A Performance Analysis of Interleaving

Mark Vogel
B. Eng, University of Victoria, 1990

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Applied Science

in the Department of
Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

[redacted]
Dr. Vijay K. Bhargava (Electrical and Computer Engineering)

[redacted]
Dr. Qiang Wang (Electrical and Computer Engineering)

[redacted]
Dr. Micaela Serra (Computer Science)

[redacted]
Dr. Meyer Nahon (Mechanical Engineering)

SUPERVISOR: V. K. BHARGAVA

Abstract

An analysis of interleaving based on performance benefits and performance costs is presented. Interleaving offers a communication system protection against bursty errors by re-ordering data before it is sent on the channel. The original order is regained by the deinterleaver after the channel and any error burst which may have occurred will be distributed by the deinterleaving process. Two measures of the extent of an interleaver's re-ordering are introduced. The cost of interleaving is assessed in terms of delay and memory requirements. These performance measures are considered in a full analysis and comparison of four interleaving methods: block interleaving, convolutional interleaving, pseudo-random interleaving and helical interleaving.

Examiners:

[redacted]

Dr. Vijay K.Bhargava (Electrical and Computer Engineering)

[redacted]

Dr. Qiang Wang (Electrical and Computer Engineering)

[redacted]

Dr. Micaela Serra (Computer Science)

[redacted]

Dr. Meyer Nahon (Mechanical Engineering)

Table of Contents

List of Figures

Acknowledgments

Many thanks are due to Vijay Bhargava for his tremendous support, faith and especially patience in seeing this thesis through to it conclusion. Thanks also to the CITR laboratory in general for providing the atmosphere in which to work, and to David Peterson in particular for bringing interleaving to my attention.

This research was funded in part by the Natural Sciences and Engineering Research Council through a post-graduate scholarship; their support is very much appreciated.

To my mother and father for their unconditional support of all my endeavors; thanks for giving me the freedom to choose.

Finally, my deepest thanks to Anita for absorbing the endless angst and to Seaman/Poet Hewett for his many bottles of inspiration.

To laughter-silvered wings

Chapter 1

Introduction to Interleaving

A major concern for many designers of communications systems is the often bursty quality of the errors present on real communications channels. Bursty errors are errors which occur on the channel consecutively. An error burst of length x consists of x consecutive errors. Real channels generate bursts of errors through a wide variety of different mechanisms. A land mobile channel is prone to fading as vehicles pass through differing areas of signal coverage caused by buildings and terrain. A compact disk suffers scratches which cover an entire sequence of data. A hostile jammer generates a burst of noise over a wireless channel which corrupts an entire portion of a message. A Slow Frequency Hop Spread Spectrum system allows two users to hop to the same frequency bin simultaneously, adversely affecting all the data those users transmit while in that bin. There are numerous such examples of channel errors that can be considered bursty in nature.

Very powerful error correction schemes will be overcome by a large number of consecutive symbol errors. Block codes such as Reed-Solomon codes can only

absorb so many errors per codeword before failing. Convolutional codes are better able to correct separated errors; for a given number of channel symbol errors, consecutive channel errors will result in more decoded errors than well distributed channel errors within the same constraint length. These problems arising from a bursty channel justify an investigation into a method for converting a bursty channel into a facsimile of a discrete, memoryless, symmetric channel. Interleaving is such a method.

The configuration of a simplified communications system using interleaving is shown in Figure 1.1

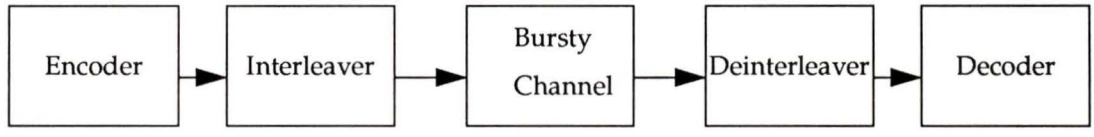

FIGURE 1.1: An Interleaved Communications System

Assuming digital data, the encoder adds redundancy to the data stream for error detection and correction. The data is then interleaved and transmitted across the channel where it is subject to bursts of errors. At the receiver the data is deinterleaved which returns the original ordering and distributes the error bursts. Finally, the data is decoded where the now separated errors are corrected.

The basic role of the interleaver in a communications system is to provide separation on the channel of adjacent symbols leaving the encoder. In this way a burst

of consecutive errors on the channel does not affect consecutive symbols in the data outside of the channel. Ramsey, in his description of what are now called convolutional interleavers [1], defines (n_2, n_1) interleavers as reordering a sequence so that no contiguous sequence of n_2 symbols in the reordered sequence contains any symbols that were separated by fewer than n_1 symbols in the original ordering. More specifically, if $a_{z_1}, a_{z_2}, a_{z_3}, \ldots$ is the sequence of data at the output of the interleaver and z_i denotes the position of each symbol at the input of the interleaver then

$$|z_i - z_j| > n_1 \quad \text{for} \quad |i - j| \leq n_2 - 1 \tag{1.1}$$

The $n_2 - 1$ term arises due to the definition of separation. For the purposes of this thesis, two symbols separated by x symbols will have $x - 1$ symbols between them. Thus adjacent symbols have separation 1.

A more contemporary description, applicable to block codes, defines the *depth*, I, of an interleaver as one less than the shortest burst length which can hit two adjacent symbols in the input data stream. The task of a system designer is to choose an interleaving depth large enough to combat the greatest tolerable error burst expected in the channel.

To reorder the data sequence before decoding, a deinterleaver is employed. The deinterleaver could be described as the inverse of the interleaver, and is by definition an interleaver itself. Together they constitute the working interleaver-

deinterleaver pair, hereafter described as the interleaver system. Whenever interleaving is mentioned, it implies both interleaving and deinterleaving.

The advantages of interleaving do not come for free. Because consecutive symbols arriving at the interleaver do not leave the interleaver consecutively, there must be some delay, d_i, associated with the interleaving process. There will then, for the same reason, be a delay in the deinterleaving process, d_d. Define the total delay due to interleaving as

$$D = d_i + d_d \tag{1.2}$$

The interleaving delay increases with an increase in the depth of the interleaver. Note that the delays, d_i and d_d, are not constant, rather, different symbols will experience different delays passing through the interleaver and deinterleaver. However, so that the symbol order is preserved, the total delay, D, will necessarily be the same for every symbol. For the purposes of this thesis, delay will be measured in symbol periods and storage is measured in symbol spaces.

Wherever there is delay introduced in a system there is also the need for storage. Define the total memory required, S, as

$$S = S_i + S_d \tag{1.3}$$

where S_i is the required interleaver memory and S_d is the required deinterleaver memory. During the delay associated with interleaving a symbol must be stored somewhere within the interleaving system. This gives rise to the following relation

$$S \geq D \tag{1.4}$$

An interleaver for which $S = D$ is optimum in terms of storage; the required interleaver memory cannot be reduced any further.

Designing an interleaved system becomes a balance between the required level of re-ordering and the resulting delay and memory requirements. Depending upon the application, delay is of varying importance. For digitally encoded music, a large delay is acceptable as the user is not aware of how long the signal has spent in the system. For wireless voice communication, a large delay is unacceptable as a gap between request and reply in a conversation becomes very unsettling. This thesis takes the position that delay is of primary importance. An interleaving method which provides the minimum delay for a given depth and (n_2, n_1) performance will be considered a superior interleaver.

All previous work which has been done on interleaving either does not offer an explicit comparison of the various interleaving methods, offers only a qualitative comparison[6], or creates some empirical graphs for comparison purposes [2]. The purpose of this thesis is to establish a basis of interleaving performance measures for which general solutions can be derived for several interleaving methods. Using the derived general solutions, one class of interleaver can be quantitatively compared to another in order that a decision can be made regarding the superiority of one interleaving method over another.

In the subsequent chapters, four interleaving methods are analyzed in terms of delay, depth and (n_2, n_1) performance. Block interleaving, convolutional interleaving, pseudo-random interleaving and helical interleaving are described and analyzed. In particular, general expressions for the (n_2, n_1) performance of each method are derived for block interleaving and convolutional interleaving which differ from similar expressions given in [6]. Additionally, there is no previous work which presents general expressions for the (n_2, n_1) performance of variable depth helical interleaving, expressions which are derived and presented here. The various methods are directly compared based on the derived expressions of performance. This comparison confirms the previously assumed superiority of helical interleaving and describes the previously unassumed equivalency of helical and convolutional interleaving.

Chapter 2

Block Interleaving

A block interleaver can be described as an N by I matrix into which data is written column by column and read out row by row. The deinterleaver is an N by I matrix into which data is written row by row and read out column by column, thus regenerating the original order. The configuration of a block interleaver/deinterleaver pair is shown in Figure 2.1.

Using Ramsey's (n_2, n_1) notation, the block interleaver is a $(I-1, N)$ or $(I, N-1)$ interleaver. This can be shown by considering the output stream of the interleaving matrix of Figure 2.1. In general, two adjacent symbols read out of the matrix are from the same position in adjacent columns. This gives a spacing on the input stream of N symbols. A third symbol read out of the interleaver will be separated on the input stream by N symbols from the second symbol and by $2N$ symbols from the first. This pattern continues until the last column has been read from. At this point the next symbol read out is from the first column and one row down from the previous symbol. Eventually, the Ith symbol is read out one column previous and one row down from the first symbol which was read out of the

matrix. The separation of these two symbols as they were written to the matrix is $N-1$, which is the minimum pairwise separation on the input stream of any sequence of I symbols on the output stream. Alternatively, if the analysis is stopped at a sequence of $I-1$ symbols on the output stream, no pair of symbols is separated on the input stream by less than N symbols.

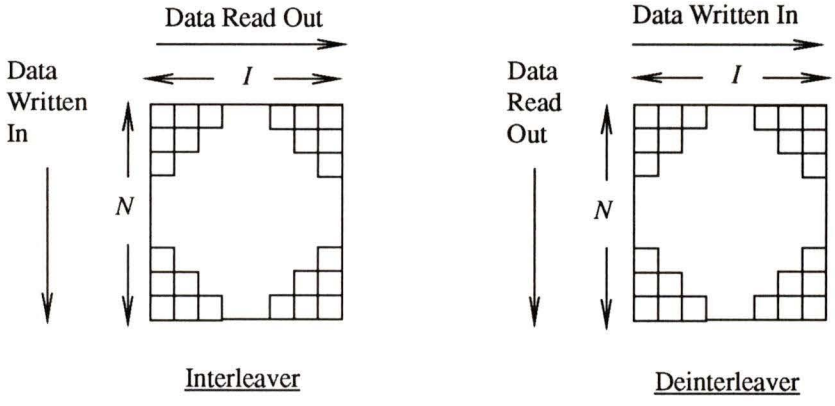

FIGURE 2.1: Block Interleaver/Deinterleaver Pair

Any burst of I errors or less on the channel will give no adjacent symbol errors on the output of the deinterleaver. The parameter I is the interleaving depth.

a	*f*	*k*	*p*
b	*g*	*l*	*q*
c	*h*	*m*	*r*
d	*i*	*n*	*s*
e	*j*	*o*	*t*

FIGURE 2.2: A 5 by 4 Block Interleaver

As an example, consider an $N = 5$ by $I = 4$ block interleaver. This configuration can be represented by the matrix in Figure 2.2, where the positions in the actual

sequence of data are represented by $a, b, c, \ldots, r, s, t$.

Figure 2.3 shows the sequence of data at three stages in the interleaved system: passing into the interleaver, crossing the channel, and passing out of the deinterleaver. Also shown is a burst of $n_2 = I = 4$ adjacent errors corrupting the channel data stream, represented by X's. Note how the regenerated data leaving the deinterleaver has separated the errors by at least $n_1 = N - 1 = 4$ symbols.

