УДК 621.391

ПЕРЕМЕЖЕНИЕ В КАНАЛЬНОМ КОДИРОВАНИИ: СВОЙСТВА, СТРУКТУРА, СПЕЦИФИКА ПРИМЕНЕНИЯ

А. Ю. Баринов

Череповецкое высшее военное инженерное училище радиоэлектроники, 162622, г. Череповец, пр-т Советский, д. 126

Статья поступила в редакцию 17 декабря 2018 г., после доработки – 24 декабря 2018 г.

Аннотация. Перемежение является крайне важной процедурой в канальном кодировании для получения максимальной дистанции между кодовыми словами. В статье представлен обзор результатов исследований принципов перемежения с учетом особенностей помехоустойчивых кодов и требуемых условий использования. При этом особое внимание уделено исследованию *S*-случайного перемежителя. Предложены варианты технической реализации перемежителей. Рассмотрено применение перемежения для обеспечения защиты информации.

Ключевые слова: перемежение, перемежитель, псевдослучайный перемежитель, *S*-случайный перемежитель, свойства перемежителя, структура перемежителя, применение перемежения, помехоустойчивость, защита информации

Abstract. Interleaving is an extremely important procedure in channel coding where we look for the longest distance between codewords. Nowadays there are many kinds of interleavers and their design depends upon the component codes and the application. However, in most cases, theory of interleavers is examined in a general or abstract way that can be equivocated.

In this paper, the author thoroughly examined interleavers from combination of mathematical and engineering standpoints. The basic properties of interleavers such as period, memory, latency, spreading factor and dispersion are presented. The proposed classification captures two parent classes of block and convolutional interleavers with division according to random, pseudorandom and deterministic generation process. Special attention is paid to *S*-random interleaver. Several schemes

for efficient hardware realization of interleaving are presented. In addition, the graphical representation of the popular interleavers is depicted.

The author considers the usage of interleavers for error protection. Whereas in a conventional serial concatenated code an interleaver is used merely to spread out burst errors, the interleaver in modern iteratively decodable codes plays a far more important role. In general, in modern codes, deterministic interleavers can perform equal or better than pseudorandom interleavers if the size of the interleaver is small, and pseudorandom interleavers perform better than deterministic interleavers when the size of the interleaver is medium or large. Nevertheless capacity-approaching performances for modern codes are achieved using pseudorandom interleavers with period of order several thousand bits.

In conclusion, the author considers secondary usage of interleavers – for information security. Wireless telecom systems can be intercepted and monitored, as a result there are many technics of protection. A very high level of security may be obtained using combination of cryptography and steganography technics with channel coding. Enhanced security may be obtained using large pseudorandom interleavers in the channel coding. The evaluation of *S*-random interleavers diversity is also presented.

Key words: interleaving, interleaver, pseudorandom interleaver, *S*-random interleaver, interleaver properties, interleaver structure, applications of interleaving, noise immunity, information security

Введение

Современные информационно-телекоммуникационные системы (ИТС) трудно представить без методов канального кодирования, обеспечивающих надежную передачу, хранение и обработку информации. При этом в канальном кодировании используются методы помехоустойчивого кодирования и предполагается реализация методов скремблирования и перемежения.

В настоящее время существует обширная теория кодирования, которая постоянно обновляется. Ознакомиться с данной теорией можно через соответствующие исследования, в том числе и через обзорные статьи.

Исследованию псевдослучайных последовательностей (ПСП) положенных в основу скремблирования, посвящено много работ. Сравнительно полное системное изложение принципов, свойств, областей применения помехоустойчивого кодирования можно найти в работах [1, 2] и других. Однако те же сведения о перемежении часто рассматриваются в общих чертах и допускают неоднозначную трактовку.

Цель работы – представить обзор результатов исследований принципов перемежения с учетом особенностей помехоустойчивых кодов и требуемых условий использования.

1. Свойства перемежения

Термин перемежение (чередование) понимается как в широком, так и в специальном смыслах.

В широком смысле перемежением называется изменение по установленному правилу естественного порядка элементов некоторой последовательности.

В специальном смысле перемежение представляет собой такое изменение порядка следования элементов последовательности, при котором любые два ее элемента, находящиеся в пределах расстояния s элементов друг от друга, оказываются разнесенными на расстояние не меньше s.

Перемежение является обратимым безызбыточным преобразованием, а процедуру, обратную перемежению, принято называть деперемежением (восстановлением). В результате деперемежения восстанавливается естественный порядок следования элементов.

Основное назначение методов перемежения-деперемежения – повышать помехозащищенность ИТС. Наряду с этим перемежение усложняет структуру сигналов, что затрудняет несанкционированный доступ к содержащейся в них информации.

Процедуру перемежения реализует перемежитель – устройство с конечным числом состояний, имеющее один вход и один выход, принимающее последовательности элементов в фиксированном алфавите и вырабатывающее

выходную последовательность того же алфавита, идентичную входной во всем, кроме порядка следования элементов [3]. Каждому перемежителю соответствует деперемежитель – устройство, восстанавливающее оригинальный порядок следования элементов.

Перемежитель может быть периодическим, у которого с течением времени перестановки неизменны, и, в противном случае, непериодическим. На практике используются периодические перемежители, в силу того, что реализация непериодических достаточно сложна.

Существуют разные способы представления перемежителей, наиболее распространенными из них являются вектор перестановки (таблица перемежения), матрица перестановки, графические зависимости, уравнения, а также непосредственно функциональная схема.

Например, перемежителю, сдвигающему $i-ый$ входной элемент на $\pi(i)-ую$ выходную позицию, соответствует вектор перестановки $\mathbf{\Pi}=[\pi(1),\pi(2),\ldots,\pi(K)]$, где K размер периода перемежителя.

Исследование и описание свойств перемежителей представлено в работах [3-5]. Можно выделить основные свойства перемежителей:

1. Периодичность означает, что для периода $K \geq 1$ выполняется равенство

$$\pi(i+K)=\pi(i)+K,\ \forall\ i\in[1,2,...,K].$$

Период (длина, размер) – это минимальный промежуток времени K элементарных посылок, через который используемое перемежителем правило перестановки повторяется.

2. Емкость памяти – это размерность в битах массива (регистров) перемежителя. Память требуется для временного хранения в ней каждого из перемежаемых блоков данных и во многих случаях необходима для хранения таблицы перестановок перемежителя.

3. Задержка – это максимальный промежуток времени между моментом поступления элемента на входе перемежителя и появлением данного элемента на его выходе:

$$delay_{\pi} = \max_{i} \left| \pi(i) - i \right|, \ \forall \ i \in [1, 2, ..., K].$$

4. Рассеивание (spreading factor) – (s, r). Перемежитель имеет фактор рассеивания (s, r), если любые два элемента исходной последовательности, находящиеся на расстоянии в пределах s элементов друг от друга, после перемежения оказываются на расстоянии не меньшем r элементов.

То есть, для (s, r) должно выполняться условие:

$$\text{если } \left| i - j \right| \le s, \text{ то } \left| \pi(i) - \pi(j) \right| \ge r,$$

где s – это наибольшее натуральное число, такое, что $s \le r$; $i, j \in [1, 2, ..., K]$ – позиции элементов исходной последовательности, $i \ne j$.

В случае (s, s), фактор рассеивания часто называют рассеиванием s.

В случае $(s = 1, \ r = 1)$, по другому $s = 1$, перемежитель представляет собой любую перестановку.

В случае $(s = 1, r)$, величину $J = r + 1$ называют глубиной (степенью) перемежения. Глубина перемежения – минимальное расстояние в элементах на выходе перемежителя, между двумя любыми элементами, которые были соседними на входе перемежителя.

5. Дисперсия характеризует «случайность» перестановки перемежителя. Высокая дисперсия указывает на разнообразие переставленных расстояний между элементами, поэтому для обеспечения высокой дисперсии следует избегать регулярной структуры перестановки.

Для перемежителя существует следующее множество:

$$D(\pi) = \left\{ (j - i, \ \pi(j) - \pi(i)) \right\}, \ \forall \ 1 \le i < j \le K$$

Дисперсия перемежителя $\left| D(\pi) \right|$ – это мощность множества $D(\pi)$.

В случае тождественной перестановки, когда $\pi(i) = i$, $\forall \ 1 \le i < j \le K$:

$$D(\pi) = \left\{ (1, 1), (2, 2), \ldots, (K - 1, K - 1) \right\},$$

и дисперсия минимальна $\left| D(\pi) \right| = K - 1$.

Возможное количество вариантов сочетаний i и j равняется $K(K - 1) / 2$, следовательно:

$$K-1 \le \left|D(\pi)\right| \le K(K-1)/2 .$$

Для характеристики степени случайности перемежителя используется нормированная дисперсия:

$$d = \frac{2\,|\,D(\pi)\,|}{K(K-1)} . \qquad (1)$$

Чем ближе к единице значение нормированной дисперсии d, тем более случайной является структура перестановки.

Два последних свойства можно рассмотреть на простом примере перемежителя $\mathbf{\Pi} = [5, 3, 6, 1, 4, 2]$.

Из представленного вектора перестановки видно, что перемежитель имеет фактор рассеивания $(s = 1, r = 2)$, глубину $J = 2$.

