A Theory of Interleavers

Kenneth Andrews*
andrews@ee.cornell.edu

Chris Heegard*
heegard@ee.cornell.edu

Dexter Kozen†
kozen@cs.cornell.edu

Abstract

An interleaver *is a hardware device commonly used in conjunction with error correcting codes to counteract the effect of burst errors. Interleavers are in widespread use and much is known about them from an engineering standpoint.*

In this paper we propose a mathematical model that provides a rigorous foundation for the theoretical study of interleavers. The model captures precisely such notions as block and convolutional interleavers, spread, periodicity, causality, latency, and memory usage.

Using this model, we derive several optimality results on the latency and memory usage of interleavers. We describe a family of block interleavers and show that they are optimal with respect to latency among all block interleavers with a given spread. We also give tight upper and lower bounds on the memory requirements of interleavers.

1 Introduction

Interleaving is a standard signal processing technique used in a variety of communications systems. An *interleaver* is a hardware device that takes symbols from an fixed alphabet as the input and produces the identical symbols at the output in a different temporal order. The classical use for interleaving is to disperse sequences of bits in a bitstream so as to minimize the effect of burst errors introduced in transmission.

Error correcting codes can correct errors successfully as long as there are not too many errors in a single codeword. However, errors sometimes tend to be *bursty* in the sense that there can be a local concentration of many errors, too many for typical error correction schemes to handle. This situation occurs for example in (i) burst error channels such as wireless communications channels, and (ii) *concatenated coding*, in which the first stage of decoding generates burst errors, such as in Viterbi decoders [4, 7]. Another very recent application of interleaving is in parallel concatenated turbo coding [1].

There are two classical kinds of interleavers, commonly referred to as *block* and *convolutional* [4, 7]. In a block interleaver, the input data is written along the rows of a memory configured as a matrix, and then read out along the columns. A variation of a block interleaver is a *pseudorandom block interleaver*, in which data is written in memory in sequential order and read in a pseudorandom order [6, 2]. In a *convolutional interleaver*, the data is multiplexed into and out of a fixed number of shift registers [5, 3].

In this paper, we propose a mathematical model that is motivated by these classical designs and the need to understand the foundations of interleaving in the realm of turbo coding. Our model captures the notions of block and convolutional interleavers as well as various notions such as spread, periodicity, causality, latency, delay, and memory usage.

Using this model, we study the latencies of a popular family of interleavers called *block interleavers.* We identify a certain class of block interleavers and show that they are optimal with respect to latency among all block interleavers of a given spread. More specifically, we show that for spread s,

(i) there do not exist block interleavers with period less than s^2;

(ii) there are exactly two block interleavers of period s^2, and both have latency $2s^2 - 2s$;

(iii) the interleaver $a \mapsto as + 1 \bmod s^2 + 1$ has period $s^2 + 1$ and latency $2s^2 - 3s + 1$;

(iv) this latency is optimal among block interleavers of spread s.

The interleavers described in (iv) were introduced in [2].

In addition to these results, we also derive exact upper and lower bounds on memory requirements of

*School of Electrical Engineering, Cornell University, Ithaca, NY 14853-7501, USA

†Department of Computer Science, Cornell University, Ithaca, NY 14853-7501, USA

interleavers. We characterize the number of memory cells necessary to implement a given interleaver and describe a hardware implementation that achieves this bound.

2 Mathematical Framework

In the information theory literature, an *interleaver* is usually understood to be a single input, single output finite-state device that takes sequences of symbols in a fixed alphabet and produces an output sequence over the same alphabet that is identical to the input sequence except for order. Every interleaver has a corresponding *de-interleaver* that acts on the output of the original interleaver and puts the symbols back into their original order (with a possible time delay).

For our study, the input/output alphabet is irrelevant; it is only the permutation on time instants that is important. In other words, if the input sequence is $\ldots, a_{-2}, a_{-1}, a_0, a_1, a_2, a_3, \ldots$ and the output sequence is $\ldots, b_{-2}, b_{-1}, b_0, b_1, b_2, b_3, \ldots$, then we focus on the permutation $\pi : \mathbb{Z} \to \mathbb{Z}$ such that $a_i = b_{\pi(i)}$. Thus, for the purposes of this paper, we define an *interleaver* to be a periodic permutation $\pi : \mathbb{Z} \to \mathbb{Z}$ of the integers. Here *permutation* means that the map π is one-to-one and onto, and *periodic* means that for some $p \geq 1$,

$$\pi(x+p) \quad = \quad \pi(x) + p$$

for all x. The number p is called a *period* of π. Equivalently, π is periodic with period p if it commutes with D^p under composition, where D is the successor function[1]

$$D \quad \stackrel{\text{def}}{=} \quad \lambda x.x + 1.$$

Periodicity models the fact that interleavers are realized as finite-state devices.

The only interleavers of period 1 are the powers of D. These are called *delay interleavers*. The *fundamental period* of an interleaver is the gcd of all its periods.

The family of all interleavers forms a group under composition. The lcm of the periods of two interleavers is a period of their composition.

2.1 Shifting

A *shift* of an interleaver π is an interleaver of the form

$$D^k \circ \pi \circ D^m \quad = \quad \lambda x.\pi(x+m) + k$$

for some k, m. We say that interleavers π and ρ are *shift equivalent*, or that ρ is a *shift* of π, if there exist integers k, m such that

$$\rho \quad = \quad D^k \circ \pi \circ D^m.$$

This relation identifies two interleavers if one can be obtained from the other by shifting the input or the output or both.

