ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное
образовательное бюджетное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»

О. С. КОГНОВИЦКИЙ
В. М. ОХОРЗИН
И. А. НЕБАЕВ

ТЕОРИЯ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

Часть 2

СВЕРТОЧНЫЕ КОДЫ. ТУРБОКОДЫ

УЧЕБНОЕ ПОСОБИЕ

СПб ГУТ)))

САНКТ-ПЕТЕРБУРГ
2015

УДК 621.391(075.8)
ББК 32.811.4я73
К57

Рецензенты:
доктор технических наук, профессор, начальник кафедры специальных информационных технологий *А. И. Примакин*,

кандидат технических наук, доцент Военной академии связи *Б. И. Бабич*

Утверждено редакционно-издательским советом СПбГУТ в качестве учебного пособия

Когновицкий, О. С.

К57 Теория помехоустойчивого кодирования. Часть 2. Сверточные коды. Турбокоды / О. С. Когновицкий, В. М. Охорзин, И. А. Небаев ; СПбГУТ. – СПб., 2015 – 64 с.

Излагаются основы принципов построения сверточных, каскадных и турбокодов, широко используемых в современных системах телекоммуникаций и изучаемых в учебных дисциплинах СПбГУТ: «Математическая теория помехоустойчивого кодирования», «Основы теории передачи дискретных сообщений», «Теория кодирования» и т. д.

Предназначено для студентов, обучающихся по направлению 11.03.02 «Инфокоммуникационные технологии и системы связи». Изложенные материалы могут быть использованы для подготовки студентов, обучающихся по всем техническим направлениям подготовки.

УДК 621.391(075.8)
ББК 32.811.4я73

СОДЕРЖАНИЕ

1. СВЕРТОЧНЫЕ КОДЫ

1.1. Основы построения сверточных кодов

Сверточные коды относятся к классу непрерывных кодов. В основе построения этих кодов лежит рекуррентный принцип кодирования и декодирования, когда на вход кодера поступает непрерывная последовательность информационных символов источника, а с выхода кодера снимается также непрерывная последовательность символов, являющихся функцией входных символов и структуры кодера. На вход декодера поступает непрерывная последовательность символов из канала связи (возможно, искаженная ошибками), а на выходе восстанавливается (возможно, с ошибками, но, как правило, меньшими чем канальные) последовательность информационных символов. Сверточные коды являются наиболее распространенным классом непрерывных кодов. Операция формирования выходной последовательности по заданной входной последовательности является для большинства сверточных кодов линейной. Однако на практике, например в модемных технологиях, применяются также нелинейные сверточные коды.

Первенство в создании непрерывных кодов принадлежит нашим соотечественникам – Л. М. Финку и В. И. Шляпоберскому (1955) [10, 11]. Они предложили и реализовали в отечественной аппаратуре передачи данных систематические непрерывные коды, названные ими «цепными кодами» в силу рекуррентного способа формирования избыточных элементов.

Дальнейшее развитие рекуррентных кодов связано с именами зарубежных ученых Д. В. Хагельбергера [1, 2], Дж. Возенкрафта [3, 4], А. Витерби. [5, 6], а также отечественных – А. Э. Нейфаха и К. Ш. Зигангирова.

По мере развития и усложнения принципа формирования рекуррентной последовательности для описания свойств создаваемых кодов и способа формирования последовательности избыточных элементов стали использовать понятия из теории инвариантных линейных систем. С точки зрения этой теории процедура кодирования реализуется с помощью операции линейной дискретной свертки последовательностей импульсных характеристик, подобно формированию кодовых комбинаций групповых кодов умножением кодируемой последовательности на порождающую матрицу кода. Такая процедура и дала название непрерывным рекуррентным кодам – сверточные.

Важнейшими достоинствами сверточных кодов являются следующие [12].

1. Сверточные коды позволяют производить кодирование и декодирование информационных потоков непрерывно во времени.

2. Сверточные коды не нуждаются в блоковом фазировании.

3. Применение сверточных кодов позволяет обеспечить высокую надежность передаваемой информации.

Принцип построения двоичных сверточных кодов рассмотрим на примере простого цепного кода [11].

Пусть кодированию подлежит последовательность информационных элементов $\{a\} = (a_1, a_2, a_3, ..., a_i, a_{i+1}, ...)$. Процедура кодирования цепным кодом заключается в том, что каждый проверочный элемент $b_{i,k}$ формируется как сумма по модулю двух информационных элементов, отстоящих друг от друга на $t = k - i$ шагов, где t – шаг сложения:

$$a_i \oplus a_k = b_{i,k};$$
$$a_{i+1} \oplus a_{k+1} = b_{i+1,k+1};$$
$$.....$$
$$a_k \oplus a_{k+t} = b_{k,k+t};$$
$$a_{k+1} \oplus a_{k+t+1} = b_{k+1,k+t+1}.$$

Как видно, каждый информационный элемент a_k участвует в вычислении двух проверочных элементов $b_{i,k} = b_{k-t,k}$ и $b_{k,k+t}$. В канал связи передается непрерывная последовательность, в которой за каждым информационным элементом следует проверочный. Избыточность цепного кода при этом будет равна 1/2.

Принцип декодирования цепного кода состоит в следующем.

1. Из принятых информационных элементов $a_i^{'}$ по тому же правилу, что и на передаче, формируются контрольные элементы c, а именно: $a_{k-t}^{'} \oplus a_k^{'} = c_{k-t,t}$.

2. Полученные контрольные элементы сравниваются с соответствующими проверочными элементами $b^{'}$, принятыми из канала связи, т. е. $c_{k-t,t}$ с $b_{k-t,t}^{'}$, $c_{k-t+1,t+1}$ с $b_{k-t+1,t+1}^{'}$ и т. д.

3. При отсутствии ошибок в двоичном канале контрольные элементы c и принятые проверочные элементы $b^{'}$ будут совпадать. При наличии обнаруживаемых ошибок сравниваемые элементы полностью совпадать не будут, в то же время в результате такого сравнения возможно исправление некоторых ошибок. Так, например при ошибке в информационном элементе $a_k^{'} = a_k \oplus 1$ контрольные элементы $c_{k-t,t}$ и $c_{k,k+t}$ не будут совпадать с проверочными элементами $b_{k-t,t}^{'}$ и $b_{k,k+t}^{'}$ соответственно. Такое несовпадение свидетельствует о том, что возникла ошибка в информационном элементе, входящем в обе проверки, т. е. в элементе $a_k^{'}$. Этот элемент может быть исправлен путем его инвертирования.

4. При ошибочном приеме проверочного элемента, например $b'_{k-t,t}$, и при безошибочном приеме информационных элементов $a'_{k-t} = a_{k-t}$ и $a'_k = a_k$ несовпадение будет только в одном сравнении проверочного элемента $c_{k-t,t}$ и контрольного элемента $b'_{k-t,t}$, а другие проверочные и контрольные элементы, отстоящие на t левее и правее от $b'_{k-t,t}$, будут совпадать. Тем самым, ошибка будет локализована в проверочном элементе $b_{k-t,k}$, что позволяет произвести ее исправление (в случае необходимости).

Таким образом, для исправления ошибочно принятого информационного элемента, например a'_k, необходимо, чтобы безошибочными были информационные элементы a'_{k-t} и a'_{k+t} и проверочные элементы $b'_{k-t,t}$ и $b'_{k,k+t}$ соответственно.

Кроме того, для цепного кода будет справедливым следующее его свойство: при шаге сложения t код может исправить любую пачку ошибок длиной $b \le 2t$ при условии, что каждый проверочный элемент будет передаваться в канал с задержкой $(3t + 1)$ тактовых интервалов, и что между последней ошибкой данного пакета и первый ошибкой следующего пакета ошибок будет не менее $(3b + 1)$ неискаженных элементов [11].

В обобщенном виде двоичный линейный сверточный кодер состоит, как правило, из k параллельных регистров сдвига, включающих v ячеек памяти каждый, блока параллельных сумматоров по mod 2, входы каждого из которых связаны с выходами определенных ячеек памяти регистров в соответствии с коэффициентами связи $hi, j = (0,1)$, где 1 означает наличие связи, а 0 – отсутствие.

Работает кодер следующим образом: на каждом такте работы кодера в k регистров сдвига параллельно поступает очередной блок из k двоичных информационных символов источника, и одновременно регистры сдвига освобождаются от k символов, содержащихся в их крайних правых ячейках памяти. В течение этого же такта формируется n выходных символов, которые поочередно считываются при помощи коммутатора и подаются в канал связи в виде комбинации (c_1, c_2, c_3, …, c_n). Следовательно, если ru – скорость последовательного поступления символов в кодер, то при отсутствии задержек во времени скорость последовательной передачи символов в канал связи должна быть равна $r_k = \frac{n}{k} r_u$. Таким образом, на k двоичных информационных символов на входе кодера формируются n символов на выходе кодера, откуда видно, что отношение $R = \frac{k}{n}$ определяет относительную скорость такого сверточного кода.

Ниже будет рассмотрен пример сверточного кода со скоростью $R = \frac{2}{3}$.

Для упрощения понимания свойств и принципов работы сверточных кодов рассмотрим наиболее простой вариант сверточных кодов с одним регистром сдвига и со скоростью $R = \frac{1}{n}$, т. е. на один информационный символ на входе с выхода кодера считываются n символов. При этом длину регистра сдвига (число ячеек K) называют кодовым ограничением. На рис. 1.1 представлен пример двоичного сверточного кода со скоростью $R = 1/2$ и длиной кодового ограничения $K = 3$. Выходные символы формируются на выходах верхнего и нижнего сумматоров по mod2. Связи ячеек памяти с сумматорами задаются *порождающими* полиномами кода.

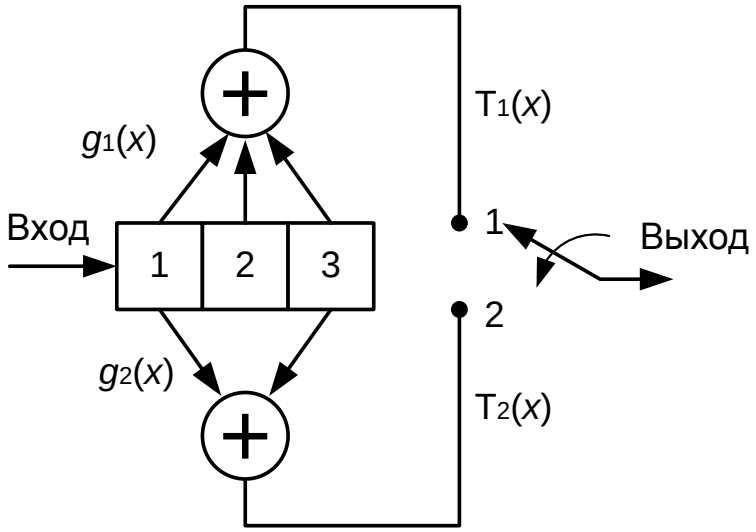

Рис. 1.1. Сверточный кодер со скоростью $R = 1/2$ и кодовым ограничением $K = 3$.