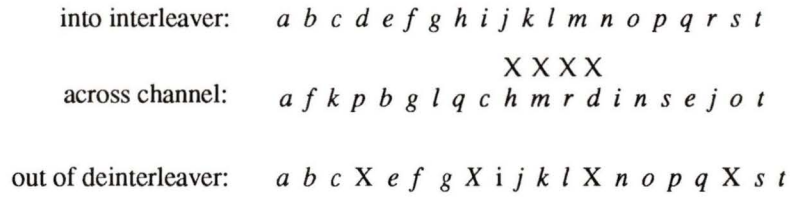

FIGURE 2.3: Burst Error Handling of 5 by 4 Block Interleaver

Block interleavers are especially well suited to block codes. The interleaver can be designed to match the column length, N, to the codelength, n, thus guaranteeing that an error burst of I symbols or less on the channel will affect any given codeword no more than once. A t random error correcting code can, with interleaving, correct t random errors and bursts of up to $I \times t$ errors.

The simplest implementation of a block interleaver would use two N by I RAM's. While one RAM is being written to, the other is being read from. At the point where the written RAM is full and the read RAM is empty, the roles of both RAM's switch and the cycle continues. This method guarantees a continuous flow of data through the interleaved system.

For the block interleaver to operate successfully, the data must be written into the deinterleaver in the same order as it was read out of the interleaver. This requires a synchronization effort, which, for the block interleaver, must be done modulo NI.

The delay, d_i, for each symbol between being written into the RAM and being read out of the RAM depends upon its position in the interleaving matrix. For instance, considering the matrix description shown in Figure 2.1, the upper left symbol of the interleaving matrix is written to the RAM and is not read out until the entire matrix has been filled, resulting in a delay of NI. Similarly, the lower right symbol is the last to be written into the matrix and also the last to be read, again giving a delay of NI. For both symbols the delay through the deinterleaver is also NI with the same explanations.

As another example, the lower left symbol in the matrix is the Nth symbol written into the RAM and number $(N-1)I+1$ to be read out. This gives a delay of $N(I-1)+(N-1)I+1$. The same symbol is number $(N-1)I+1$ to be written into the deinterleaver and the Nth to be read out. This gives a deinterleaving delay of $I+N-1$. The total delay due to interleaving is therefore $2NI$, the same as the previous example. Similar calculations can be made for every symbol in the matrix.

If the rows of the interleaver matrix are labelled from top to bottom as 0 to $N-1$ and the columns are labelled from left to right as 0 to $I-1$ then for a particular row n and column i, d_i can be calculated to be

$$d_i = NI + (I-1)\,n - (N-1)\,i \tag{2.1}$$

Using the same notation and calculation, the delay through the deinterleaver is given by

$$d_d = NI + (N-1)\,i - (I-1)\,n \tag{2.2}$$

Summing equation (2.1) and equation (2.2) gives the total interleaving delay for any symbol passing through the simple two RAM block interleaver

$$D_{\text{simpleblock}} = d_i + d_d = 2NI \tag{2.3}$$

The delay for a block interleaver can be reduced somewhat by beginning the read out onto the channel before all the data has been written into the RAM. Obviously, the limiting factor in this procedure is that no symbol can be read before it has been written into the matrix. With some observation of the read/write pattern of the interleaving matrix, this means the readout of the matrix can begin no earlier than would result in the upper right symbol of the matrix being simultaneously written in and read out. The read out can start, therefore, $I + N - 1$ symbols before the RAM has been fully written, for both the interleaver and the deinterleaver. The minimum block interleaving delay becomes

$$\begin{aligned} D_{\text{blockmin}} &= 2NI - 2\,(I + N - 1) \\ &= 2\,(N-1)\;(I-1) \end{aligned} \tag{2.4}$$

Using a simple two RAM block interleaver, the 5 by 4 example discussed earlier would have a total delay of 40 symbols. The minimum block interleaving delay is 24 symbols, a reduction of 40%. This appears to be a spectacular improvement,

however, it is due primarily to the small size of the example. A 31 by 30 block interleaver shows a minimum delay improvement of only 6.5% over the basic delay.

The storage required for a two RAM block interleaver is $2NI$, for a total storage for the interleaver/deinterleaver pair of $4NI$. Given the total delay for both interleaver and deinterleaver in equation (2.4), this storage requirement is more than double the optimum amount. If storage is at a premium, it can be reduced to NI using a single RAM [2]. This gives a total storage of $S = 2NI$ which is much closer to the optimum condition as given by equation (1.4). The single RAM method, in addition to reducing storage requirements, lowers configuration complexity by simultaneously reading from and writing to each position in the addressing sequence.

The single RAM method works by labelling the addresses of an N by I RAM matrix from 0 as the top left location down the successive columns through $NI-2$ and adding the symbol ∞ to label the lower right location. With the RAM matrix labelled this way the addressing sequence follows the pattern

$$0, N^k, 2N^k, \ldots, (NI-2)\,N^k, \infty \tag{2.5}$$

where the index k is initially zero and is incremented by one on each pass through the addressing sequence. All addressing is calculated modulo $(NI-1)$. The labelling of the deinterleaver RAM is identical and the addressing sequence of the deinterleaver follows the pattern

$$0, I^k, 2I^k, \ldots, (NI-2)\, I^k, \infty \tag{2.6}$$

On the $k = 0$ pass, the RAM matrix is filled with the incoming data symbols as soon as they are available. On subsequent passes, the addressing sequences given in (2.5) and (2.6) are followed as data symbols are first read from an address and new data immediately written to the same address. Note that the addressing period will be NI times the multiplicative order of N modulo $(NI-1)$ for the interleaver and NI times the multiplicative order of I modulo $(NI-1)$ for the deinterleaver. However, since $NI \equiv 1$ modulo $(NI-1)$, the interleaver and deinterleaver addressing periods will be the same, as is necessary for any synchronization efforts to work.

To confirm that this storage saving addressing scheme works as a block interleaver, three things must be verified. There must be a one to one mapping of symbols entering to symbols leaving the system, adjacent symbols entering the interleaver must be separated by I symbols on the channel, and the order of symbols entering the system must be restored by the deinterleaver.

By reading out and writing to the same RAM address each time an address is visited, a symbol can never be read more than once. At each pass through the address sequence every location in the RAM is visited. Therefore, a one to one mapping is guaranteed.

Adjacent symbols entering the interleaver are, in general, mapped to positions xN^k and $(x+1)\, N^k$ where x can be in the range from 0 to $(NI-2)$. On the next

pass, the interleaver reads those symbols with the next iteration of the address sequence from their positions now labelled yN^{k+1} and $(y+s)N^{k+1}$ where s represents the separation between the symbols as they are read into the channel. Because the positions are the same modulo $(NI-1)$ the following relations hold

$$\begin{aligned} xN^k &= yN^{k+1} && \therefore x = yN \\ (x+1)N^k &= (y+s)N^{k+1} && \therefore (x+1) = (y+s)N \\ \text{substituting for } x: & && yN+1 = yN+sN \\ & && \therefore sN = 1 \end{aligned} \tag{2.7}$$

As was previously noted, $NI \equiv 1$ modulo $(NI-1)$, therefore $s = I$, which is the required spacing on the channel of adjacent symbols entering the interleaver.

By the same procedure as in (2.7) it can be shown that symbols separated on the channel will be restored to their correct adjacent positions by the deinterleaving sequence of (2.6). The one RAM interleaving scheme is therefore a fully functional block interleaver.

Chapter 3

Convolutional Interleaving

Whereas a block interleaver is conceived around the idea of a memory matrix, convolutional interleavers are based on delay lines or shift registers, the prevalent memory devices of the early 70's. Convolutional interleavers were described by Ramsey in 1970 [1] and Forney in 1971 [3]. Both treatments will be discussed here.

3.1 Forney's Convolutional Interleaver

Forney's interleaving structure is shown in Figure 3.1. The incoming data is multiplexed onto M registers of varying delay. The delay of each register in symbols is given inside the boxes of Figure 3.1. Every time an incoming symbol is written into a register, a symbol is read off the other side of the same register onto the channel.

It is readily observed that the first symbol into the interleaver is passed directly onto the channel with zero delay. The second symbol into the interleaver is delayed by b symbols in its register. However, the register is only clocked every M symbols, therefore the second symbol is actually delayed by bM symbols

before it is read onto the channel. The third symbol into the interleaver is delayed by $2bM$ symbols and so on to the Mth symbol which is delayed by $(M-1)\,bM$ symbols. The cycle then repeats starting with the $(M+1)$ th symbol receiving zero delay.

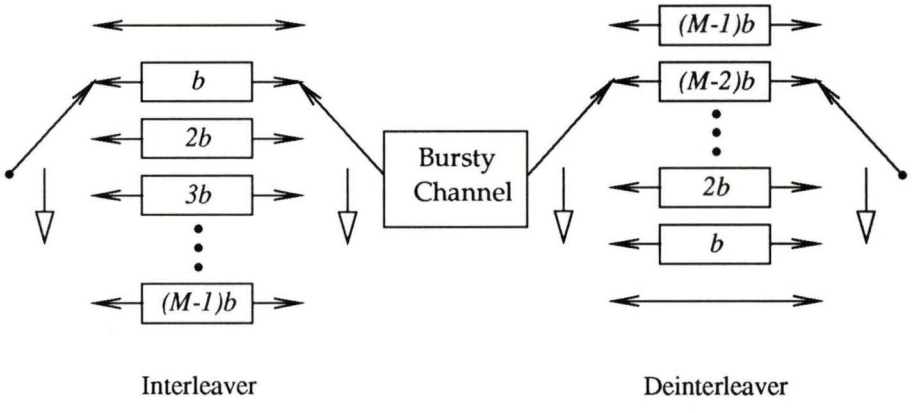

FIGURE 3.1: Forney's Convolutional Interleaver

The deinterleaver performs the same function as the interleaver, however, the ordering of the delays is reversed. The first of M symbols into the deinterleaver is delayed by $(M-1)\,bM$ symbols. The Mth symbol is passed through with no delay. For the interleaving system to work there must be synchronization between the banks of delay lines of the interleaver and the deinterleaver. A symbol passing through the first line of the interleaver must pass through the first line of the deinterleaver and so on. The two groups of M delay lines must be effectively "lined up". Consequently, synchronization of Forney's convolutional interleaver is modulo M symbols.

Any symbol which experiences a delay of $d_i = ibM$ symbols at the interleaver

will be delayed by $d_d = (M-1-i)\,bM$ symbols at the deinterleaver. The total delay for Forney's interleaver is then

$$D_{\text{Forney}} = (M-1)\,bM \tag{3.1}$$

A simple observation of Figure 3.1 and the previous discussion on delay shows that each symbol passes through $(M-1)\,b$ storage locations during the entire interleaving process. There are M possible paths through the interleaving system, therefore the total storage required is

$$S_{\text{Forney}} = M\,(M-1)\,b \tag{3.2}$$

Comparing (3.1) and (3.2) gives $S_{\text{Forney}} = D_{\text{Forney}}$ meaning the optimum storage is achieved for this interleaving configuration.

It is now shown that Forney's convolutional interleaver, using Ramsey's (n_2, n_1) notation, is at least an $(M, bM-1)$ interleaver. Consider any two consecutive symbols leaving the interleaver. The second symbol will, in general, have been delayed bM symbols longer since it was written to the interleaver than the first symbol was delayed in its passage through the interleaver. (Unless the first symbol passed through the last register and the second symbol passed through the initial, zero delay, register, in which case the first symbol will have been delayed $M\,(M-1)\,b$ symbols longer than the second). Because the registers the two symbols passed through are sampled one symbol time apart, the two consecutive symbols out of the interleaver must have been separated by $bM-1$ (or more) symbols on the input stream. This gives $n_1 = bM-1$.

Calculating n_2 is a matter of determining the range of consecutive symbols leaving the interleaver for which the inter-symbol spacing on the input stream was at least $bM - 1$. Consider any symbol as it leaves the interleaver. As discussed above, the next symbol to leave will have occurred $bM - 1$ symbols earlier on the input stream. The third symbol will have occurred $2bM - 2$ symbols earlier. In general, two symbols leaving the interleaver separated by x symbols will have been separated by $xbM - x$ symbols on the input stream, provided both symbols leave during the same pass down the bank of delay lines. Again, two symbols separated by x symbols have $x - 1$ symbols between them. From this it is easy to show that during any one pass down the delay lines of the interleaver, any range of output symbols has pairwise separation on the input stream of at least $bM - 1$.