Вычисление дисперсии начинается с расчета всех пар, для которых значения на входе перемежителя находятся на расстоянии 1 друг от друга:

$$(2-1, \pi(2)-\pi(1)) = (1,-2),$$
$$(3-2, \pi(3)-\pi(2)) = (1,3),$$
$$(4-3, \pi(4)-\pi(3)) = (1,-5),$$
$$(5-4, \pi(5)-\pi(4)) = (1,3),$$
$$(6-5, \pi(6)-\pi(5)) = (1,-2)$$

Итак, первые элементы множества $D(\pi)$: $(1,-2), (1,3), (1,-5)$. Далее вычисляются оставшиеся пары, для которых значения на входе находятся на расстояниях 2, 3, 4 и 5. В результате, полный список множества имеет вид:

$$D(\pi) = \{(1,-2), (1,3), (1,-5), (2,1), (2,-2), (3,-4), (3,1), (4,-1), (5,-3)\} .$$

Тогда $|\,D(\pi)\,| = 9$ и согласно (1) нормированная дисперсия $d = 0,6$.

На практике требования к свойствам конкретного перемежителя согласуют с выбранным методом помехоустойчивого кодирования. В общем случае желательно, чтобы перемежитель имел большое рассеивание, высокую дисперсию, малую задержку, малую память и был простым в реализации [5].

Важно заметить, что указанные свойства взаимосвязаны и требования к ним противоречат друг другу. Поэтому в настоящее время существует множество видов перемежения, различающихся по обеспечению выигрыша от

кодирования, сложности реализации и многим другим параметрам. Для каждой конкретной системы с помехоустойчивым кодированием может быть выбран собственный вид перемежения, учитывающий специфику данной системы.

2. Структура перемежения

Перемежение можно выполнять только над множеством элементов. Следовательно, поступающие элементы предварительно сохраняются в буфере перемежителя. В зависимости от организации буфера разделяют блочные и сверточные перемежители. Данное разделение положено в основу классификации перемежителей, представленной на рис. 1.

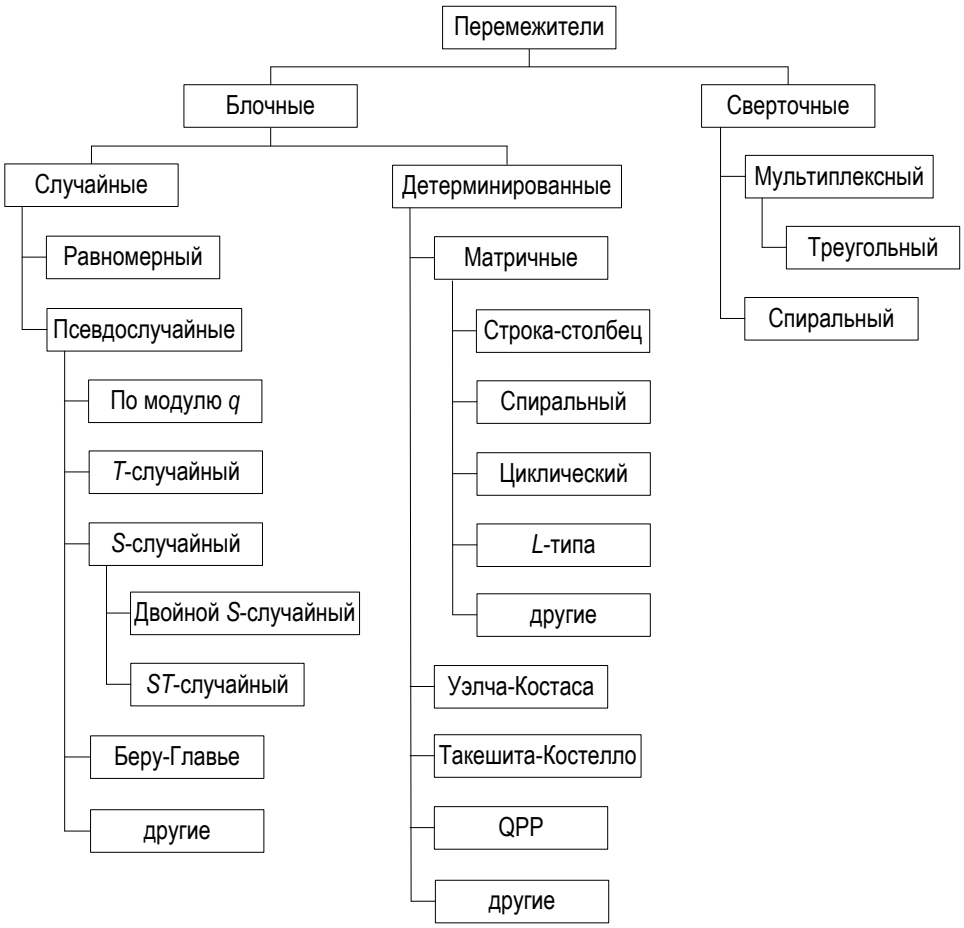


Рис. 1. Классификация перемежителей

Блочный перемежитель предусматривает сохранение поступающего блока (кадра) из K элементов в буфере, и устройство производит одну и ту же перестановку каждого блока независимо от других блоков.

Сверточный перемежитель имеет начальное состояние внутри буферного окна и выполняет перестановку первого блока последовательности с учетом

начального состояния, а в дальнейшем выполняет перестановку с учетом элементов, оставшихся в данном окне от предыдущего блока.

Сверточное перемежение

Среди сверточных перемежителей различают традиционный сверточный перемежитель, мультиплексный перемежитель, спиральный перемежитель.

Традиционный сверточный (треугольный) перемежитель работает следующим образом. Входные данные последовательно подаются на набор линий задержки (регистров сдвига) возрастающих длин (треугольная структура). При поступлении каждого нового элемента последовательности входной коммутатор переключается на новую линию задержки, куда поступает следующий элемент, в то время как самый старый элемент этой линии поступает на выход. Коммутаторы на входе и выходе работают синхронно. Пример реализации такого перемежителя на четырех линиях задержки $[0, D, 2D, 3D]$ представлен на рис. 2, а порядок следования элементов после перемежения, при $D = 1$, следующий (* - параметр начального состояния):

1,*,*,*,5,2,*,*,9,6,3,*,13,10,7,4,17,14,11,8,21,18,15,12,25,…

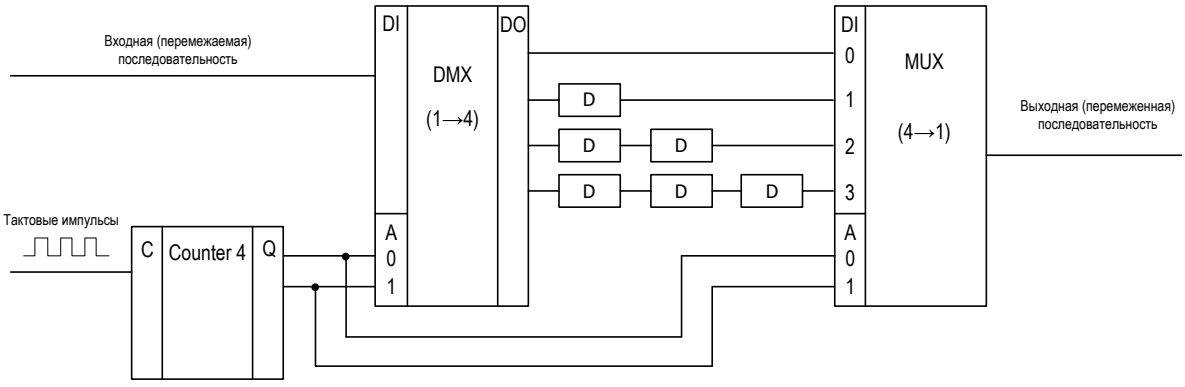


Рис. 2. Реализация традиционного сверточного перемежителя

Обобщением традиционного сверточного перемежителя является сверточный мультиплексный (табличный) перемежитель, который состоит из набора регистров с независимо программируемой задержкой (произвольная структура). Например, для набора регистров с задержкой [0, 4, 2, 5] получится следующая последовательность на выходе:

1,*,*,*,5,*,*,*,9,*,3,*,13,*,7,*,17,2,11,*,21,6,15,4,25,10,19,8,…

Спиральный сверточный перемежитель [6] можно представить следующим образом. Входная последовательность делится на блоки размера $M \times N$, далее каждый блок записывается в матрицу, состоящую из N столбцов и «бесконечного» количества строк по спирали. Затем содержимое матрицы считывается по M строкам. Например, для блоков размера 4×3 (рис. 3) получится следующая последовательность на выходе:

1,*,*,2,5,*,3,6,9,4,7,10,13,8,11,14,17,12,15,18,21,16,19,22,…

1	*	*
2	5	*
3	6	9
4	7	10
13	8	11
14	17	12
15	18	21
16	19	22
...	20	23
...	...	24
...	...	...

Рис. 3. Матрица 4×3 спирального сверточного перемежителя

Блочное перемежение (случайное и псевдослучайное)

В зависимости от процесса генерирования перестановки блочные перемежители делятся на случайные, псевдослучайные, детерминированные (алгебраические).

Случайные перемежители устраняют регулярные зависимости в кодированной последовательности. Перестановка элементов случайного перемежителя генерируется с помощью датчика случайных чисел в диапазоне, ограниченном периодом перемежения. Хорошей статистической моделью случайного перемежителя является равномерный (uniform) перемежитель [7]. Равномерный перемежитель выполняет перестановку входного блока размера

K элементов и веса w любым из $C_K^w = \dfrac{K!}{w!(K-w)!}$ различных вариантов с равной вероятностью $1/C_K^w$.