Для верхнего и нижнего сумматоров эти полиномы имеют вид $g_1(x) = 1 + x + x^2$ и $g_2(x) = 1 + x^2$ соответственно. Полиномы часто записывают в цифровом сокращенном виде. Так, для кодера на рис. 1.1 каждые три отвода (три двоичных коэффициента полиномов $g_1(x)$ и $g_2(x)$) обозначают восьмеричным цифровым кодом как (7, 5). Полиномы $g_1(x)$ и $g_2(x)$ названы порождающими потому, что они формируют выходную последовательность на выходе кодера по заданной на входе информационной последовательности. Если информационную последовательность представить в виде многочлена $s(x)=s_0 + s_1x + s_2x^2 + \ldots$, где s_i – информационные символы (для двоичных кодов 0 и 1), то выходные последовательности будут получены путем умножения $s(x)$ на порождающие полиномы $g_1(x)$ и $g_2(x)$, т. е.:

$$T_1(x) = s(x)\cdot g_1(x);\ \ T_2(x) = s(x)\cdot g_2(x). \qquad (1.1)$$

Представленный на рис. 1.1 сверточный код по своей классификации является *несистематическим*, так как в выходной последовательности в явном виде не присутствуют информационные символы последовательности $s(x)$. Для *систематического* сверточного кода один из полиномов связи (либо $g_1(x)$, либо $g_2(x)$) должен быть одночленом, равным x^i, где $i = 0$, 1 или 2.

Другим представлением сверточного кода является порождающая матрица G, которая строится по полиномам $g_1(x)$ и $g_2(x)$ и является полубесконечной. В двоичном преставлении каждая строка матрицы G является реакцией кодера на поданную на вход 1 при условии, что все ячейки регистров сдвига предварительно были установлены в 0. Такую реакцию часто называют откликом кодера. Так, если на вход рассматриваемого сверточного кода (рис. 1.1) подать 1, за которой последуют нули, т. е. $s(x) = 1$, то на выходе кодера появится последовательность 11 10 11 00 00 …, которая и будет реакцией или откликом кодера. При этом первой строкой порождающей матрицы G и будет отклик на 1. Каждая последующая строка будет тем же откликом, но сдвинутым относительно предыдущей на n символов, в данном примере на $n = 2$. Таким образом, порождающая матрица рассматриваемого несистематического сверточного кода будет иметь вид:

$$G = \begin{bmatrix} 11 & 10 & 11 & 00 & 00 & 00 & \dots & & \\ 00 & 11 & 10 & 11 & 00 & 00 & 00 & \dots & \\ 00 & 00 & 11 & 10 & 11 & 00 & 00 & 00 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}. \qquad (1.2)$$

Легко проверить, что выходная последовательность Y будет получена как произведение произвольной непрерывной входной последовательности в виде вектор-строки X на порождающую матрицу G, т. е. $Y = X \cdot G$. Поэтому выходная последовательность кодера может быть представлена как цифровая свертка входной информационной последовательности и отклика кодера (отсюда название кодов – сверточные).

Следовательно, выходная последовательность Y получается как сумма по модулю 2 соответствующих строк порождающей матрицы G. Например, в рассматриваемом случае выходная последовательность, соответствующая входной вида $X = 1110000\ldots$, будет равна сумме по модулю 2 строк 1, 2 и 3 матрицы G, т. е. $Y = 11\ 01\ 10\ 01\ 11\ 00\ 00\ \ldots$

Рассмотрим теперь другие наглядные способы представления сверточного кода: кодовое дерево, решетчатая диаграмма и диаграмма состояний, отображающих связь между входными и выходными последовательностями.

Например, *кодовое дерево* для линейного сверточного кодера, показанного на рис. 1.1, имеет вид, представленный на рис. 1.2.

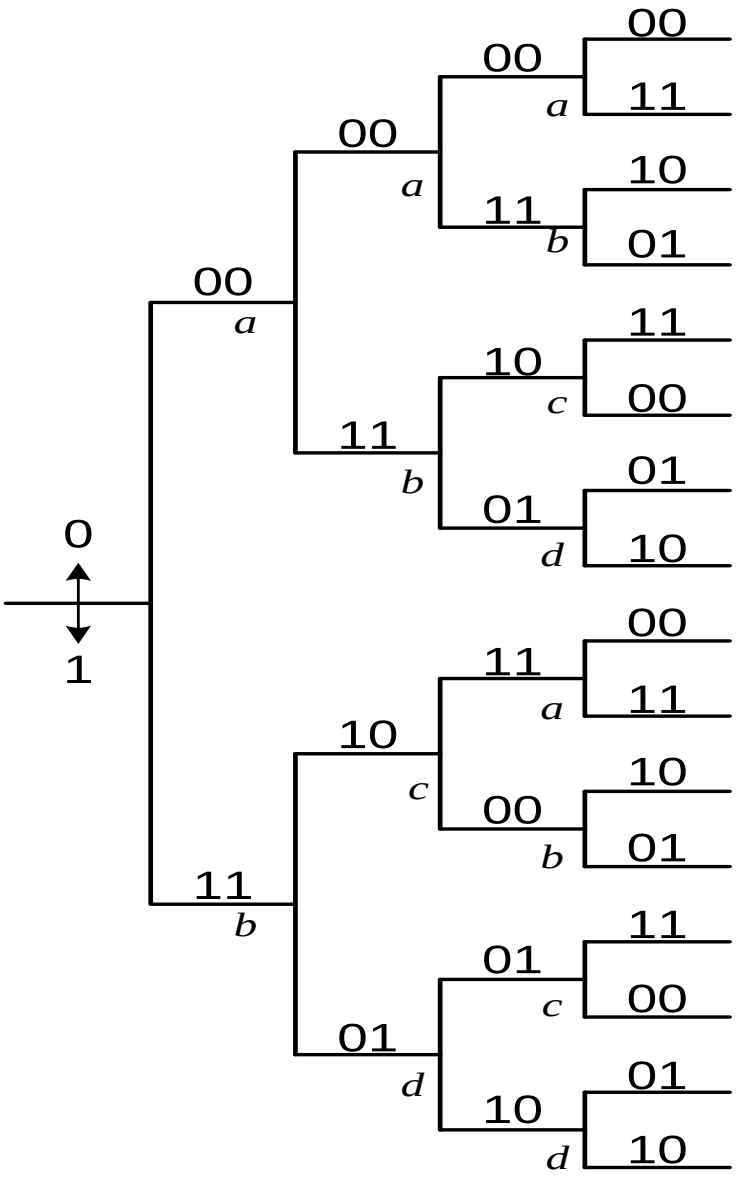

Рис. 1.2. Древовидная диаграмма сверточного кода
для кодера на рис. 1.1 со скоростью кода $R = 1/2$
и длиной кодового ограничения $K = 3$

Кодовое дерево построено в предположении, что кодер первоначально находится в нулевом состоянии (во всех ячейках нули). Правило построения кодового дерева следующее: если на вход кодера поступает 0, то соответствующая ветвь дерева строится вверх, а если 1 – то вниз, при этом рядом с ветвями ставится пара символов, считываемых с выходов 1 и 2 кодера. Диаграмма показывает, что, если первый входной символ 0, то на выходе появится пара выходных символов 00, а если первый входной символ 1, то выходная последовательность будет 11. Предположим, что вторым входным

символом будет 0. Тогда первый единичный символ со следующим тактом работы кодера продвинется по регистру с первой (левой) ячейки во вторую, а второй входной символ 0 запишется в первую ячейку регистра. Таким образом, в ячейках регистра установится состояние 010 (левый символ в первой ячейке, самый правый символ (0) – в третьей). При этом, в соответствии с порождающими полиномами $g_1(x)$ и $g_2(x)$, на первом выходе кодера появится 1, а на втором выходе – 0, т. е. пара символов 10. Заметим, что двухбитовая последовательность на выходе кодера для каждого входного бита определяется значением самого входного символа и состоянием первых двух ячеек регистра сдвига, которое может быть обозначено как a = 00, b = 10, c = 01, d = 11. При записи состояния первый (левый) двоичный символ соответствует состоянию первой (левой) ячейки регистра сдвига на рис. 1.1, а второй символ – состоянию второй ячейки регистра сдвига. Третий (правый) символ регистра не влияет на формирование выходной пары символов, так как он покидает регистр во время записи входного символа в первую ячейку регистра.

На кодовом дереве выходные пары двухсимвольных комбинаций записаны над путями, а состояния кодера (первых двух ячеек регистра) обозначены буквами *a, b, c, d.* Некоторые авторы [7] эти состояния обозначили цифрами 0, 1, 2, 3. Таким образом, для рассматриваемого примера сверточного кода (рис. 1.1) входные информационные последовательности отображаются пошаговым продвижением вдоль пути (вверх – 0, вниз – 1), а выходные последовательности отображаются двухбитовыми символами вдоль ветвей дерева.

Здесь уместно подробнее остановиться на понятии *кодового ограничения* для сверточных кодов. В известных фундаментальных монографиях по помехоустойчивому кодированию [7, 8] применительно к сверточным кодам со скоростью $R = 1/n$ кодовое ограничение определяется как число информационных символов в тактовый момент i, которые влияют на формирование n выходных символов в этот же тактовый момент i. Как было показано выше, для сверточного кода с $R = 1/n$ и длиной регистра сдвига из K ячеек памяти n выходных символов в i-й тактовый момент определяются текущим входным символом и предшествующим состоянием первых (младших) $(K - 1)$ ячеек регистра, т. е. предыдущими $(K - 1)$ информационными элементами, поступившими ранее на вход кодера. Таким образом, кодовое ограничение будет равно $(K - 1) + 1 = K$, длине регистра сдвига.