Moving from the end of one pass to the beginning of the next, the symbols which are leaving the interleaver change from being older and older in terms of when they were written into the interleaver to being the most recent symbol into the interleaver. The pairwise separation on the input stream of two symbols read out in consecutive passes down the delay lines must be determined. Labelling the delay lines of Figure 3.1 from 1 to M and calling the position of one symbol leaving the interleaver on the first pass z and the position of the other symbol leaving the interleaver on the second pass y, expressions for the pairwise separation are obtained as follows. Between position z and position M on the first pass the pairwise separation is $(M - z)\,bM - (M - z)$. Between position M on the first pass and position 1 on the second pass, the pairwise separation changes by $-1 - (M - 1)\,bM$ where the negative sign on the second term indicates a separa-

tion on the input stream that has moved ahead in time rather than behind. Between position 1 and position y on the second pass, the pairwise separation changes by $(y-1)\,bM-(y-1)$. Summing these three expressions and simplifying gives the total pairwise distance as

$$d_{\text{pair}} = (y-z)\,(bM-1)-M$$

The value n_2 will be one less than the first spacing which causes $|d_{\text{pair}}| < bM-1$. This gives

$$|d_{\text{pair}}| = |(y-z)\,(bM-1)-M| < bM-1$$

which after scaling by $bM-1$ gives

$$\left|(y-z)-\frac{M}{bM-1}\right| < 1 \tag{3.3}$$

This inequality is false for all y less than z, or for spacing on the output stream of M or less. For y equal to z, or for a spacing of $M+1$, (3.3) is false if $b = 1$ and true if $b > 1$. For $y = z+1$ (3.3) is true for all values of b. This analysis shows that Forney's convolutional interleaver is a $(M, bM-1)$ interleaver for $b > 1$ and a $(M+1, bM-1)$ interleaver for $b = 1$.

The depth of Forney's convolutional interleaver can be similarly calculated. If the bank of M delay lines is used to transmit codewords of length $n = M$ then any burst on the channel of $bM+1$ errors or less affects any one codeword only once. This gives the interleaving depth, $I = bM+1$. Proving this consists of showing that a block of M consecutive symbols read into the interleaver's delay lines from

line 1 to line M are pairwise separated by at least $bM + 1$ symbols at the output of the interleaver.

Using the moment the first symbol of the block of M symbols leaves the interleaver as a reference, the spacing of the subsequent $M - 1$ symbols can be calculated. The first symbol experiences zero delay and is written to the channel immediately. The second symbol is blocked by a b symbol delay line clocked every M symbols, giving a spacing of bM symbols plus the one symbol spacing that initially existed. The third symbol is blocked by a $2b$ symbol delay line clocked every M symbols, giving a spacing of $2bM$ symbols plus the initial two symbol spacing.

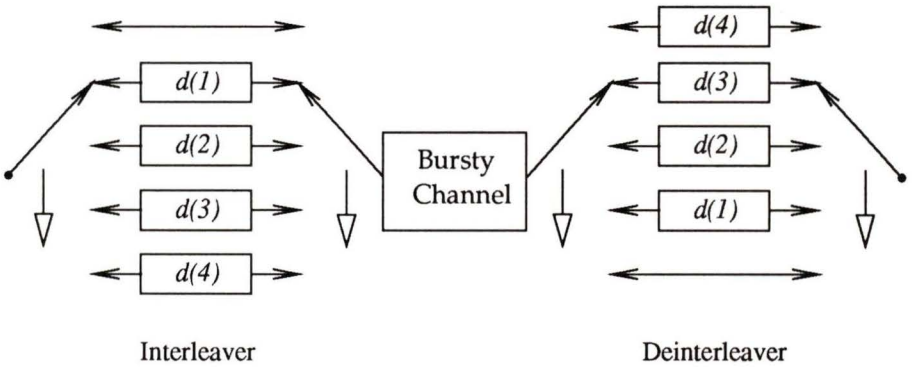

FIGURE 3.2: Forney's Interleaver for M=5, b=1

In general, the xth symbol of a block of M symbols aligned with the M delay lines is separated from the first symbol of the block by $(x-1)\,bM + (x-1)$ on the output of the deinterleaver. Choosing any pair of symbols from the input block it can be shown that the minimum spacing on the output of the deinterleaver occurs between symbols which were adjacent on the input, and this minimum spacing,

and thus the interleaving depth of Forney's interleaver, is $bM + 1$.

An example of Forney's interleaver is given in Figure 3.2 for $M = 5$ and $b = 1$. This configuration is therefore a $(6, 4)$ interleaver with interleaving depth $I = 6$. To show that these performance parameters are valid, label each block of 5 symbols entering the interleaver a_x, b_x, c_x, d_x, e_x where the subscript x starts at 1 and is incremented by 1 for each block read into the interleaver. The first block is a_1, b_1, c_1, d_1, e_1, the second block is a_2, b_2, c_2, d_2, e_2 and so on. The data flow into the interleaver and the data flow onto the channel are shown in Figure 3.3 simultaneously versus time measured in symbol periods.

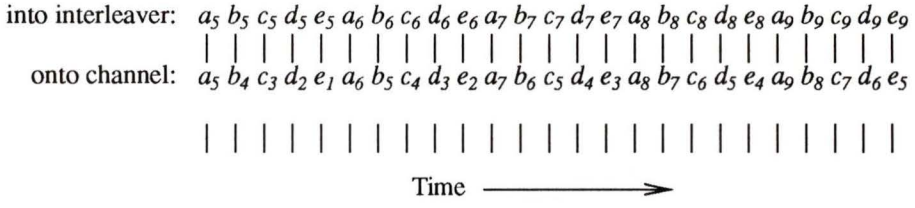

FIGURE 3.3: Data Flow for Forney's Interleaver

Notice how as a symbol is read into the interleaver, a symbol from the same position in a block, but from a different block, is written onto the channel. There is a certain order preservation in this behavior which is intuitively satisfying. By inspection it can be seen that no set of 6 contiguous symbols on the channel were pairwise separated on the input to the interleaver by less than 4 symbols. Also note that an error burst of 6 symbols would only hit one symbol of any particular 5 symbol code block.

Forney generalized his interleaving scheme [4] with what he called a modular

periodic interleaver. A description of this interleaver is included here for completeness without any analysis or proof of its function. In Figure 3.1, the delay lines of the interleaver are selected sequentially from delay 0 to delay $(M-1)\,b$. The modular interleaver places designer control over the order in which the delay lines are used. If the convolutional interleaver is labelled a $\{M, b\}$ interleaver then the modular interleaver is labelled a $\{M, b, m\}$ interleaver where m defines the ordering of delay in the following way.

The order of delays, d_i with i running from 0 to $M-1$, is given by $d_i = r_i b$ where $r_i = im$ modulo M. This produces reasonable results with all M delays from 0 to $(M-1)\,b$ used once and only once if three conditions on m are met:

1. $1 \le m \le M-1$
2. M and m are relatively prime.
3. M and $mb+1$ are relatively prime.

From this it can be seen that a straight convolutional interleaver is a $\{M, b, 1\}$ interleaver using the modular interleaver notation.

3.2 Ramsey's Convolutional Interleaver

Forney's interleaver is realized as a bank of M autonomous shift registers. Ramsey envisioned his interleaver differently, conceiving instead a single long shift register tapped along its length for input or output. As will be shown, particular realizations of Ramsey's interleaver can be made to perform the same function as Forney's interleaver.

Ramsey describes four interleavers which perform the interleaving and deinterleaving tasks for various combinations of n_2 and n_1. The designer must first determine the desired (n_2, n_1) performance of the interleaving task, then choose the interleaver which matches those values. The four types of interleavers and the ranges of (n_2, n_1) they cover are

Type I	n_1 and $n_2 + 1$ relatively prime, $n_1 > n_2 + 1$.
Type II	n_2 and $n_1 + 1$ relatively prime, $n_2 > n_1 + 1$.
Type III	n_1 and n_2 relatively prime.
Type IV	n_1 and n_2 relatively prime.

If a type I (n_2, n_1) interleaver is required, a type II (n_1, n_2) interleaver will perform the deinterleaving task. Conversely, a type I (n_1, n_2) interleaver is the deinterleaver for a type II (n_2, n_1) interleaver. The same relationship holds for the type III and type IV interleavers, that is, a type III (n_2, n_1) interleaver uses a type IV (n_1, n_2) interleaver for its deinterleaver while a type III (n_1, n_2) interleaver is the deinterleaver for a type IV (n_2, n_1) interleaver. Only the type I and II interleavers will be described here. The configurations of type III and IV are similar to types I and II respectively.

A type I (n_2, n_1) interleaver is shown in Figure 3.4. It consists of an $n_2(n_1 - 1) + 1$ stage shift register with output taps at the first and last stage and at every $(n_1 - 1)$ th stage in between, giving a total of $n_2 + 1$ taps. The output taps are sampled beginning with the last stage of the shift register and moving tap by tap to the first stage of the shift register. The cycle then repeats beginning with the last stage again. Each symbol period a symbol is shifted into the register from the

input stream, the next tap is connected to the output stream and a symbol is written onto the channel. The delay experienced by each symbol passing through this interleaver varies from zero delay for symbols read from tap n_2 to a delay of $n_2(n_1 - 1)$ for symbols read from tap 0. In general, a symbol read from tap x will have been delayed by $(n_2 - x)(n_1 - 1)$ symbols in the interleaver.

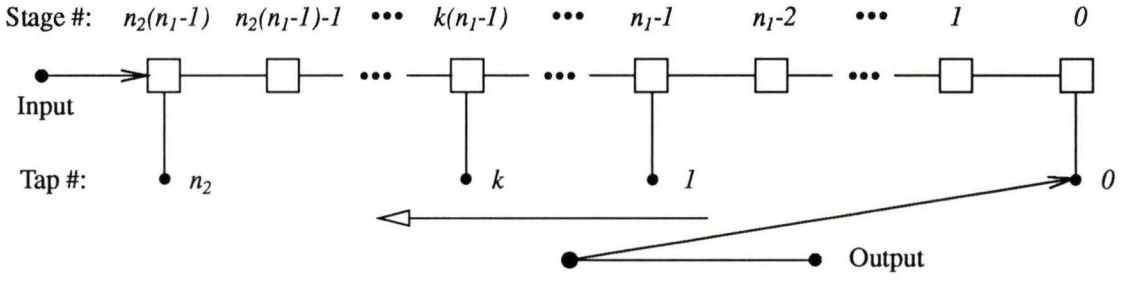

FIGURE 3.4: Ramsey's Type I Interleaver

It is not immediately evident by inspection of Figure 3.4 that a proper interleaving function is performed. Two properties of this interleaver will now be proved: (1) the interleaver is indeed an (n_2, n_1) interleaver, and (2) every input symbol is output once and only once.

To show property (1) assume the output is being read from tap 0 with the data stored in shift register stages 0 through $n_2(n_1 - 1)$ labelled a_k through $a_{k+n_2(n_1-1)}$. Tap 0 is read which produces output a_k. A new symbol is shifted into the interleaver and the output is read from tap 1. Tap 1 originally contained symbol a_{k+n_1-1}, but because of the shift that accompanies each new symbol, tap 1 produces symbol a_{k+n_1}. This cycle continues giving the output sequence

$$a_k, a_{k+n_1}, a_{k+2n_1}, \ldots, a_{k+jn_1}, \ldots, a_{k+(n_2-1)n_1}, a_{k+n_2n_1} \tag{3.4}$$

The last symbol in this sequence comes from tap n_2. The next symbol in the sequence will come from tap 0, $n_2 (n_1 - 1)$ stages earlier in the input data sequence. This means the next symbol in the sequence is $a_{k+n_2n_1-n_2(n_1-1)+1} = a_{k+n_2+1}$ where the addition of 1 to the symbol number accounts for the shift which precedes each tap being read. The sequence thus continues from (3.4) as

$$a_{k+n_2+1}, a_{k+n_2+n_1+1}, a_{k+n_2+2n_1+1}, \cdots, a_{k+n_2+jn_1+1}, \cdots \tag{3.5}$$

Obviously, any set of symbols from either sequence (3.4) or sequence (3.5) has the required pairwise separation of n_1 symbols on the input stream. However, n_2 contiguous symbols which includes symbols from both (3.4) and (3.5) must be more closely examined.

