Abstract—This work considers the design and performance of a stream-oriented approach to turbo codes which avoids the need for data framing. The stream paradigm applies to both serial and parallel turbo codes using continuous, free-running constituent encoders along with continuous, periodic interleavers. A stream-oriented turbo code based on parallel concatenated convolutional codes (PCCC) is considered and interleaver design criteria are developed for both block and nonblock periodic interleavers. Specifically, several nonblock interleavers, including convolutional interleavers, are considered. Interleaver design rules are verified using simulations where it is shown that nonblock interleavers with small-to-moderate delay and small synchronization ambiguity can outperform block interleavers of comparable delay. For large-delay designs, nonblock interleavers are found which perform within 0.8 dB of the capacity limit with a synchronization ambiguity of N = 11.

Index Terms—Convolutional interleaver, parallel concatenated convolutional codes (PCCC), periodic interleaving, synchronization, turbo codes.

I. INTRODUCTION

TURBO coding is a channel coding scheme based on the concatenation of convolutional codes along with interleaving and a suboptimum, iterative decoding rule. Whether implemented as Parallel Concatenated Convolutional Codes (PCCC) [1] or Serial Concatenated Convolutional Codes (SCCC) [2], turbo codes are one of the most powerful coding techniques yet discovered, offering performance approaching the capacity limit with modest decoding complexity.

Traditionally, turbo codes have been viewed as block codes, with research focusing on block interleaver design and block decoding architectures. However, in applications where the message is long, such as circuit-switched communications, stream-oriented coding schemes are advantageous since data-framing, which is necessary for block and deinterleaver synchronization, is avoided. Stream-oriented turbo codes, like the industry standard convolutional codes, can also be used in communications links where framing is present provided the delays can be tolerated. In this work, a stream paradigm for turbo codes is explored, termed stream-oriented turbo codes [3], [4]. The stream approach generalizes to both PCCC and SCCC, using “free-running” constituent encoders, continuous or synchronous periodic interleavers and a continuous pipelined decoder.1

Aside from potential synchronization advantages, the stream paradigm also offers hope of performance improvements for turbo codes. In [6], it was shown that continuous encoding used in the stream paradigm was superior to block encoding over the ensemble of all block interleavers in a PCCC. Furthermore, the stream paradigm provides compatibility for turbo codes with any periodic interleaver, which can be either block or nonblock. To date, only block interleavers have been used for turbo code design, despite the availability of many nonblock interleavers, such as convolutional interleavers and Ramsey interleavers [7]–[9]. These nonblock interleavers potentially offer improved performance over block interleavers with comparable delay, but with smaller periods which simplifies the problem of deinterleaver synchronization. With stream-oriented turbo codes, the performance and design of nonblock interleavers can now be studied and will be the central theme of this work.

The paper is organized as follows. In Section II, the encoding and decoding of stream-oriented PCCC is described and periodic interleavers discussed, including block and nonblock interleavers. In Section III, design rules for general periodic interleavers are developed with attention to specific nonblock periodic interleavers. Section IV reports simulation results verifying design rules and demonstrating the performance of both block and nonblock periodic interleavers in a stream-oriented PCCC. A summary is provided in Section V.

II. STREAM-ORIENTED PCCC

A. Stream-Oriented Encoding

While the stream paradigm for turbo codes applies to both PCCC and SCCC, we focus in this work on PCCC. The traditional PCCC encoder is constructed from two or more recursive systematic convolutional (RSC) codes and an interleaver(s). For encoding, one constituent encoder operates on the input sequence while the other constituent encoders operate on permutations of the input. As in [6], PCCC encoding can be either block-wise or continuous. For block encoding, the data sequence is segmented into nonoverlapping blocks of length L, with each block encoded and later decoded independently. Block interleavers are required and the constituent encoders are forced to the same state (usually the zero state) at the beginning of every block encoding using either trellis termination or trellis truncation. For block encoding, the PCCC can be viewed as a linear, time-invariant block code [10].

1Continuous encoding and decoding of turbo codes was proposed for PCCC in [5], where continuous constituent decoding algorithms were developed.
With block encoding, trellis termination appends extra symbols to the input block to force the constituent encoders to the desired state [11]. Trellis truncation, on the other hand, is simply a hard reset of the encoder states to the desired state at the appropriate time. With trellis termination, both the initial and final states of the encoder are known, which can benefit decoding. However, an energy penalty is incurred due to the transmission of the tail symbols. With trellis truncation, the initial states of the encoders is known at the decoder without an energy penalty, but the final encoder states must be assumed or “learned,” which can slightly degrade performance [12], [5].

In contrast to block encoding, continuous encoding removes all restrictions on the constituent encoders, allowing them to be “free-running” for all time. If block interleavers are used, the resulting PCCC can be viewed as linear, time-varying block code or a large-constraint length, time-varying convolutional code [6]. Without an imposed block structure, continuous encoding admits both block and nonblock interleavers for turbo code design by viewing the interleaver as a single input/single output scrambling device or synchronous interleaver. From [8], any periodic interleaver, block or nonblock can be constructed in this fashion and thus used in a stream-oriented turbo code as shown in Fig. 1.

### B. Interleavers: Theory and Notation

For turbo code design, the interleaver is a key element since the code properties, e.g., the free distance and weight spectrum, are functions of the permutation [13], [14]. Therefore, an understanding of interleaver structure and properties is central to intelligent design and analysis of turbo codes.

An interleaver is a device which permutes a sequence of symbols in some deterministic manner. Mathematically, an interleaver \( \Pi \) scrambles an infinite sequence \( x[k] \to \{x[k]\}_{k=-\infty}^{\infty} \) to produce a permuted sequence \( y = \{y[k]\}_{k=-\infty}^{\infty} \). The interleaver is described by a permutation law \( \pi(k) \) with \( y[k] = x[\pi(k)] \). Associated with every interleaver \( \Pi \) is a deinterleaver \( \Pi^{-1} \) which reorders the permuted sequence, with \( \Pi^{-1} \) producing a sequence \( z[k] = \{z[k]\}_{k=-\infty}^{\infty} \) such that \( z[k] = x[k - D] \), i.e.,

\[
\pi^{-1}(\pi(k)) = k - D.
\]

From [15], the interleaver/deinterleaver delay \( (D) \) or span, which is of primary interest for turbo codes, is defined as

\[
D = \max_k \{k - \pi(k)\} - \min_k \{k - \pi(k)\}.
\]

From [7], an interleaver is either periodic or aperiodic, with most interleavers of interest being periodic. For periodic interleavers, the permutation law is a periodic function of time modulo the interleaver period \( N \), where \( N \) is defined as the smallest positive integer such that

\[
\pi(k) \mod N = \pi(k + N) \mod N.
\]

Periodic interleavers can further be subdivided into block interleavers and nonblock interleavers [7]. Block interleavers, such as the row/column interleaver, are perhaps the most common periodic interleavers. However, nonblock interleavers, such as the convolutional interleaver, are commonly used in communications systems such as cable modems, due to their scrambling properties and advantages in deinterleaver synchronization over block interleavers with comparable scrambling properties [7], [16].

Any interleaver has a simple representation using a delay function \( d_k(\pi(k)) \), where \( \pi(k) = k - d_k(k) \) [15]. For periodic interleavers, the delay function is very convenient since \( d_k(k) = d_k(k + N) \) allowing a periodic interleaver to be completely described by an \( N \)-length delay vector \( \mathbf{d}_k = \{d_k(k)\}_{k=0}^{N-1} \). For periodic interleavers, the deinterleaver is also periodic with period \( N \) and can be described by the delay vector \( \mathbf{d}_k^{-1} \). Since \( \pi^{-1}(\pi(k)) = k - D \), a relation between the delay functions of an interleaver and deinterleaver is given by

\[
d_k(k) + d_k^{-1}(k - d_k(k)) = D, \quad 0 \leq k < N. \quad (2)
\]

From (2), we have a condition for a valid periodic interleaver/deinterleaver system and a periodic function \( d_k(k) \) corresponds to an interleaver if and only if a periodic function \( d_k^{-1}(k) \) exists to satisfy (2) (e.g., an associated deinterleaver exist). The relation in (2) can be used to systematically construct pseudo-random periodic interleaver/deinterleaver systems.