Случайный перемежитель, как правило, имеет высокую дисперсию. Однако «неконтролируемая» генерация случайных чисел может привести к тому, что другие характеристики данного перемежителя будут недостаточны для обеспечения высокой помехоустойчивости передачи.

Для управления выбором перестановок в псевдослучайных перемежителях на случайную перестановку элементов накладываются определенные ограничения.

Например, в нечетно-четных (odd-even) случайных перемежителях, ограничение заключается в следующем. Элементы, находящиеся на четных позициях, должны отображаться на четные позиции и наоборот. Обобщением таких «mod 2» перемежителей являются «mod q» случайный перемежитель. Данные перемежители обычно используют совместно с перфорацией.

В T-случайных перемежителях действует следующее условие. Элементы, находящиеся на расстояниях L, $2L$, $3L$, … , TL друг от друга, не могут находиться на расстояниях L, $2L$, $3L$, … , TL друг от друга в новом перемеженном порядке.

Отличными характеристиками обладает S-случайный перемежитель [8] (где $S = 1, 2, 3, \ldots$), который представляет собой разновидность псевдослучайного перемежителя с ограничением на фактор рассеивания $(s = S, r = S + 1)$, то есть:

$$\text{если } |i - j| \le S \text{, то } |\pi(i) - \pi(j)| > S,\ i, j \in [1, 2, \ldots, K],\ i \ne j. \qquad (2)$$

Алгоритм построения случайного перемежителя предлагается на рис. 4. Исходными данными являются: параметр S и вектор перестановки случайного перемежителя, обозначенный как $\mathbf{P} = [p(1), p(2), \ldots, p(K)]$. В алгоритме используются следующие обозначения: $[i = k(l)n] = (i = k, k + l, k + 2l, \ldots, n)$ – последовательность значений целочисленной переменной i (l – шаг

последовательности); $p(l)=[\ ]$ – исключение l-го элемента из вектора $\mathbf{P}$; $length(\mathbf{P})$ – текущий размер вектора $\mathbf{P}$.

Результатом работы алгоритма является вектор перестановки S-случайного перемежителя $\mathbf{\Pi}=[\pi(1),\pi(2),...,\pi(K)]$, в котором обеспечивается условие (2).

Время работы алгоритма увеличивается с увеличением параметра S, и алгоритм не гарантирует успешный результат. Однако обычно возможно найти перемежитель для $S<\sqrt{K/2}$ за разумное время [8].

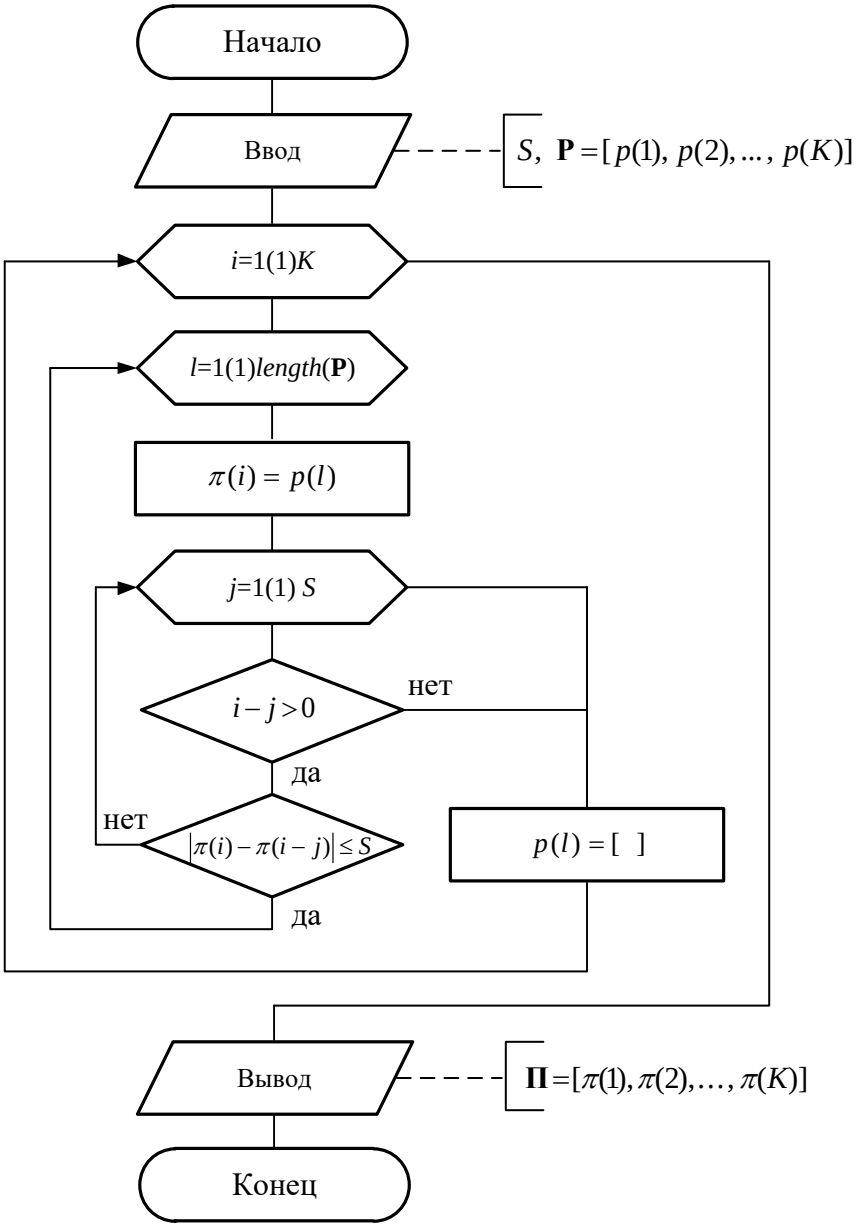


Рис. 4. Алгоритм построения S-случайного перемежителя

В настоящее время известно много модификаций *S*-случайных перемежителей, например [9, 10]. Также за счет дополнительных ограничений можно получить: двойной (2-step) *S*-случайный перемежитель; *ST*-случайный пермежитель; нечетно-четный (odd-even) *S*-случайный перемежитель и другие.

Двойной *S*-случайный перемежитель представляет собой *S*-случайный перемежитель с ограничением на расстояние между позицией входного элемента i и новой (перемеженной) его позицией на выходе $\pi(i)$:

$$\left|i-\pi(i)\right|>S_2,$$

что приводит к снижению корреляции между входом и выходом.

ST-случайный перемежитель идентичен *S*-случайному, с той разницей, что все элементы входной последовательности, находящиеся на расстояниях *L*, 2*L*, 3*L*, … , *TL* друг от друга, в новом перемеженном порядке находятся на расстоянии большем *S*.

Недостатком случайных и псевдослучайных перемежителей является невоспроизводимость процесса генерирования перестановок. Таблица перемежения таких устройств генерируется предварительно и целиком сохраняется в памяти кодера/декодера. На рис. 5 изображен вариант аппаратной реализации перемежения, где вектор перестановки $\mathbf{\Pi}=[\pi(1),\pi(2),\ldots,\pi(K)]$ хранится в ПЗУ (ROM) в виде K двоичных $n-$разрядных слов.

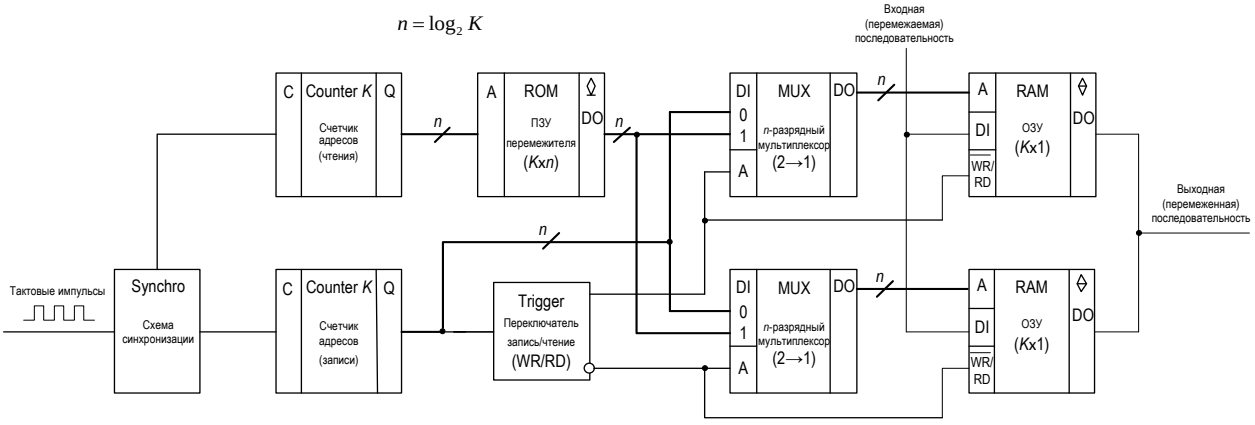


Рис. 5. Структурная схема блочного перемежителя

Однако, при соответствующем межкомпонентном соединении (перемежении контактных соединений) можно избежать хранения перемежителя в ПЗУ. Данный вариант реализации предлагается на рис. 6.

