constructed permutations will be implemented using a 3-D array. The floors along the z-direction will be indexed $0, 1, ..., (a-1)$, the rows along the x-direction will be indexed $0, 1, ..., (p-1)$ and the columns along the y-direction will be indexed $0, 1, ..., (b-1)$.

First, the data is written into the 3-D array "pillar"-wise (along the z-coordinate). Then each row r on floor f is cyclically shifted to the right in by $r \times f$. Finally, the data is read out column-wise (in y-direction). Figure 5 depicts this procedure schematically for a good permutation with $a = b = p = 3$ and Figure 6 shows how this happens in a step-by-step and "floor-by-floor" manner.

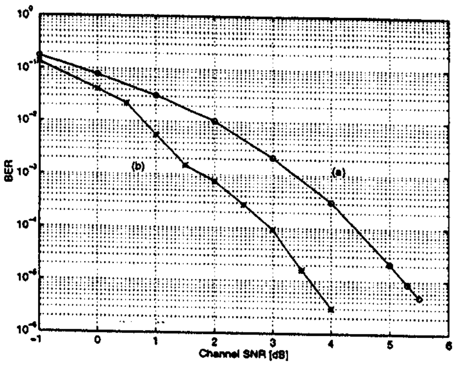

Figure 7: Performance of two concatenated systems with different interleavers: (a) standard block interleaver (b) interleaver from construction in Section 3.

3.2 Performance Improvement

First, we consider a serial concatenation of two simple (11, 5) block codes with a generator matrix

$$\mathbf{G} = \begin{bmatrix} 1&0&0&0&0&0&1&1&1&1&1 \\ 0&1&0&0&0&1&0&1&1&1&1 \\ 0&0&1&0&0&1&1&0&1&1&1 \\ 0&0&0&1&0&1&1&1&0&1&1 \\ 0&0&0&0&1&1&1&1&1&0&1 \end{bmatrix}. \qquad (21)$$

The standard system uses a uniform 5×11 block interleaver between the two encoders. The second system uses constructed good interleaver of size $5 \times 11 \times 11$. Performance results are obtained for the two systems on a discrete time additive white Gaussian noise channel with binary inputs ($Channel\ SNR = \frac{E_s}{\sigma^2} = \frac{1}{\sigma^2}$). Decoding, in both cases, is done iteratively, using an iterative decoder considered in [3].

Figure 7 presents the simulation results of the BER after five decoding iterations. (Most of decoding improvement was achieved after this number of iterations.) We can see that at $BER = 10^{-5}$ the constructed interleaver gains about 1.5 dB against the standard one. Another way to view the improvement is to observe that at channel $SNR = 3.7$ dB improved interleaving lowers the BER by a factor of about 100.

The other test system is similar to existing digital recording applications, such as CD and DVD. It contains a $[28, 24] \times [32, 28]$ product Reed-Solomon code followed by a simple modulation code and a precoded discrete-time duobinary partial-response channel with additive white Gaussian noise. The receiver side contains a ML decoder for the PR sequence and a modulation decoder with AZD (ambiguity zone detection) that declares uncertain bytes as erased. The iterative decoder uses the idea from [3], when it tries to resolve erasures and correct errors by iterating over the two RS decoders.

Figure 8 presents the simulation results of the BER after six decoding iterations. (Most of decoding improvement was achieved after this number of iterations.) We can see that at $BER = 10^{-6}$ the constructed interleaver gains about 0.6 dB against the standard one. One can

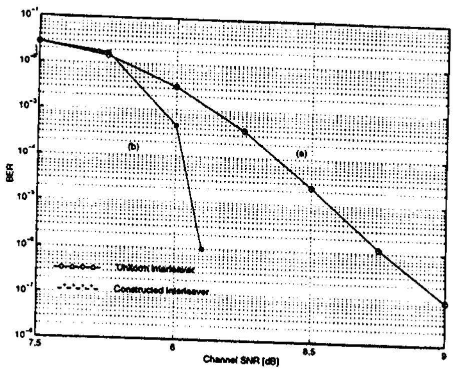

Figure 8: Performance of two system with two Reed-Solomon and different interleavers: (a) standard block interleaver (b) interleaver from construction in Section 3.

also see that the BER is lowered by several orders of magnitude due to better interleaving. Currently, we were not yet able to simulate error rates close to the ones used by the recording systems, i.e., $10^{-12} - 10^{-14}$, though a simple extrapolation of the curves in Figure 8 would indicate improvement in the order of dB's.

4 Conclusion and Further Work

We have defined good interleavers/permutations for parallel and serial concatenation of two block codes. For both cases, we have proved a lower bound on their size that is essentially cubic in the block size of the component codes. We have then constructed a class of good permutations which are asymptotically optimal with respect to the lower bound.

A simple way to implement constructed permutations using a 3-D array was described. Finally, we observed significant performance improvement from using constructed interleavers in two concatenated systems as compared to standard uniform block interleavers.

The future work includes extending definitions and results to concatenation of convolutional codes as well as block and convolutional codes. Finally, the case of a general concatenation (more than two codes) will be considered.

Acknowledgement

The present work has been supported, in part, by the National Science Foundation, the New Jersey Commission on Science and Technology, Asahi Chemical Co. Ltd and the Ogasawara Foundation for the Promotion of Science and Engineering.

One of the authors would like to thank Peter Hegarty for his useful suggestions and the prime number theorem reference.

References

[1] J. D. Andersen and V. V. Zyablov, " Interleaver Design for Turbo Codes", *Turbo Codes Symposium*, Bretagne, France, *Conference Record*, pp. 154-156, Sept. 1997.

[2] L.R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. on Information Theory*, vol. IT-20, pp. 284-287, 1974.

[3] J. Bajcsy and H. Kobayashi, "Error Control of Generalized Concatenated Systems, " *PACRIM'97*, Victoria, B.C., Canada, *Conference Record*, pp. 749-752, 1997.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes (I)," *Proc. ICC '93*, pp. 1064-1070, Geneva Switzerland, May 23-26, 1993.

[5] C. Berrou and A. Glavieux, "Turbo Codes: General Principles and Applications", *Sixth Tirrenia Workshop on Digital Communication*, pp. 215-226, 1993.

[6] P. J. Cameron and J. H. van Lint, *Designs, Graphs, Codes and their Links*, Cambridge University Press, 1991.

[7] G. D. Forney Jr., *Concatenated Codes*, MIT Press, 1966.

[8] J. Hagenauer, P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications," *GLOBECOM 1989*, Dallas, Texas, *Conference Record*, pp. 1680-1686.

[9] J. Hokfelt and T. Maseng, " Methodical Interleaver Design for Turbo Codes", *Turbo Codes Symposium*, Bretagne, France, *Conference Record*, pp. 212-215, Sept. 1997.

[10] H. Kobayashi and D. T. Tang, "On Decoding of Correlative Level Coding Systems with Ambiguity Zone", *IEEE Trans. Communications*, Vol. COM-19, pp. 467-477, Aug. 1971.

[11] S. Lin, and D. J. Costello, Jr, *Error Control Coding: Fundamental and Applications*, Prentice-Hall, 1983.

[12] J. L. Ramsey, "Realization of Optimum Interleavers," *IEEE Trans. on Information Theory*, vol. IT-16, pp. 338-345, 1973.

[13] H. E. Rose, *A Course in Number Theory*, Oxford University Press, 1994.