We say that interleavers π and ρ are *strongly shift equivalent*, or that ρ is a *strong shift* of π, if there exists an integer n such that

$$\rho \quad = \quad D^{-n} \circ \pi \circ D^n;$$

This relation identifies two interleavers if one can be obtained from the other by shifting the input and the output the same distance. The strong shift equivalence class of π contains exactly p elements, where p is the fundamental period of π.

2.2 Causality

An interleaver is *causal* if for all x, $\pi(x) \geq x$. This property models the fact that in an actual implementation, a symbol cannot come out before it goes in. An interleaver is *minimal causal* if it is causal and if $\pi(x) = x$ for some x. Every interleaver has a unique (up to strong shift equivalence) minimal causal shift.

In the literature on interleavers, it is common to restrict attention to causal interleavers, because noncausal ones are not realizable. However, for theoretical purposes, the property of causality can sometimes be a red herring. The key properties of interleavers are shift-invariant, and it often simplifies the mathematics considerably to ignore causality. We will see several examples of this below.

2.3 Delay and Latency

For an interleaver π, define

$$\begin{aligned} \ell_\pi^+ &\stackrel{\text{def}}{=} \max_x \pi(x) - x, \\ \ell_\pi^- &\stackrel{\text{def}}{=} \min_x \pi(x) - x \quad = \quad -\ell_{\pi^{-1}}^+. \end{aligned}$$

The quantity ℓ_π^+ is called the *delay* of π. This is the maximum time delay between the arrival of an input symbol and the time it should be produced as an output. An interleaver is causal iff $\ell_\pi^- \geq 0$ and is minimal causal iff $\ell_\pi^- = 0$.

A related concept is *latency*, which we define to be the shift-invariant quantity

$$\ell_\pi \quad \stackrel{\text{def}}{=} \quad \ell_\pi^+ - \ell_\pi^-.$$

[1] Here we are using standard λ-notation for functions: $\lambda x.M(x)$ is the function that on input a produces $M(a)$.

In other words, denoting the action of π on $\mathbb{Z}$ by arrows, it is the sum of the lengths of the longest leftward and longest rightward arrow.[2] An interleaver is causal iff $\ell_\pi^+ \geq \ell_\pi$ and is minimal causal iff $\ell_\pi^+ = \ell_\pi$. Thus the minimal causal shift of π has the minimum delay among all causal shifts of π.

The de-interleaver corresponding to π is just the inverse π^{-1}. It follows immediately from the definitions that π and π^{-1} have the same latency.

In a real implementation, we would use a causal shift of π for interleaving and a causal shift of π^{-1} for de-interleaving. Define the *minimum total delay* of an interleaver π to be the minimum sum of delays for interleaving and de-interleaving using causal shifts of π and π^{-1}. Since the delays of both these causal shifts are at least the latency of π, one might expect the minimum total delay to be at least twice the latency. Somewhat surprisingly, it turns out that it is exactly the latency.

Theorem 1 *The minimum total delay of an interleaver is equal to its latency. It is achieved by composing minimal causal shifts of the interleaver and its inverse.*

Proof. Let $D^m \circ \pi$ be a causal shift of π; then $m \geq -\ell_\pi^-$. Let $D^k \circ \pi^{-1}$ be a causal shift of π^{-1}; then $k \geq -\ell_{\pi^{-1}}^-$. Before we can compose these two interleavers, we must take a strong shift of one of them so that the outputs of the first interleaver line up with the inputs of the second. Strongly shifting $D^k \circ \pi^{-1}$ by m, we obtain

$$D^m \circ D^k \circ \pi^{-1} \circ D^{-m}.$$

Composing this with $D^m \circ \pi$ gives

$$D^m \circ D^k \circ \pi^{-1} \circ D^{-m} \circ D^m \circ \pi \quad = \quad D^{m+k},$$

which is just the identity shifted by $m + k$; this is the total delay. This number is minimized by taking $m = -\ell_\pi^-$ and $k = -\ell_{\pi^{-1}}^- = \ell_\pi^+$, in which case $m + k = \ell_\pi$, the latency of π. □

Example 2 Consider the minimal causal interleaver

$$\pi(k) \quad = \quad \begin{cases} k+2 & \text{if } k = 0 \bmod 3, \\ k & \text{if } k = 1 \bmod 3, \\ k+1 & \text{if } k = 2 \bmod 3 \end{cases} \tag{1}$$

with latency 2. Its inverse is

$$\pi^{-1}(k) \quad = \quad \begin{cases} k-1 & \text{if } k = 0 \bmod 3, \\ k & \text{if } k = 1 \bmod 3, \\ k-2 & \text{if } k = 2 \bmod 3. \end{cases}$$

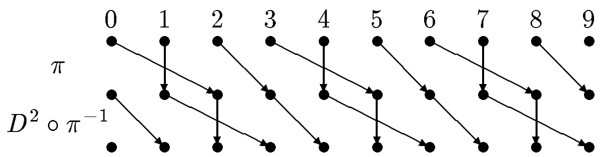
[2]In case there are no leftward arrows, read "negative of the length of the shortest rightward arrow" for "length of the longest leftward arrow."