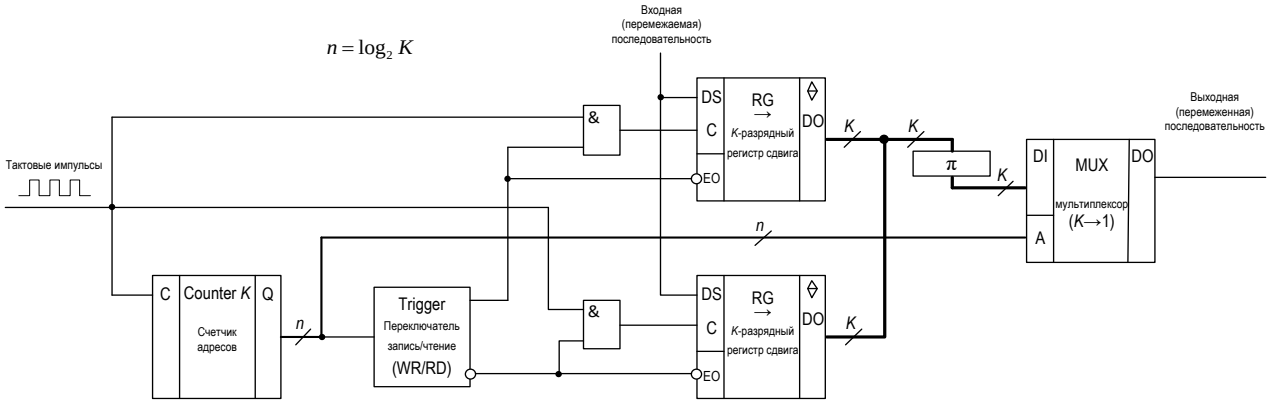


Рис. 6. Структурная схема блочного перемежителя без использования ПЗУ

В представленной схеме блок π определяет соединение между регистрами сдвига и мультиплексором на выходе.

Например, для $K = 20$, $S = 2$, возможен следующий вектор перестановки S-случайного перемежителя:

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

В этом случае содержание блока π имеет вид, изображенный на рис. 7.

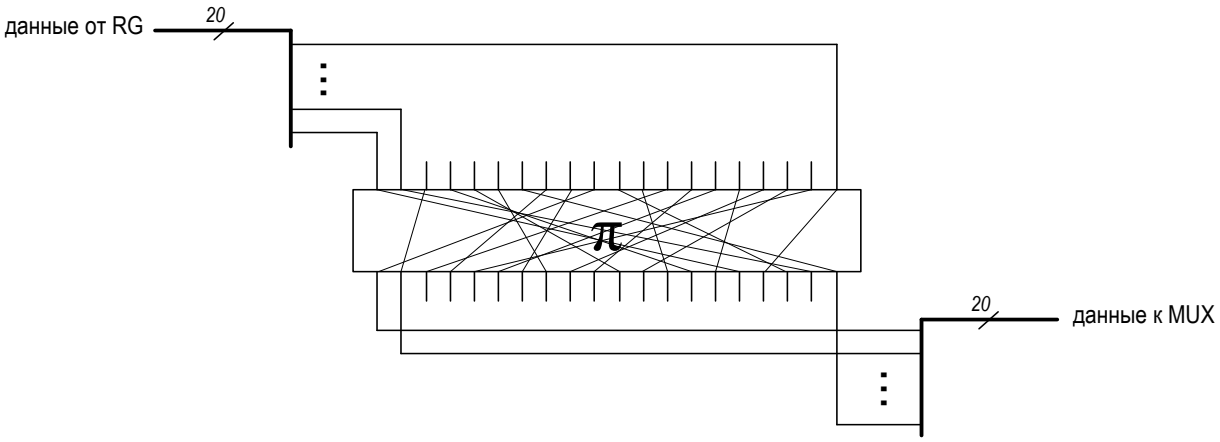


Рис. 7. Пример структуры блока π

Блочное перемежение (детерминированное)

Детерминированные перемежители обладают точными характеристиками упорядочивания перемеженной последовательности, поэтому зависимость позиций элементов в перемеженном блоке от их позиций в исходном блоке описывается уравнениями и, как правило, имеет четко выраженную структуру.

Широко распространенными детерминированными перемежителями являются матричные перемежители. Работу данных перемежителей можно представить как запись и чтение элементов из матриц-накопителей в разных направлениях.

Традиционным матричным перемежителем является перемежитель вида «строка-столбец», когда входные блоки данных длины $K = M \times N$ записываются в строки (столбцы) матрицы размера $M \times N$, а считываются по столбцам (строкам). Значения M и N выбираются в зависимости от используемого помехоустойчивого кода и ожидаемых характеристик канала связи. Данный перемежитель определяется уравнением:

$$\pi(i) = ((i-1) \bmod N)M + \lfloor (i-1)/N \rfloor + 1, \quad i = 1, 2, ..., MN .$$

При спиральном матричном перемежении данные записываются в матрицу построчно, а считываются по диагонали [11]. Для матрицы размера $M \times N$ спиральный матричный перемежитель определяется уравнением

$$\pi(i) = (i-1)(N+1) \bmod (MN) + 1, \quad i = 1, 2, ..., MN .$$

При циклическом матричном перемежении последовательность элементов записывается в столбцы матрицы. Затем каждая i-ая строка циклически сдвигается на $(i\text{-}1)D$ позиций, где D – фиксированная заранее установленная задержка. Далее элементы считываются из матрицы по столбцам. Например, для матрицы размера 4×5 и $D = 2$ получится следующая последовательность на выходе: 1,14,7,20,5,18,11,4,9,2,15,8,13,6,19,12,17,10,3,16.

Перемежители L-типа и их модификации [5] можно представить как запись и считывание данных в системе, включающей несколько матричных перемежителей вида строка-столбец.

Из других детерминированных перемежителей можно отметить перемежители: Уэлча-Костаса (Welch-Costas) [12], Такешита-Костелло (Takeshita-Costello) [13] и QPP (Quadratic Permutation Polynomial) перемежитель [14]. Примечательно, что перемежитель Уэлча-Костаса имеет нормированную дисперсию $d=1$, но рассеивание $s=1$.

Так, QPP перемежитель на периоде K определяется как:

$$\pi(i)=(f_1 i+f_2 i^2)\bmod K,\quad i=1,2,...,K,$$

где коэффициенты f_1 и f_2 удовлетворяют условиям:

1. Коэффициент f_1 и размер блока K являются взаимно простыми числами.

2. Все простые множители K также являются множителями f_2.

Достоинством детерминированных перемежителей является то, что их можно представить уравнениями и реализовать на основе небольшого алгоритма (комбинационной логической схемы) для выполнения перестановок без дополнительной памяти и задержки. Детерминированные перемежители могут иметь большое рассеивание, однако их недостатком является низкое значение дисперсии, при этом зависимость между рассеиванием и дисперсией обратная [4].

На рис. 8 показано графическое представление векторов перестановок **П** для разных видов детерминированных и случайных перемежителей.

Перемежитель строка-столбец имеет большое рассеивание, но низкую дисперсию. Например, при $M=N$, $d=\dfrac{4}{N(N+1)}$ [12]. Случайный перемежитель, наоборот, может иметь наименьшее рассеивание $s=1$ при высокой дисперсии, $d\approx 0{,}81$. В свою очередь *S*-случайный перемежитель сочетает большое рассеивание с высокой дисперсией ($d\approx 0{,}81$). Проведенные в [4] эксперименты показали, что у *S*-случайного перемежителя параметр *S* и дисперсия не зависят друг от друга.