Besides the permutation law and delay function, an interleaver can also be described by a mapping function \( \rho(k) \) where the interleaver input at time \( k \) maps to a position \( \rho(k) \) in the output with \( x[k] = y[\rho(k)] \). It can be shown using the delay functions that

\[
\rho(k) = \pi^{-1}(\pi(k)) + D = k + D - d_k^{-1}(k).
\]

The interleaver mapping function is central to the analysis and design of turbo codes since it describes the scrambling of certain “bad” input sequences.

For the construction of periodic interleavers, Ramsey has noted that every causal periodic interleaver can be constructed using a tapped shift register along with a commutator or multiplexer as shown in Fig. 2 [8]. If the desired permutation is noncausal (i.e., \( d_k(k) < 0 \) for at least one \( 0 \leq k < N \)), there exist a causal equivalent interleaver which can be implemented using the circuit of Fig. 2 [15]. In the interleaver circuit, the shift-register length is equal to the interleaver/deinterleaver delay \( D \) and the delay functions serve as select signals for the multiplexers. For deinterleaver synchronization, an \( N \)-fold

---

2 Continuous encoding is called continuous co-decoding in [6].
ambiguity exists which motivates the use of interleavers having small periods.

1) Block Interleavers: The most common type of periodic interleaver is the block interleaver, which accepts blocks of symbols and performs an identical permutation on each block. A periodic interleaver \( \Pi \) is a block interleaver if and only if \( \exists (L, I) \in \mathbb{Z}^2 \) such that for every integer \( i \), the vector

\[
[\pi(I + iL), \pi(I + iL + 1), \ldots, \pi(I + (i + 1)L - 1)]
\]

contains \( L \) consecutive integers [15]. The smallest integer \( L \) which satisfies this condition is called the interleaver block length. Thus, a block interleaver can be described by an \( L \)-length permutation vector \( \pi_b = \{\pi_b(k)\}_{k=0}^{L-1} \) where \( \pi_b \) is simply a permutation of the integer vector \([0, 1, \ldots, (L - 1)]\). Using the permutation vector, the permutation law can be written as in [15]

\[
\pi(k) = L \left\lfloor \frac{k}{L} \right\rfloor + \pi_b(k \mod L)
\]

\[
= k - k \mod L + \pi_b((k \mod L) - L),
\]

(3)

(4)

While the interleaver length \( L \) is the most cited parameter for turbo codes, the period \( N \) and delay \( D \) are also of interest. Using (4), it can be shown that the period of a block interleaver equals its length \( (N = L) \) where \( L \) is defined above. Furthermore, the delay function of a block interleaver is bounded by

\[
-(L - 1) \leq d_x(k) \leq (L - 1)
\]

and the interleaver delay \( D \) is bounded by

\[
0 \leq D \leq 2(L - 1).
\]

A corollary result is that a periodic interleaver with period \( N \) and delay \( D = 2(N - 1) \) is not a block interleaver.

In the literature, a variety of block interleavers have been used for time diversity in communication systems as well as concatenated coding schemes, including turbo codes. Examples include row/column interleavers, helical interleavers [17], congruence based interleavers [18], and interleavers based on circular shifting [13]. As an example, the permutation vector for a block interleaver based on circular shifting is given by

\[
\pi_b(k) = (ak + r) \mod L, \quad 0 \leq k < L
\]

(5)

where the step size \( a \) and offset \( r \) are chosen relatively prime to \( L \).

Block interleavers, such as row/column and circular shifting interleavers, have relatively simple closed-form permutation maps and regular permutation patterns. For best performance, turbo codes typically require interleavers with more irregular or random-like permutation maps which do not have simple closed forms. Examples of such interleavers include interleavers constructed using Pseudo-random Noise (PN) sequences, \( S \)-random interleavers from [13] which are pseudo-randomly constructed to be an \((S, S)\)-interleaver\(^4\) and other pseudo-random interleaver constructions. In particular, irregular interleavers appear necessary for the near-capacity performance of turbo codes, but the lack of a closed-form permutation map complicates analysis and design.

As a periodic interleaver, a block interleaver can be constructed using the circuit in Fig. 2. However, a common block interleaver construction uses two memory buffers of length \( L \) where a block of symbols is written into one buffer while the preceding block, residing in the alternate buffer, is being read according to \( \pi_b(k) \). In this fashion, the read/write operations alternate between registers and the interleaving delay is fixed at \( D = 2L \) [15]. This block interleaver realization avoids causality issues, but requires extra delay and storage compared to the causal equivalent minimum realization of [15].

As an example, consider the implementation of an \( m \times n \) row/column interleaver. This interleaver envisions an \( m \times n \) matrix, where the symbols are written by rows and read out by columns. The interleaver length and period are \( N = L = mn \) and the permutation law is

\[
\pi(k) = mn \left\lfloor \frac{k}{mn} \right\rfloor + n(k \mod m) + k \mod mn \frac{mn}{m}.
\]

(6)

\(^4\)The terms block interleaver and row/column interleaver have been used interchangeably in the literature. Here, the term block interleaver describes a broad class of interleavers, of which the row/column interleaver is a special case.

\(^5\)From [8], an \((S_2, S_1)\)-interleaver has the property that if \( 0 < |i - j| < S_2 \), then \( |\pi(i) - \pi(j)| \geq S_1 \).
Here,
\[ d_{\pi}(k_{\min}) = d_{\pi,\min} = -(m-1)(n-1) \]
at \( k_{\min} = (n-1) \) and \( d_{\pi,\max} = (m-1)(n-1) \). Thus, the interleaver is noncausal with delay
\[ D = 2(m-1)(n-1) > L = mn. \]
As in [15], a causal equivalent minimum realization exists with
permutation law \( \Pi_{CE} \) given by
\[ \pi_{CE}(k) = (m-1)(n-1) + \pi(k). \]

For \( m = n = 16 \), the causal equivalent minimum realization has a delay of \( D = 450 \), compared to a delay of \( D = 512 \) for the two-buffer realization.

2) Nonblock Interleavers: The distinction between block
and nonblock interleavers is similar to the distinction between
block and convolutional codes [7]. In many communications
systems, nonblock interleavers, such as the convolutional
interleaver, are preferred over block interleavers since the deinterleaver synchronization problem is simplified by keeping the
interleaver period small. For turbo codes, nonblock interleavers
with small periods coupled with the synchronization scheme
proposed in [19] provide “self-synchronizing” turbo decoders
without the overhead necessary for conventional block synchro-
nization techniques. Furthermore, the motivation to use
these nonblock interleavers is enhanced if their performance
in turbo systems is comparable or better than block interleaver
performance with comparable delay.

The literature on nonblock interleavers traces back to early
work by Ramsey and Forney in the 1970s. In [8], the convolu-
tional interleaver and shift-register nonblock interleavers were
introduced. Forney also discusses the convolutional interleaver
in [9], while generalizing the concept and introducing the idea of
concatenated interleavers in [20]. We now consider in more de-
tail several of these nonblock interleavers and develop a pseudo-
random nonblock interleaver construction.