Shifting the output by 2 gives the minimal causal interleaver

$$D^2 \circ \pi^{-1}(k) \quad = \quad \begin{cases} k+1 & \text{if } k = 0 \bmod 3, \\ k+2 & \text{if } k = 1 \bmod 3, \\ k & \text{if } k = 2 \bmod 3, \end{cases} \tag{2}$$

also with latency 2. Composing (1) and (2) gives

$$D^2 \circ \pi^{-1} \circ \pi \quad = \quad D^2.$$



□

2.4 Spread

We say that an interleaver π has *spread* s, t if $|\pi(x) - \pi(y)| \geq t$ whenever $|x - y| < s$. Intuitively, π has spread s, t if any two input symbols in an interval of length s are separated by a distance of at least t at output. An interleaver has spread s, t iff its inverse has spread t, s.

In this paper we focus on the case $s = t$. We abbreviate spread s, s by spread s. The property of having spread s is invariant under shift and inverse.

It is desirable to have large spread, since this is how we counteract burst errors. In practice, a typical spread might be 18. However, larger spreads entail longer latencies. We analyze this tradeoff in Section 3.

2.5 Memory

Intuitively, the memory required by a causal interleaver π is the maximum number of input symbols that must be remembered from some time $i - 1$ to time i. Formally, for causal interleavers, we can define this to be

$$\max_i |\{x \mid x < i \text{ and } \pi(x) \geq i\}|.$$

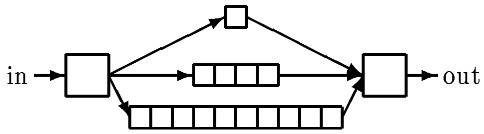
In Section 4 we give a shift-independent definition and show that for block interleavers, this quantity is exactly $-\ell_{\pi'}^-$, where π' is a permutation interleaver shift-equivalent to π. We also give tight upper and lower bounds on memory usage for any causal interleaver.

2.6 Examples of Interleavers

2.6.1 Multiplexed Interleavers

A *multiplexed shift register interleaver* [5, 3, 4] is described in the literature as a hardware device that

1. de-multiplexes the input sequence into p subsequences,
2. introduces a uniform delay on each subsequence,
3. multiplexes the p results.



These devices can be implemented efficiently in hardware using a set of p shift registers, where the lengths of the registers determine the delay.

In terms of our model, a multiplexed interleaver with period p is one that is shift equivalent to an interleaver with $p \mid \pi(x) - x$ for all x. Such an interleaver π is uniquely determined (up to shift equivalence) by the integers $k_i = (\pi(x) - x)/p$, $0 \le i \le p-1$.

2.6.2 Block Interleavers

Another wide class of interleavers in common use are the *block interleavers.* An interleaver π is a *block interleaver of blocksize* p if p is a period of π and there is an interval of length p whose image under π is also an interval of length p. The blocksizes of a block interleaver π are exactly the periods of π.

The property of being a block interleaver of blocksize p is invariant under shift and inverse.

Any block interleaver is shift equivalent to a *permutation interleaver.* A permutation interleaver is given by a permutation on $\{0, 1, \ldots, p-1\}$ repeated with period p. For example, the period 3 interleaver

$$k \quad \mapsto \quad \begin{cases} k+1 & \text{if } k = 0 \bmod 3, \\ k+2 & \text{if } k = 1 \bmod 3, \\ k+3 & \text{if } k = 2 \bmod 3 \end{cases}$$

is $D^3 \circ \rho \circ D^{-1}$, where ρ is the permutation interleaver described by the permutation $(0)(1\,2)$ in cycle notation.

Not all interleavers are block interleavers. For example, the interleaver

$$k \quad \mapsto \quad \begin{cases} k+1 & \text{if } k = 0 \bmod 3, \\ k+5 & \text{if } k = 1 \bmod 3, \\ k+9 & \text{if } k = 2 \bmod 3 \end{cases}$$

is not. However, if p is a period, then for any $x, y \in \{0, 1, \ldots, p-1\}$, $x \neq y$, the residues of $\pi(x)$ and $\pi(y)$ modulo p must be distinct. Thus an arbitrary interleaver is completely determined (up to shift equivalence) by a permutation on $\{0, 1, \ldots, p-1\}$ along with a delay of a multiple of p on each element. In other words, every interleaver is shift equivalent to a composition of a permutation interleaver and a multiplexed interleaver.

An interleaver that is not a block interleaver is called a *convolutional interleaver.*

3 Block Interleavers of Minimal Latency

The following are our main results on latency.

Theorem 3 *For block interleavers of spread s,*

(i) there are none with period $s^2 - 1$ or less;

(ii) up to shift equivalence, there are exactly two with period s^2, and both have latency $2s^2 - 2s$;

(iii) the permutation interleaver $a \mapsto as+1 \bmod s^2+1$ has period s^2+1 and latency $2s^2 - 3s + 1$;

(iv) this latency is optimal among block interleavers of spread s.

We remark that the interleaver $a \mapsto as \bmod s^2 + 1$ has nonoptimal latency $2s^2 - 2s$, the same as the interleavers of (ii).

Proof. (i) Let π be an arbitrary block interleaver of spread s. By shift-invariance, we can assume that π is a permutation interleaver. Let $I = \pi(I) = [0, p-1]$, where p is a period of π.