For a sequence of n_2 symbols to cross the boundary between (3.4) and (3.5) the first symbol must be a_{k+jn_1}, where $2 \le j \le n_2$. The last symbol in the sequence will then be $a_{k+n_2+(j-2)n_1+1}$. The lowest symbol from sequence (3.4) is a_{k+jn_1} and the highest symbol from sequence (3.5) is $a_{k+n_2+(j-2)n_1+1}$. In order that these symbols are separated by n_1 symbols on the input stream

$$k + jn_1 - (k + n_2 + (j-2)n_1 + 1) \ge n_1$$

which gives

$$2n_1 - n_2 - 1 \ge n_1 \quad \text{or} \quad n_1 \ge n_2 + 1$$

which is one of the conditions given earlier for proper functioning of a type I interleaver.

To show property (2), assume an arbitrary tap j is being sampled for output when symbol a_0 is first shifted into the interleaver. Symbol a_0 will reside at tap k after $(n_2 - k)\ (n_1 - 1)$ shifts. The output at this time will be sampled from tap $\{j + (n_2 - k)\ (n_1 - 1)\}$ mod $(n_2 + 1)$. In order for a_0 to be read as output from tap k, the following must hold

$$\{ (k) - (j + (n_2 - k)\ (n_1 - 1)) \} \bmod (n_2 + 1) = 0$$

which after subtracting all factors of $(n_2 + 1)$ gives

$$\{ (k+1)\, n_1 - j - 1 \} \bmod (n_2 + 1) = 0$$

$$\text{or} \quad (kn_1) \bmod (n_2 + 1) = C \quad \text{where} \quad C = (n_1 - j - 1) \bmod (n_2 + 1) \tag{3.6}$$

For (3.6) to be satisfied for one and only one value of k, n_1 and $(n_2 + 1)$ must be relatively prime. This is one of the conditions for proper functioning of a type I interleaver, therefore each symbol into the interleaver is output once and only once.

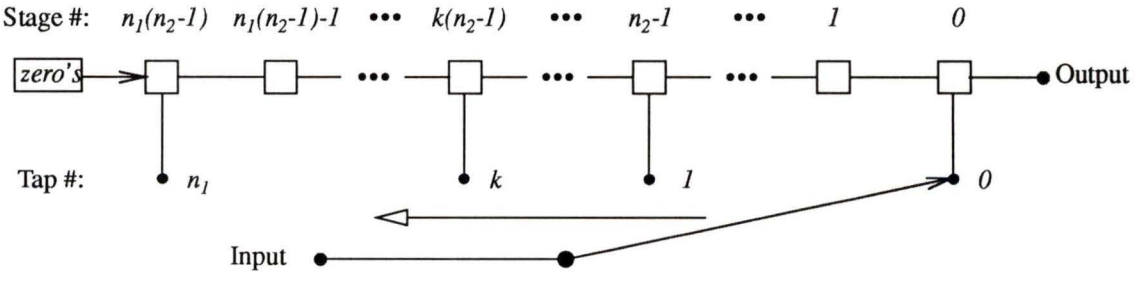

FIGURE 3.5: Ramsey's Type II Interleaver

The deinterleaver for a type I interleaver is a type II interleaver. A type II (n_2, n_1) interleaver is shown in Figure 3.4. This is a $n_1\ (n_2 - 1) + 1$ stage shift register with

input taps at the first and last stage and at every $(n_2 - 1)$ th stage in between, giving a total of $n_1 + 1$ taps. Symbols are read into the interleaver beginning with the last stage of the shift register and moving tap by tap to the first stage of the shift register. For every symbol read into the interleaver, the shift register shifts another symbol to the output. A symbol input to tap x experiences a delay of $x(n_2 - 1)$ before being read out of the interleaver.

In order for a type I (n_2, n_1) / type II (n_1, n_2) combination to act as an interleaver/deinterleaver pair, there must be synchronization between the output taps of the interleaver and the input taps of the deinterleaver. A symbol leaving tap x of the interleaver must enter tap x of the deinterleaver for all x, $0 \le x \le n_2$. The total delay for any symbol passing through a type I / type II interleaving system connected to operate in either direction is therefore

$$D_{\text{Ramsey}} = (n_2 - x)(n_1 - 1) + x(n_1 - 1) = n_2(n_1 - 1) \quad \text{type I to type II}$$

$$D_{\text{Ramsey}} = x(n_2 - 1) + (n_1 - x)(n_2 - 1) = n_1(n_2 - 1) \quad \text{type II to type I} \qquad (3.7)$$

By inspection the total storage required by Ramsey's interleaver is

$$S_{\text{Ramsey}} = 2[n_2(n_1 - 1) + 1] \quad \text{type I to type II}$$

$$S_{\text{Ramsey}} = 2[n_1(n_2 - 1) + 1] \quad \text{type II to type I} \qquad (3.8)$$

This storage requirement is twice the optimum given by the delays of (3.7). Storage requirements can be reduced by a more complex shifting algorithm for the shift register which eliminates the undesirable property of data which has already been used or unused zeros taking up space in the interleaver's shift register.

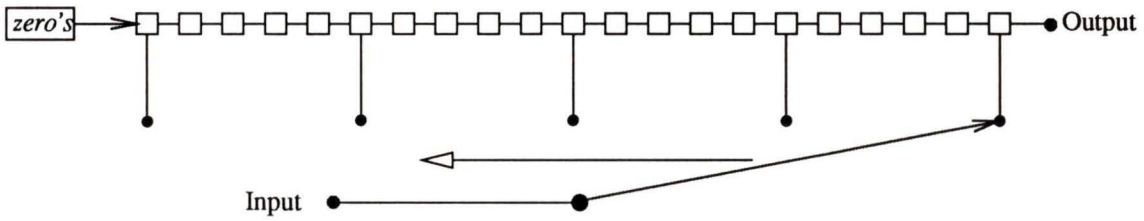

FIGURE 3.6: A Type II (6,4) Interleaver

It was mentioned earlier that Ramsey's interleaver and Forney's interleaver can perform the same interleaving function. The example of a Forney interleaver shown in Figure 3.2 is an $n_2 = 6, n_1 = 4$ interleaver. These parameters demand the use of a type II $(6, 4)$ interleaver with 5 input taps and 21 stages. Such an interleaver is shown in Figure 3.6. The data flow for this example is identical to the Forney example and is shown in Figure 3.3. The deinterleaver for Figure 3.6 is a type I, $n_2 = 4, n_1 = 6$ interleaver with 5 output taps and 21 stages.

Chapter 4

Pseudo-Random Interleaving

Both the convolutional and block interleaving techniques are effective against random noise bursts. However, they share the potential weakness of an output sequence with a relatively short period. For this reason, convolutional and block interleaving are placed in the class of interleavers called periodic interleavers. Periodic interleavers have the property that any two adjacent symbols from a particular point in the period of the input sequence are separated by C symbols at the output. An intelligent jammer pulsing its energy on the channel every C symbols will create an error burst in the data after deinterleaving, effectively blanking out the data stream.

Even in a non-military environment, there are situations where a periodic noise source could blank out a periodic interleaver. Radar transmitters, inefficient motors which arc every revolution and the ignition systems on motor vehicles are examples of potential periodic noise sources. Such benign sources can, however, be countered by designing the interleaver with a different period from that of any anticipated noise.

The solution to countering an intelligent jammer is to transmit the data in a perfectly random fashion. Such a scheme is difficult to realize if only for the need of coherent deinterleaving. A compromise is to transmit the data according to a very long pseudo-random sequence. Two such schemes as well as a simple method to prevent short bursts after deinterleaving are presented here.

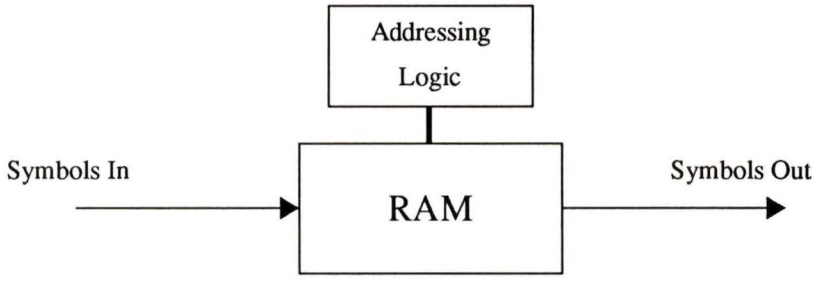

FIGURE 4.1: Basic Configuration of Pseudo-Random Interleaver

The basic configuration of a pseudo-random interleaver, which is, in fact, a simple schematic for any interleaver, is shown in Figure 4.1. The symbols on the input stream are read into the RAM and then written to the channel in a random fashion. Any random addressing logic is suitable so long as it guarantees that every symbol read into the RAM is written out once and only once. One method [5] which guarantees this uses the recursive addressing sequence given by

$$A_{n+1} = (aA_n + c) \mod L \tag{4.1}$$

for a RAM size of L. For sequence (4.1) to have period L with all values from 0 to $L-1$ represented (a maximum length sequence), a and c must be chosen according to

1. $a$and c less than L
2. c and L relatively prime
3. all prime factors of L must also be factors of $(a-1)$
4. if 4 is a factor of L, 4 must also be a factor of $(a-1)$

To increase the effective period of the interleaver, valid pairs of a and c can be stored and swapped into sequence (4.1) every L symbols. For p stored pairs of a and c, the output sequence is p permutations of sequence (4.1) for an effective period of pL.

Sequence (4.1) is not perfectly random, in fact, it exhibits certain regularities at large submultiples of L. Because of this, any noise source operating with a period which divides L may, if the phase of the noise is aligned with the regularities of sequence (4.1), defeat the interleaving effort and corrupt the data stream. To counter this problem it is suggested [6] that sequence (4.1) be further mixed according to the output of a length m, where $2^m = L$, linear feedback shift register. The shift register would be tapped according to the coefficients of an mth degree primitive polynomial to produce a maximum length sequence.

Another addressing sequence [2] is also based on an m bit maximum length linear feedback shift register. In this case the lower $(m-1)$ bits of an m-bit shift register provide the addressing sequence for an interleaving RAM of size 2^{m-1}. Again, the shift register is wired according to the coefficients of an mth degree primitive polynomial. The sequence has period $2^m - 1$ with all addresses from 1 to $2^{m-1} - 1$ appearing twice per period and address 0 appearing once per period. The shift register state giving address 0 is a 1 followed by $(m-1)$ 0's. This state

receives special attention and is labelled β.

Each of $2^m - 1$ contiguous symbols through the interleaver experiences a unique delay of from one to $2^m - 1$ symbols. All the states of the shift register are 1 or 0 followed by the $(m-1)$ bits which represent the address. When a symbol is read in, it must wait for the shift register to produce the address of its location again to be read out. This will happen when the shift register produces a state equivalent to the one when the symbol was read in, except with the first bit flipped. If α is a root of the mth degree polynomial for which the shift register is wired, the state of the shift register at time t can be labelled α^t. If a symbol is read in during state $\alpha^t \neq \beta$, it will be read out during state α^τ where

$$\alpha^\tau = \alpha^t + \beta \tag{4.2}$$

The symbol delay passing through the interleaver is then $d = \tau - t$.
Equation (4.2) gives $\alpha^d = 1 + \beta\alpha^{-t}$ which is equivalent to

$$\alpha^t = \frac{\beta}{1 + \alpha^d} \tag{4.3}$$

For any delay, d, from one to $2^m - 2$, there is a unique value of t which solves equation (4.3). This leaves state β to be considered. State β occurs once every period, therefore a symbol read in during state β experiences a delay of $2^m - 1$. This has shown that the delays of symbols passing through the interleaver are uniformly distributed amongst the integers from 1 to $2^m - 1$.