Отдельные авторы определяют длину кодового ограничения по-другому, а именно [7] как логарифм по основанию 2 от числа состояний кодера, которое при длине регистра K и скорости $R = 1/n$ будет равно 2^{K-1}. При таком определении длина кодового ограничения будет на 1 меньше числа ячеек регистра сдвига, т. е. $(K - 1)$.

Однако большинство авторов дают предыдущее определение кодового ограничения как длину регистра сдвига сверточного кода со скоростью $R = 1/n$.

Таким образом, множество путей на кодовом дереве представляет собой множество разрешенных кодовых последовательностей, которые однозначно определяются входными последовательностями. Просмотрев структуру кодового дерева на рис. 1.1, можно заметить, что число разрешенных кодовых последовательностей растет экспоненциально в зависимости от длины входной информационной последовательности. Например, при длинах входной последовательности 1, 2, 3, …, длины выходных последовательностей соответственно будут равны 2, 4, 6, …, а количество разрешенных последовательностей 2, 4, 8, … соответственно. Таким образом, для простого сверточного кода со скоростью $R = 1/2$ при длине входной последовательности i длина выходной последовательности в двоичных разрядах будет равна $2i$, а общее количество разрешенных последовательностей – 2^i. Из этого следует, что общая доля разрешенных последовательностей на i-м шаге экспоненциально падает в зависимости от длины входной последовательности i как $2^i/2^{2i} = 2^{-i}$, а доля запрещенных последовательностей соответственно экспоненциально растет как $(1 - 2^{-i})$.

Аналогично для двоичного сверточного кода со скоростью $R = 1/n$ при длине входной последовательности i длина выходной последовательности в двоичных разрядах будет равна ni с общим числом разрешенных последовательностей 2^i из 2^{ni} возможных. Таким образом, доля разрешенных последовательностей на i-м шаге экспоненциально падает в зависимости от длины входной последовательности i как $2^i/2^{ni} = 2^{-(n-1)i}$, а доля запрещенных − соответственно экспоненциально растет как $(1-2^{-(n-1)i})$.

Кодовое дерево можно считать удобной и наглядной формой, позволяющей легко представить простую процедуру декодирования. Суть этой процедуры заключается в выборе пути в кодовом дереве, который ближе всего по расстоянию Хемминга отстоит от принятой последовательности. Такое декодирование можно назвать декодированием по принципу максимального правдоподобия. Но при этом остается неясной реализация процедуры сравнения принятой последовательности с возможными путями на дереве. В частности, какой длины необходимо выбирать принятую последовательность для сравнения. Ведь, как было показано ранее, с ростом длины последовательности число путей растет по экспоненциальному закону, и поэтому задача оценки по максимуму правдоподобия будет сложно реализуемой. Однако эта задача достаточно просто решается, если кодовое дерево заменить решетчатой структурой [6, 7].

Рассмотрим построение *решетчатой структуры*, например для сверточного кода на рис. 1.1. Из кодового дерева (рис. 1.2) видно, что после первых трех шагов (трех входных символов), соответствующих длине кодового ограничения $K = 3$, возможные состояния кодера повторяются. Из двух узлов, обозначенных буквой *a*, формируются одинаковые последовательности. То же мы наблюдаем и для другой пары узлов, помеченных одной и той же буквой. Поэтому информационное содержание кодового дерева не изменится, если объединить в нем повторяющиеся узлы и исходящие из них пути. В этом случае вместо разрастающегося вширь кодового дерева мы получим другую компактную структуру, которую Форни назвал решетчатой (рис. 1.3).

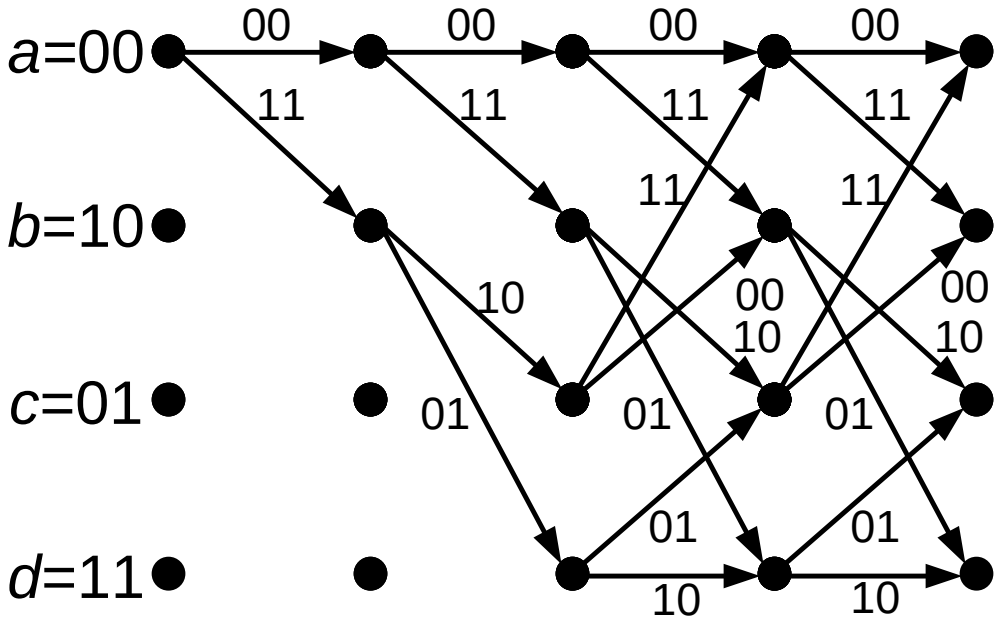

Рис. 1.3. Решетка кодера для сверточного кода
со схемой на рис. 1.1 ($R = 1/2$, $K = 3$)

Таким образом, для сверточного кода со скоростью $R = 1/2$ решетка представляет собой ориентированный граф с периодически повторяющейся структурой, в которой вершины (узлы) графа соединены ребрами. Между процедурой кодирования сверточным кодом и решеткой имеется взаимно однозначное соответствие, которое задается следующими правилами:

– все вершины (узлы) на одном уровне (по горизонтали) соответствуют какому-либо внутреннему состоянию кодера;

– ребро, исходящее из каждой вершины, соответствует одному из возможных двоичных символов, поступивших на вход кодера – верхнее для 0 и нижнее для 1;

– над каждым ребром отмечены значения пар двоичных символов на выходе кодера;

– последовательность ребер (путь на решетке) определяет, с одной стороны, последовательность двоичных символов, поступивших на вход кодера, а с другой – последовательность двоичных пар символов, появившихся на его выходе.

Кроме перечисленных выше параметров, сверточный код характеризуется расстоянием по Хеммингу между выходными кодовыми последовательностями. От распределения расстояний между кодовыми последовательностями, как и в блочных кодах, зависят корректирующие свойства сверточного кода. В сверточных кодах, как правило, присутствует и нулевая полубесконечная выходная последовательность. Тогда для линейных сверточных кодов распределение расстояний между разрешенными кодовыми последовательностями может быть определено через весовой спектр w_i кода, что, в свою очередь, позволит оценить корректирующие свойства кода. Минимальное расстояние по Хеммингу называется *свободным расстоянием* сверточного кода $d_{\text{св}}$. Следовательно, минимальный вес $w_{\min}$ линейного сверточного кода будет равен свободному расстоянию $d_{\text{св}}$.

Таким образом, в режиме обнаружения ошибок свободное расстояние *dсв* позволяет оценить, какое наименьшее количество ошибок должно произойти в канале для того, чтобы одна кодовая последовательность перешла в другую и ошибки не были обнаружены. В режиме исправления ошибок сверточный код может гарантированно исправить ошибки кратностью от 1 до $t = \left\lfloor \frac{d_{\text{св}} - 1}{2} \right\rfloor$ включительно, $\lfloor \ \rfloor$ – целая часть дроби.

Минимальное расстояние Хемминга, т. е. свободное расстояние $d_{\text{св}}$, найдем как расстояние между двумя различными, но сливающимися путями. Для этого воспользуемся решеткой на рис. 1.3, соответствующей сверточному коду на рис. 1.1. Пусть один из двух путей идет через узлы *a* и будет полностью нулевым, т. е. ему также будет соответствовать входная последовательность, состоящая из одних 0. Предположим, что в другой последовательности один 0 заменен на 1. Тогда, как видно на рис 1.3, с этого момента путь с узла *a* пойдет далее через узлы *b* и *c*, а затем снова сольется с узлом *a*. При этом расходящимся в узле *a* и снова сходящимся в том же узле *a* участкам будут соответствовать верхний путь (00 00 00) для входных символов 000 и нижний (11 10 11) – для входных символов 100. Длина каждого из участков равна длине реакции (отклика) кодера на 1. Сравнивая эти два участка, видим, что минимальное расстояние Хемминга между ними равно 5. Из решетчатой диаграммы легко заметить, что точно такое же свободное расстояние будет между участками, расходящимися, например, в узле *a* и сливающимся в узле *b*, *c* или *d*. Зону между участками со свободным расстоянием иногда называют минимальным просветом.

Таким образом, мы построили сверточный код со свободным расстоянием $d_{\text{св}}$ = 5, который способен гарантированно исправить до 2 ошибок на участке, длиной равной реакции кодера. Однако условия и процедура исправления ошибок сверточными кодами существенно отличается от блочных помехоустойчивых кодов.

Поиск хороших сверточных кодов (с наибольшим $d_{\text{св}}$ при заданных R и кодовом ограничении K) обычно осуществляется перебором всех порождающих полиномов на ЭВМ.

Еще более компактной, чем решетка, является *диаграмма состояний* сверточного кода. Диаграмма состояний – это направленный граф переходов из одного состояния в другое. В качестве примера на рис. 1.4 показана диаграмма состояний для кодера, представленного на рис. 1.1. Эта диаграмма показывает, что возможные переходы таковы:

$$a \xrightarrow{0(00)} a, a \xrightarrow{1(11)} b, b \xrightarrow{0(10)} c, b \xrightarrow{1(01)} d, c \xrightarrow{0(11)} a, c \xrightarrow{1(00)} b, d \xrightarrow{0(01)} c, d \xrightarrow{1(10)} d,$$

где $\alpha \xrightarrow{1(a_0 a_1)} \beta$ означает переход из состояния α в β, когда входной символ 1, а выходные – (a_0, a_1). Пунктирная ветвь (ребро) на графе означает, что входной символ 1, а сплошная ветвь − входной символ 0. Эти же символы (0 или 1) на графе для наглядности стоят рядом с ветвью. Два символа у каждой ветви на диаграмме состояний представляют выходные биты.