3) Convolutional Interleavers (CI): The convolutional in-
terleaver (CI) or Type I Ramsey interleaver of [8] has been used
for mitigating the effects of bursty channels for coded systems
[9]. A CI has a serial implementation as in Fig. 2 [8] with a
permutation law which is a function of the period \( N \) and spacing
parameter \( B \)
\[ \pi_{CI}(k) = k - NB(k \mod N). \]

From (7), it can be seen that the CI has a very regular permuta-
tion map with a delay of \( D = NB(N-1) \).

As in [7], CIs are often compared to row/column interleavers
in terms of their scrambling properties. In Figs. 3–6, the circuits
and permutation laws are shown for a causal minimum realization
of a \( 4 \times 4 \) row/column block interleaver \( L = N = 16, D = 20 \)
and a CI with period \( N = 5 \) and spacing parameter
\( B = 1 \) \( (D = (N-1)NB = 20) \). In Figs. 3 and 4, the shift-register
implementations of the interleavers and the commutator hopping
patterns are shown. From the CI circuit, it is observed that
the CI is a nonblock interleaver since \( D = 20 > 2(N-1) = 8 \).
In Figs. 5 and 6, the block structure of the \( 4 \times 4 \) row/column
design is evident, as is the lack of block structure for the CI.

For turbo codes, a CI is attractive for several reasons. First,
the closed-form mapping function (8) of the CI makes it more
amenable for design and optimization. Secondly, the deinter-
leaver synchronization problem is simplified versus block in-
terleavers of comparable delay since the period \( N \) is generally
much smaller than the delay \( D \approx N^2B \) (see Figs. 5 and 6).
A CI also has many attractive scrambling properties as noted in
[8]. First, a CI is a minimum-delay \( (NB, N) \)-interleaver, where
from [8], \( S_2, S_1 \)-interleaver is an interleaver where
\[ |i - j| < S_2 \rightarrow |\pi(i) - \pi(j)| \geq S_1, \quad \forall i \neq j. \]
This spreading property has been shown useful for block inter-
leaver design in turbo codes [13].

It is observed that this interleaver must be nonblock since \( D = (N-1)NB = 20 > 2(N-1) = 8 \).
4) Generalized Convolutional Interleavers (GCI): The delay function of a CI can be expressed as $d_{GCI}(k) = N f(k)$ where $f(k) = B(k \mod N) = f(k + N)$. From (2), any nonnegative periodic function with period $N$ of the form $d_{GCI}(k) = f(k)N$ corresponds to a periodic interleaver with $d_{GCI}(k) = N(f_{\text{max}} - f(k))$. We term an interleaver of this form a Generalized Convolutional Interleaver (GCI) which has permutation law

$$\pi_{GCI}(k) = k - N f(k).$$
Without loss of generality, GCIs have \( f(0) = 0 \) and delay \( D = N f_{\text{max}} \). From (2), the GCI mapping function is
\[
\rho_{\text{GCI}}(k) = k + N f(k).
\] (10)

While CIs are very regular versions of GCIs, pseudo-random construction of \( f(k) \) results in GCIs which are irregular, which can be advantageous for turbo code design.

GCIs also offer other attractive design properties for turbo codes. First, they can be designed with spreading properties. From [20], if \( f(k) \neq f(l) \) for \( k \mod N \neq l \mod N \), then the GCI is an \((N, N)\)-interleaver. A GCI also offers more possibilities for design with given a fixed period and delay constraint. For a given period, CIs can be constructed having delays of the form \( D = (N - 1)N B \). In contrast, a GCI is much more flexible in design, allowing delays of the form \( D = N f_{\text{max}} \) where \( f_{\text{max}} \) is any nonzero integer. For example, a CI with \( N = 10 \) and \( D = 100 \) does not exist since \( D = N B (N - 1) = 90B \). However, many different GCIs can be constructed with the property \( D = 100 \) and \( N = 10 \). Consider the GCI with
\[
f = \{f(k)\}_{k=0}^{N-1} = [0, 8, 2, 6, 4, 1, 10, 7, 9, 3]
\]
which has period \( N = 10 \) and delay \( D = N f_{\text{max}} = 100 \).

5) Cascade Convolutional Interleavers (CCI): In [20], cascade-type interleavers, constructed from the serial concatenation of two (or more) periodic interleavers was proposed. If the constituent interleavers are block, then the result is a non-block interleaver. Here, we consider the concatenation of two CIs, which we term a cascade convolutional interleaver (CCI). Constructed from an inner CI with parameters \((N_i, B_i)\) and an outer CI with parameters \((N_o, B_o)\), the CCI permutation map is given by \( \pi_{\text{CCI}}(k) = \pi_i(\pi_o(k)) \), where \( \pi_i(k) \) and \( \pi_o(k) \) are the permutation maps of the inner and outer CIs, respectively.

The period and delay of a CCI are related to the constituent CIs. If \( N_i \) and \( N_o \) are relatively prime, the CCI has period \( N = N_i N_o \) and delay \( D = D_i + D_o \) [20]. However, if \( N_i \) and \( N_o \) are not prime, then the period of the CCI can be less than the product of the component periods. For example, a design with \( N_o = N_i = N \) gives
\[
\pi_{\text{CCI}}(k) = k - N (B_o + B_i) (k \mod N)
\]
where the CCI is equivalent to a CI with period \( N \) and spacing parameter \( B = B_o + B_i \). For design with delay \( D \), the period of the CCI would typically be larger than a CI of similar delay, but with a more irregular permutation map.

6) Pseudo-Random Interleavers: Given a desired period and delay, periodic interleavers can be pseudo-randomly constructed. If the delay \( D > 2(N - 1) \), the resulting interleaver will be non-block. We now focus on pseudo-random interleaver constructions using the interleaver/deinterleaver delay functions. Not every pseudo-randomly generated delay function \( d_p(k) \) is a proper interleaver. For example, a design with \( N = 2, D = 3 \), and delay function in vector form given by \( d_p = [0, 3] \) is not a proper interleaver. For this design, \( \pi(0) = 0 = \pi(3) \) indicating that all even-indexed symbols are duplicated, while all odd-indexed symbols are deleted. As illustrated by this example, the pseudo-random construction must also incorporate the interleaver/deinterleaver properness condition of (2).

We now present an algorithm for the construction of a random periodic interleaver with period \( N \) and delay \( D \). The algorithm is based on (2) and jointly constructs the interleaver and deinterleaver delay functions. Without loss of generality, the algorithm assumes \( d_p(k) = 0 \), giving \( d_{p-1}(k) = D \). In the algorithm pseudo-code below, the variable \( \text{set}(k) \) is used to indicate which values of \( d_{p-1}(k) \) have been assigned (or set) and \( r\text{and}(\cdot) \) refers to a pseudo-random number generator with integer outputs in the range \([0, D]\).

**Variables:**
- \( N \)-Interleaver/deinterleaver period
- \( D \)-Interleaver/deinterleaver delay
- \( d_p(k) \)-Interleaver delay function
- \( d_{p-1}(k) \)-Deinterleaver delay function

**Initialization:**
- \( d_p(0) = 0 \)
- \( d_{p-1}(0) = D \)
- \( \text{set}(0) = \text{TRUE} \)
- for \( k = 1 \) to \( N - 1 \)
- \( \text{set}(k) = \text{FALSE} \)

**Recursion:**
- for \( k = 1 \) to \( N - 1 \)
  - \( t = \text{rand}(\cdot) \)
  - while \( \text{set}((k-t) \mod N) == \text{TRUE} \)
  - \( t = \text{rand}(\cdot) \)
  - end
  - \( d_p(k) = t \)
  - \( d_{p-1}((k-t) \mod N) = D - t \)
  - \( \text{set}((k-t) \mod N) = \text{TRUE} \)
  - end
- \% Ensure interleaver has delay \( D \)
  - \( t = D \)
  - \( k = 1 + \text{rand}(\cdot) \mod (N - 1) \)
  - \( d_p(k) = t \)
  - \( d_{p-1}((k-t) \mod N) = 0 \)

The algorithm guarantees a causal periodic interleaver of period \( N \) and delay \( D \), which will be nonblock if \( D > 2(N - 1) \). By pseudo-random construction, these interleavers typically have irregular permutations maps. Furthermore, the algorithm can be modified to generate \((S, S)\)-interleavers by adding a check of the spreading condition. The attainable values of \( S \) are determined by the period \( N \) and the delay \( D \). In [13], it was found that block interleavers typically require \( S \leq \sqrt{N}/2 \). We have observed empirically that \((S, S)\) nonblock interleavers can be generated provided \( S \leq \min\{N, \lfloor \frac{N}{2} \rfloor\} \).