Certainly $p \ge s$, since otherwise we would have

$$\begin{aligned} |\pi(a+p) - \pi(a)| &= \pi(a) + p - \pi(a) = p < s, \\ |(a+p) - a| &= p < s, \end{aligned}$$

contradicting spread s. Thus $|I| \ge s$.

Define an *s-interval* to be a subinterval of I of length s. Define an *s-anti-interval* to be a subset of I of size s such that no two elements are of distance less than s from each other. A necessary condition for π to have spread s is that it send s-intervals to s-anti-intervals. This condition is not sufficient, for we must also worry about what happens at the boundary of adjacent blocks.

Since $p \ge s$, $s-1 \in I$ and every element of I is contained in at least one s-interval. Let J be an s-interval containing $\pi^{-1}(s-1)$. Then $\pi(J) \subseteq I$ is an s-anti-interval. Let $\pi(J) = \{a_0, a_1, \ldots, a_{s-1}\}$ with

$$a_0 < a_1 < \cdots < a_{s-1}.$$

Note that $s-1 = a_0$, since no s-anti-interval containing $s-1$ can contain any smaller element.

Since $\pi(J)$ is an s-anti-interval, we must have

$$a_{i+1} - a_i \quad \geq \quad s, \quad 0 \leq i \leq s-2.$$

Combining these in a telescoping sum, we have

$$\begin{aligned} a_{s-1} &= a_0 + \sum_{i=0}^{s-2} a_{i+1} - a_i \\ &\geq s - 1 + \sum_{i=0}^{s-2} s \\ &= s - 1 + (s-1)s \\ &= s^2 - 1. \end{aligned}$$

Thus $s^2 - 1 \in I$, so $|I| \geq s^2$.

(ii) We show there are exactly two permutation interleavers of spread s and period s^2, namely

$$is + j \quad \mapsto \quad (s-1-j)s + i, \quad 0 \leq i, j \leq s-1 \quad (3)$$

and its inverse

$$is + j \quad \mapsto \quad js + (s-1-i), \quad 0 \leq i, j \leq s-1. \quad (4)$$

The permutations (3) and (4) can be described intuitively as follows. Let I be the interval $[0, s^2-1]$. Arrange the elements of I in an $s \times s$ matrix R in row major order; thus $R_{ij} = is + j$, $0 \leq i, j \leq s-1$. The permutations (3) and (4) are obtained by rotating R a quarter turn counterclockwise and clockwise, respectively.

For $0 \leq i, j \leq s-1$, let

$$\begin{aligned} I_i &= \{is, is+1, is+2, \ldots, is+s-1\}, \\ J_j &= \{j, s+j, 2s+j, \ldots, (s-1)s+j\}. \end{aligned}$$

The sets I_i and J_j are the elements appearing in the i-th row and j-th column of R, respectively. Note that the sets I_i are s-intervals (among others), and the sets J_j are s-anti-intervals (among others). As observed above in (i), a necessary condition for a permutation $\pi : I \to I$ to have spread s is that it send s-intervals to s-anti-intervals.

Now let π be an arbitrary permutation of I of spread s. We will show that π must be either (3) or (4).

By inspecting the matrix R, it can be observed that each of $s-1$ and s^2-s is contained in exactly one s-anti-interval, namely J_{s-1} and J_0, respectively. Since every distinct s-interval must go to a distinct s-anti-interval under π, there can be at most one s-interval containing $\pi^{-1}(s-1)$ and at most one s-interval containing $\pi^{-1}(s^2-s)$. But there are only two elements of I contained in exactly one s-interval, namely $0 \in I_0$ and $s^2-1 \in I_{s-1}$. Thus either

A. $\pi(0) = s^2 - s$ and $\pi(s^2-1) = s-1$, in which case $\pi(I_0) = J_0$ and $\pi(I_{s-1}) = J_{s-1}$; or

B. $\pi(0) = s-1$ and $\pi(s^2-1) = s^2-s$, in which case $\pi(I_0) = J_{s-1}$ and $\pi(I_{s-1}) = J_0$.

These two cases will give rise to (3) and (4), respectively.

In case A, we claim more generally that

$$\pi(I_i) \quad = \quad J_i, \quad 0 \leq i \leq s-1. \quad (5)$$

We have just shown this for $i = 0$ and $i = s-1$. Proceeding by induction, suppose we have shown it for $0 \leq i \leq k-1$, $k < s-1$. Let $c = \pi^{-1}(s^2 - s + k)$. There is exactly one s-anti-interval containing $s^2 - s + k$ disjoint from $J_0, \ldots, J_{k-1}$, namely J_k. Since every distinct s-interval containing c disjoint from $I_0, \ldots, I_{k-1}$ must go to a distinct s-anti-interval disjoint from $J_0, \ldots, J_{k-1}$ under π, there can be at most one s-interval containing c disjoint from $I_0, \ldots, I_{k-1}$. There are only two elements of I for which this is true, namely ks and s^2-1, and we have already argued that $\pi(s^2-1) = s-1 \notin J_k$, therefore $c = ks$ and $\pi(I_k) = J_k$.