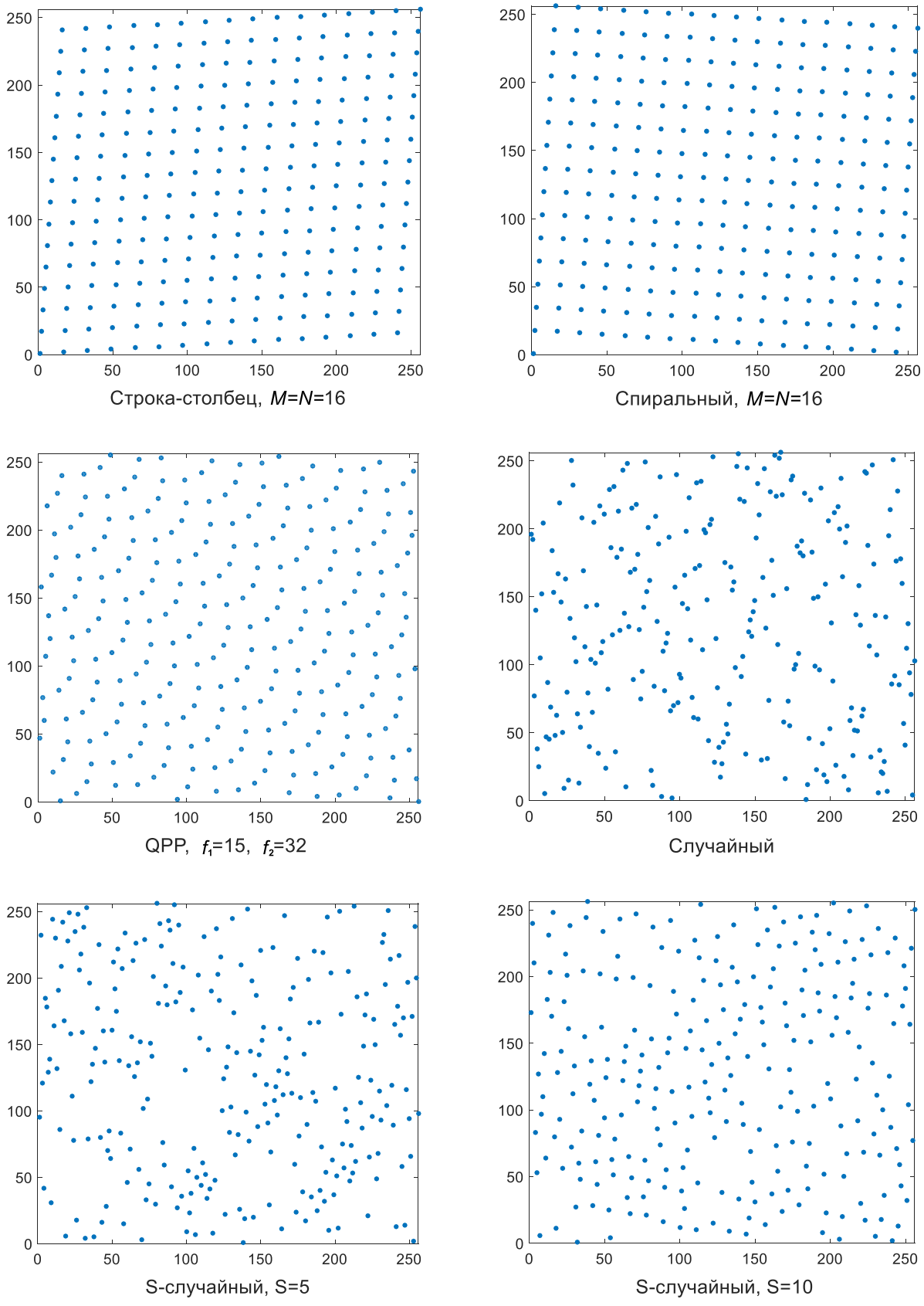
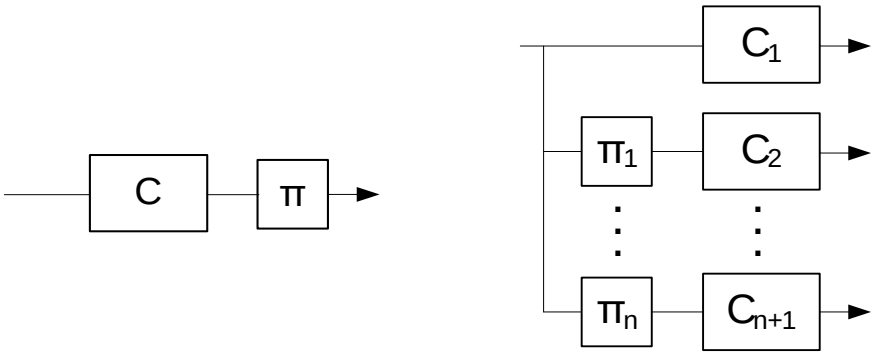


Рис. 8. Графики зависимости $\pi(i)$ от i для видов перемежителей при $K = 256$

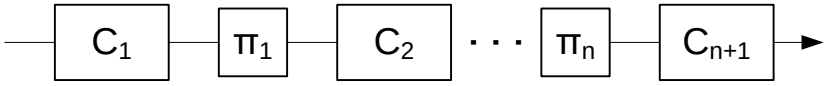
3. Специфика применения перемежения

Перемежение является крайне важной процедурой в канальном кодировании для получения максимальной дистанции между кодовыми словами и может быть как внешним, так и внутренним.

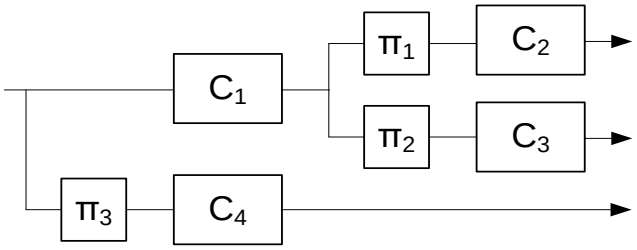
Внешний перемежитель находится на выходе кодера (рис. 9а). Внутренний перемежитель является важной частью системы кодирования, имеющей в своем составе несколько компонентных кодеров (рис. 9б, в, г).



а) б)



в)



г)

Рис. 9. Место перемежителя в структуре кодера:
а) Код-перемежение б) Параллельный составной код
в) Последовательный составной код г) Гибридный код

Стоит заметить, что общие схемы на рис. 9 содержат много каскадов кодирования и перемежения. Однако каждый новый каскад уменьшает скорость составного кода, и схемы, где $n>1$, используются редко.

Расположение перемежителя относительно кодера помехоустойчивого кода зависит от того, с какой целью применяют перемежение.

3.1 Классическое применение перемежения

Большинство блочных и сверточных кодов хорошо противостоят случайным независимым ошибкам, но не справляются с пакетами ошибок. Под пакетом ошибок понимается группирование ошибок в одной ограниченной области кодового слова. Длина пакета ошибок – это количество элементов в данной области. Например, если после передачи двоичной последовательности …10101101101…, принятая последовательность будет …10**01**1**0**0**0**101…, тогда вектор ошибок представляет собой …00110101000…, то есть содержит 4 ошибки в пакете длины 6.

В этом случае одно из возможных решений состоит в рассредоточении пакетов ошибок при помощи внешнего перемежителя (рис. 9а). При таком подходе последовательность на выходе кодера подвергается перемежению до передачи по каналу и восстанавливается перед декодированием.

Распространенными внешними перемежителями являются детерминированные перемежители: традиционные блочные и сверточные перемежители.

Например, перемежитель $M \times N$ «строка-столбец» имеет фактор рассеивания $(s = N, r = M)$. При таком способе перемежения любой пакет ошибок, который содержит не больше N последовательных элементов, переходит на выходе деперемежителя в одиночные ошибки, каждые две из которых оказываются на расстоянии не менее M элементов друг от друга. В то же время, любой пакет ошибок длины bN $(b > 1)$ переходит в пакеты ошибок длины, не большей $\lceil b \rceil$ элементов, каждые два из которых разнесены на расстояние не менее чем $M - \lfloor b \rfloor$ элементов.

Для блочных турбокодов, опционально, в качестве внешнего перемежителя применяют модификацию перемежителя вида «строка-столбец» – спиральный перемежитель.

Таким образом, перемеженный помехоустойчивый код имеет повышенную способность к исправлению пакетов ошибок.

3.2 Применение перемежения в кодах-произведениях и каскадных кодах

Важным этапом в развитии теории кодирования явилось появление последовательного составного кодирования (рис. 9в, $n=1$), когда данные сначала кодируются внешним (относительно канала связи) кодом C_1, а затем внутренним кодом C_2. При этом между внешним и внутренним кодером включается перемежитель, осуществляющий перестановку элементов внешнего кода. Ключевая идея заключается в том, что внутренний и внешний коды могут быть относительно короткими, легко декодируемыми кодами, тогда как составной код – длинный и мощный.

Так, последовательное соединение блочных систематических (n_i, k_i, d_i) кодов C_i, $i=1,2$, через $k_2 \times n_1$ перемежитель (как правило вида строка-столбец) позволяет получить код-произведение $C_p = C_1 \otimes C_2$ с параметрами $(n_1 n_2, k_1 k_2, d_1 d_2)$ [15]. Код C_p может исправлять, дополнительно, все (одиночные) пакеты ошибок длиной до $B = \max\{n_1 t_1, n_2 t_2\}$, где $t_i = \lfloor (d_i - 1)/2 \rfloor$, для $i=1,2$.

Широкое распространение получили каскадные коды [16]. В общем случае внешний код C_1 является недвоичным (n_1, k_1, d_1) кодом Рида-Соломона над $\mathbf{GF}(2^{k_1})$. Кодовые слова кода C_1 записываются в память перемежителя. На выходе перемежителя считываются символы и поступают на вход внутреннего кода C_2. Внутренний код может быть двоичным или недвоичным блочным или сверточным кодом. Если используется двоичный блочный (n_2, k_2, d_2) код C_2, то каскадный код $C = C_1 \times C_2$ является двоичным блочным $(n_1 n_2, k_1 k_2, d_1 d_2)$ кодом. На практике используется несколько видов перемежителей. Наибольшее распространение нашел каскадный код, состоящий из кода Рида-Соломона (255, 223, 33) и сверточного кода скорости $R=1/2$ с длиной кодового ограничения $n_c = 7$, использующий сверточный перемежитель [2].

Таким образом, в каскадных кодах (и кодах-произведениях) внутренний перемежитель преследует две цели:

(1) он требуется в качестве буфера для преобразования кодовых слов внешнего кода в векторы, размерность которых соответствует размерности для внутреннего кода;

(2) что еще более важно, перемежение позволяет разбить пакеты ошибок, в том числе созданные внутренним декодером (в случае ошибочного декодирования).

3.3 Применение перемежения в современных кодах

Современный этап теории кодирования ознаменовался появлением кодовых конструкций, позволяющих работать вблизи пропускной способности канала без памяти. В литературе эти кодовые схемы получили название турбоподобные (turbo-like) коды (ТК). Важными турбоподобными кодами являются: турбокоды [17], блочные турбокоды [18], низкоплотностные помехоустойчивые (Low Density Parity Check – LDPC) коды [19], коды повторения-накопления (Repeat-Accumulate – RA) [20].