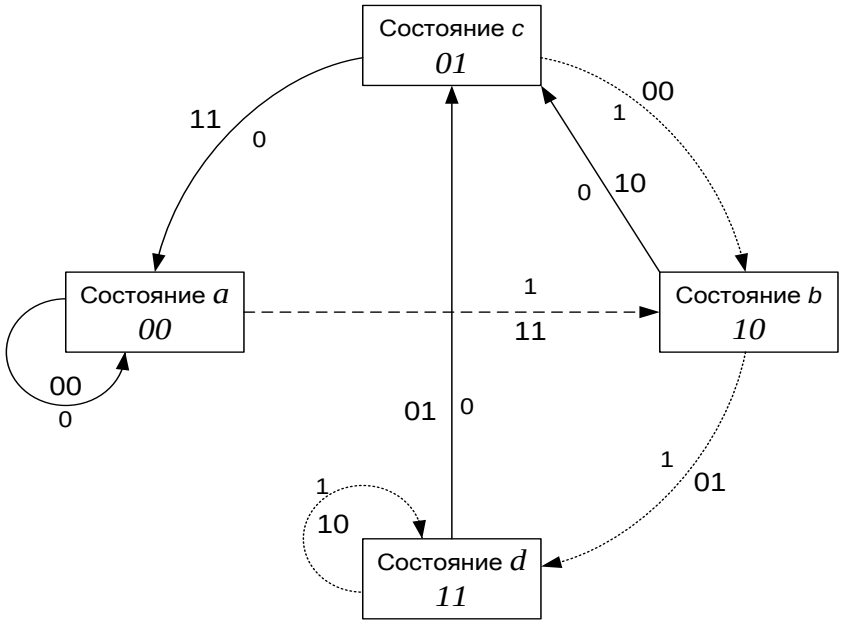

Рис. 1.4. Диаграмма состояний для сверточного кода

Для более точной оценки корректирующих свойств сверточного кода с помощью диаграммы состояний и на основе теории направленных графов можно довольно легко найти весовые характеристики кодовых последовательностей, т. е. определить дистанционные свойства сверточного кода [6, 7]. Для этого на диаграмме состояний сверточного кода, например, представленной на рис. 1.4, все ребра пометим символами D в степени 0, 1 или 2 в зависимости от веса пары бит, стоящей у данного ребра. Кроме того, условимся, что будем рассматривать дистанционные свойства относительно нулевого пути решетчатой диаграммы (рис. 1.3), т. е. все ненулевые пути, исходящие из узла a (состояние 00) и снова сливающиеся впервые в этом же узле. Тогда мы можем представить диаграмму состояний (рис. 1.4) в виде, показанном на рис. 1.5.

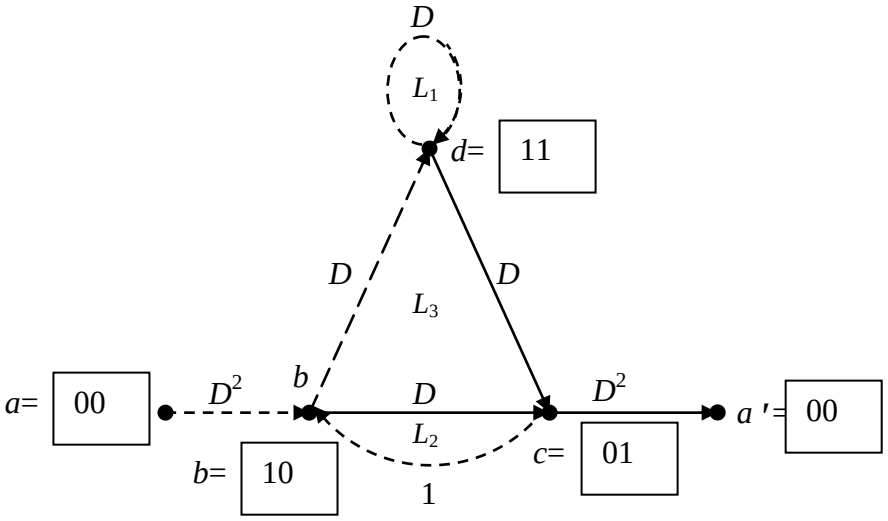

Рис. 1.5. Диаграмма состояний
(на ребрах при D^i степень i указывает на вес выходной пары бит)

Для определения весового спектра (нумераторов весов) сверточного кода необходимо найти передачу из исходящего узла a в конечный узел a' графа как функцию от переменной D. Одно из решений этой задачи, предложенное Витерби [6], состоит в решении системы уравнений относительно искомой передачи. Второе – это применение формулы Мэзона – Циммермана [9]. Первый способ целесообразно применять для сложных направленных графов, а второй – для сравнительно простых графов, например, как на рис. 1.5. На основе формулы Мэзона – Циммермана передача из узла a в конечный узел a' в общем виде может быть представлена следующим выражением:

$$T_{(a\to a')} = \frac{\sum\limits_i P_i \prod\limits_j \left(1 - L_j\right)}{\prod\limits_{m=1}^{k} \left(1 - L_m\right)} = \frac{\sum\limits_i P_i \prod\limits_j \left(1 - L_j\right)}{1 - \sum\limits_{m=1}^{k} L_m + \left(\sum\limits_{i,j} L_i L_j\right)^* - \left(\sum\limits_{i,j,z} L_i L_j L_z\right)^* + \ldots}, \quad (1.3)$$

где P_i – передача i-го прямого пути (без замкнутых циклов) из узла a в узел a';

L_j – передача j-й петли на графе;

k – количество петель на графе.

В числителе выражения (1.3) под знаком произведения $\prod_j$ должны находиться только те сомножители $(1 - L_j)$, у которых петля L_j не соприкасается с прямым путем P_i. В знаменателе выражения (1.3) звездочки обозначают тот факт, что среди членов соответствующих сумм в произведениях L_iL_j, $L_iL_jL_z$ и т. д. должны отсутствовать соприкасающиеся или пересекающиеся петли на графе.

Исходя из вышесказанного, передача из узла a в узел a' графа (рис. 1.3), имеющего три петли L_1 (в узле d), L_2 (с переходами между узлами $b \to c \to b$) и L_3 (с переходами между узлами $b \to d \to c \to b$), будет равна

$$T_{(a \to a')} = \frac{\{abca'\}(1 - L_1) + \{abdca'\}}{1 - (L_1 + L_2 + L_3) + L_1L_2}. \qquad (1.4)$$

В представленном выражении (1.4) в фигурных скобках условно записаны передачи прямых путей из узла a в конечный узел a' без петель.

Так как передачу из узла a в конечный узел a' необходимо выразить как функцию от переменной D, то в представленной формуле (1.4) пути в фигурных скобках и петли L_i запишем через весовые коэффициенты соответствующих ребер графа. Так, передача частного прямого пути $\{abca'\}$ будет равна произведению весовых коэффициентов ребер $\{ab\}$, $\{bc\}$ и $\{ca'\}$, т. е. произведению $D^2\, D\, D^2 = D^5$. Аналогично передача пути $\{abdca'\}$ будет равна D^6. По такому же принципу передачи петель L_i заменим на произведение весовых коэффициентов соответствующих ребер графа: $L_1 = D$, $L_2 = D$, $L_3 = D^2$. Подставив в (1.4) значения прямых путей из узла a в узел a' и петель, получим передачу

$$T_{(a \to a')}(D) = \frac{D^5}{1 - 2D} = D^5 + 2D^6 + 4D^7 + \ldots + 2^k D^{k+5} + \ldots \qquad (1.5)$$

в виде производящей функции от переменной D, которая представляет собой распределение весов путей, исходящих из узла a и снова сливающихся в узле a. Веса путей представлены степенью у символов D.

Таким образом, из (1.5) видно, что по отношению к нулевому пути для рассматриваемого примера сверточного кода существует один путь с весом 5; 2 пути с весом 6; 4 пути с весом 7 и т. д., т. е. 2^k путей с весом $(k + 5)$. Очевидно, что знание распределения весов необходимо для аналитической оценки вероятностей правильного и неправильного декодирования после-

довательностей, но этого недостаточно. Необходимо также знать не только веса, но и длины путей. А для оценки эквивалентной вероятности битовой ошибки (вероятности остаточной ошибки после применения сверточного кода) еще надо знать, сколько ошибок порождает неправильно декодированная последовательность сверточного кода в исходной информационной последовательности. Методика оценки эквивалентной вероятности ошибки в блоковых (n, k)-кодах для непрерывных кодов неприменима. Эти характеристики для сверточных кодов также можно найти с помощью направленного графа, поставив у ребер соответствующие переменные и найдя, по аналогии с предыдущим, передачу из узла a в узел a'. Пусть переменная M соответствует одному такту работы кодера, когда для кода со скоростью $R = 1/2$ на вход поступает один информационный символ, а с выхода кодера будет считано 2. В общем случае для кода с $R = k/n$ за один такт работы кодера на вход поступит k информационных символов, а с выхода будет считано n двоичных символов. Переменной N пусть соответствует вес входной информационной последовательности, поступившей на вход кодера за один такт его работы. Тогда, для рассматриваемого здесь примера сверточного кода (рис. 1.1) со скоростью $R = 1/2$ и его решетчатой диаграммы (рис. 1.3), граф на рис. 1.5 примет вид, представленный на рис. 1.6.

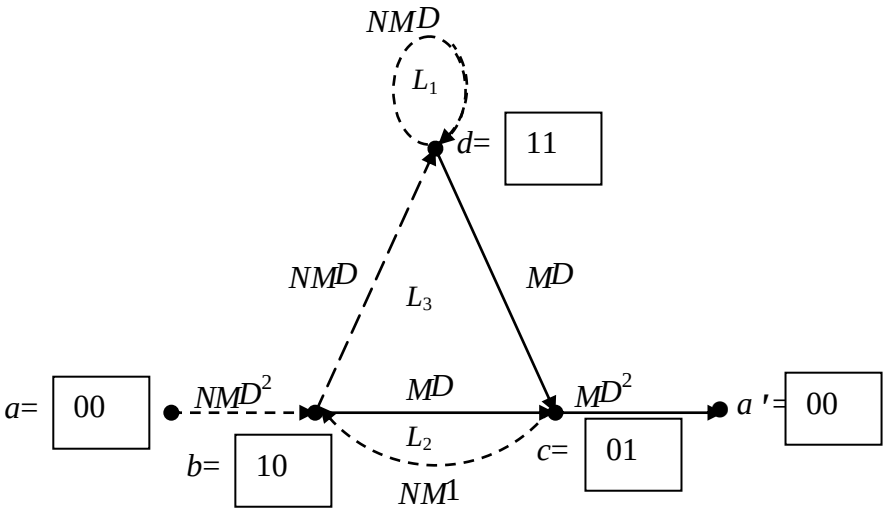

Рис. 1.6. Диаграмма состояний
(на ребрах при D^i степень i указывает на вес выходной пары бит)

Как видно из этого рисунка, у ребер, кроме переменной D, появился символ M, соответствующий одному такту работы кодера, и переменная N в первой степени у того ребра, которому соответствует входной информационный бит, равный 1 (пунктирные ребра), т. е. $N^1 = N$. Если входной бит равен 0, то переменная N у соответствующего ребра (сплошные ребра) не проставляется, так как $N^0 = 1$.