Every symbol which experiences a delay of d through the interleaver must expe-

rience a delay of $2^m - 1 - d$ through the deinterleaver to keep the total symbol delay constant at $2^m - 1$ and maintain the symbol order. To accomplish this the interleaver and deinterleaver are identical. A symbol leaves the interleaver from a particular address and enters the deinterleaver at the same address. The symbol experiences a delay of d through the interleaver and because any one address is only accessed twice per sequence period of $2^m - 1$ and the sequencing of interleaver and deinterleaver are synchronous, the symbol must wait in the deinterleaver for an additional $2^m - 1 - d$ symbol times before the address comes up in the sequence again.

A symbol entering the interleaver during state β of the shift register will be delayed the full $2^m - 1$ symbols in the interleaver. To avoid this symbol receiving an additional $2^m - 1$ symbol delay at the deinterleaver, the deinterleaver will have to be specially set up to pass a symbol directly with zero delay during state β.

Any pseudo-random scheme like the ones described above have an interleaving depth of 1 and are no better than $n_2 = 2, n_1 = 1$ interleavers. This is because a random scrambling of the data does not guarantee that two consecutive symbols on the input stream are not transmitted to the channel consecutively. A scheme [7] designed with block codes in mind to interleave randomly but still guarantee that symbols of a particular codeword are separated with high probability by at least b symbols will now be presented.

To strive for separation, b, of symbols in any given codeword, each address in the interleaving sequence is selected according to the following rules.

1. All RAM addresses must be selected before reselecting any of them. Addresses already selected are banned until all addresses have been used.
2. When a symbol is written to the channel, no other symbol of the same codeword should be written out for $(b-1)$ selections. Addresses containing the other symbols of the codeword are temporarily banned for $(b-1)$ address selections.
3. If no addresses are available because of rule 1 and rule 2, only rule 1 need apply.

At each step in the sequence, an address is chosen randomly from the set of addresses allowed by the above three rules. When all addresses in the RAM have been selected all bans are lifted and a new permutation of the addressing sequence is generated with the same rules.

The minimum RAM size for this method is $N \times b$ where N is the length of a codeword. With an $N \times b$ RAM, these rules produce an output which is identical to that of a block interleaver. Evidently, a RAM size of $N \times (b+a)$ is required, where $a > 0$ to allow variable distance between two symbols of the same codeword. The distance performance is reputedly quite good. If $a = 1$, $N = 15$ and $b = 280$, the distance is never lower than 270 and is lower than $b = 280$ for only 0.1% of the symbols.

The deinterleaver for this scheme must be explicitly constructed from the permutations that compose the interleaving sequence. If $N(b+a)$ symbols are stored in the interleaver during one permutation of the interleaving sequence, they will all be written to the channel and then the deinterleaver during the next permutation

of the interleaving sequence. During the third permutation the symbols will be read out of the deinterleaver in order. Because each permutation of the addressing sequence lasts $N(b+a)$ symbols, the total delay due to this interleaving scheme for any symbol is $2N(b+a)$.

The final topic in this section is a simple method [8] to protect random error correcting codes such as convolutional codes from errors in consecutive symbols which occur when an error burst on the channel exceeds the interleaving depth of a block interleaver. The solution to this problem is to transmit the rows of the interleaving block in a random fashion designed to maximize the number of rows transmitted between adjacent rows of the interleaving block.

If a burst of errors on the channel is long enough to affect a rows of the interleaver output, then the following algorithm will provide the necessary separation between affected rows to prevent consecutive errors at the deinterleaver output. N is the length of the interleaver column, d is the greatest common factor of N and a, and the rows are labelled from 0 to $N-1$. R_i gives the row number to send at time i, where one row is sent per time interval.

For $j = d-1$ to 0
  For $k = 0$ to $\left(\frac{N}{d}-1\right)$
    $R_i = (ak+j) \bmod N$
    $i = i+1$

This algorithm can be shown to be optimal for the purpose of providing the maximum spacing between adjacent rows.

Chapter 5

Helical Interleaving

The block interleaver was shown to have a very simple structure, both conceptually and operationally. However, the block interleaver suffers from an overly large delay for the amount of burst protection it offers. A slight modification to the block structure yields a helical interleaver [9]. It will be shown that the helical interleaving structure offers delay and storage advantages over the block interleaver.

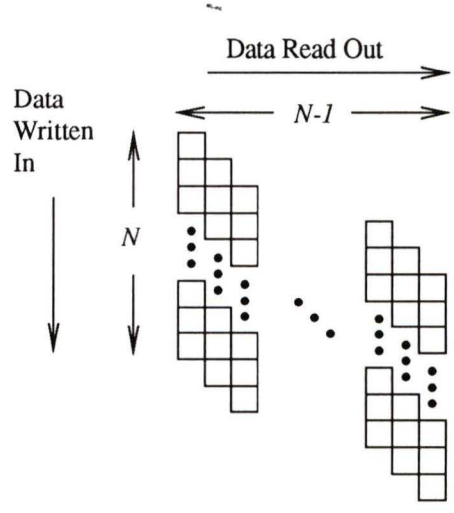

FIGURE 5.1: Helical Interleaving

Figure 5.1 depicts a generalized helical interleaver. Like a block interleaver, data

is written in column by column and read out row by row. Unlike a block interleaver, the beginning of each column of the helical interleaver is one row lower than the beginning of the previous column and the number of columns must be exactly $N-1$, where N is the number of symbols in each column. The restriction on the number of columns will later be relaxed somewhat, but for the basic helical interleaver described in [9] the number of columns is $N-1$. As was the case for the block interleaver, the number of columns is equivalent to the interleaving depth, I. The deinterleaving structure of a helical interleaver is the same as the interleaver except data is written into the deinterleaver row by row and read out column by column.

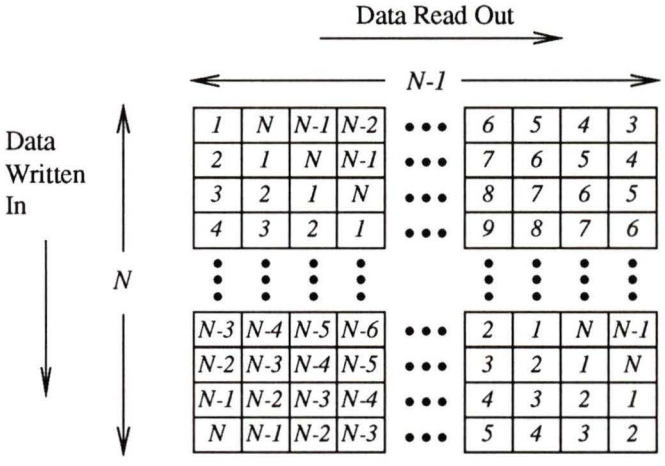

FIGURE 5.2: Helical Interleaver in Matrix Format

It will help, both from a visualization and an implementation perspective, to depict the helical interleaver in a matrix format. A helical interleaver confined to an N by $N-1$ block is shown in Figure 5.2. Each column accepts symbols in the labelled order from 1 to N. Once one column is filled, the next column begins

accepting symbols. This sequence continues until the last column is filled at which point symbols are written to the first column again. Symbols are read onto the channel row by row at the same time as symbols are written to the columns. The read/write sequence is synchronized so that the upper left symbol of the matrix is written in and read out simultaneously. Once this synchronization is arranged, the restriction of $N-1$ columns guarantees that the first symbol of every column is written in and read out simultaneously.

a_1	e_2	d_3	c_4
b_1	a_2	e_3	d_4
c_1	b_2	a_3	e_4
d_1	c_2	b_3	a_4
e_1	d_2	c_3	b_4

FIGURE 5.3: An N=5, I=4 Helical Interleaver

The data flow through a helical interleaver is best illustrated with an example. An $N = 5, I = 4$ helical interleaver is shown in Figure 5.3. For this example the same notation as was used in Figure 3.3 for Forney's convolutional interleaver is used. Each block of five symbols in the input data stream is labelled a_x, b_x, c_x, d_x, e_x where the subscript x represents the ordering of the block in the data stream. Figure 5.3 shows the contents of the interleaving matrix just after the fifth symbol of the fourth block, e_4, has been written in.

The data flow starting when symbol a_5 is written into the interleaver is shown in Figure 5.4. Note how the first symbol of every five symbol block is written into

the interleaver and read out of the interleaver onto the channel during the same symbol period. Note also that any of the symbols from a_1 to e_4 which are not shown to be read out in Figure 5.4 would have been read out during the previous 20 symbol read/write cycle.

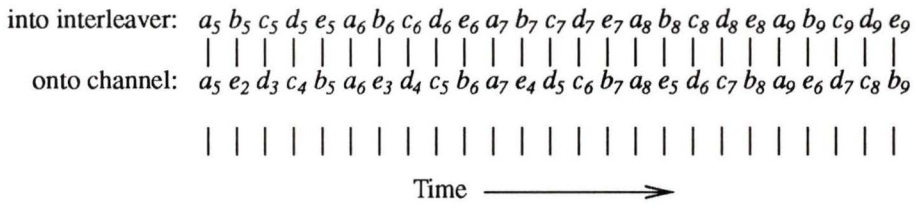

FIGURE 5.4: Data Flow for Helical Interleaver Example

The performance of a helical interleaver can be measured using Ramsey's (n_2, n_1) notation. To clarify the following evaluation it will help to refer to Figure 5.2. Consider the output sequence from a helical interleaver. In general the output sequence is of the form $\ldots, N, N-1, N-2, \ldots, 3, 2, 1, N, N-1, \ldots$ where the numbers in the sequence represent the column positions of symbols in the interleaver as they are read out. A symbol read out of position x in column y will always be followed by a symbol read out of position $x-1$ in column $y+1$ provided the following substitutions are made: if $x-1 = 0$, $x \leftarrow N$ and if $y+1 = N$, $y \leftarrow 1$.

Two adjacent symbols read out of the interleaver will have been separated on the input stream in two possible ways. The most likely possibility is that the two symbols are from two N symbol blocks which were adjacent on the input stream. In this case, for any column position x of the first symbol, the symbol separation

on the input stream will have been $(N-x) + (x-1)$ or $N-1$. The other possibility is that the first symbol is read from position 1 of column y and the second symbol is read from position N of column $y+1$. Because the first symbol of every column is written in and read out simultaneously, the second symbol of the pair is part of an N symbol block which was written to the interleaver $N-2$ blocks before the block containing the first symbol of the pair. This gives a symbol separation on the input stream of $(N-3)N+1$ for this special case.

For any value of N greater than 3, that is, for any realistic interleaving application, the symbol separation on the input stream of any adjacent pair of symbols on the output stream will have been at least $N-1$. Therefore, n_1 will equal $N-1$ for some range of output symbols, n_2. To find n_2, consider a general sequence of output symbols. The second symbol in the sequence will have entered the interleaver, or occurred, $N-1$ symbols later than the first. The third symbol will have occurred $N-1$ symbols later than the second and $2(N-1)$ symbols later than the first. This sequence of relationships will continue until the xth output symbol comes from position 1 of an interleaver column. This symbol will have occurred $(x-y)(N-1)$ symbols later than the yth symbol in the sequence, provided $x > y$. The $x+1$th symbol comes from position N of a column, and as described previously, will have entered the interleaver $(N-3)N+1$ symbols earlier than the xth symbol in the sequence. The zth symbol of the sequence, where $z > x+1$, will have occurred $(z-(x+1))(N-1)$ symbols later than the $x+1$th symbol. Combining these terms gives an expression for the separation on the input stream of two symbols of a sequence from the output stream where the position in the

sequence of the first symbol is y and the position of the second symbol is z.

$$(z-y-1)\ (N-1)\ -\ (N-3)\,N-1 \tag{5.1}$$

(5.1) is valid only if a symbol from column position 1 followed by a symbol from column position N forms part or even all of the sequence between symbol y and symbol z. For y to z not containing symbols from column position 1 followed by column position N, the separation is given simply by

$$(z-y)\ (N-1) \tag{5.2}$$

(5.1) and (5.2) are positive for symbol z occurring later on the input stream than symbol y and negative for symbol z occurring earlier than symbol y.