На вопрос о том, какие из современных кодов лучше, нет однозначного ответа. В низкоскоростных приложениях (в плохих каналах) при скорости кодов $R \le 1/2$ предпочтение отдается турбокодам, а при скорости кодов $R > 1/2$ (в хороших каналах) турбокоды все больше вытесняются блочными турбокодами и особенно LDPC кодами [5, 21, 22]. Особое место занимает семейство RA кодов, которое сочетает достоинства турбокодов (линейная сложность кодирования) и LDPC кодов (линейная сложность декодирования), поэтому в будущих приложениях может заменить LDPC коды (и возможно даже турбокоды) [5].

Турбоподобное кодирование основано на двух фундаментальных идеях: построение длинных кодов с кодовыми словами, обладающими квазислучайными свойствами и построение относительно несложных декодеров, использующих принцип итеративного декодирования с «мягкими» решениями.

Случайность в процессе кодирования вносит внутренний перемежитель. В силу этого, максимальная эффективность ТК достигается при использовании случайных или псевдослучайных перемежителей большой длины $K = 10^3 \div 10^7$. Так, в оригинальной работе [17], использовался псевдослучайный перемежитель Беру-Главье (Berrou-Glavieux) [23] с периодом $K = 65536$.

Однако, чем больше период перемежителя, тем больше вносимая в систему задержка. В некоторых приложениях, критичных к времени задержки, приходится использовать ТК с внутренним перемежителем относительно небольшой длины. Как показал ряд исследований [5, 12, 24], если период перемежителя в ТК небольшой, то есть $K < 10^3$, то детерминированные внутренние перемежители могут быть эффективнее псевдослучайных.

Таким образом, в зависимости от приложений, в ТК могут использовать длинные (или средней длины) псевдослучайные перемежители и короткие детерминированые перемежители.

Перемежение в турбокодах

Турбокоды [17] строятся на основе нескольких относительно простых, как правило идентичных, рекурсивных систематических сверточных кодеров соединенных параллельно через перемежители (рис. 9б). Для повышения скорости турбокода может применяться перфорация отдельных проверочных элементов.

В процессе декодирования турбокодов выполняются ряд итераций декодирования, на каждой из которых происходит блочное декодирование каждого из компонентных кодов, после чего результат декодирования с учетом перемежения используется декодером другого компонентного кода для получения более точных решений.

Перемежитель является самой ответственной частью турбокода для достижения высокой эффективности при итеративном декодировании. Задача перемежителя – декоррелировать последовательности на входах компонентных кодеров, тем самым уменьшить корреляцию между проверочными элементами кодового слова турбокода. Благодаря этому:

(1) оказывается огромное влияние на минимальное кодовое расстояние турбокода (если на выходе одного компонентного кодера проверочная комбинация с низким весом, то на выходе другого компонентного кодера с большой вероятностью будет проверочная комбинация высокого веса);

(2) уменьшается вероятность ошибки декодирования на каждой итерации («плохая» комбинация ошибок для одного из компонентных декодеров с большой вероятностью будет исправлена другим компонентным декодером, и наоборот).

Перемежение в последовательных турбокодах

Если техника итеративного «турбо» декодирования применяется к последовательным составным кодам (рис. 9в), то такие коды называют последовательными турбокодами. В качестве компонентных кодов здесь могут быть как сверточные, так и блочные коды. В отличие от турбокода, в последовательном турбокоде не удаляются проверочные элементы, соответствующие проверкам проверок.

Роль перемежителя в данных турбокодах следующая:

(1) оказать влияние на минимальное кодовое расстояние турбокода, то есть если как следует перемешать кодовые слова внешнего кода C_1, то минимизируется «плохое» расположение ненулевых элементов (которое могло привести к низкому весу проверочных комбинаций внутреннего кода C_2);

(2) рандомизировать позиции информационных элементов в каждом компонентом кодовом слове, что приведет к уменьшению корреляции между внешней информацией, которой обмениваются компонентные декодеры.

Важным видом последовательных турбокодов являются блочные турбокоды [18]. В качестве компонентных кодов в них часто используют короткие, легко декодируемые коды, такие как коды с контролем четности, коды Хэмминга, коды БЧХ (Боуза-Чоудхури-Хоквингема) и некоторые другие. Компонентными кодами могут быть как идентичные, так и разные коды.

Перспективным видом последовательных турбокодов являются RA коды, где внешний код – это код повторение скорости $1/q$, а внутренний код – это

рекурсивный сверточный код (аккумулятор) скорости 1 с генератором вида $1/(1+D)$. Развитием RA кодов являются прекодированные RA (Accumulate Repeat Accumulate – ARA, Flexible-LDPC – F-LDPC) коды [25, 26] и нерегулярные RA (Irregular Repeat Accumulate – IRA) коды [27].

С другой стороны, структура RA кодов позволяет отнести их к LDPC кодам и, следовательно, к ним можно применить итеративный алгоритм декодирования линейной сложности.

Перемежение в LDPC кодах

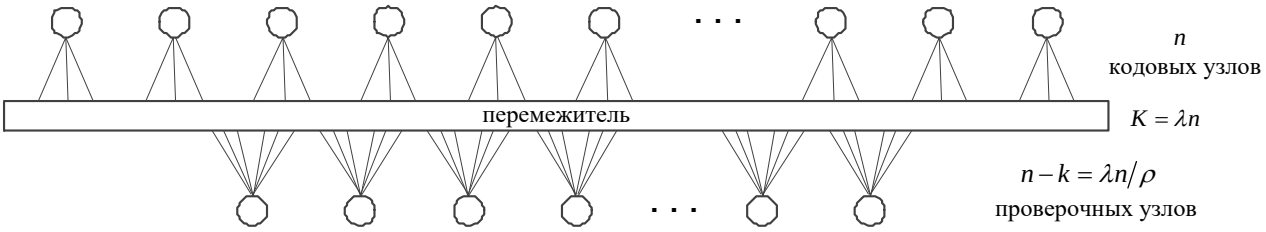
LDPC коды представляют собой линейные блочные (n, k, d) коды, задаваемые с помощью очень сильно разряженной проверочной матрицы $\mathbf{H}_{[n-k,k]}$, то есть для двоичного случая данная матрица имеет низкое отношение числа единиц к числу нулей.

Если LDPC код регулярный, то во всех столбцах и строках проверочной матрицы имеется одинаковое количество единиц – λ и ρ соответственно. В общем случае, когда количество единиц в каждом столбце и/или строке проверочной матрицы не фиксировано, LDPC код является нерегулярным.

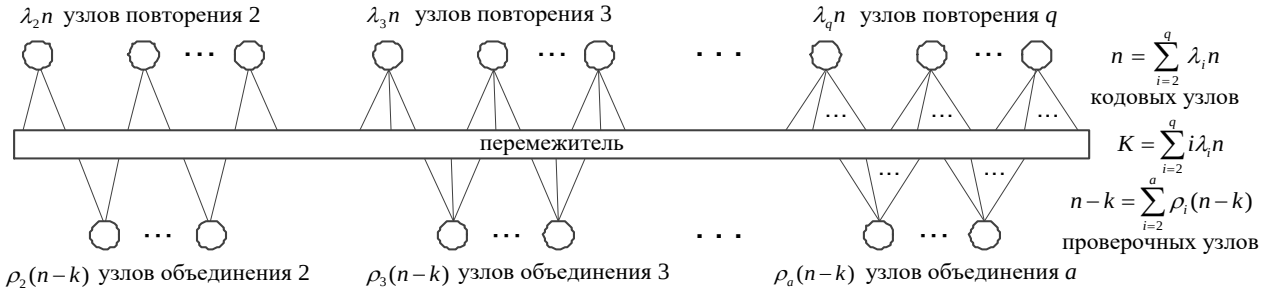
Проверочной матрице LDPC кода ставится в соответствие граф Таннера, в котором для представления строк и столбцов проверочной матрицы используются определенным образом связанные между собой кодовые и проверочные узлы. В основе декодирования LDPC кода лежит итеративный обмен мягкими решениями между кодовыми и проверочными узлами графа.

С другой стороны, LDPC код можно представить в виде введенного в [28] нормального графа (normal graph). Такое представление (рис. 10) позволяет рассматривать данный LDPC код в виде последовательного составного кода, где присутствует внешний код – код-повторение, перемежитель, а также внутренний код – простой код с проверкой на четность [2, 29].

Тогда LDPC коды можно разделить на случайные LDPC, в которых компонентный перемежитель случайный (псевдослучайный) и структурированные LDPC коды, где перемежитель детерминированный.



а) Регулярный $\lambda = 3,\ \rho = 6$



б) Нерегулярный

Рис. 10. LDPC код

Важной характеристикой LDPC кода является величина цикла минимальной длины или обхват соответствующего графа. Многие исследователи считают, что присутствие в графе циклов (особенно коротких, длины 4 и 6) свидетельствует о заложенной в структуру кода избыточности, не приводящей к улучшению его помехоустойчивых свойств. Однако нежелательный эффект влияния коротких циклов ослабляется с увеличением длины кода и значительно снижается для длинных LDPC кодов ($> 10^3$ бит) [29].

В связи с этим в относительно коротких LDPC кодах детерминированные перемежители, устраняющие короткие циклы, по своей эффективности сравнимы с псевдослучайными перемежителями. Представителями данных детерминированных перемежителей являются, например, перемежители *L*-типа и их модификации [5]. В случае длинных LDPC кодов сказывается низкая дисперсия детерминированных перемежителей, поэтому, несмотря на невысокую вероятность появления коротких циклов, применение псевдослучайных перемежителей приводит к превосходным характеристикам [2].