Передачи петель у нового графа будут $L_1 = NMD$, $L_2 = NM^2D$, $L_3 = N^2M^3D^2$.

Подставив значения передач прямых путей и петель в выражение (1.4), найдем передачу графа из узла a в узел a' как функцию от переменных D, N и M:

$$T_{(a\to a')}(D,N,M) = \frac{NM^3D^5}{1-NM(1+M)D} = \\ = NM^3D^5 + N^2M^4(1+M)D^6 + \ldots + N^iM^{2+i}(1+M)^{i-1}D^{4+i}, \tag{1.6}$$

где $i \geq 1$.

Из выражения (1.6) следует, что в рассматриваемом сверточном коде со скоростью $R = 1/2$ для $i = 1$ существует один путь, исходящий из узла a и снова попадающий в этот же узел, вес которого равен 5 (степень при переменной D), а его длина 6 (3 такта работы кодера – степень у переменной M – по 2 бита на выходе с каждым тактом). Именно этот путь имеет минимальное расстояние Хемминга по отношению к нулевой последовательности, и, следовательно, свободное расстояние $d_{\text{св}} = 5$. Декодировав такой путь, получим соответствующую входную информационную последовательность длиной 3 бита (3 такта работы кодера по 1 биту на входе с каждым тактом) с одной 1 (степень у символа N) и двумя нулями на входе кодера. Это означает, что в этот путь могла быть декодирована нулевая последовательность из 6 нулей (3 нуля на входе кодера) вследствие возникновения в канале передачи 5 ошибок. При этом в восстановленной информационной последовательности из 3 бит будет только одна ошибка.

Аналогично из второго слагаемого ($i = 2$) в выражении (1.6) следует, что существуют также 2 пути, исходящие и сходящиеся в узле a, с весом 6. Один из них имеет длину 8 = (4 × 2), а другой – 10 бит. Степень при N говорит о том, что обоим этим путям соответствуют входные информационные последовательности с двумя 1, одна входная последовательность имеет длину 4, а другая 5 (степени при M). Очевидно, что в эти последовательности с весом 6 может быть декодирована нулевая последовательность в результате возникновения в канале 6 ошибок, в то время как в восстановленной информационной последовательности, по сравнению с нулевой, будут только 2 ошибки.

Последовательно увеличивая $i \geq 1$, из общего выражения слагаемого в (1.6)

$$N^iM^{2+i}(1+M)^{i-1}D^{4+i}$$

можно определить все пути, которые начинаются в нулевом состоянии и снова сливаются с ним, не попадая в него в промежуточные моменты, со-

ответствующие им веса и длины входных и выходных последовательностей. В принципе, этих сведений достаточно, чтобы оценить вероятностно-временные характеристики сверточного кода со скоростью $R = 1/2$.

Аналогично может быть построен и проанализирован граф для скорости $R = 1/n$. При этом следует иметь в виду, что длина входной информационной последовательности будет равна j (j – степень при M), а выходной – jn символов. Если же скорость кода $R = k/n$, то длина входной последовательности будет jk. Разумеется, и в том, и в другом случаях распределение весов будет различным и зависящим от структуры сверточного кода.

1.2. Алгоритм декодирования Витерби

Задача декодирования сверточного кода, как упоминалось ранее, заключается в нахождении пути, который ближе всего по расстоянию Хемминга к принимаемой последовательности, т. е. декодирование по принципу максимального правдоподобия. Наиболее простой путь решения поставленной задачи был впервые предложен А. Витерби еще в 1967 г. [5]. Впоследствии он получил широкую известность как алгоритм Витерби. В основе алгоритма Витерби лежит выбор наиболее правдоподобного маршрута на решетчатой диаграмме сверточного кода, при этом в качестве метрики используется именно расстояние Хемминга.

В качестве примера рассмотрим декодирование на основе алгоритма Витерби для сверточного кода с $R = 1/2$ и кодовым ограничением $K = 3$, схема кодера и решетчатая диаграмма которого показаны рис. 1.1 и 1.3. Предположим, что информационная последовательность состояла из одних нулей, тогда с выхода кодера в канал будет передана нулевая последовательность 00 00 00 00 00…

Предположим далее, что на длине реакции кодера (6 бит) возникли две ошибки, и принятая последовательность с момента возникновения первой ошибки имеет вид 10 00 10 00 00 00 … (рис. 1.7).

На первом уровне принятая пара бит (10) сравнивается с двумя ребрами, исходящими из узла a. Оба они имеют одинаковую метрику (расстояние Хемминга), равную 1. Происходит запоминание этих двух путей. На втором уровне, продвигаясь по решетке, происходит сравнение новой пары бит (00) с четырьмя ребрами, исходящими из двух предшествующих, сохраненных в памяти декодера, путей. При этом метрики накапливаются. В результате запоминаются уже 4 пути: верхний путь в узел a с метрикой 1, следующий путь в узел b с метрикой 3 и два нижних пути (в узлы c и d) с одинаковыми метриками 2. Начиная с третьего уровня, в каждый из 4 узлов поступают по два ребра, поэтому сравниваются по два пути, сливающиеся

в каждой вершине. Из двух сохраняется тот, накопленная метрика которого лучше, а другой отбрасывается. Таким образом, из 8 путей сохраняются только 4. Эти оставшиеся пути называют *выжившими*. Для наглядности отброшенные пути на третьем уровне зачеркнуты крестиками. Например, при поступлении из канала третьей пары бит (10), в узел *a* приходит путь с метрикой 2 из узла *a* и второй путь с метрикой 3 из узла *c*, который отбрасывается из-за большего расстояния Хемминга. Аналогично поступаем с путями, входящими в другие узлы. Остаются только 4 выжившие пути с метриками 2, 2, 3 и 2 в узлах *a, b, c* и *d* соответственно. На рис. 1.7 при приеме 4, 5 и 6-й пар (00 00 00) для упрощения диаграммы отбрасываемые пути не показаны. Как мы видим, на каждом из уровней, начиная с третьего, остаются только 4 «выживших» пути, следовательно, и запоминаются только эти 4 пути, а не все их множество на кодовом дереве. Это существенно упрощает реализацию процесса декодирования по алгоритму Витерби. Следует учесть, что в определенных случаях, как это показано на 4-м уровне, в узлы приходят по два пути с одинаковыми метриками. В этом случае равновероятно выбирается один из них. На 6-м уровне имеется один «выживший» путь с метрикой 2, а остальные три – с метрикой 4. Это говорит о том, что с большой вероятностью правильный путь нулевой (на решетчатой диаграмме выделен жирными ребрами), а две единицы в принятой последовательности были ошибками, которые сверточный код исправил.

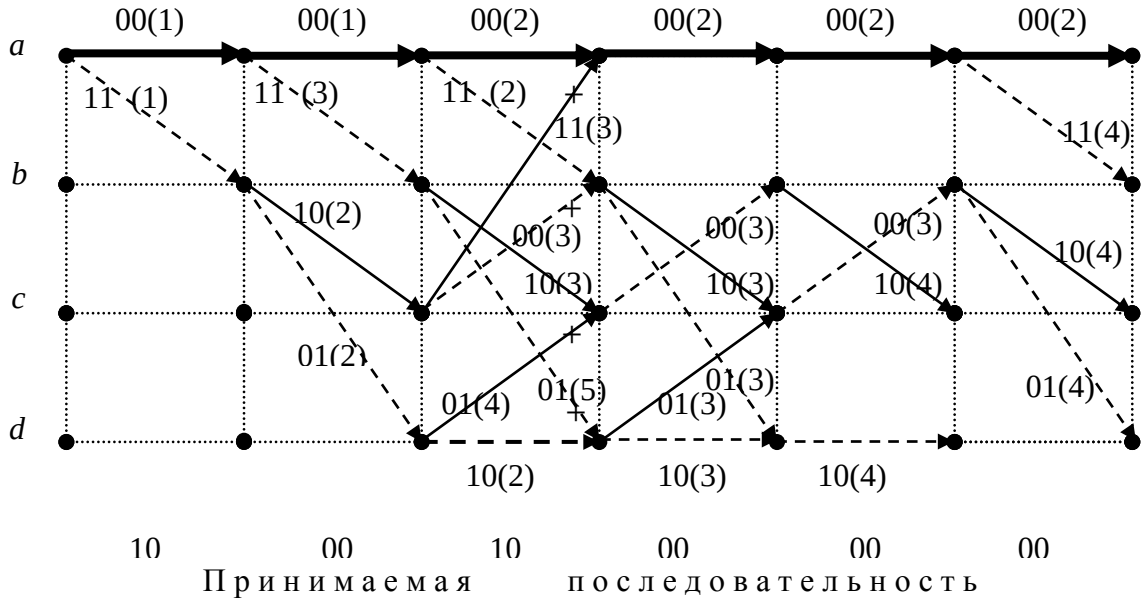

Рис. 1.7. Пример декодирования исправляемой комбинации ошибок по алгоритму Витерби

Таким образом, двигаясь по решетчатой диаграмме, декодер запоминает на каждом уровне в каждом состоянии только по одному выжившему пути (всего их 4) и расстояния от них до принятой последовательности. На определенном шаге принимается решение в пользу пути, имеющего наименьшее расстояние от принятой последовательности. Двигаясь по выбранному пути на решетчатой диаграмме обратно к началу, можно восстановить исходную информационную последовательность, при этом нижнему пунктирному ребру на диаграмме будет соответствовать 1, а верхнему сплошному – 0.