By experimentation with expressions (5.1) and (5.2) it can be shown that a sequence of $N-2$ or less output symbols will always have pairwise separation on the input stream of at least $N-1$ symbols. A sequence of $N-1$ output symbols will have the first and last symbols separated by only $N-2$ symbols on the input stream unless the sequence begins with a symbol from position N or position $N-1$ of a column. A sequence of N output symbols will have the first and last symbols separated by only one symbol on the input unless the sequence begins with a symbol from position N of a column. All $N+1$ symbol output sequences will contain at least one pair separated by only one symbol on the input stream.

Because an (n_2, n_1) interleaver must guarantee the stated performance for all output sequences, a particular sequence with better than average performance must be ignored. Consequently, a helical interleaver is both a $n_2 = N-2 = I-1$,

$n_1 = N-1$ or a $n_2 = N-1 = I$, $n_1 = N-2$ interleaver. The example shown in Figure 5.3 is therefore a $(3, 4)$ or a $(4, 3)$ interleaver with interleaving depth $I = 4$.

The symbol delay due to helical interleaving will now be discussed. Investigating the delay also provides some insight into the design of the deinterleaver. Consider any of the $N-1$ codewords shown in Figure 5.2. As mentioned earlier, the first symbol of any codeword is simultaneously written to the interleaver and read out to the channel, giving a delay of zero for symbol 1 of any codeword. The second symbol is written to the interleaver and read out $N-2$ symbol periods later. The third symbol is written to the interleaver and read out $2\ (N-2)$ symbol periods later. In general, the xth symbol of an N symbol codeword experiences a delay through the interleaver, in symbol periods, of

$$d_i = (x-1)\ (N-2) \tag{5.3}$$

For example, the 5 symbols of any codeword written into the interleaver shown in Figure 5.3 will be delayed in order from symbol a to symbol e by 0, 3, 6, 9 and 12 symbol periods respectively. Every codeword passing through the interleaver experiences the same sequence of delays. Because of this the deinterleaver need only acquire synchronization modulo N, the length of a codeword.

The sequence of delays also indicates how the read/write sequence of the deinterleaver must be arranged. An interleaver is designed to provide variable delay for symbols across the channel. The deinterleaver then serves to force the total delay

through the interleaving system to be constant for every symbol. This constant total delay guarantees that the order of symbols is maintained. To achieve a constant total interleaving delay while minimizing this delay is the goal of a good deinterleaver.

The delay through the helical interleaver ranges from zero delay for the first symbol of a codeword to $(N-1)\ (N-2)$ for the last symbol. Ideally, the deinterleaver will add no more delay to the last symbol, giving a total interleaving delay of $(N-1)\ (N-2)$ for each symbol. A deinterleaver which achieves this will delay the first symbol of a codeword by $(N-1)\ (N-2)$ symbol periods, the second symbol by $(N-2)\ (N-2)$ symbol periods and so on to a delay of zero for the last symbol of a codeword. In general, the xth symbol of a codeword should experience a delay through the deinterleaver, in symbol periods, of

$$d_d = (N-x)\ (N-2) \tag{5.4}$$

The data flow through the deinterleaver is arranged so that symbols are written from the channel into the deinterleaver row by row. At the same time symbols are read out onto the datastream column by column and thus codeword by codeword. To achieve minimum and constant total delay for each symbol, the read/write sequence is synchronized so that the last symbol of each codeword is simultaneously written in and read out of the deinterleaver. This synchronization guarantees the correct distribution of delays for each symbol of each codeword.

a_1	e_{-2}	d_{-1}	c_0
b_1	a_2	e_{-1}	d_0
c_1	b_2	a_3	e_0
d_1	c_2	b_3	a_4
e_1	d_2	c_3	b_4

FIGURE 5.5: An N=5, I=4 Helical Deinterleaver

As an example, consider Figure 5.5 which represents an $N = 5, I = 4$ helical deinterleaver just after the fourth symbol of the fifth row, b_4, has been written in from the channel. The data flow starting when symbol a_5 is written into the deinterleaver is shown in Figure 5.6.

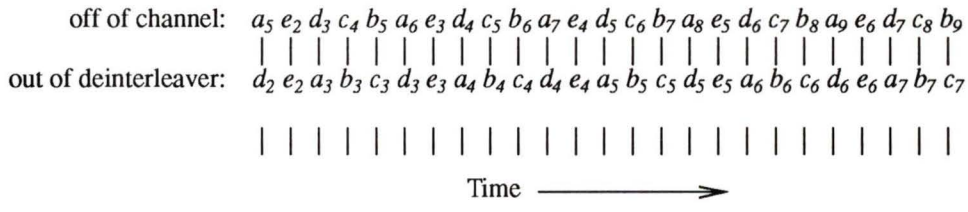

FIGURE 5.6: Data Flow for Helical Deinterleaver Example

Note how the last symbol of every five symbol block is written into the deinterleaver and read out of the deinterleaver onto the data stream during the same symbol period. Also, observe how the delay for a symbol passing through the interleaver in Figure 5.4 plus the delay for that symbol passing through the deinterleaver in Figure 5.6 is equal to 12 for every symbol.

5.1 Variable Depth Helical Interleaving

Further investigation of the form of helical interleaving reveals that it is not necessary to restrict the interleaving depth to $N-1$. The unique quality of a helical interleaver is to provide the same pattern of delays for the symbols of any N symbol block passing through it. The delays follow the sequence $d_i = (x-1)\ (I-1)$, where x is the position from 1 to N of a symbol in a block and I is the interleaving depth. The function of variable depth interleaving [10] is to maintain this pattern of delays without requiring $I = N-1$.

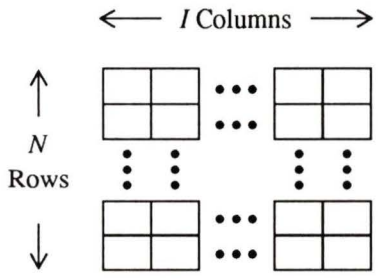

FIGURE 5.7: A General Helical Interleaver

A generalized helical interleaver in matrix format is shown in Figure 5.7. The standard helical read/write sequence will apply, that is, the first symbol of every column is written into and read out of the interleaving matrix simultaneously. Symbols are written into the matrix one column at a time at the same time as symbols are read out of the matrix onto the channel row by row from the top down. If the columns are numbered from 0 to $I-1$ and the rows are numbered from 0 to $N-1$, and the first codeword is written into column 0 starting from the upper left position in the matrix, then the second codeword will be written into column

N modulo I starting at row N div I. The function x div y represents the value when x is divided by y with the remainder ignored. This placement of the second codeword ensures that its first symbol is written and read simultaneously.

In general, the kth codeword will be written into column $(k-1)N$ modulo I starting at row $[(k-1)N$ div $I]$ modulo N. For this ordering to work, all I available columns must be used before any column need be used again. More strictly, each column must be selected exactly I codewords since the last time that column was selected. If I is relatively prime to N, the above condition will be met. This gives the only restriction on the depth of a helical interleaver.

a_1	c_4	e_3	g_2
b_1	d_4	f_3	a_2
c_1	e_4	g_3	b_2
d_1	f_4	a_3	c_2
e_1	g_4	b_3	d_2
f_1	a_4	c_3	e_2
g_1	b_4	d_3	f_2

FIGURE 5.8: An N=7, I=4 Helical Interleaver

For example, if a data stream is to be helically interleaved into columns of length $N = 7$, the choice of available depths is far larger than just $I = N - 1 = 6$. Because I need only be relatively prime to $N = 7$, the possible choices for interleaving depth are 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 22 ... Any value of I not divisible by 7 is a candidate, giving a virtually unlimited number of selections to the designer. A helical interleaver with $N = 7$ and $I = 4$ is shown in

Figure 5.8. Note how the columns are written in not sequentially but rather in a manner which guarantees that the first symbol, a_x, of every column is written in and read out simultaneously.

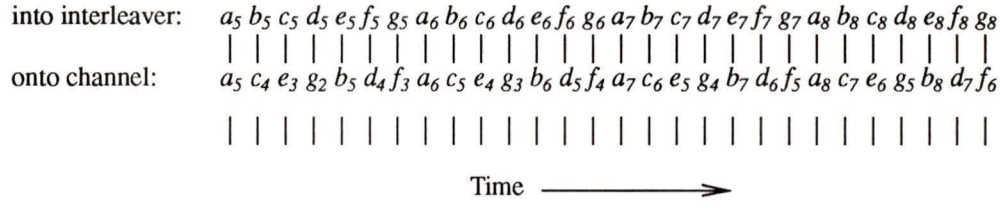

FIGURE 5.9: Data Flow for N=7, I=4 Helical Interleaver

Figure 5.9 shows the data flow starting when symbol a_5 is written into the interleaver. Note how the sequence of delays for each symbol follows the pattern $(x-1)\ (I-1)\ =\ 3\,(x-1)$ where x is the position of the symbol in the column.

a_1	b_6	c_4	a_2	b_7	c_5	a_3
b_1	c_6	a_4	b_2	c_7	a_5	b_3
c_1	a_6	b_4	c_2	a_7	b_5	c_3

FIGURE 5.10: An N=3, I=7 Helical Interleaver

As a second example, consider columns of length $N\ =\ 3$. This gives possible helical interleaving depths, I, of 2, 4, 5, 7, 8, 10, 11, 13 ... Choosing $I\ =\ 7$ gives the helical interleaver shown in Figure 5.10. The data flow starting when symbol a_8 is written into the interleaver is shown in Figure 5.11. The sequence of delays for individual symbols passing through this interleaver is $6\,(x-1)$.

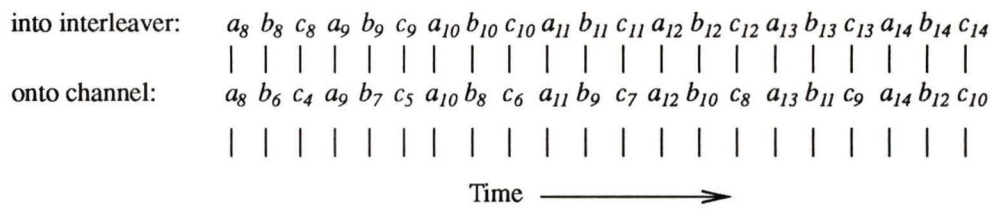

FIGURE 5.11: Data Flow for N=3, I=7 Helical Interleaver

The design of the deinterleaver for variable depths is the same as for the previously discussed case of $I = N - 1$. The read/write sequence is synchronized so that the last symbol of each column is written in and read out simultaneously. This gives a sequence of delays through the deinterleaver of $d_d = (N - x)\ (I - 1)$. Summing this expression with the expression arrived at for the interleaver delay, d_i, gives

$$D = (N - 1)\ (I - 1) \tag{5.5}$$

the general expression for the total interleaving delay due to helical interleaving. Thus the total interleaving delay for the interleaver of Figure 5.8 is 18 symbol periods and that of Figure 5.10 is 12 symbol periods.

The interleaving depth of variable depth helical interleaving is obviously I. The performance in terms of Ramsey's (n_2,n_1) notation requires somewhat more analysis. Consider any symbol exiting a variable depth helical interleaver, and call it α_x. Symbol α_x will be from position α of column x. Now consider a sequence of n_2 output symbols beginning with α_x. Call the sequence β. If $n_2 = I + 1$, sequence β will most likely end with symbol $(\alpha + 1)_x$, the symbol following α_x in column x. This is a separation of $n_1 = 1$ on the input stream to the interleaver

and is not a useful value. If $n_2 = I$, sequence β will contain one symbol from each column of the interleaver. The closest symbol on the input stream to α_x of the symbols in β will define the value of n_1 for $n_2 = I$.

Within any output sequence of length I or less, the smallest pairwise separation on the input stream of symbols in the sequence will arise from symbols resident in adjacent columns. Adjacent in this sense means columns which were written to the interleaver one after the other. Sequence β will, in general, contain one symbol from the column preceding column x and one symbol from the column following column x. Whichever of these two symbols is closest to symbol α_x on the input stream will define n_1.

Symbol α of column $x+1$ will always be written into the interleaver N symbols after symbol α of column x. If sequence β can be considered as the first of a series of I symbol virtual output rows, then symbol α_{x+1} will end up N div I rows below symbol α_x. This leads to the observation that within sequence β, the symbol from column $x+1$ is, in fact, symbol $\alpha - N$ div I. This gives a possible value for n_1 of $N - N$ div I, assuming the minimum pairwise spacing occurs between α_x and the symbol from the subsequent column.