3.4 Применение перемежения для обеспечения защиты информации

ИТС требуются каналы передачи данных, которые обеспечивают высокий уровень защиты от несанкционированного информационного доступа. В таких каналах заинтересован каждый, особенно большие организации, «утечка» информации из которых может привести к потрясениям, компрометации, финансовым потерям и другим серьезным последствиям.

Генерирование и формирование сигналов современных ИТС предполагает реализацию последовательности этапов преобразований сообщений [30], каждый из которых вносит в структуру сигналов определенные закономерности. При этом неопределенность хотя бы одного из этапов исключает возможность информационного доступа к ИТС.

Максимальную неопределенность вносят этапы, следующие после демодуляции сигнала. Прежде всего, этапы криптографической, стеганографической защиты и канального кодирования (рис. 11).

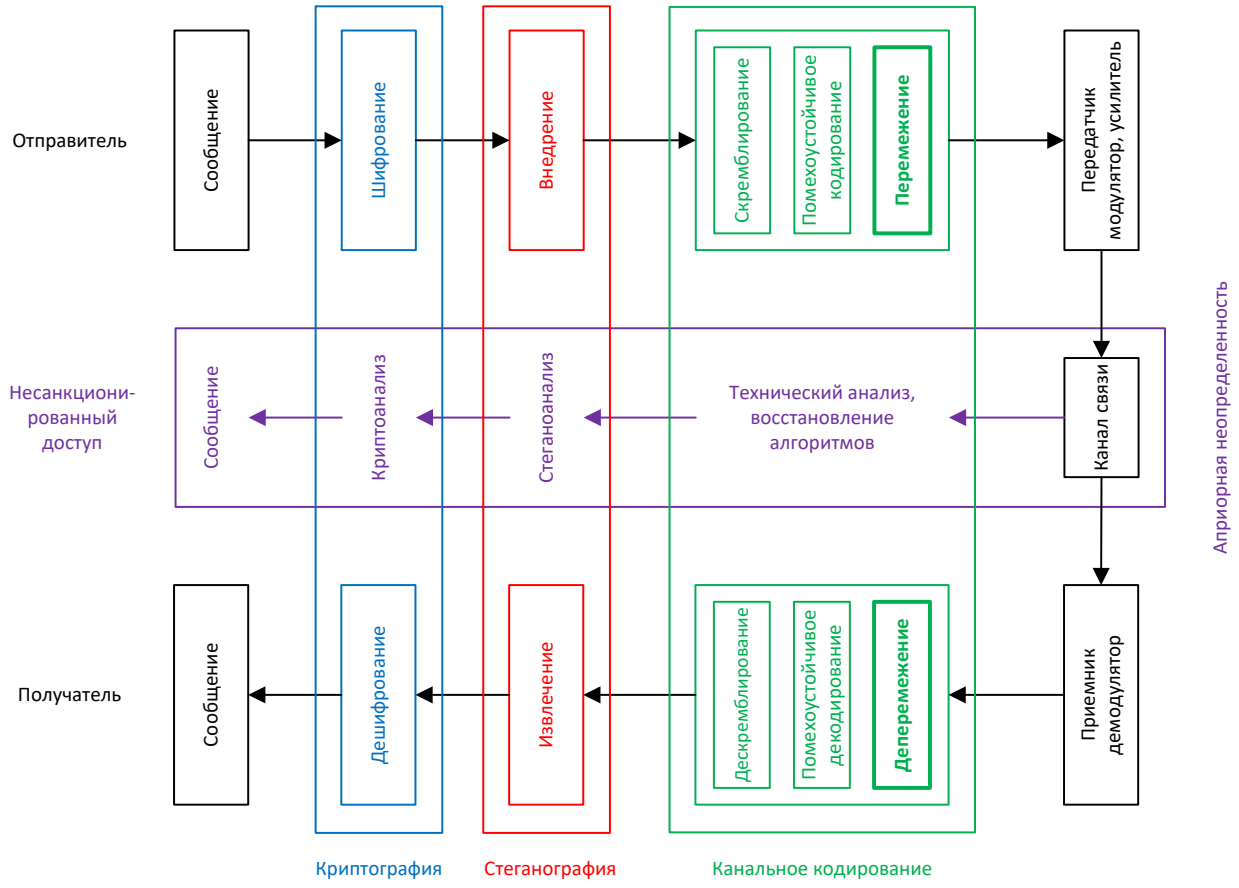


Рис. 11. Защищенная система связи

Большая неопределенность относительно первых двух из перечисленных этапов очевидна и обусловлена их назначением. Значительная неопределенность относительно канального кодирования обусловлена спецификой вскрытия канальных кодов (работа в шумах, наличие перфорации и так далее) и сложностью ее решения, определяемой большим количеством типов канальных кодов и многообразием их характеристик.

В свою очередь современные канальные коды построены на основе скремблирования и составного помехоустойчивого кодирования, включающего перемежение. Следовательно, в случае несанкционированного доступа для снятия неопределенности относительно канального кодирования, необходимо обнаружить и восстановить алгоритмы его генерирования.

Поскольку компонентные помехоустойчивые коды, как правило, простые и в большинстве случаев регламентированы, то задача их выявления может быть решена в том числе и путем прямого перебора возможных вариантов.

Задача анализа скремблеров заключается в определении полиномов генераторов ПСП. Количество данных полиномов ограничено свойством неприводимости, а их таблицы представлены в специальных математических справочниках. Также известно множество публикаций, посвященных вопросу идентификации скремблеров, например [31-33].

Известные работы, посвященные идентификации перемежителей, например [34-39], имеют те или иные недостатки. С другой стороны, перемежители отличаются широким многообразием, так как теоретически может быть задействован почти любой из $K!$ вариантов перестановок (где период перемежителя $K = 10^2 \div 10^7$). Кроме того, задача выявления перемежителей осложняется наличием помех и возможной перфорацией.

В связи с этим, неизвестный перемежитель является ключевым компонентом, который вносит максимальный вклад в априорную неопределенность канального кодирования.

Для лучшего понимания общего количества перемежителей на периоде K можно воспользоваться приближенной формулой Муавра-Стирлинга:

$$K! \approx K^K e^{-K} \sqrt{2\pi K},$$

Например, $10^2! \approx 9{,}33 \cdot 10^{157}$, $10^4! \approx 2{,}846 \cdot 10^{35659}$ – астрономические величины, не поддающиеся обработке современными ЭВМ.

Тем не менее, на практике в качестве перемежителя в современных канальных кодах часто используется *S*-случайный перемежитель. К тому же множество *S*-случайных перемежителей включает в себя множество используемых на практике детерминированных перемежителей. В этой связи, полезно иметь представление о количестве возможных вариантов *S*-случайных перемежителей.

При построении *S*-случайного перемежителя (см. алгоритм на рис. 4) на первой итерации $i = 1$ выбор $\pi(1)$ может быть выполнен любым из K вариантов, включая идентичное соответствие. С учетом условия (2) на второй итерации $i = 2$ для выбора $\pi(2)$ существует как минимум $(K - 1 - 2S)$ вариантов. На итерации $i = 3$ может быть не меньше $(K - 2 - 4S)$ вариантов выбора $\pi(3)$ и так далее:

$$V_1 = K$$

$$V_2 \geq K - 2S - 1 > K - 2S^2 - 1$$

$$V_3 \geq K - 4S - 2 > K - 2S^2 - 2$$

…

$$V_S \geq K - 2(S-1)S - (S-1) > K - 2S^2 - (S-1)$$

$$V_{S+1} \geq K - 2S^2 - S$$

$$V_{S+2} \geq K - 2S^2 - (S+1)$$

…

$$V_{K-2S^2} \geq K - 2S^2 - (K - 2S^2 - 1) = 1$$

…

$$V_K = 1,$$

где V_i – количество вариантов на каждой итерации $i = 1, 2, ..., K$.

В результате, для *S*-случайного перемежителя общее количество вариантов векторов перестановок V можно оценить снизу как:

$$V=\prod_{i=1}^{K}V_i > K\underbrace{\prod_{i=1}^{S-1}(K-2Si-i)}_{>\prod_{i=1}^{S-1}(K-2S^2-i)}\prod_{i=S}^{K-2S^2-1}(K-2S^2-i) > K\prod_{i=1}^{K-2S^2-1}(K-2S^2-1),$$

следовательно, справедливо неравенство:

$$V > (K-2S^2-1)!\,K\,. \qquad (3)$$

Например, пусть имеется $K=10^4$, тогда $S_{\max}=70$ и $V>199!\cdot 10^4$. Посчитав факториал, окончательно получим $V>3{,}943\cdot 10^{376}$. Если для $K=10^4$ взять $S=35$, то в этом случае $V>1{,}087\cdot 10^{26002}$.

Некоторые другие примеры представлены в таблице 1.