Как мы видим, процедура декодирования по алгоритму Витерби является, на первый взгляд, довольно простой. Однако реализация этого алгоритма сопряжена с определенными трудностями. В частности, не ясен момент, когда следует принимать решение о выбранном пути, т. е. интервал времени, в течение которого должен быть обработан участок принимаемой последовательности и выбран путь. Обычно этот интервал называют *глубиной декодирования* сверточного кода. Понятно, что она зависит от помеховой обстановки в канале и не может быть вычислена заранее. Поэтому при практической реализации в декодере устанавливается некоторая фиксированная глубина декодирования. Как показано в [7], глубина декодирования должна устанавливаться в пределах от $5m$ до $10m$, где m – логарифм по основанию 2 от числа состояний кодера (одна из трактовок длины кодового ограничения). Очевидно, от глубины декодирования зависит и емкость памяти, требуемой для запоминания путей и их метрик в процессе декодирования.

Рассмотренный вариант сверточного кодирования со скоростью $R = 1/2$ наиболее прост, его свойства и реализация легко распространяются на коды со скоростью $R = 1/n$. Сложнее обстоит дело с кодами со скоростями $R = k/n$, где $k > 1$. Во-первых, существенно увеличивается число путей на решетчатой диаграмме – из каждого узла для двоичных кодов исходят 2^k ребер; во-вторых, в каждый узел решетки поступает также 2^k путей, что усложняет реализацию алгоритма декодирования Витерби по поиску наиболее правдоподобного пути. Однако, несмотря на все это, принципиальные свойства сверточного кодирования остаются прежними. Приведем пример сверточного кода с $R = 2/3$ и кодовым ограничением $K = 4$, рассмотренный в [7]. Схема кодера показана на рис. 1.8. С каждым тактом работы кодера на два его входа поступают два символа: на один из входов символ $I_1(x)$, а на второй – символ $I_2(x)$. На выходах трех сумматоров по модулю 2 формируются выходные символы $T_1(x)$, $T_2(x)$ и $T_3(x)$. Порождающие многочлены для верхних ячеек $Я_{11}$ и $Я_{12}$ будут $g_{11}(x) = g_{12}(x) = 1 + x$ и $g_{13}(x) = 1$, для нижних ячеек $Я_{21}$ и $Я_{22}$ соответственно будут $g_{21}(x) = 0$ и $g_{22}(x) = x$; $g_{23}(x) = 1 + x$.

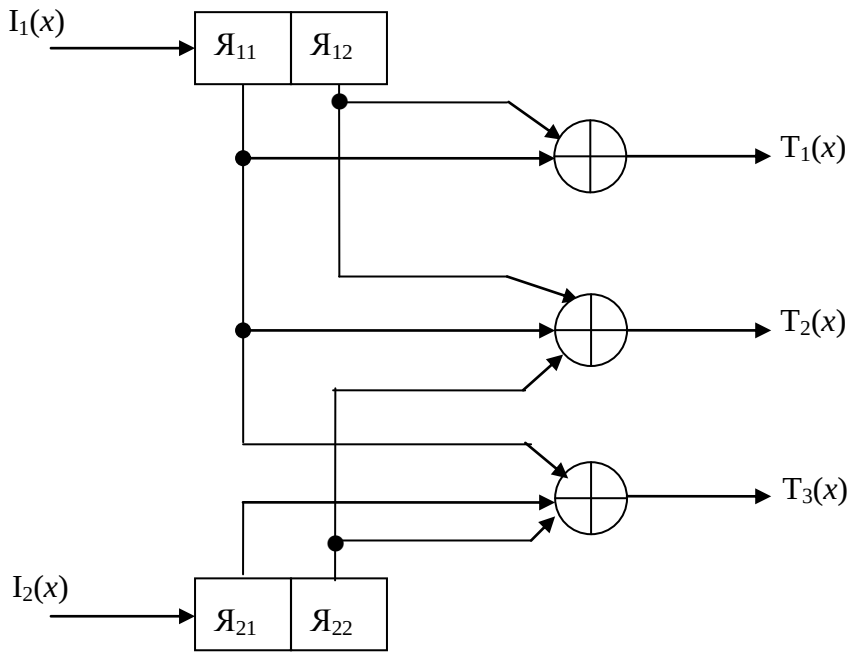

Рис. 1.8. Схема сверточного кодера
с $R = 2/3$ и длиной кодового ограничения $K = 4$

Тогда порождающая матрица кода в полиномиальном представлении будет иметь вид:

$$G(x) = \begin{bmatrix} g_{11}(x) & g_{12}(x) & g_{13}(x) \\ g_{21}(x) & g_{22}(x) & g_{23}(x) \end{bmatrix} = \begin{bmatrix} 1+x & 1+x & 1 \\ 0 & x & 1+x \end{bmatrix}. \qquad (1.7)$$

Следовательно, выходные последовательности на выходах сумматоров могут быть получены в результате умножения матрицы входных последовательностей $[I_1(x)I_2(x)]$ на порождающую матрицу $G(x)$:

$$\begin{aligned} \begin{bmatrix} T_1(x) & T_2(x) & T_3(x) \end{bmatrix} &= \begin{bmatrix} I_1(x) & I_2(x) \end{bmatrix} \cdot G(x) = \\ &= \begin{bmatrix} I_1(x) & I_2(x) \end{bmatrix} \begin{bmatrix} 1+x & 1+x & 1 \\ 0 & x & 1+x \end{bmatrix}, \end{aligned} \qquad (1.8)$$

где операции умножения и сложения производятся по модулю 2.

Пусть, например, $I_1(x) = 1 + x^2$, а $I_2(x) = x$. Тогда последовательности на выходах сумматоров по модулю 2 в соответствии с (1.8) будут следующими:

$$\begin{bmatrix} T_1(x) & T_2(x) & T_3(x) \end{bmatrix} = \begin{bmatrix} \left(1+x^2\right) & x \end{bmatrix} \begin{bmatrix} 1+x & 1+x & 1 \\ 0 & x & 1+x \end{bmatrix} =$$

$$= \begin{bmatrix} \left(1+x+x^2+x^3\right) & \left(1+x+x^3\right) & (1+x) \end{bmatrix}.$$

Двоичные коэффициенты при x^i представляют собой двоичные последовательности на выходах сумматоров, так, на выходе первого сумматора (T_1) будет последовательность (1111000…), на выходе второго сумматора (T_2) – последовательность (1101000…) и на выходе T_3 – последовательность (1100000…). Выходные символы должны поочередно побитно считываться с сумматоров, образуя тройки (T_1 T_2 T_3). Таким образом, на выходе кодера появится последовательность: (111 111 100 110 000 000 …).

Представленная форма порождающей матрицы (1.7) в виде порождающих многочленов удобна в математическом плане, но для реализации более рациональной является векторная форма представления порождающей матрицы по аналогии с (1.2). Для рассматриваемого сверточного кода (рис. 1.8) порождающая матрица G в векторном представлении будет иметь вид

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}, \tag{1.9}$$

где полубесконечные матрицы G_1 и G_2 в двоичном виде будут:

$$G_1 = \begin{bmatrix} 111 & 110 & 000 & 000 & 000 & \dots \\ & 111 & 110 & 000 & 000 & \dots \\ & & 111 & 110 & 000 & \dots \\ & & & 111 & 110 & \dots \\ & & & & \dots & \dots \end{bmatrix}, \quad G_2 = \begin{bmatrix} 001 & 011 & 000 & 000 & 000 & \dots \\ & 001 & 011 & 000 & 000 & \dots \\ & & 001 & 011 & 000 & \dots \\ & & & 001 & 011 & \dots \\ & & & & \dots & \dots \end{bmatrix}. \tag{1.10}$$

Тогда выходная последовательность, составленная из трехбитовых комбинаций (T_1 T_2 T_3), будет получена как произведение вектор-строки из двух входных последовательностей I_1 и I_2 на матрицу G, т. е.

$$\{T_1 T_2 T_3\} = [I_1\ I_2] \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = [I_1\ I_2] \begin{bmatrix} 111 & 110 & 000 & 000 & 000 & \dots \\ & 111 & 110 & 000 & 000 & \dots \\ & & 111 & 110 & 000 & \dots \\ & & & 111 & 110 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 001 & 011 & 000 & 000 & 000 & \dots \\ & 001 & 011 & 000 & 000 & \dots \\ & & 001 & 011 & 000 & \dots \\ & & & 001 & 011 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}.$$

Для рассмотренного выше примера входные последовательности в двоичном виде будут I_1 = (10100…) и I_2 = (01000…). Тогда выходная последовательность будет получена как сумма первой и третьей строк матрицы G_1 и второй строки матрицы G_2, т. е. (111 111 100 110 000 000 …).

На рис. 1.9 представлена решетчатая диаграмма рассматриваемого сверточного кода с R = 2/3 и K = 4. Из диаграммы следует, что в каждом из 4 узлов необходимо сравнивать не два, а 4 пути и выбирать из них лучший из «выживших». Это существенно усложняет реализацию алгоритма Витерби. Но существуют сверточные коды, обеспечивающие повышение скорости без существенного усложнения алгоритма декодирования. К таким кодам относятся *перфорированные* сверточные коды [7, 8]. Процедура перфорации заключается в удалении (выкалывании) определенных кодовых символов. Например, если в сверточном коде с R = 1/2 и K = 3 (рис. 1.1) в каждых двух парах из четырех выходных символов удалять по одному символу, то мы получим код с R = 2/3, т. е. на каждые два входных символа на выходе кодера будет 3.