In case B, a symmetric argument shows that

$$\pi(I_i) \quad = \quad J_{s-1-i}, \quad 0 \leq i \leq s-1. \quad (6)$$

Now π^{-1} is also a permutation interleaver of spread s and period s^2, thus satisfies either (5) or (6). But if π satisfies (5), then π^{-1} cannot satisfy (5), because $s^2 - s \in I_{s-1}$ but $\pi^{-1}(s^2-s) = 0 \notin J_{s-1}$. Thus π^{-1} must satisfy (6); that is,

$$\pi^{-1}(I_i) \quad = \quad J_{s-1-i}, \quad 0 \leq i \leq s-1.$$

Inverting, we have

$$\pi(J_j) \quad = \quad I_{s-1-j}, \quad 0 \leq j \leq s-1. \quad (7)$$

Using the fact that row I_i and column J_j intersect in the single element $is+j$, (3) follows immediately from (5) and (7).

By a symmetric argument, π and π^{-1} cannot simultaneously satisfy (6), thus if π satisfies (6) then π^{-1} must satisfy (5); in other words,

$$\pi(J_j) \quad = \quad I_j, \quad 0 \leq j \leq s-1.$$

This and (6) imply (4).

To verify that the interleavers (3) and (4) indeed have spread s, we need only observe the form of the image of any s-interval contained in I, as well as any interval of length s straddling a boundary between two blocks. In all cases the rotation of R (or two adjacent copies of R) a quarter turn in either direction takes such a set to an s-anti-interval.

(iii) Consider the permutation interleaver

$$a \quad \mapsto \quad as + 1 \bmod s^2 + 1, \quad 0 \leq a \leq s^2.$$

In particular,

$$\begin{aligned} s - 1 \quad &\mapsto \quad s^2 - s + 1, \\ s^2 - s + 1 \quad &\mapsto \quad 2. \end{aligned}$$

By inspection, it can be ascertained that these are the longest leftward and rightward arrows, respectively, giving a latency of $2s^2 - 3s + 1$. As we show in (iv), this is the optimal latency for block interleavers with spread s.

(iv) Consider an arbitrary permutation interleaver π of spread s and period p:

$$\pi : [0, p - 1] \quad \to \quad [0, p - 1].$$

We will show that π has latency at least $2s^2 - 3s + 1$. By (ii) we can assume without loss of generality that $p \geq s^2$. As in (ii), subdivide the interval $[0, p - 1]$ into disjoint contiguous s-intervals $I_0, I_1, \ldots, I_{s-1}, \ldots$ where

$$I_k \quad \stackrel{\text{def}}{=} \quad [ks, (k + 1)s - 1]$$

(the last subinterval will be shorter if p is not a multiple of s). Since $\pi(I_0)$ is an s-anti-interval, all $\pi(i)$ for $i \in I_0$ must occupy different I_k. Thus at least one such $\pi(i)$ lies in I_k for some $k \geq s - 1$; in other words,

$$\max\{\lfloor \pi(i)/s \rfloor \mid i \in I_0\} \quad \geq \quad s - 1.$$

Thus

$$\begin{aligned} \max\{\pi(i) - i \mid i \in I_0\} \quad &\geq \quad \min I_{s-1} - \max I_0 \\ &= \quad (s - 1)s - (s - 1) \\ &= \quad s^2 - 2s + 1. \end{aligned} \tag{8}$$

The same argument holds for π^{-1}, since the periods, latencies and spreads of π and π^{-1} are the same. Thus

$$\max\{\pi^{-1}(i) - i \mid i \in I_0\} \quad \geq \quad s^2 - 2s + 1. \tag{9}$$

Let $I_0 = \{a_0, a_1, \ldots, a_{s-1}\}$, where the elements are numbered in order of their images under π:

$$\pi(a_0) \quad < \quad \pi(a_1) \quad < \quad \cdots \quad < \quad \pi(a_{s-1}).$$

If $\pi(a_0) \notin I_0$, we are done: in this case,

$$\begin{aligned} \max\{\pi(i) - i \mid i \in I_0\} \quad &\geq \quad \min I_s - \max I_0 \\ &= \quad s^2 - (s - 1) \\ &= \quad s^2 - s + 1, \end{aligned} \tag{10}$$

and the latency is at least the sum of (9) and (10), which is $2s^2 - 3s + 2$. Similarly, number the elements of $I_{-1} = [-s, -1]$ as $a_{-1}, a_{-2}, \ldots, a_{-s}$, where

$$\pi(a_{-s}) \quad < \quad \cdots \quad < \quad \pi(a_{-2}) \quad < \quad \pi(a_{-1}).$$

As above, if $\pi(a_{-1}) \notin I_{-1}$, we are done.

Assume therefore that $\pi(a_0) \in I_0$ and $\pi(a_{-1}) \in I_{-1}$. Either

$$a_0 - a_{-1} \quad \geq \quad s \quad \text{or} \tag{11}$$

$$\pi(a_0) - \pi(a_{-1}) \quad \geq \quad s, \tag{12}$$

since π has spread s. By the symmetry between π and π^{-1}, we can assume (12) without loss of generality; if (12) is false but (11) is true, interchange π and π^{-1} throughout.