Таблица 1. Количество перестановок V в зависимости от K, S

S \ K	256	1024	4096	16384	32768	65536
$S_{\max}=\lceil\sqrt{K/2}\rceil-1$	11	22	45	90	127	181
$V>$	$1{,}59\cdot 10^{12}$	$1{,}3\cdot 10^{76}$	$4{,}9\cdot 10^{59}$	$1{,}98\cdot 10^{340}$	$8{,}5\cdot 10^{1162}$	$4{,}08\cdot 10^{14}$
$\lceil S_{\max}/2\rceil$	6	11	23	45	64	90
$V>$	$3{,}1\cdot 10^{338}$	$6{,}8\cdot 10^{1924}$	$9{,}6\cdot 10^{9262}$	$4{,}43\cdot 10^{53458}$	$6{,}75\cdot 10^{115354}$	$2{,}83\cdot 10^{210118}$

Неравенство (3) и представленные в таблице 1 примеры показывают, что количество вариантов векторов перестановок для *S*-случайного перемежителя, в том числе в случае $S=S_{\max}$, является астрономической величиной, не поддающейся обработке современными ЭВМ при прямом переборе.

Таким образом, длинный псевдослучайный перемежитель является компонентом высокой априорной неопределенности, а его использование повышает уровень защиты ИТС и ограничивает возможности при несанкционированном доступе.

Заключение

Перемежение является одной из передовых техник канального кодирования и активно используется на линиях связи. Перемежитель

представляет собой перестановку, обладающую рядом свойств и структурой. Высокая надежность и скрытность ИТС во многом обеспечивается за счет правильно выбранного перемежителя.

На сегодняшний день известно множество видов перемежения, различающихся по обеспечению выигрыша от кодирования, сложности реализации и многим другим параметрам. В зависимости от ожидаемого эффекта в канальных кодах применяют блочные перемежители (детерминированные и случайные) и сверточные перемежители.

По результатам настоящего исследования можно сделать обоснованный вывод, что отличными характеристиками обладает *S*-случайный перемежитель и его модификации. Данный перемежитель сочетает большое рассеивание с высокой дисперсией, тем самым вносит значительный вклад в обеспечение энергетического выигрыша и защиту информации на этапе канального кодирования.

Кроме того, согласно предложенной в работе структурной схеме, *S*-случайный перемежитель может быть реализован без использования ПЗУ и лишней задержки.

Литература

1. Lin S., Costello D.J., Jr. *Error control coding*, 2nd ed. Pearson Prentice Hall, New Jersey, 2004, 1260 p.
2. Costello D.J., Jr, Forney G.D., Jr. Channel coding: The road to channel capacity. *Proc. IEEE,* 2007, Vol. 95. No. 6, pp. 1150-1177.
3. Andrews K., Heegard C., Kozen D. *A theory of interleavers. Technical report TR97-1634,* Department of Computer Science, Cornell University, 1997.
4. Dolloff J., Kenz Z., Rische J., Rogers D.A., Smih L., Mosteig E. *Algebraic interleavers in turbo codes.* Applied Mathematical Sciences Summer Institute, Department of Mathematics & Statistics, California State Polytechnic University, Pomona 3801 W. Temple Ave. Pomona, California, 2006, 76 p.
5. Johnson S.J. *Iterative error correction. Turbo, low-density parity-check and*

repeat accumulate codes. Cambridge University Press, New York, 2010, 335 p.

6. Helical interleaver [online].
   Available at: https://www.mathworks.com/help/comm/ref/helicalinterleaver.html
7. Bendetto S., Montorsi G. Unveiling turbo codes: Some results on parallel concatenated coding schemes. *IEEE Trans. Inform. Theory*, 1996, Vol. IT-42, pp. 409-428.
8. Dolinar S., Divsalar D. Weight distributions for turbo codes using random and nonrandom permutations. 1995. *TDA Progress Report* 42-122, Jet Propulsion Laboratory, Pasadena, California, pp. 56-65.
9. Popovski P., Kocarev L., Risreski A. Design of flexible-length S-random interleaver for turbo codes, *IEEE Commun. Letter*, 2004, Vol. 8, No. 7, pp. 461-463.
10. Dinoi L., Bendetto S. Design of fast prunable S-random interleavers. *IEEE Trans. Wireless Commun*., 2005, Vol. 4, No. 5, pp. 1-9.
11. Matrix Helical Scan Interleaver [online]. Available at:
    https://www.mathworks.com/help/comm/ref/matrixhelicalscaninterleaver.html
12. Heegard C., Wicker S.B. *Turbo Coding*. Kluwer Academic Publishers, Boston, 1999, pp. 54-55.
13. Takeshita O.Y., Costello D.J., Jr. New classes of algebraic interleavers for turbo-codes. *Proc. 1998 IEEE ISIT*, Boston, 1998, pp. 419.
14. Sun J., Takeshita O.Y. Interleavers for turbo codes using permutation polynomial over integer rings. *IEEE Trans. Inform. Theory*, 2005, Vol. 51, No 1, pp. 101-119.
15. Elias P. Error-free coding. *IRE Trans. Inform. Theory*, 1954, Vol. IT-4, pp. 29-37.
16. Forney G.D., Jr. *Concatenated Codes*. MIT Press, Cambridge, 1966.
17. Berrou C., Glavieux A., Thitimasjshima P. Near Shannon limit error-correcting coding and decoding: Turbo codes. *Proc. IEEE Int. Conf. on Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.
18. Pyndiah R.M. Near-optimum decoding of product codes: block turbo codes. *IEEE Transactions on Communications*, 1998, Vol. 46, No 8, pp. 1003-1010.
19. Gallager R.G. Low-density parity-check codes. *IRE Trans. Inform. Theory*, 1962,

Vol. IT-8, pp. 21-28.

20. Divsalar D., Jin H., McEliece R.G. Coding theorems for turbo-like codes, *Proc. 1998 Allerton Conf*. Allerton, IL, September 1998, pp. 201–210.
21. Кудряшов Б.Д. Основы теории кодирования. – СПб.: БХВ-Петербург, 2016. – 400 с.
22. Голиков А.М. Модуляция, кодирование и моделирование в телекоммуникационных системах. Теория и практика. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2016. – 516 с.
23. Berrou C., Glavieux A. Near optimum error correcting coding and decoding, *IEEE Trans. Comm*., 1996, Vol. 44, No. 10, pp. 1261-1271.
24. Vucetic B., Yuan J. Turbo codes: principles and applications, Kluwer Academic, Norwell, MA, 2000, 312 p.
25. Abbasfar A., Divsalar D., Kung Y. Accumulate-repeat accumulate codes, *Proc. IEEE Global Telecommunications Conference (GLOBECOM 2004)*, Dallas, TX, December 2004, pp. 509–513.
26. Halford T.R., Bayram M., Kose C., Chugg K.M., Polydoros A. The F-LDPC family: High-performance flexible modern codes for flexible radio, *in Proc. ISSSTA 2008*, Bologna, Italy, September 2008, pp. 376–380.
27. Jin H., Khandekar A., McEliece R. Irregular repeat-accumulate codes, *In Proc. 2nd Int. Symp. Turbo codes and related topics*, Brest, France, September 2000, pp. 1–8.
28. Forney G.D., Jr., Codes on graphs: Normal realizations, *IEEE Trans. Inform. Theory*, 2001, Vol. IT-13, pp. 520-548.
29. Moreira J.C., Farrell P.G. Essentials of error control coding, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, England, 2006, 361 p.
30. Скляр Б. Цифровая связь. Теоретические основы и практическое применение, 2-е изд. / Б. Скляр ; Пер. с англ. Е.Г. Грозы [и др.]; под редакцией А.В. Назаренко – М. : Вильямс, 2003. – 1104 с.
31. Canteaut A., Filiol E. Ciphertext only reconstruction of stream ciphers based on combination generators, Berlin, Springer, 2001, 16 p.

32. Cluzeau M. Reconstruction of a linear scrambler, *IEEE Trans. on Computers*, 2007, Vol. 56, No. 8, pp. 1283-1293.
33. Кривоногов А.С., Кривоногова Е.О. Алгоритм идентификации вида скремблирования бинарных данных // Известия Вузов России. Радиоэлектроника. 2017. № 6. С. 10-14.
34. Sicot G., Houcke S. Blind detection of interleaver parameters. *Proc. IEEE ICASSP*, March 2005, pp. 829-832.
35. Cluzeau M, Finiasz M, Tillich J, Methods for the reconstruction of Parallel Turbo Codes. in *Proc. IEEE Int. Symp. Information Theory (ISIT '10)*, Austin, Texas, USA, June 2010, pp. 2008–2012.
36. Tillich J., Tixier A., Sendrier N., Recovering the interleaver of an unknown Turbo-Code, in *Proc. IEEE Int. Symp. Information Theory (ISIT '14)*, Honolulu, Hawaii, USA, 2014, pp. 2784–2788.
37. Баринов А.Ю. Методы анализа турбоподобных кодов с учетом идентификации их компонентных перемежителей // Наукоемкие технологии. 2016. № 12. С. 4–11.
38. Баринов А.Ю., Асеев А.Ю. Модифицированная математическая модель системы генерирования перемеженной дискретной последовательности турбоподобного кода // Вестник Череповецкого государственного университета. 2017. №6 (81). С. 9–18. DOI 10.23859/1994-0637-2017-6-81-1
39. Баринов А.Ю. Идентификация перемежителей турбокодов на основе их полиномиального и матричного представления // Информация и Космос. 2018. № 2. С. 61–66.

Для цитирования:
А. Ю. Баринов. Перемежение в канальном кодировании: свойства, структура, специфика применения. Журнал радиоэлектроники [электронный журнал]. 2019. №1. Режим доступа: http://jre.cplire.ru/jre/jan19/13/text.pdf
DOI 10.30898/1684-1719.2019.1.13