Правила перфорации могут быть разными, но чаще всего выбирают периодическую перфорацию. При этом правило удаления выходных символов может быть задано в виде двоичной матрицы перфорации P [8], в которой 0 указывают на удаляемые символы выходной последовательности. Пусть для сверточного кода с R = 1/2 и K = 3 (рис. 1.1) матрица перфорации P имеет вид:

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

Такая матрица означает, что в каждых двух парах выходных символов $\left(T_1^i, T_2^i, T_1^{i+1}, T_2^{i+1}\right)$ каждый второй символ второй пары будет удален, т. е. на выходе кодера появятся только три символа $\left(T_1^i, T_2^i, T_1^{i+1}\right)$. Здесь нижние индексы при T указывают на символы на выходах первого и, соответственно, второго сумматоров кодера, а верхний индекс i указывает на i-й тактовый момент работы кодера. Таким образом, выходная последовательность кодера на рис. 1.1

$$T = \left\{..., T_1^i, T_2^i, T_1^{i+1}, T_2^{i+1}, T_1^{i+2}, T_2^{i+2}, T_1^{i+3}, T_2^{i+3} T_1^{i+4}, T_2^{i+4}, T_1^{i+5}, T_2^{i+5}, ...\right\} \quad (1.11)$$

будет преобразуется в перфорированную последовательность

$$T_p = \left\{..., \left(T_1^i, T_2^i, T_1^{i+1}\right)\left(T_1^{i+2}, T_2^{i+2}, T_1^{i+3}\right)\left(T_1^{i+4}, T_2^{i+4}, T_1^{i+5}\right)...\right\}. \quad (1.12)$$

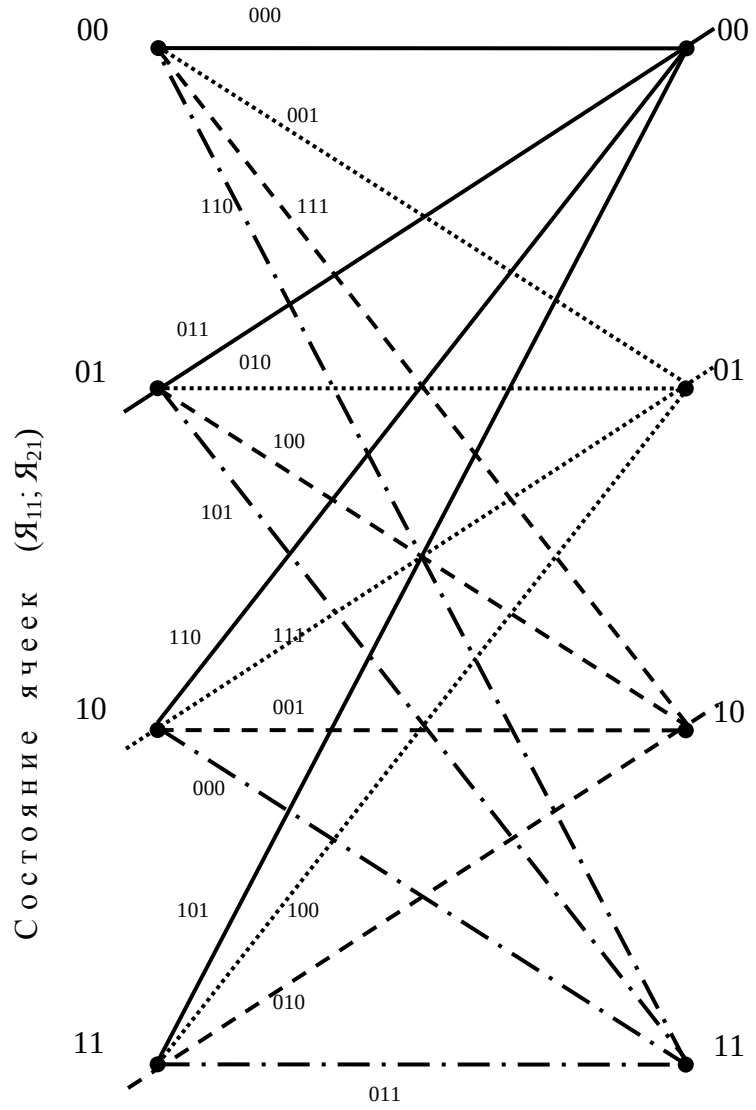

Рис. 1.9. Решетчатая структура сверточного кода с $R = 2/3$ и $K = 4$
(на диаграмме парам различных входных символов ($I_1(x)$; $I_2(x)$)
соответствуют различные линии:
для пары (00); для пары (01)
для пары (10); для пары (11)

Положительным свойством полученного перфорированного сверточного кода со скоростью $R = 2/3$ и $K = 3$ является то, что его декодер, реализующий алгоритм Витерби, остается таким же, как и для кода с $R = 1/2$ и $K = 3$. В то же время реализация перфорированного сверточного кода требует более сложной системы синхронизации. Действительно, если код с $R = 1/2$ требовал синхронизации по парам бит, то полученный перфорированный код с $R = 2/3$ требует дополнительной синхронизации на передающей стороне по четырем битам (по двум парам), а на приемной – по трем битам перфорированной последовательности. Кроме того, вводятся дополнительные функции декодера в зависимости от способа декодирования. Так, например, один из способов декодирования состоит в восстановлении

(*деперфорации*) специальных символов стираний на приемной стороне на удаленных позициях. При декодировании позиции с символами стираний не учитываются при вычислении метрик ребер на решетчатой диаграмме. Второй, такой же по сути, способ декодирования основан на матрице перфорации, отличающийся тем, что он не требует деперфорации удаленных позиций.

Еще одной проблемой реализации перфорированного сверточного кода является увеличение глубины декодирования.

Наконец, необходимо иметь в виду и следующее немаловажное свойство перфорированных сверточных кодов: чем большей скорости мы достигаем путем перфораций, тем ниже становится корректирующая способность сверточного кода. Это видно из представленной в [8] таблицы применительно к стандартному сверточному коду NASA с $R = 1/2$, памятью кодера (длиной регистра) 6 и порождающими многочленами $(g_1, g_2) = (171,133)$ в восьмеричной системе записи. Ниже приведена указанная таблица, показывающая соотношение между скоростью и минимальным кодовым расстоянием (свободным расстоянием) $d_{\min}$ по Хеммингу в зависимости от матрицы перфорации (табл. 1.1).

Таблица 1.1

Матрицы перфорации стандартного сверточного кода скорости $R = 1/2$

Скорость	Матрица перфорации	Кодовая последовательность на выходе кодера	$d_{\min}$
1/2 (без перфорации)	$\begin{bmatrix}1\\1\end{bmatrix}$	$\left(T_1^i, T_2^i\right)$	10
2/3	$\begin{bmatrix}1 & 0\\1 & 1\end{bmatrix}$	$\left(T_1^i, T_2^i, T_2^{i+1}\right)$	6
3/4	$\begin{bmatrix}1 & 0 & 1\\1 & 1 & 0\end{bmatrix}$	$\left(T_1^i, T_2^i, T_2^{i+1}, T_1^{i+2}\right)$	5
5/6	$\begin{bmatrix}1 & 0 & 1 & 0 & 1\\1 & 1 & 0 & 1 & 0\end{bmatrix}$	$\left(T_1^i, T_2^i, T_2^{i+1}, T_1^{i+2}, T_2^{i+3}, T_1^{i+4}\right)$	4
7/8	$\begin{bmatrix}1 & 0 & 0 & 0 & 1 & 0 & 1\\1 & 1 & 1 & 1 & 0 & 1 & 0\end{bmatrix}$	$\left(T_1^i, T_2^i, T_2^{i+1}, T_2^{i+2}, T_2^{i+3}, T_1^{i+4}, T_2^{i+5}, T_1^{i+6}\right)$	3

В табл. 1.1, как и ранее, T_1 и T_2 являются символами на выходах первого и второго сумматоров соответственно. Индексы i, $(i+1)$, … соответствуют i-му, $(i + 1)$-му, … тактам работы кодера.

Рассмотренные выше методы декодирования сверточных кодов основаны на применении так называемых жестких решений. Эффективность сверточных кодов может быть повышена, если применить декодирование с мягким решением [8].

2. КАСКАДНЫЕ КОДЫ

2.1. Назначение и цели каскадного кодирования

В развитии и совершенствовании помехоустойчивого кодирования можно выделить несколько этапов. В 1949 г. К. Шеннон доказал фундаментальную теорему, в которой утверждается, что дискретный канал с шумом имеет точно определяемую пропускную способность, и что с помощью подходящих кодов можно передавать информацию с любой скоростью, меньшей пропускной способности канала, и при этом вероятность ошибки после декодирования будет произвольно малой. Взаимосвязь достижимой вероятности ошибки и скорости передачи информации явилась фактором, определяющим прогресс в области помехоустойчивого кодирования. Изобретение блокового кода, исправляющего однократную ошибку, в 1950 г. Р. Хеммингом явилось началом помехоустойчивого кодирования. Эффективность кодов Хемминга в отношении исправления ошибок была низкой. Разработчикам аппаратуры передачи данных требовались коды, исправляющие большое число ошибок. Выход был найден использованием обнаруживающих ошибки кодов с многократным повторением информации при весьма низкой скорости передачи. Возможность практического применения кодов, исправляющих ошибки, нашел П. Элайес в 1954 г. в виде конструкции из двух или более помехоустойчивых блоковых кодов, получившей название итеративного кодирования. Суть итеративного кодирования состоит в двукратном или многократном кодировании передаваемой информации, представленной в виде таблицы двоичных символов, одним или различными двоичными кодами (кодирование по строкам и столбцам в простейшем случае). При этом минимальное кодовое расстояние результирующего кода равняется произведению минимальных кодовых расстояний используемых кодов. Одним из важных достоинств итеративных кодов является возможность итеративной процедуры декодирования принятой кодовой комбинации.

Это был первый опыт создания составных кодов. Итеративные коды, использующие в качестве составных коды Хемминга с минимальным кодовым расстоянием 3 или 4, позволяют получить требуемую достоверность в каналах с независимыми ошибками при достаточно высокой скорости передачи. В каналах с группированием ошибок требуемая эффективность итеративных кодов достигается искусственной декорреляцией ошибок (перемежением). Надежное исправление ошибок итеративными кодами выполняется в результате точного определения координат ошибки с помощью используемых составных кодов.

Бурное развитие теории помехоустойчивого кодирования в 50-е и 60-е гг. XX в. связано с именами И. Рида, Г. Соломона, У. Питерсона,

Э. Берлекэмпа, Д. Форни и др. [8, 13, 16]. Именно Форни [14] является автором каскадных кодов. В 1966 г. он предложил новую конструкцию составных кодов, которая на длительный период времени решила проблему защиты от ошибок в дискретных каналах связи. В отличие от итеративного кодирования составными кодами при каскадном кодировании являются двоичные и недвоичные коды. На первой ступени кодирования кодируемый двоичный массив представляется в виде последовательности элементов расширенного двоичного поля, которая кодируется недвоичным кодом, а на следующей ступени каждый символ полученной недвоичной кодовой комбинации представляется двоичной последовательностью и кодируется двоичным кодом. Эта процедура может быть повторена многократно. Выполненные Форни исследования, показали, что для двоичного симметричного канала при любой скорости передачи R, не превосходящей пропускной способность канала, и при любом выборе числа ε существует каскадный код, со скоростью передачи равной или меньшей R и с вероятностью ошибки не превосходящей ε. Резюмируя, можно сказать, что назначение и цель каскадного кода состоит в следующем: в результате комбинации недвоичных и двоичных кодов создается новый составной код, исправляющий ошибки высокой кратности с поэтапной и поэтому относительно простой реализацией процедур кодирования и декодирования. При этом в реальных каналах с группированием ошибок при соответствующем подборе режимов комбинируемых двоичных и недвоичных кодов можно получить более высокую эффективность результирующего кода, чем в каналах с независимыми ошибками.