The second spacing to be investigated is that between α_x and the symbol from the preceding column, column $x-1$, that occurs in sequence β. Again, considering the virtual output rows defined by sequence β, it is obvious that α_{x-1} occurred at least in the previous row. Then, by the same reasoning applied for the symbol from column $x+1$, symbol α_{x-1} occurred N div $I+1$ rows above symbol α_x.

Therefore, within sequence β the symbol from column $x-1$ is symbol $\alpha + N \text{ div } I + 1$. The spacing on the input stream between symbol $(\alpha + N \text{ div } I + 1)_{x-1}$ and symbol α_x is $N - N \text{ div } I - 1$, which is smaller than the spacing derived above between α_x and the symbol from the subsequent column. A general, variable depth helical interleaver is thus shown to be a $n_2 = I$, $n_1 = N - N \text{ div } I - 1$ interleaver.

For values of n_2 less than I it is difficult to determine n_1 as was done for the simple helical interleaver with $I = N - 1$. It is not so easy to generally predict for all cases exactly what columns contribute to an output sequence of less than I symbols. The derived result does predict the previously obtained values of (n_2,n_1) for the helical interleaver with $I = N - 1$. Substituting this value for I to the general expressions above gives $n_2 = N - 1$, $n_1 = N - 1 - 1 = N - 2$.

The example in Figure 5.8 is therefore a $n_2 = 4$, $n_1 = 5$ interleaver. The example in Figure 5.10 is therefore a $n_2 = 7$, $n_1 = 2$ interleaver. These performance measures can be visually confirmed by inspecting the output sequence of each example.

The storage required for helical interleaving depends upon the implementation used. The simplest method is to use an $N \times I$ symbol memory for the interleaver and an $N \times I$ symbol memory for the deinterleaver and adjust the addressing logic of each to match the read/write sequencing discussed earlier in this chapter. This method ensures the minimum total helical interleaving delay of $(N-1)\ (I-1)$ using more than twice the minimum total storage requirement.

A unique addressing scheme to eliminate wasted storage has been proposed [11]. The method only deals with the specific helical interleaver with $I = N - 1$, and there is no deinterleaver design described. Using their scheme, the authors imply a total storage requirement of $(N-1)\ (N-2)$, the minimum amount. This is at a slight cost in delay, apparently resulting in a total interleaving delay of $(N-1)\ (N-2) + 2$.

Chapter 6

Comparing Interleaving Methods

The interleaving methods described in this thesis will now be compared. There were four basic performance measures discussed for each type of interleaving: storage, delay, depth and Ramsey's (n_2, n_1) description of interleaved data. As memory becomes cheaper, faster and more compact, the limitation of storage is less of a concern. Also, for each interleaving method there are different ways to implement the interleaver which are more or less memory efficient than others. Generally, there will be an implementation which can achieve the minimum memory requirement (equation (1.4)) for the interleaver delay, therefore to compare delays will also be a comparison of storage requirements.

A final comment on storage is that it is not necessary for the same implementation to be used for both the interleaver and deinterleaver. If a system's receiver has stricter memory requirements than its transmitter, a more memory efficient algorithm can be used for the deinterleaver. It is the distribution of the interleaved data across the channel which defines a particular interleaving, not the implementation that achieves that distribution.

In addition to the performance measures mentioned above as a basis for comparison, there is the issue of synchronization. Each interleaving method requires that the data from the interleaver enter the deinterleaver at exactly the correct position so that the data is reordered correctly. The length of the data which must be synchronized was described as the synchronization modulo. The smaller the synchronization modulo, the more rapid the initial synchronization acquisition and the sooner synchronization can be regained should it be lost. Speedy acquisition and reacquisition of synchronization translates directly into lower data losses.

Block interleaving has poor synchronization requirements. Synchronization must be obtained for an entire $N \times I$ symbol block of data. For a simple synchronization scheme using one marker per block of data, every marker missed during initial or subsequent synchronization attempts will result in an entire block of data being lost. More complex synchronization algorithms are, of course, possible, but at the cost of greater complexity and a drop in information rate due to extra marker overhead.

Convolutional and helical interleaving have much better characteristics for synchronization. Synchronization need only be obtained on a block of M or N symbols respectively, due to the output delay symmetry of both methods. Less data is lost gaining initial synchronization and the recovery time from a loss of synchronization is shorter as compared with block interleaving. More will be said on the subject of synchronization in Chapter 7.

Block, convolutional and helical interleaving will now be compared in terms of

the delay cost versus the benefits of interleaving depth and (n_2, n_1) performance. Pseudo-random interleaving is not included in this comparison because it is a specialized method whose use transcends the basic measure of data distribution versus delay. Due to its very nature, the pseudo-random interleaver cannot guarantee anything better than a depth of 1 and $n_2 = 2, n_1 = 1$. However, where there exists an intelligent jammer the other interleaving methods may fail completely and it is in this capacity that pseudo-random interleaving may stand out as the superior interleaving choice in a hostile environment.

The delay and interleaving performance of three methods of interleaving are summarized in the table below.

Interleaving	**Delay**	**(n_2,n_1)**	**Depth**
Block	$2(N\text{-}1)(I\text{-}1)$	$(I\text{-}1, N)$ or $(I, N\text{-}1)$	I
Forney's Convolutional	$(M\text{-}1)bM$	$(M\text{+}1, bM\text{-}1)$ for $b\text{=}1$	$bM\text{+}1$
Helical	$(N\text{-}1)(I\text{-}1)$	$(I, N\text{-}N\text{div}I\text{-}1)$	I

The three interleaving schemes will be compared in pairs. For each pair of schemes the depth will be compared for a fixed delay or the delay will be compared for a fixed depth. The values of (n_2, n_1) will be compared for a fixed delay.

To equate the delay for the block and convolutional interleavers make $2(N-1)(I-1) = (M-1)bM$. Assuming $b = 1$ and solving for I gives

$$I = \frac{(M-1)M}{2(N-1)} + 1$$

To further simplify the expression, make $N = M$, a valid decision if a block code of length $M = N$ is being interleaved. This gives $I = M/2 + 1$. Thus for the same delay, interleaving same-length codewords, block interleaving will have depth $I = M/2 + 1$ while convolutional interleaving gives depth $M + 1$, almost twice the depth for a given delay. In terms of (n_2, n_1), using the same assumptions as above, for a given delay the block interleaver is either a $(\frac{M}{2}, M)$ or a $(\frac{M}{2} + 1, M - 1)$ interleaver while a convolutional interleaver is a $(M + 1, M - 1)$ interleaver. Again, the convolutional interleaver offers almost twice the protection across the reordered sequence for the same length of input sequence.

The helical interleaver is compared to the block interleaver by fixing the depth and comparing the delay. Assume the same blocklength, N. For a fixed depth of I a block interleaver has a delay of $2\,(N-1)\,(I-1)$ while the helical interleaver has a delay of $(N-1)\,(I-1)$, half that of the block interleaver. Fixing the delay of helical and block interleaving gives

$$I_{\text{block}} = \frac{I_{\text{helical}} + 1}{2}$$

giving values for (n_2, n_1) of $(\frac{I-1}{2}, N)$ or $(\frac{I+1}{2}, N-1)$ for the block interleaver versus $(I, N - N \operatorname{div} I - 1)$ for the helical interleaver.

Comparing the helical interleaver to the convolutional interleaver will be done by fixing the delay. Assuming $b = 1$, $N = M$ and fixing the delay gives $I = M + 1$. Substituting gives a depth of $M + 1$ for the helical interleaver, the same depth as for the convolutional interleaver. Using the same substitution gives (n_2, n_1)

equal $(M+1, M - M\text{div}I - 1)$ for the helical interleaver, the same as for the convolutional interleaver because, for this comparison, $I > M$ making $M\text{div}I$ equal 0.

Based on the these comparisons both helical and convolutional interleaving are significantly better than block interleaving while convolutional interleaving and helical interleaving are identical in performance. These statements must be qualified somewhat by clarifying that all applications will have different interleaving requirements and one method may be more suitable than another for a particular application. Block interleavers, though operating with double the delay of a comparable convolutional or helical interleaver, are the most flexible in terms of size. For any data column length, any interleaving depth is possible. Compare this to the helical interleaver which is restricted to depths which are relatively prime to the column length. Most restrictive of all is the convolutional interleaver, where the depth can only be multiples of the "column" length M plus one.

However, these restrictions are mainly the concern of applications where interleaving depth is the important performance measure, applications such as interleaving block codes. For synchronization purposes it is desirable to match the codeword length to the parameter M if convolutional interleaving is used, or to the parameter N if helical interleaving is used. If the interleaving application uses convolutional codes, Ramsey's (n_2, n_1) performance measure is more applicable than interleaving depth. In this case convolutional interleaving can be used by modifying parameters M and b to achieve the desired performance according to Ramsey's measure. Helical interleaving still offers greater flexibility in this

respect as the values for n_1 and n_2 are more independent of one another than are the values for convolutional interleaving.

A case has been made [12] that helical interleaving is equivalent to convolutional interleaving modified by a reverse permutation circuit. In the referenced paper, the author demonstrates how the output sequence from an $M = 4, b = 1$ convolutional interleaver can be manipulated to look like the output sequence from an $N = 4, I = 3$ helical interleaver. The manipulation involves reversing the output sequence of the convolutional interleaver in sets of four beginning with the first output symbol, then delaying the resulting sequence by 3 symbols. This reordering does indeed cause the output sequence to match that of the helical interleaver, however, reordering of data is the whole function of interleaving and to change the ordering of an interleaving output sequence changes the interleaving method and its resulting performance.

Consider the performance of the two interleavers used in [12]. The convolutional interleaver would give depth 5, $n_2 = 5$, $n_1 = 3$ for a delay of 12 while the helical interleaver would give depth 3, $n_2 = 3$, $n_1 = 2$ for a delay of 6. Clearly these are not "equivalent" interleavers. A more appropriate case could have been made by comparing an $M = 4, b = 1$ convolutional interleaver with a $N = 4, I = 5$ helical interleaver. These two interleavers can be shown to have identical performance characteristics and identical ordering of the output symbols. In fact, with a little investigation, it becomes clear that the convolutional interleavers described in this comparison simply define a subset of the possible interleavers realizable

through helical interleaving.

For some perspective on actual applications, the costs and benefits of interleaving for a realistic example will be described. A wireless voice communication system needs protection against error bursts encountered on the channel. The system already uses a (63, 43) Reed Solomon code capable of correcting ten six bit symbol errors per codeword. Data is transmitted along the channel using binary FSK at a rate of 64 kilobits per second. With no interleaving the system can absorb 60 bit, or 940 μs, error bursts with the Reed Solomon code alone. Unfortunately, due to a complex transmission environment, there are frequent error bursts of up to 4.5 ms duration on the channel. An 4.5 ms error burst means the loss of 288 bits on the channel, over four times what the Reed Solomon code can absorb.

Solving this problem with interleaving requires an interleaving depth of 5 or more. Block interleaving the codelength 63 Reed Solomon code to a depth of five would result in a total interleaving delay of 496 symbols, or 46.5 ms. Using helical interleaving halves that delay to 23.3 ms. Considering that the total one way delay for an entire voice communication system should be less than 90 ms [13] and in the future less than 30 ms, the block interleaver would be deemed unacceptable for this example while the helical interleaver is a possible solution. A convolutional interleaver would be inappropriate here as setting the parameter M to 63 gives an interleaving depth of at least 64, much greater than what is required and resulting in an interleaving delay of 122 ms.

Chapter 7

Recommendations for Future Work

7.1 Better Interleaving Performance Measures

The interleaving performance measures used in this thesis are interleaving depth and Ramsey's (n_2, n_1) rating. Interleaving depth simply measures the distance on the channel between symbols which were adjacent entering the interleaver. From the perspective of a block code, interleaving depth indicates the longest error burst which will affect a particular codeword only once. This is a relatively simple measure of interleaving performance.