Since $\pi(I_0)$ is an s-anti-interval, we must have

$$\pi(a_{i+1}) - \pi(a_i) \quad \geq \quad s, \quad 0 \leq i \leq s - 2.$$

Combining these in a telescoping sum, we have

$$\begin{aligned} \pi(a_{s-1}) \quad &= \quad \pi(a_0) + \sum_{i=0}^{s-2} \pi(a_{i+1}) - \pi(a_i) \\ &\geq \quad \pi(a_0) + (s - 1)s, \end{aligned} \tag{13}$$

and symmetrically,

$$\pi(a_{-1}) \quad \geq \quad \pi(a_{-s}) + (s - 1)s. \tag{14}$$

By (13), we have

$$\begin{aligned} \pi(a_{s-1}) - a_{s-1} \quad &\geq \quad \pi(a_{s-1}) - (s - 1) \\ &\geq \quad \pi(a_0) + (s - 1)s - (s - 1) \\ &= \quad \pi(a_0) + s^2 - 2s + 1. \end{aligned} \tag{15}$$

Symmetrically, using (14), we also have

$$\begin{aligned} a_{-s} - \pi(a_{-s}) \quad &\geq \quad -s - \pi(a_{-s}) \\ &\geq \quad -s - \pi(a_{-1}) + (s - 1)s \\ &= \quad -\pi(a_{-1}) + s^2 - 2s. \end{aligned} \tag{16}$$

The latency of π is at least the sum of (15) and (16):

$$\begin{aligned} &(\pi(a_{s-1}) - a_{s-1}) + (a_{-s} - \pi(a_{-s})) \\ &\quad \geq \quad (\pi(a_0) + s^2 - 2s + 1) + (-\pi(a_{-1}) + s^2 - 2s) \\ &\quad = \quad \pi(a_0) - \pi(a_{-1}) + 2s^2 - 4s + 1 \\ &\quad \geq \quad s + 2s^2 - 4s + 1 \qquad \text{by (12)} \\ &\quad = \quad 2s^2 - 3s + 1. \end{aligned}$$

□

4 Memory-Optimal Interleavers

We have defined the memory required by a causal interleaver π to be the maximum number of input symbols that must be remembered from some time $i-1$ to time i; in other words, the maximum over all integers i of the quantity

$$|\{x \mid x < i \text{ and } \pi(x) \geq i\}|. \tag{17}$$

For example, for the multiplexed shift register interleavers described in Section 2.6.1, the memory required is no more than the sum of the lengths of the shift registers.

We argue below that the quantity (17) is independent of i, and that it is both a lower and an upper bound on the memory needed to implement π. For the latter, we describe a hardware implementation that uses exactly this many memory cells.

We are also interested in minimizing the memory usage over all causal shifts π. We show that memory usage is minimized by the unique minimal causal shift of π. Thus the minimal causal shift of a given interleaver optimizes both memory and total delay among all causal shifts of π.

As above, we will find it mathematically convenient to ignore causality. For any interleaver π, causal or not, and $x, y \in \mathbb{Z}$, define

$$\begin{aligned} e_\pi(x,y) &\stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } y = \pi(x) \\ 0 & \text{otherwise,} \end{cases} \\ m_\pi(i,j) &\stackrel{\text{def}}{=} \sum_{\substack{x<i \\ y \geq j}} e_\pi(x,y) - \sum_{\substack{x \geq i \\ y<j}} e_\pi(x,y). \end{aligned}$$

Intuitively, if we draw arrows from x to $\pi(x)$, then $m_\pi(i,j)$ is the number of arrows going from $i-1$ or before to j or after less the number of arrows going from i or after to $j-1$ or before. Because of periodicity, there are only finitely many such arrows, so $m_\pi(i,j)$ exists. Note that $m_\pi(i,i)$ is the number of arrows that cross over a vertical line between $i-1$ and i from left to right less the number that cross from right to left. If π is causal, there are no arrows that cross from right to left; as already observed, in this case $m_\pi(i,i)$ is a lower bound on the number of memory cells required to implement π.

The following are some basic properties that follow from these definitions.

Lemma 4 *For any interleaver π and integers x, y, i, j, k, m,*

(i) $e_{\pi^{-1}}(x,y) = e_\pi(y,x)$;

(ii) $e_{D^k \circ \pi \circ D^m}(x,y) = e_\pi(x+m, y-k)$;

(iii) $m_{\pi^{-1}}(i,j) = -m_\pi(j,i)$;

(iv) $m_\pi(i+m, j+k) = m_\pi(i,j) + m - k$;

(v) $m_{D^k \circ \pi \circ D^m}(i,j) = m_\pi(i,j) + m + k$.

Proof. All these properties follow in a straightforward way from the definitions. We prove (iv) explicitly. First,

$$\begin{aligned} m_\pi(i,j+1) &= \sum_{\substack{x<i \\ y \geq j+1}} e_\pi(x,y) - \sum_{\substack{x \geq i \\ y<j+1}} e_\pi(x,y) \\ &= \sum_{\substack{x<i \\ y \geq j}} e_\pi(x,y) - \sum_{\substack{x<i \\ y=j}} e_\pi(x,y) \\ &\quad -\Big(\sum_{\substack{x \geq i \\ y<j}} e_\pi(x,y) + \sum_{\substack{x \geq i \\ y=j}} e_\pi(x,y)\Big) \\ &= \sum_{\substack{x<i \\ y \geq j}} e_\pi(x,y) - \sum_{\substack{x \geq i \\ y<j}} e_\pi(x,y) \\ &\quad - \sum_{x<i} e_\pi(x,j) - \sum_{x \geq i} e_\pi(x,j) \\ &= m_\pi(i,j) - \sum_x e_\pi(x,j) \\ &= m_\pi(i,j) - 1. \end{aligned}$$