C. Stream-Oriented Iterative Decoding

For the decoding of stream-oriented PCCC, there are several candidate decoding architectures. In this work, we appeal to the continuous pipelined decoding architecture suggested in [1]. Here, each decoding module in the pipeline is composed of two constituent decoders, called soft-input/soft-output (SISO) [5] modules, along with appropriate buffering for data align-
Fig. 7. Conventional stream-oriented PCCC decoding module.

Fig. 8. Conventional stream-oriented PCCC decoding architecture.

ment and interleaving/deinterleaving. Each decoding module is continuous and capable of one iteration.

Within each decoding module, the SISO decoder uses a continuous version of the maximum a posteriori (MAP) or Log–MAP decoding algorithm. Examples include sliding-window versions of the Log–MAP such as SW-Log–MAP (SW-LM), SW-Max-Log–MAP (SW-MLM) as well as the soft-output Viterbi algorithm (SOVA). The complexity and memory requirements of the constituent decoders vary with algorithm and implementation, which can be either SISO or multiple-input/multiple-output (MIMO) [21], [5], [22]. Provided that the window length is sufficiently large, the sliding-window version of the Log–MAP algorithm incurs no performance degradation.

A decoding module is shown in Fig. 7 for a stream-oriented PCCC using two constituent encoders. The global decoder, capable of iterations, connects decoding modules in series as shown in Fig. 8. For binary phase-shift keying (BPSK) modulation, the variables, , and correspond to the noisy matched-filter outputs for the systematic and parity symbols. The variables and correspond to the extrinsic information and the log-likelihood ratio, respectively [1]. The pipelined decoding architecture shown in Fig. 8 can support stream-oriented PCCC decoding using either block or nonblock periodic interleavers.

III. PERIODIC INTERLEAVER DESIGN FOR PCCC

Having discussed periodic interleavers, we now consider the problem of interleaver design for turbo codes. Specifically, we consider design for a stream-oriented PCCC using two, identical constituent RSCs. The interleaver should attempt to scramble input sequences which produce low parity weight in one encoder, such that large parity weight is produced by the other encoder. The result is a code with a small number of low-weight codewords or a "thin" weight spectrum [14]. It is the weight-spectrum properties of turbo codes that enable performance close to the Shannon limit for moderate bit error rates (BER $\geq 10^{-6}$) [23]. The interleaver should also attempt to create a sufficiently large free distance for the code. Although difficult to achieve with a PCCC, a large free distance leads to good asymptotic performance (i.e., low error floor) [24].

Ideally, we desire the “best” interleaver for a given period and delay with the criteria being the free distance or the multiplicity of low-weight codewords. Unfortunately, testing all input sequences is required to choose an interleaver based on this criterion. Interleaver optimization for all input sequences is too difficult and feasible design methods have instead attempted to optimize for certain input types. The focus of this optimization is typically low-weight inputs, which produce low-parity weight in a single constituent encoder. While the design methodology is not strictly optimum, it is motivated by results in [24] and [14], which suggest that weight-2 and other low-weight inputs commonly result in low-weight codewords in a PCCC.

While interleaver design for low-weight inputs can be effective, the interleaver design requires a certain amount of irregularity for turbo codes. This phenomenon refers back to random coding ideas, where Shannon showed that random block codes are good with high probability as the block length grows. Battail has noted that with large pseudo-random block interleavers, turbo codes can be considered “random-like” codes with weight distribution which are “close” to that of random codes and in simulation have been shown capable of near-capacity performance [23]. Even for small delay, pseudo-random block interleavers designed with a spreading condition (i.e., $\{S, S\}$-interleavers) have been shown effective in simulation [13].
In the sections that follow, the design of periodic interleavers will be considered for a stream-oriented PCCC. General design rules will be developed based on the permutation of low-weight input sequences. As in [13] and [18], weight-2 and weight-4 inputs will be considered. For the case of block interleavers, the impact of continuous encoding will be investigated and comments made on the problem of interleaver “edge effects.”

A. Encoder Terminology

Periodic interleaver design for low-weight inputs requires a review of notation from [25]. From [25], a finite-weight input to an RSC which produces finite parity weight (and thus finite code weight) is termed a finite codeword (FC) input.7 A finite-weight input to an RSC should clearly be avoided since or equivalently if is a cyclic weight term. For design based on weight-4 inputs, represents the total number of nonmerged trellis sections in both constituent trellises for a given input . For a weight-2 FC input, we can write the weight-2 effective separation as a function of the nonzero bit indexes for the input with

\[ \Delta_2(k, k + iP) = |\rho(k) - \rho(k + iP)| + |iP| = |iP - d_{\text{err}}(k + iP) + d_{\text{err}}(k)| + |iP|, \]  

(11)

The free distance produced among all weight-2 inputs is proportional to the minimum weight-2 effective separation

\[ \Delta_{\text{min},2} = \min_{\forall i,k;i \neq 0} \{\Delta_2(k, k + iP)\}. \]  

(12)

From [26], can be written

\[ d_{\text{free},2} = 2 + 2(g_t + g_m) + g_c \left( \frac{\Delta_{\text{min},2}}{P} \right) \]  

(13)

where is a transient weight term, is a merge weight term, and is a cyclic weight term. For design based on weight-2 FC inputs, the free distance due to weight-2 inputs is maximized by maximizing and . For interleaver design, should be maximized or made sufficiently large to ensure large code weight for all weight-2 inputs.

For periodic interleavers, several observations can be made regarding . Using (10) and the fact that if the period of the encoder is a multiple of the period of the interleaver, then . From this observation, the case of should clearly be avoided since the worst case is the case of a multiple of . For all weight-2 FC inputs, the free distance due to weight-2 inputs is maximized by maximizing and . For interleaver design, should be chosen relatively prime to prevent low-weight codewords resulting from weight-2 inputs.

Regardless of the primeness of and , a weight-2 FC input of the form will always be GFC input with by the periodicity of the interleaver. This observation leads to an upper bound on for all periodic interleavers,

\[ d_{\text{free},2} \leq 2 + 2(g_t + g_m + g_c N). \]  

(14)

The upper bound in (14) for a PCCC is based only on the period of the interleaver and the properties of the constituent encoder. It indicates that regardless of the permutation map, the period of the interleaver needs to be sufficiently large to avoid low-weight codewords due to weight-2 inputs. The choice of is especially critical when is small, as is the case for four-state encoders ( ) [6].

If an interleaver achieves the upper bound for , we say it is optimum for the scrambling of weight-2 inputs since it maximizes . It should be noted that this optimality places no constraints on the interleaver delay and experiments show that the bound can be achieved quite easily as the delay is allowed to grow large. For block interleavers, which have delay

In [13], these sequences were called self-terminating because the trellis path of the constituent encoder leaves and returns to the zero state without a tail sequence.

6 Here, is a dummy variable used to represent the sequence in polynomial form and should not be confused with the delay of the interleaver.