The (n_2, n_1) measure comes closer to describing the level of re-ordering performed by an interleaver. However, to say an interleaver is an $n_2 = x$, $n_1 = y$ interleaver describes the worst case performance. Out of all possible x symbol sequences in an output period of an interleaver, one sequence might have a symbol pair separated on the input sequence by y symbols. All other pairs of all other sequences are separated by more than y symbols on the input sequence, yet the interleaver is still only an $n_2 = x$, $n_1 = y$ interleaver. Hypothetically, another interleaver may have every symbol pair from every x symbol output sequence

separated by y symbols on the input sequence, making this interleaver an $n_2 = x$, $n_1 = y$ interleaver also. Clearly, the first interleaver is better than the second, but this is not reflected in Ramsey's (n_2, n_1) performance measure.

An improved performance measure would include some statistical information in addition to the (n_2, n_1) information. An interleaver could be described as a 30% $n_2 = x_1, n_1 = y_1, 60\%$ $n_2 = x_2, n_1 = y_2, 100\%$ $n_2 = x_3, n_1 = y_3$ interleaver. Such a notation would provide a better sense of the nature of an interleaved data sequence.

7.2 Berlekamp's Synchronization Algorithm

In the course of the research towards this thesis, particular interest was initially taken in a novel synchronization algorithm presented by Berlekamp [2] in his larger discussion on helical interleaving. Berlekamp's synchronization method includes a certain character, the synch character, as the last character of each $N \times I$ character block which exits the interleaver. The method assumes symbol synchronization has already been achieved at the receiver and that the synch character is not unique. Any character from the symbol alphabet can be used for the synch character.

The receiver has $N \times I$ counters which are initially set to the value s. Each character which enters the deinterleaver is assigned a unique counter until all $N \times I$ counters are used up. Subsequent characters are then assigned counters sequen-

tially beginning again with the first counter. Thus, characters which are $N \times I$ characters apart on the data stream are assigned the same counter. As a character enters the deinterleaver, its counter is incremented by one if the character is the synch character or its counter is decremented by one if the character is not the synch character. Once a counter reaches the value zero it goes no lower. If a counter reaches some upper threshold, T, it goes no higher and is declared to be the counter for the synch character, the last character of each $N \times I$ block. The read write sequencing for the incoming data is rearranged to reflect this declaration and synchronization of the interleaving process is achieved.

Berlekamp shows that this synchronization scheme is extremely robust at maintaining the correct interleaving synchronization. Using starting values for the counters of s=12, an upper threshold for the counters of T=16 and a probability of character error of 0.10, this scheme will have acquired synchronization after 20 $N \times I$ blocks have passed through. Once synchronization is acquired, Berlekamp claims the mean free time between mis-synchs would be 1.9×10^5 centuries. This claim is based on a probability of the incorrect counter being incremented of $\frac{1}{32}$, which matches the case of 32-ary data transmitted across an error free channel. Despite the optimistic conditions of the above example, it is clear that once synchronization is established, Berlekamp's synchronization method will probably never lose synch for an appropriate value of T and most real channel error conditions.

There are two problems with Berlekamp's synchronization method, however.

The obvious one is the length of time, or more correctly, the amount of data it takes to acquire synch. For many communications applications there is not the provision of a large buffer to store incoming data until its arrangement is decided upon by a synchronization decision 20 blocks later. Generally, for each symbol received a symbol will be expected at the output with the minimum of delay. Thus for the above example, 20 N x I blocks will be thrown out before correct information appears at the output. The solution to this is not to wait for a counter to attain the value T but rather to monitor all the counters and immediately declare whichever counter is the highest at a given time to be the location of the synch character and arrange the read/write sequencing of the deinterleaver accordingly. In most cases, the correct counter will become evident long before it reaches the value T, particularly if T is a large value like 16. Synchronization will in this manner be acquired relatively quickly, while the power of the method for synchronization maintenance is not compromised.

The other problem with Berlekamp's method is that it ignores the possible problem of insertions or deletions on the channel. If a character is inadvertently skipped or an extra character added to the data stream, Berlekamp's synchronization scheme will be immediately defeated as it simply counts the characters to judge where in the deinterleaver each character belongs. Using Berlekamp's synch decision, fully T blocks of data will be lost regaining synch. Even using the maximum counter rather than the counter equal to T as the synch location would involve the loss of $\frac{T}{2}$ blocks of data. These losses are under ideal conditions with no probability of channel symbol error.

It becomes clear that to use lower values of T would result in less data lost for an insertion or a deletion. However, lowering T also makes the synchronization maintenance susceptible to random errors. It would appear that for given channel conditions, an optimum value of T for the synchronization algorithm exists which is small to minimize lost data from insertions and deletions, and yet is large enough to protect the synchronization from random channel errors. A large number of simulations were run to investigate this and other problems. Even though this investigation was abandoned, some interesting results were achieved which will be briefly presented here.

A computer simulation was created implementing an $N = 31, I = 30$ helical interleaver over a 32-ary symbol alphabet. One hundred $N \times I$ blocks of pseudo-random data were generated for each test, interleaved, corrupted in some manner on a sequential channel, and deinterleaved after synchronization had been acquired and maintained. The original data could then be compared to the deinterleaved data to obtain results on the damage which had occurred due to the corrupted channel.

Tests were run measuring the number of N symbol lines of data which were lost during the initial synchronization for values of T running from 1 to 20, values of s running from 0 to $T-1$ and independent probability of symbol error on the channel, p_s, running from $\frac{0}{128}$ to $\frac{65}{128}$ in $\frac{5}{128}$ increments. Five sets of such tests were run for five different positions within a block at which symbols were first presented to the deinterleaver. The choice of value for s, the starting value of the counters, had

little effect on the time to acquire synchronization except where s was zero, one or $T-1$ where the boundaries of zero and T for the counters would affect the natural progression of the counters. The five sets of tests all showed that values of T as low as 2 were just as efficient at acquiring synchronization as much higher values of T. This would be a direct consequence of selecting the synch position immediately for whichever counter is the highest. From the perspective of acquiring synchronization, then, there is no hindrance to the desire for a small T in order that deletions and insertions result in minimal data loss.

p_s	T	p_s	T	p_s	T
5/128	3	30/128	4	55/128	6
10/128	3	35/128	5	60/128	11
15/128	3	40/128	5	65/128	19
20/128	3	45/128	6		
25/128	6	50/128	7		

FIGURE 7.1: Synch Threshold Values for Variety of Symbol Error Probabilities

What will require a larger value of T is the need for protection against synchronization loss due to the probability of symbol error on the channel, p_s. Tests were run for values of T running from 1 to 20, values of s running from 0 to $T-1$ and independent probability of symbol error on the channel, p_s, running from $\frac{5}{128}$ to $\frac{65}{128}$ in $\frac{5}{128}$ increments. The total number of N symbol lines lost for 100 blocks sent across the channel was measured as an indication of synchronization loss due to random symbol errors on the channel. The lowest value of T for which no data was lost due to loss of synchronization or for which the data loss did not improve

for any higher value of T is given in Figure 7.1 for each value of p_s tested.

The data lost when synchronization is lost is generally equal to $\frac{T}{2} \times N \times I$, where $N \times I$ is the distance between synch characters. As discussed above, minimizing T will reduce the data loss. However, there is another parameter which can be adjusted to improve data loss during synchronization loss and that is the distance between synch characters. One of the benefits of helical interleaving is that synchronization need only be obtained mod N. Therefore, there are many more options for the placement of the synch character than simply every $N \times I$ characters in the data stream.

The deinterleaver read/write sequence is arranged so that the last character of each N symbol line is simultaneously read in and written out of the deinterleaver. If synch characters are placed periodically at the end of every t lines and $N \times t$ counters are set up, then the action for the synchronization algorithm to take should a counter indicate that it is reading into a synch position is to, if necessary, alter the read/write sequence to ensure that the character is read in and written out simultaneously. Assuming a correct decision was made regarding the location of the synch position, the subsequent read/write sequencing after this action will guarantee synchronized deinterleaving.

Experiments were run for the same simulation as the previous examples except in this case the density of synch characters was varied. It appears that the density of synch characters must divide I evenly in order for the algorithm to work, so synch characters were placed at the end of every line, every second line, every

third line, every fifth line, every sixth line, every ten lines, every fifteen lines and the old case of every thirty lines for a total of 8 different experimental synchronization arrangements. Numerous tests were run for the usual range of channel symbol error probabilities to try to determine optimum synch character densities for a given channel. The more synch characters used per block the lower the overall information rate of the communications system, but the lower the data loss in the case of synchronization loss. The results of the experiments were inconclusive, although there was a certain trend showing lower data loss with the higher synch character densities for higher channel symbol error probabilities.

The tests that were run in the investigation of Berlekamp's synchronization algorithm were first draft experiments. They simply showed trends which may or may not be valid given the simplifications made. Though there was some attempt to isolate the effects of different data loss mechanisms, there is room for improvement. More detailed simulations and, most importantly, theoretical verification would be required to validate the results that were obtained in these endeavors. However, it is safe to say that for a particular interleaving application, the synchronization algorithm can and should be optimized according to the expected channel conditions over which the communications system operates.

References

[1] J. L. Ramsey, "Realization of Optimum Interleavers", *IEEE Trans. on Information Theory*, vol. IT-16, pp. 338-345, May 1970.

[2] "Interleaved Coding for Bursty Channels", Prepared by Cyclotomics, Final Report on Phase I, Small Business Innovation Research, NSF Grant No. ECS-8260180, April 1983.

[3] G. D. Forney, Jr., "Burst-Correcting Codes for the Classic Bursty Channel", *IEEE Trans. on Commun. Technol.*, vol. COM-19, pp. 772-781, Oct 1971.

[4] G. D. Forney, Jr., "Interleavers", US Patent Number 3,652,998, March 28, 1972.

[5] I. Richer, "A Simple Interleaver for use with Viterbi Decoding", *IEEE Trans. Commun.*, vol. COM-26, pp. 406-408, March 1978.

[6] G. C. Clark, Jr., J. B. Cain, *Error-Correction Coding for Digital Communications*, Plenum Press, 1981.

[7] M. M. Darmon, P. R. Sadot, "A New Pseudo-Random Interleaving for Anti-jamming Applications", *MilCom '89*, Boston, Mass., November, 1989, pp. 1.2.1-1.2.5.

[8] E. Dunscombe, F. C. Piper, "Optimal Interleaving Scheme for Convolutional Coding", Electronics Letters, Vol. 25, pp. 1517-1518, October 1989.

[9] E. R. Berlekamp, P. Tong, "Interleavers for Digital Communications", US Patent Number 4,559,625, December 17, 1985.

[10] "Final Report on Variable-Depth Helical Interleaving", Prepared by Cyclotomics for Office of Naval Research, Contract Number N00014-84-C-0407, June 1987.

[11] "Coding for Frequency Hopped Spread Spectrum Satellite Communications", Prepared for Communications Canada under SSC Contract No. 36001-2-3572/01-SS, April 1993.

[12] D. T. Chi, "Helical Interleavers", *International Telemetering Conference*, Las Vegas, Nevada, October, 1990.

[13] S. Chia, "The Universal Mobile Telecommunication System", *IEEE Communications Magazine*, Vol. 30, no. 12, p. 59, December, 1992.

VITA

Surname: VOGEL Given Names: MARK

Place of Birth: Montreal, Quebec Date of Birth: April 25, 1967

Educational Institutions Attended:

University of Victoria	1985 to 1993

Degrees Awarded:

Bachelor of Engineering	Department of Electrical and Computer Engineering, University of Victoria	1990

Honours and Awards:

NSERC Post-Graduate Scholarship	1990 to 1992
President's Research Scholarship	1990 to 1992

Publications:

G. Li, T. A. Gulliver, I. F. Blake, Q. Wang, V. K. Bhargava and M. Vogel, *Coding for Frequency Hopped Spread Spectrum Satellite Communications*, Prepared for Communications Canada under SSC Contract No. 36001-2-3572/01-SS, April 1993.

Partial Copyright License

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis/Dissertation: **A Performance Analysis of Interleaving**

Author 

(Signature)

Mark Vogel

September 23, 1993

(Date)