By iterating,

$$m_\pi(i,j+k) = m_\pi(i,j) - k.$$

Using this fact and (iii),

$$\begin{aligned} m_\pi(i+m,j+k) &= m_\pi(i+m,j) - k \\ &= -m_{\pi^{-1}}(j,i+m) - k \\ &= -m_{\pi^{-1}}(j,i) + m - k \\ &= m_\pi(i,j) + m - k. \end{aligned}$$

□

It follows immediately from Lemma 4(iv) that for all $i, j \in \mathbb{Z}$, $m_\pi(i,i) = m_\pi(j,j)$, thus this quantity depends only on π. We denote this number by m_π.

Define a *cycle* of an interleaver π to be a minimal nonempty set of elements of $\mathbb{Z}$ closed under π and π^{-1}. A cycle is *nontrivial* if it contains more than one element. If π is causal, then every nontrivial cycle is infinite. If π is minimal causal, then it has trivial cycles as well.

Example 5 The minimal causal interleaver

$$k \quad \mapsto \quad \begin{cases} k+4 & \text{if } k = 0 \bmod 3, \\ k+2 & \text{if } k = 1 \bmod 3, \\ k & \text{if } k = 2 \bmod 3 \end{cases}$$

has two nontrivial cycles

$$\{\ldots,-6,-2,0,4,6,10,12,16,18,22,24,\ldots\} \quad (18)$$
$$\{\ldots,-5,-3,1,3,7,9,13,15,19,21,25,\ldots\} \quad (19)$$

as well as infinitely many trivial cycles

$$\ldots,\{-7\},\{-4\},\{-1\},\{2\},\{5\},\{8\},\{11\},\{14\},\ldots$$

□

It is apparent that the number of nontrivial cycles of a causal interleaver π is just m_π, since for any i, there is exactly one $x \in \mathbb{Z}$ in each nontrivial cycle with $x < i$ and $\pi(x) \geq i$.

Theorem 6 *(i) If π is minimal causal, then m_π is minimum among all causal shifts of π. Thus the minimal causal shift minimizes both the total delay and memory among causal shifts of a given interleaver.*

(ii) For a minimal causal block interleaver π with block $I = [0, p-1]$ and $\pi(I) = [m, m+p-1]$, $m_\pi = m$.

(iii) For a causal interleaver π, m_π memory cells are necessary and sufficient for implementing π.

Proof. (i) Let π be minimal causal. It follows from Lemma 4(v) that $m_{D^k \circ \pi} = m_\pi + k$, thus m_π is minimum among causal shifts of π.

(ii) Let us call an interleaver ρ *zero-memory* if $m_\rho = 0$. By Lemma 4(v), every interleaver π has a unique (up to strong shift equivalence) zero-memory shift

$$\pi' \quad = \quad D^{-m_\pi} \circ \pi.$$

Then

$$\ell^-_{\pi'} \quad = \quad \ell^-_\pi - m_\pi.$$

If π is minimal causal, then $\ell^-_\pi = 0$, therefore $\ell^-_{\pi'} = -m_\pi$. The result for block interleavers follows from this and the additional observation that permutation interleavers are zero-memory: at any block boundary, there are no arrows crossing in either direction.

(iii) Let π be causal. We have already argued that m_π is a lower bound on memory usage. We now describe a simple hardware implementation of π using synchronous clocked logic that achieves this bound. Our device uses m_π shift registers of length 1, one for each nontrivial cycle of π. The idea is that at any time i, the cell associated with a given cycle c holds the symbol that was input at time $x < i$ and will be output at time $\pi(x) \geq i$, where x lies on c. Since $\pi(x)$ lies on c as well, the symbol vacates the cell at the same time the next symbol comes in. Trivial cycles require no memory; the symbols are passed directly from input to output with no time delay.

For example, consider the interleaver of Example 5. Let c and d be the shift registers associated with the nontrivial cycles (18) and (19), respectively. The following is a protocol that tells at each time instant which register the input should be written to and the output should be read from.

Time	*Write to/Read from*
⋮	⋮
−3	d
−2	c
−1	direct
0	c
1	d
2	direct
3	d
4	c
5	direct
6	c
7	d
8	direct
9	d
10	c
⋮	⋮

Note that this protocol is periodic with period 6.

In general, the protocol will be periodic with period some multiple of p, say np, which could be exponential in the number of cycles. A basic hardware implementation simply requires a cyclic counter of period np with combinatorial logic to convert the counter value to a memory address as described by the table above. □

As noted, the period np can be exponential in the number of cycles, which could make this implementation impractical. Moreover, the counter must maintain its own state, which requires additional internal memory of size $\log(np)$. We present two alternative implementations that address these issues. The first stores symbols as just described, but uses an array of smaller cyclic automata, each no larger than the number of cycles, to generate the memory addresses. This implementation uses the traditional memory architecture, in which each symbol is stored at a fixed addressable location; this makes it necessary to maintain extra state information beyond the minimum required to identify the current interleaver phase $\{0, 1, \ldots, p-1\}$. Our second implementation uses a shift register architecture and only a cyclic counter of length p.