7 The idea of maximizing was suggested in [27] and is tied to maximizing the period of the RSC.
$D < 2(N - 1)$, interleavers have been found which achieve $\Delta_{\text{min}, 2} \approx \sqrt{2NP} < 2NP$ [26]. It will be shown that nonblock interleavers, such as the CI, can achieve the $2NP$ bound for sufficiently large delay.

2) Weight-4 FC Inputs: Interleavers optimized for weight-2 inputs are not necessarily the “best” interleavers, as noted in [13]. In fact, there are many FC inputs which can, and most likely will, result in a lower free distance for the PCCC. In [13], it was shown that a class of weight-4 FC inputs of the form

$$u(D) = (1 + D^{4P})D^k + (1 + D^{iP})D^r$$

with $r > k$, $r \neq k + iP$, and $r + jP \neq k + iP$ are often troublesome.\footnote{It should be noted that these weight-4 sequences are not the only weight-4 FC inputs. For example, an RSC with $d(D) = 1 + D + D^2 + D^3$ has a period $P = 4$, yet an input of the form $u(D) = 1 + D + D^2 + D^3$ is a weight-4 FC input.} For scrambling with regular interleavers such as row/column and circular shifting interleavers, it has been shown that these weight-4 FC inputs result in a relatively small free distance for a PCCC [13], [18].

Unlike weight-2 inputs, analysis of these sequences is more difficult and a computer search which attempts to find $\Delta_{\text{min}, 4}(u)$ is more useful. Recall that $\Delta_4$ is the sum of lengths of all nonmerged trellis sections from constituent encoder trellises for a given weight-4 input $u$. Using $\Delta_{\text{min}, 4}$, $d_{\text{free}, 4}$ can be upper-bounded by assuming one parity bit for every nonmerged trellis section giving a loose upper bound of $d_{\text{free}, 4} \leq 4 + 2\Delta_{\text{min}, 4}$. For interleavers with small periods ($N < 50$), computer searches over this class of weight-4 FC inputs can be done efficiently and will give insights into potential interleaver weaknesses.

C. Block Interleaver Design

To this point, interleaver design has focused on general periodic interleavers. We now partition the design problem into block and nonblock interleaver design and first focus on block interleavers. In particular, the effect of continuous versus block encoding on block interleaver design for PCCC is of interest.

For continuous encoding, block interleaver design is concerned only with the scrambling of FC inputs. On the other hand, block encoding requires that both FC and non-FC inputs be considered since low-weight codewords can result from premature termination or truncation of the constituent encoders. In particular, block interleaver design with block encoding must avoid interleaver “edge effects,” where the interleaver maps symbols at the end of the block to the end of the interleaved block resulting in low-weight codewords. A block interleaver can be said to suffer “edge effects” if for some design distance $w > 0$, $\pi_L(L - k) \geq L - w$ for some value $1 \leq k \leq w$. For an interleaver with edge effects, truncating or terminating the constituent trellis can prematurely limit the parity weight produced by certain input sequences (e.g., $u(D) = D^{iP}$), resulting in low-weight codewords.

As an example, consider an interleaver with $\pi_L(L - 1) = L - 1$ which is used in a $R = 1/3$ PCCC with block encoding. The probability of a randomly chosen block interleaver having this property is $(L - 1)!/L! = 1/L$. Using trellis truncation, the PCCC has $d_{\text{free}} = d_{\text{free}, 1} \leq 3$ resulting from an input sequence $u = [0 \cdots 0]$, regardless of the constituent encoders and the interleaver length. With trellis termination, the same weight-1 input sequence would be appended with $\nu$ tail bits to force termination. Assuming that all tail bits and parity bits were nonzero, the free distance of the PCCC is upper bounded by $d_{\text{free}} \leq 3 + 3\nu$. While the free distance using trellis termination is larger than for truncation, it still can be quite small. For example, using four-state RSC ($\nu = 2$) gives an upper bound on the free distance of $d_{\text{free}} \leq 9$. For continuous encoding, edge effects are not problematic since there is no premature truncation of the parity sequences.

The impact of edge effects for PCCC is evident in the results of [6], where it was shown via an average upper bound over all block interleavers of fixed $L$ that continuous encoding/decoding outperformed block-encoded PCCC with trellis termination and trellis truncation. The fraction $1/L$ of the interleavers in the ensemble having $\pi_L(L - 1) = L - 1$ (and $d_{\text{free}} = 3$) dominate the bound, resulting in poor performance. For continuous encoding and trellis termination, the ensemble performance is dominated by weight-2 FC inputs [24]. Since the free distance due to weight-2 FC inputs will generally be greater than 3 for constituent codes of interest, the ensemble performance is better. The superiority of continuous encoding over trellis truncation and termination can be attributed to the premature termination of non-FC inputs which results in a larger multiplicity of low-weight codewords.

The analysis in [6] demonstrates the performance of a uniform or “average” block interleaver while saying nothing regarding the performance of a specific interleaver. For a specific interleaver, if edge effects are present, continuous encoding and block encoding with trellis termination will significantly outperform block encoding with trellis truncation. Without edge effects, which is often the case for large block interleavers, all encoding methods will perform equally well as will be demonstrated in simulation.

D. Nonblock Interleaver Design

Unlike block interleavers, nonblock interleaver design is not concerned with edge effects, focusing only on scrambling of FC inputs. Here, we consider the design of several specific nonblock interleavers based on low-weight FC inputs and on irregular pseudo-random constructions with spreading properties. The interleavers of interest include convolutional and generalized convolutional interleavers as well as both CCIs and random nonblock interleavers. In cases where closed-form permutation maps exist, optimization is performed by analyzing low-weight FC inputs. For other cases, interleaver design and performance are studied by simulation.

1) Convolutional Interleaver Design: For CI design, we wish to formulate criteria for choosing “good” values of $N$ and $B$. We begin with weight-2 FC inputs [24] and optimize $N$ and $B$ based on maximizing $d_{\text{free}, 2}$. We then extend our analysis to check for weaknesses due to weight-4 FC inputs.
a) Weight-2 FC inputs: From (7) and (11), the weight-2 effective separation function for a CI is

\[
\Delta_2(k, k+iP) = |iP + NB[(k+iP) \mod N - k \mod N]| + iP. \tag{15}
\]

A weight-2 FC input is a GFC input if and only if

\[
\Delta_2(k, k+iP) \mod P = 0.
\]

By examining these sequences using \( \rho_{\text{eff}}(k) \), we can make several observations regarding the proper choice of \( N \) and \( B \).

**Proposition 1:** For a CI with \( \gcd(NB, P) = 1 \)

\[
\Delta_2(k, k+iP) \geq \min\{2NP, NBP\}
\]

for all weight-2 GFC inputs and thus \( \Delta_{\text{min},2} = 2NP \) for \( B \geq 2 \).

**Proof:** If \( \gcd(NB, P) = 1 \), then \( u(D) \) is a GFC input if and only if

\[
(k+iP) \mod N - k \mod N = iP \tag{16}
\]

with \( 0 \leq |l| \leq \bigg\lfloor \frac{N}{P} \bigg\rfloor \). We now consider the cases of \( l = 0, l > 0 \) and \( l < 0 \).

- **Case 1** \( l = 0 \): From Euclid’s Remainder Theorem, the solutions are \( i = rN \) where \( r \in Z, \forall k \in Z \). Since \( r \neq 0 \), all weight-2 FC inputs satisfying (16) with \( l = 0 \) are of the form \( u(D) = (1 + D^{(rN)P})D^k \) and

\[
\Delta_2(k, k+rNP) = 2r|NP| \geq 2NP.
\]

- **Case 2** \( l > 0 \): Here, \( l > 0 \) implies that

\[
(k+iP) \mod N = iP + k \mod N.
\]

Since \( (k+iP) \mod N \geq 0 \), we only need to consider values of \( k \in (0, 1, N-1-lP) \). For these values of \( k \), the solutions are \( i = l+rN \) where \( r \in (0, 1) \). Inputs of the form \( u(D) = (1 + D^{(l+r)NP})D^k \) with \( k \mod N \in (0, 1, N-1-lP) \) are GFC inputs with

\[
\Delta_2(k, k+(l+rN)P) = (l+rNP)P + IPNB \geq 2P + NBP > NBP.
\]

- **Case 3** \( l < 0 \): Using arguments as in Case 2, solutions exist for \( k \in (lP, lP+1, \ldots, N-1) \) with \( i = rN-l \) where \( r \in (0, 1) \). For these solutions

\[
\Delta_2(k, k+(rN-l)P) = (rN-l)P + |(rN-l)P - NBP| \geq NBP.
\]

From these three cases, we have that for all weight-2 GFC inputs, the minimum effective separation is

\[
\Delta_{\text{min},2} \geq \min\{2NP, NBP\} = 2NP, \quad \text{for } B \geq 2. \quad \square
\]

**Proposition 2:** For a CI with \( \gcd(NB, P) = 1 \) and \( N < P \), a weight-2 FC input \( u(D) = (1 + D^{IP})D^k \) is a GFC input if and only if \( i = lN \) with \( I \in (1, 2, \ldots) \) and we have \( \Delta_{\text{min},2} = 2NP \).

**Proof:** Since \( N < P \), we have that

\[
|(k+iP) \mod N - k \mod N| < N < P.
\]

Therefore, for all GFC inputs \( (k+iP) \mod N - k \mod N = 0 \) is the only solution of (16). From Case 1, the values of \( k \) and \( i \) which satisfy this condition are \( \forall k \in Z, i = rN \) where \( r \in (1, 2, \ldots) \). Thus, all weight-2 GFC inputs have

\[
\Delta_2(k, k+rNP) = 2rNP \geq 2NP. \quad \square
\]

From Propositions 1 and 2, if \( \gcd(NB, P) = 1 \) and \( N < P \) or \( B \geq 2 \), a CI is optimum for weight-2 inputs over the class of all periodic interleavers with period \( N \) in that it maximizes \( d_{\text{free},2} \). Furthermore, the effective separation and \( d_{\text{free},2} \) grow linearly in \( N \) at a rate of \( 2N \). In [13], similar results for weight-2 FC inputs were shown for block interleavers based on circular shifting. For these designs, the effective separation grew at a rate roughly \( \sqrt{2N} \), with a delay which is approximately \( D = 2N \). With this delay assumption, these block interleavers have a \( d_{\text{free},2} \) which grows roughly as \( \sqrt{D} \). For the CI, the delay is approximately \( D = (N-1)NB \approx N^2B \) giving a growth rate for \( d_{\text{free},2} \) of approximately \( 2\sqrt{D/B} = 2\sqrt{D/B} \). Comparing the growth rates for block and convolutional interleavers, it is observed that CI’s provide a better tradeoff between \( d_{\text{free},2} \) and delay provided \( B \leq 4 \).

With proper design, CIs can achieve very large values for \( d_{\text{free},2} \). As an example, consider the bound in (14) for a 16-state RSC with generator \( (33/31)_8 \), for which \( g_1 = 1, g_2 = 8, \) and \( g_m = 1 \). The period of this encoder is \( P = 15 \) and by choosing \( N = 14 \), we get \( d_{\text{free},2} = 230 \). Large values of \( d_{\text{free},2} \) can also be achieved with smaller \( N \) and thus less delay. For example, choosing \( N = 7 \) instead results in \( d_{\text{free},2} = 118 \), with \( D = 42 \). We should note, however, that the true free distance of the code is typically much less than \( d_{\text{free},2} \) in these examples.

b) Weight-4 FC inputs: We have shown that with proper design, CIs are optimum periodic interleavers for weight-2 inputs. However, CIs are very regular and we might suspect problems with weight-4 FC inputs. As for weight-2 inputs, we can define an effective separation function \( \Delta_4(u) \) which represents the total number of nonmerged trellis sections in both constituent encoders. By searching all weight-4 inputs of the form \( u(D) = (1 + D^{IP})D^k + (1 + D^{IP})D^r \), we can upper-bound \( \Delta_{\text{min},4} \) leading to an upper bound on \( d_{\text{free},4} \).

In Fig. 9, we show results of a computer search for \( \Delta_{\text{min},4} \) versus the CI period \( N \) with different values of \( B \) and a constituent encoder with period \( P = 15 \).

From Fig. 9, we first note that Propositions 1 and 2 are confirmed, showing that \( \Delta_{\text{min},4} = 2NP \) can be achieved provided certain primeness conditions are maintained. Secondly, the weakness of CIs to weight-4 FC inputs is revealed in the fact that \( \Delta_{\text{min},4} = 4P = 60 \) for almost all CI designs. This is a result of an input of the form \( u(D) = (1 + D^{IP})D^k + (1 + D^{IP})D^r \) scrambling to another sequence of this form with two short
merges occurring in each constituent trellis for almost all values of 
and 
. The one exception that was found occurs for 
, where 
. For this case, 
 meaning that performance is likely limited by weight-2 inputs rather than weight-4 inputs. Except for this special case, CI performance, especially for large-delay designs, will be limited by weight-2 FC inputs as is the case for row/column block interleavers [13]. It is conjectured that the weight-2 weakness is a result of the regularity of the CI. We now proceed to analyze more irregular interleavers, such as GCIs, CCIIs, and pseudo-random designs to see if weight-4 performance can be improved.

E. Generalized Convolutional Interleaver Design

For a GCI, the two design elements are the period 
 and the integer-valued function 
. For weight-2 FC inputs, the effective separation function for a GCI is

\[ \Delta_2(k, k+iP) = [iP + N[f(k+iP) - f(k)]] + iP \]  

(17)

Our goal is to maximize the minimum effective separation, which leads to several observations regarding 
, 
, and the effective separation.

**Proposition 3:** For a GCI with 
 and 
, 
 if 
 and 
 for \([k, l] \leq N-1\).

**Proof:** If 
, then 
 if and only if \([f(k+iP) - f(k)] \mod P = 0\). Since \([a-b] \mod P = [a \mod P - b \mod P] \mod P\) then \(\Delta_2(k, k+iP) \mod P = 0\) if and only if 

\[ f(k) \mod P = f(l) \mod P, \quad \text{for } l \neq k \]

and \(0 \leq [l, k] \leq N - 1\). If 

\[ f(k) \mod P \neq f(l) \mod P \]

for \(k \neq l\) and \(0 \leq [k, l] \leq N - 1\), then 

\[ \Delta_2(k, k+iP) \mod P = 0 \]

if and only if \(i = rN\), giving \(\Delta_{\text{min,2}} = 2NP\) which is the upper bound for an interleaver with period 
.

If \(N < P\) and \(\gcd(N, P) = 1\), GCI designs which are optimum for weight-2 inputs can be constructed provided the interleaver delay is sufficiently large \(D \geq N(N-1)\). With this construction, the function 
 should be designed such that all possible differences \(f(k) - f(l)\) for \(k \neq l\) are prime with respect to 
. If \(N > P\), we must avoid cases where \(f(k) = f(k+P)\) since \(f(k) = f(k+P)\) results in 
. In general, \(f(k)\) can be designed without this condition provided \(f_{\text{max}} > N\) which allows the effective separation (and the minimum weight due to weight-2 inputs) to grow linearly in 
. A simple construction of an optimum GCI has period 
 which is prime with respect to 
 and 
, where 
 is relatively prime to 
 and the resulting 
 [20].