Method 1 For each $x \in \mathbb{Z}$, let c_x denote the cycle of π occupied by x. For any residue $i \bmod p$, consider the sequence of cycles

$$\ldots, c_{i-2p}, c_{i-p}, c_i, c_{i+p}, c_{i+2p}, c_{i+3p}, \ldots \tag{20}$$

associated with successive integers congruent to $i \bmod p$. If one of these cycles is trivial, then by periodicity, they all are. In this case we just mark i as representing only trivial cycles. The protocol will maintain a counter modulo p, and at time instants congruent to $i \bmod p$ will pass the input symbol directly to the output.

Otherwise, all cycles in the sequence (20) are nontrivial. We claim that the sequence is periodic with period at most the number of cycles. To see this, suppose $q > 0$ is the minimum number such that for some j congruent to $i \bmod p$, both j and $j + qp$ lie on the same cycle. Then q is at most the number of cycles, and $j + qp = \pi^m(j)$ for some m, since that is what it means for j and $j + qp$ to lie on the same cycle. Then by periodicity,

$$\pi^m(j+p) \quad = \quad \pi^m(j) + p \quad = \quad j + qp + p,$$

so $j+p$ and $j+p+qp$ lie on a common cycle as well. It follows that the sequence (20) is periodic with period q. In this case we associate with the residue i a cyclic finite-state automaton that tells at each successive integer congruent to $i \bmod p$ the next cycle in the sequence (20). In fact, the same automaton can be used for all residues of integers on a common cycle; in other words, the cyclic automata associated with $\pi^k(x) \bmod p$ and $x \bmod p$ are the same, albeit perhaps out of phase. Every nontrivial cycle is associated with exactly one state of exactly one such automaton, so that the sum of the sizes of the cyclic automata is just m_π, the number of nontrivial cycles.

In the interleaver of Example 5, $p = 3$. The residue $2 \bmod 3$ is marked as direct: the input is passed directly to the output at any time instant congruent to $2 \bmod 3$. The residues $0 \bmod 3$ and $1 \bmod 3$ are both associated with the cyclic automaton $c \leftrightarrow d$, except out of phase, indicating that at times $\ldots, 0, 3, 6, 9, \ldots$ the appropriate cycles are $\ldots, c, d, c, d, \ldots$ respectively, and at times $\ldots, 1, 4, 7, 10, \ldots$ the appropriate cycles are $\ldots, d, c, d, c, \ldots$ respectively.

Method 2 Consider a single shift register of maximum length m_π implementing a queue in which insertions are allowed at any position. An insertion at position k causes the values currently occupying positions $k-1, k-2, \ldots, 2, 1, 0$ to be shifted down, producing the symbol currently at position 0 as output, while the remainder of the register remains fixed. If $k = 0$, then the new symbol is sent directly to the output.

At each time i, a position k_i is selected and the new symbol inserted at that position in the queue. Intuitively, k_i is chosen so that the new symbol will be inserted behind all stored symbols that must be output before time $\pi(i)$. This method essentially implements a priority queue in which the priorities are the output times. However, we need not store the actual priorities; since the pattern of insertions is cyclic with period p, we can just calculate once and for all the insertion position for each residue modulo p. Specifically, this is

$$\begin{aligned} k_i &= \sum_{\substack{x < i \\ i \le y < \pi(i)}} e_\pi(x, y) \\ &= m_\pi - \sum_{\substack{x < i \\ y \ge \pi(i)}} e_\pi(x, y). \end{aligned} \tag{21}$$

It can be seen from (21) that the length of the shift register never exceeds m_π.

Continuing Example 5, we find that the appropriate insertion points are $k_0 = 2$, $k_1 = 1$, and $k_2 = 0$.

A dual implementation would insert symbols at one end of the queue and extract them from the appropriate position in the middle. This is essentially the above implementation for the interleaver

$$\lambda x. - \pi^{-1}(-x)$$

executed backwards.

5 Future Research

A number of interesting questions present themselves, perhaps the most interesting of which is to try to apply these techniques to ascertain optimal latencies for convolutional (nonblock) interleavers.

Acknowledgments

The authors would like to thank Subramanya P. N. Rao, Moss Sweedler, and Stephen Wicker for many helpful discussions on interleaving. The support of the National Science Foundation under grants NCR-9520981 and CCR-9317320 is gratefully acknowledged.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo codes. *ICC*, pages 1064–1070, 1993.

[2] S. Dolinar and D. Divsalar. Weight distributions for turbo codes using random and nonrandom permutations. Technical Report TDA Progress Report 42-121, JPL, August 1995.

[3] J. G. David Forney. Burst-correcting codes for the classic bursty channel. *IEEE Transactions on Communications*, COM-19(10):772–781, October 1971.

[4] J. George C. Clark and J. B. Cain. *Error-Correction Coding for Digital Communications.* Plenum Press, 1981.

[5] J. L. Ramsey. Realization of optimum interleavers. *IEEE Transactions on Information Theory*, IT-16(3):338–345, May 1970.

[6] I. Richer. A simple interleaver for use with Viterbi decoding. *IEEE Transactions on Communications*, COM-26(3):406–408, March 1978.

[7] S. B. Wicker. *Error Control Systems for Digital Communications and Storage.* Prentice Hall, Englewood Cliffs, NJ, 1995.