In Proposition 3, it is shown there exist GCI designs which are optimum, yet not CIs (e.g., \(f(k) = B[k \mod N]\)). Thus, GCIs can be used which have more irregular permutation maps, yet still perform well for weight-2 inputs. Unlike CIs, it is hoped that these less regular interleaver designs will perform well for weight-4 FC inputs.

1) CCI Design: The mapping function for a CCI is a complicated function of four variables \(N_c, B_0, N_i,\) and \(B_i\). However, from the general analysis of periodic interleavers, we propose...
that \( N_1 B_2 \) and \( N_2 B_3 \) should be relatively prime with respect to the period of the encoder \( P \). For more insight into the design of CCIs, we again resort to computer searches for weight-2 and weight-4 FC inputs as well as simulations.

2) Random and \( S \)-Random Nonblock Interleaver Design: The design and analysis of random nonblock interleavers is also complicated by the lack of a closed-form expression for the permutation law. However, as for \( S \)-random block interleavers, a spread condition appears to be a viable design criterion for large \( N \) and \( D \).

3) Nonblock Interleaver Comparison: The permutation map for GCIs and pseudo-random interleavers are typically more irregular than CIs and other structured interleavers (e.g., row/column block interleavers). For these irregular interleavers, we are interested in design based on the scrambling of weight-2 and weight-4 FC inputs. Due to the lack of a closed-form permutation map, we resort to a computer search to design GCI and random interleavers. We then compare “good” GCI and random designs to CI designs by studying the weight-2 and weight-4 minimum effective separation. Due to the lack of a closed-form permutation map, we resort to a computer search to design GCI and random interleavers. We then compare “good” GCI and random designs to CI designs by studying the weight-2 and weight-4 minimum effective separation. In Fig. 10, we have plotted the minimum among \( \Delta_{\text{min},2} \) and \( \Delta_{\text{min},4} \) for CI, GCI, and pseudo-random designs over a range of periods \( N \). Here, the delay is fixed at \( D = 2(N-1)N \) and constituent encoders with period \( P = 15 \) and feedback polynomial \( d(D) = (1 + D + D^4) \) or \( d = 31_8 \) are assumed. It can be seen that the GCI and random design outperform the CI, especially for large periods (and large delay). Since CIs are optimum for weight-2 inputs, the difference is found in the scrambling of weight-4 inputs, with “good” GCIs and random designs performing much better than CIs. Fig. 10 also illustrates the importance of the relative primeness of the interleaver and encoder periods. (Note that designs with \( N = 3, N = 5, N = 15, \) and \( N = 30 \) are poor designs for all interleavers considered.) For interleaver large delay, GCI and random constructions perform better for weight-4 inputs and are thus better candidates for achieving near-capacity performance with turbo codes, provided the primeness of the interleaver and encoder period is maintained.

IV. SIMULATION RESULTS AND CONCLUSIONS

In this section, simulation results are shown for stream-oriented PCCC which will reinforce the interleaver design results and analysis. The simulations are performed using coherent BPSK over an additive white Gaussian noise (AWGN) channel. The stream-oriented PCCC is an \( R = 1/3 \) code which uses identical 16- or 8-state RSCs. Puncturing of the parity streams is performed to achieve an overall code rate of \( R = 1/2 \). For decoding, the stream-oriented turbo decoder discussed in Section III is used. The constituent decoders use either the sliding-window Log–MAP (SW–LM) or the sliding-window Max-Log–MAP (SW–MLM) with a window length of \( W = 9(\nu + 1) \).

A. Block Interleaver Results

We begin by comparing the performance of block- and stream-oriented PCCC. Recall that comparable performance is expected, provided interleaver edge effects are avoided. For comparison, we consider a \( 12 \times 16 \) row/column interleaver \((N = L = 192\) and \( D = 390\)) and a PN-interleaver\(^{11} \) with

\(^{11}\)The permutation vector \( \tau_k \) of a PN-interleaver is formed using the state sequence of a PN generator.
block size of \((L = N = 1024\) and \(D = 1977\)). We use the trellis termination scheme of [11] which requires the transmission of a \(\nu = 4\)-bit information (tail) and parity sequence for each encoder. The code rate for the punctured PCCC using block encoding with trellis termination is \(R = L/(2L + 4\nu)\).\(^{12}\)

From Fig. 11, continuous encoding and block encoding with trellis termination performed much better than trellis truncation for the row/column interleaver design, which is limited by edge effects \((\pi_0(191) = 191\)\). With trellis truncation, the PCCC design has a free distance of \(d_{\text{free}} = 2\), which causes flaring of the BER curves, also called the “error floor.” Continuous or stream encoding and block encoding with trellis termination do not exhibit BER flaring. The transmission of the tail sequences for block encoding with trellis termination incurs an energy penalty of \((384/(384 + 16))\) or 0.18 dB versus continuous encoding which can be observed in the plot.

Fig. 11 also shows the performance of block- and stream-oriented PCCC using a \(L = 1024\) block PN-interleaver. It is observed that both block- and stream-oriented PCCC perform equally well since the interleaver does not suffer edge effects and the energy penalty due to termination is very small (0.07 dB). As for edge effects, the probability that a pseudo-randomly generated block interleaver has edge effects decreases with \(L\), meaning that for moderate to large \(L\), random block interleavers are expected to perform equally well for both stream- and block-oriented PCCC. For small \(L\), edge effects are more probable and stream-oriented PCCC can be used to lessen the problem of BER flaring, avoid the rate loss (and energy penalty) of trellis termination, and allow structured interleavers, such as the row/column interleaver, to be used for design.

B. Nonblock Interleaver Results

We begin our simulation study of nonblock interleavers with the CI where optimum designs have \(\gcd(NB, P) = 1\), while keeping \(N\) sufficiently large. In Figs. 12 and 13, the performance using a CI with different values of \(N\) and a fixed value of \(B = 1\) is shown. It should be noted that these results are not constrained by the interleaver delay, which is \(D = (N - 1)NB\).

In Fig. 12, it is observed that performance generally improves with \(N\) for \(N < 2P\) and \(\gcd(N, P) = 1\). However, Fig. 13 indicates that performance does not necessarily improve with \(N\) for \(N > 2P\). Here, the minimum \(E_b/N_0\) in decibels required to achieve BER = \(10^{-4}\) and \(10^{-5}\) is shown to increase as \(N\) increases. It is conjectured that this is due to a rapid increase in the number of low-weight codewords and the weight-4 weakness. Furthermore, “spikes” appear in the curves where the relative primeness condition is violated \((I_4(NB, P) = 1\) which reinforces the design rule \(\gcd(NB, P) = 1\). Furthermore, a design with \(N = 2P = 30\) is shown to provide good BER performance despite not satisfying \(\gcd(NB, P) = 1\).

Aside from the primeness condition on \(N\) and \(P\), our weight-2 and weight-4 analysis of CIs was unclear on the role of the spacing parameter \(B\). To understand the role of \(B\), we have fixed the period \(N = 14 < P = 15\) and simulated different values of \(B\) with delay \(D = 182B\). In Fig. 14, it is observed that performance generally increases with \(B\) (and \(D\)). In fact, performance improves for all \(B\) considered, without degradation. However, there is a point beyond which

---

\(^{12}\)No puncturing is performed on the tail sequences.
increasing $B$ increases delay without noticeable performance benefits. This performance limitation is directly attributable to the scrambling of weight-4 inputs.\(^\text{13}\)

\(^{13}\)This effect has also been observed for row/column and other linear block interleavers [18].

To this point, we have considered CI design without a delay constraint. We now consider the problem of designing the best CI for a given delay requirement. As a design example, we choose $D = 1260$, which constitutes a moderate delay design. The choice of $D = 1260$ is motivated by the fact that a variety of CI designs are possible which exactly achieve this delay.
In Fig. 15, the performance of almost all CI designs having $D = 1260$ is shown. The simulation uses eight-state constituent encoders with generator $(17/15)_8$ and period $P = 7$. From the BER curves, it is observed that designs with $N = P^2 = 7$ and $N = 3P = 21$ performed poorly, confirming the design rule that $N$ and $P$ should be relatively prime. Second, designs with small $N$, performed poorly as predicted by the weight-2 analysis. Finally, it was found that the best designs typically balanced the values of $N$ and $B \leq N$ with $N$ sufficiently large and $NB$ and $P$ relatively prime.
From our CI analysis, we next examine other nonblock interleavers. In particular, we compare performance of these nonblock interleavers to CI performance as well as block interleaver performance. As design examples, we simulated CI, GCI, CCI, and random nonblock interleavers having delays $D \approx 182$, $D \approx 1200$, and $D \approx 11000$. All designs have $\gcd(N, P) = 1$ with $N$ kept intentionally small for synchronization purposes. The GCI designs are based on the design conditions developed while the random designs are based on a spreading condition (i.e., $(S, S)$)-interleavers). These examples constitute small-delay designs, which are applicable to latency-sensitive applications, and moderate-to-large latency designs for space and satellite communications.

In Fig. 16 ($D \approx 182$), the stream-oriented PCCC uses 16-state RSCs with generator $(33/31)_8$ and period $P = 15$. Results are presented after 10 iterations using the SW-MLM ($W = 45$). From the figure, all nonblock interleavers considered performed equally well, provided the period was sufficiently large ($N \geq 7$). In fact, doubling the period from $N = 7$ to $N = 14$ resulted in negligible performance improvements despite the fact that $d_{\text{free}, 2}$ doubled as well. Also note that the CCI, which has a much larger period $N = 42$ performed no better than the CI, GCI, and random designs with period $N = 7$. Provided $N$ is sufficiently large and the design rules followed, random and GCI nonblock interleavers can perform as well as CIs, which are optimum for weight-2 inputs.

Fig. 16 also illustrates the potential for nonblock interleavers as compared to “good” block interleavers. The $12 \times 16$ row/column block interleaver has a delay $D = 330$ and a period $N = 192$. In Fig. 11 and from results of [28], this interleaver in a stream-oriented PCCC or block-oriented PCCC with trellis termination achieved a $\text{BER} = 10^{-5}$ at approximately 3.0 dB using the SW-MLM algorithm. From Fig. 16, all nonblock interleavers, with $N \geq 7$ perform equally well, but with half the delay and much less synchronization ambiguity. For better comparison, a CI with $N = 14$ and $B = 2$ has delay $D = 362$ which is comparable to the $D = 355$ of the $12 \times 16$ row/column block interleaver. From Fig. 13, this design achieves $\text{BER} = 10^{-5}$ at 2.5 dB, thus giving 0.5 dB of additional coding gain for approximately the same delay and much smaller synchronization ambiguity. These results are made more significant by observations in [28] where for small block lengths, row/column designs with trellis termination or continuous encoding performed as well as pseudo-random block interleavers of the same length. Our results indicate that for small delay designs, simple nonblock interleavers can operate at slightly lower signal-to-noise ratio (SNR), with less delay and synchronization ambiguity than “good” block interleavers.

In Fig. 17, nonblock interleavers with a moderate delay of $D \approx 1200$ are considered. Here, eight-state RSCs are used with 10 decoder iterations using the SW-MLM ($W = 36$). From Fig. 17, GCIs and random designs perform better than optimized CIs in the error floor region for equal $N$. This was predicted by our analysis of weight-4 inputs. It is also noted that the worst designs for CI, GCI, and random interleavers corresponded to smallest periods $N = 6$ indicating that $N$ needs to be sufficiently large. However, increasing $N$ too much was of diminishing benefit since the CCI design with $N = 187$ performed no better than GCI and random designs with $N = 30$.

For comparison to block interleavers, the best design of Fig. 17 is the GCI with period $N = 30$ which achieves
**Fig. 17.** Nonblock interleavers with $D \approx 1200$ in an $R = 1/2$ stream PCCC.

BER = $10^{-5}$ at approximately 1.75 dB. In Fig. 11, the $L = 1024$ PN-block interleaver achieved the same error rate at $E_b/N_0 = 2.0$ decibels using 16-state encoders rather than 8-state encoders. The nonblock design provides 0.25-dB performance improvement for less delay $D = 1200 < 1977$, less synchronization ambiguity $N = 30 < 1024$, and simpler constituent codes (8-state versus 16-state). Thus far, we have shown that nonblock interleavers in a stream PCCC can perform better than block interleavers for a given delay. However, we have not demonstrated that nonblock interleavers are capable of the near-capacity performance available with large, random block interleavers. For this purpose, we consider large-delay nonblock interleavers ($D \approx 11000$) which have small periods $N < 25$ for simplified deinterleaver synchronization. In Fig. 18, simulation results are shown after 15 iterations using the SW-LM ($W = 45$). Results show that for BER = $10^{-5}$ both random and GCI designs can perform within 0.8 dB of the capacity limit, which is $E_b/N_0 = 0$ dB for an $R = 1/2$ code, while maintaining small periods of $N = 11$ and $N = 22$. The GCI and random interleaver with period $N = 22$ were the best designs, while the CI with $N = 11$ and $B = 97$ was the worst. However, an important result is that performance very near the capacity limit can be achieved with large delay, nonblock interleavers having small periods ($N < 25$).

V. CONCLUSION

In conclusion, we have explored a stream-oriented paradigm for turbo codes which applies to both PCCC and SCCC. The key distinctions between this stream paradigm and the traditional block-oriented approach is the use of free-running constituent encoding, continuous interleaving, and continuous pipelined decoding.

We focused on stream-oriented PCCC, in particular the problem of interleaver design. From analysis of weight-2 inputs, it was shown that the period of the periodic interleaver should be chosen relatively prime to the period of the constituent encoders, avoiding at all cost cases where $N$ is a small multiple of $P$. This choice ensures that the free distance due to weight-2 inputs is maximized. We also developed a bound on this distance for a given periodic interleaver with period $N$ given as $d_{free,2} \leq 2 + 2(g_f + g_m + g_r N)$. Based on this bound, it was suggested and later verified in simulation that the period of the interleaver cannot be too small, or low-weight codewords result from weight-2 inputs.

For block interleaver design, it was shown that the stream approach removed the interleaver edge effect problem. This enables block interleavers which perform poorly in block-oriented PCCC to perform well in stream-oriented PCCC. For nonblock interleaver design, several special interleavers, including the CI, GCI, CCI, and random designs were considered. Design rules were developed and it was shown that with proper design CI and GCI are optimum periodic interleavers for weight-2 inputs.
Nonblock interleavers with $D \approx 11000$ in an $R = 1/2$ stream PCCC.

Fig. 18. Nonblock interleavers with $D \approx 11000$ in an $R = 1/2$ stream PCCC.

However, CIs, which are very regular interleavers, are limited by weight-4 inputs for large-delay designs.

Using a continuous pipelined decoding architecture, the performance of block and nonblock interleavers was studied in a stream PCCC. Compared to “good” block interleavers, non-block interleavers were found with superior performance and less delay and synchronization ambiguity. For large-to-moderate delay, nonblock interleavers with small synchronization ambiguity ($N < 25$) were shown to perform well, even within 0.8 dB of the capacity limit with periods as small as $N = 11$.

ACKNOWLEDGMENT

The authors wish to thank G. D. Forney for his constructive comments and for providing [20]. The authors are also very thankful for the anonymous reviewers, whose constructive comments and suggestions have helped to improve this work.

REFERENCES