Благодаря способности исправлять многократные пакеты ошибок каскадные коды нашли широкое применение в различных системах телекоммуникации, записи и хранения информации. Особенно хорошо они зарекомендовали себя в широкополосных системах радиосвязи.

2.2. Определение каскадного кода и его основные характеристики

Каскадные коды строятся по принципу поэтапного применения двух или более процедур кодирования к последовательности передаваемых информационных символов. В настоящем учебном пособии рассматриваются двухступенчатые каскадные коды. При этом символами кода последующего этапа (ступени) кодирования являются слова кода предыдущей ступени. Процедура кодирования двоичным каскадным кодом сводится к следующему. Последовательность двоичных символов передаваемого сообщения разбивается на K k-элементных блоков. Каждый k-элементный блок рас-

сматривается как символ нового (q-ичного) алфавита и подлежит кодированию (N, K) q-ичным кодом. В результате реализации процедуры кодирования (N, K)-кодом к k-элементным блокам добавляется $N - K$ избыточных k-элементных блоков или символов q-ичного алфавита. Предполагается, что эти избыточные символы имеют представление в виде k-элементных двоичных последовательностей. (N, K)-код получил название кода второй ступени или внешнего кода. Каждый из N k-элементных символов внешнего кода кодируется двоичным (n, k)-кодом первой ступени.

Код первой ступени называют также внутренним кодом. Процедура каскадного кодирования поясняется рис. 2.1. В результате кодирования получается двоичный блок длиной $N \times n$, являющийся кодовой комбинацией каскадного кода.

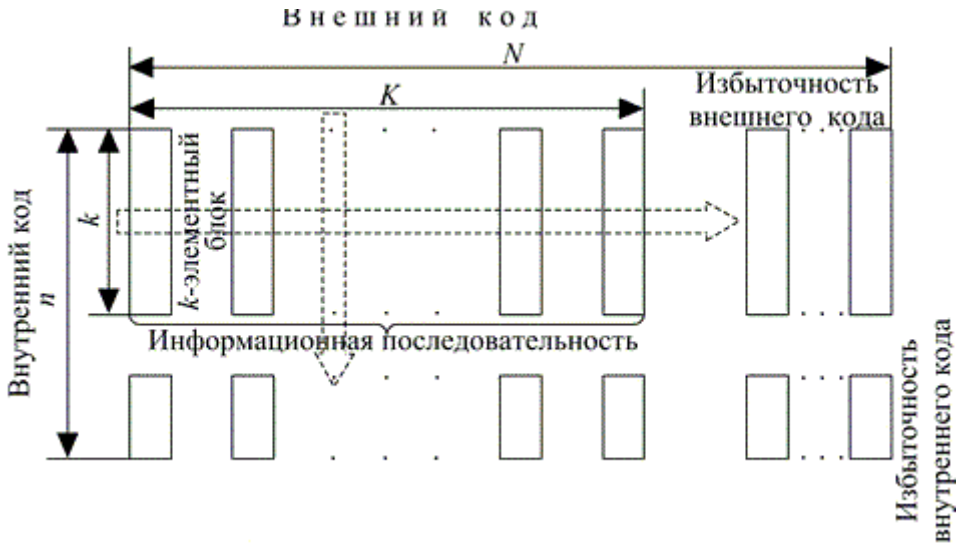

Рис. 2.1. Процедура каскадного кодирования

В теории кодирования доказано, что построенный указанным способом каскадный код является линейным, и его кодовое расстояние D_k не меньше, чем произведение кодовых расстояний внешнего (D) и внутреннего (d) кодов:

$$D_k \geq D \cdot d.$$

Двухступенчатые каскадные коды получили широкое применение в отечественной аппаратуре передачи данных. Структура двухступенчатой системы каскадного кодирования представлена на рис. 2.2.

Двоичная информационная последовательность, подлежащая кодированию каскадным кодом, поступает во внешний кодер, где разбивается на k-элементные блоки, каждый из которых рассматривается внешним кодером как q-ичный символ в двоичном представлении. Для каждых K таких q-ичных символов внешний кодер формирует $N - K$ избыточных q-ичных

символов, т. е. k-элементных блоков. Информационные и избыточные k-элементные блоки затем поступают во внутренний кодер, где преобразуются в кодовые комбинации двоичного (n, k)-кода.

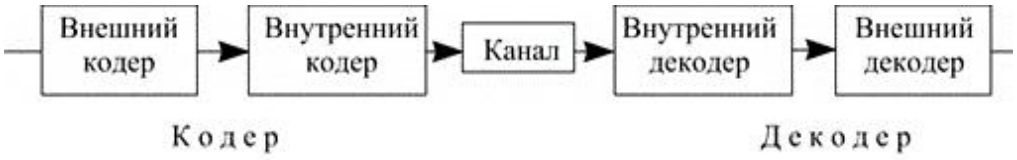

Рис. 2.2. Система каскадного кодирования

Блок длины n передается по каналу и поступает во внутренний декодер. Поток данных, поступающий на выход внутреннего декодера, состоит из k-элементных блоков, которые рассматриваются внешним декодером как символы (N, K)-кода. На выходе внешнего декодера формируются K k-элементных блоков, поступающих к потребителю информации

В качестве внутреннего кода используются двоичные блоковые или непрерывные коды. Внешний код – (N, K)-код Рида – Соломона. Коды Рида – Соломона широко распространены на практике, поскольку являются кодами с максимальным кодовым расстоянием $(D = N - K + 1)$ и имеют в качестве кодовых символов k-элементные двоичные последовательности. Коды Рида – Соломона являются подклассом кодов БЧХ и относятся к недвоичным циклическим кодам, т. е. кодам, символы которых α^i взяты из расширенного двоичного поля Галуа, содержащего $q > 2$ элементов и обозначаемого $GF(q)$, где $q = 2^k$. Информационные символы двоичного внутреннего кода представляют собою набор k-элементных двоичных последовательностей, отображающих символы $\alpha^i \in GF(2^k)$. На практике часто в качестве двоичного внутреннего кода используются укороченные двоичные (n, k)-коды БЧХ и (N, K)-коды Рида – Соломона в качестве внешнего кода. Наиболее часто в (N, K)-кодах Рида – Соломона в качестве кодовых символов используются элементы поля $GF(2^8)$, а количество информационных элементов и длина кодовой комбинации внутреннего двоичного (n, k)-кода БЧХ кратны числу 8.

Достоинством каскадных кодов является относительно низкая сложность кодирующих и декодирующих устройств, так как каскадные коды позволяют выполнить процедуры кодирования и декодирования по этапам, применяя на каждом этапе достаточно короткие, по сравнению с результирующим, коды.

Каскадные коды позволяют реализовать достаточно большое кодовое расстояние, поэтому их применение на каналах с помехами эффективно.

Поэтапная реализация процедуры декодирования позволяет рационально распределить функции между внутренним и внешним декодерами,

реализуя исправление ошибок при минимальной сложности их построения, когда внутренний декодер обнаруживает и частично исправляет ошибки, а внешний декодер исправляет ошибки и стирания.

Другое достоинство каскадных кодов состоит в том, что в силу небольших длин внутренних и внешних кодов для исправления ошибок и стираний можно использовать не только различные конструктивные методы, но и переборные.

Эффективность использования каскадных кодов в ряде случаев повышается за счет некоторой декорреляции ошибок, появляющихся в различных k-элементных блоках в результате поэтапной процедуры декодирования.

2.3. Классификация каскадных кодов

В системах передачи данных классификация каскадных кодов может быть выполнена по нескольким признакам.

1. По области применения:

– однонаправленные системы передачи данных,

– системы передачи данных с обратной связью.

2. По виду и режимам используемых помехоустойчивых кодов:

• *внешний код* – всегда код Рида – Соломона преимущественно с символами в виде элементов $GF(2^8)$ в режимах:

− исправление ошибок,

− исправление стираний,

− исправление ошибок и стираний;

• *внутренний код*:

− блоковые коды:

− короткие двоичные коды БЧХ с $d_{мин} = 4$ (как правило),

− коды Рида – Соломона;

− непрерывные, как правило, сверточные коды с различными методами декодирования:

− декодирование по алгоритму Витерби,

− последовательное декодирование.

Для внутренних блоковых и сверточных кодов предпочтительным является декодирование с использованием мягких решений на выходе демодулятора.

3. По использованию перемежения символов внутреннего кода:

– с перемежением символов (как правило, при использовании сверточных кодов с декодированием по алгоритму Витерби);

– без перемежения символов.

4. По количеству символов внешнего кода в информационной части кодовой комбинации внутреннего кода:

– один символ,

– больше одного символа.

Первый случай характерен для коротких блоковых кодов с небольшой избыточностью; второй – для блоковых кодов при необходимости достижения низкой вероятности необнаружения ошибок и всегда – для непрерывных кодов.

5. По месту использования обратной связи для исправления обнаруженных ошибок:

– при декодировании комбинации внутреннего кода,

– при декодировании комбинации внешнего кода,

– при декодировании комбинации как внутреннего, так и внешнего кода.

2.4. Примеры построения каскадных кодов и методов их декодирования

Каскадные коды нашли широкое практическое применение как инструмент высокоэффективного помехоустойчивого кодирования с простой реализацией. Простота реализации каскадных кодов достигается последовательным применением, как правило, двухступенчатого кодирования и, главное, декодирования передаваемой информации различными кодами. Первая (она же внешняя) ступень кодирования, а при декодировании заключительная, обеспечивается применением кодов Рида – Соломона. Вторая ступень кодирования (внутренняя), а при декодировании первая реализуется при более широком спектре используемых кодов. В качестве внутренних кодов используются как блоковые коды (в основном короткие), так и непрерывные. На приемном конце могут использоваться как жесткие, так и мягкие решения. Декодер внешнего кода на основе этих решений исправляет максимально возможное число ошибок. В рассматриваемых ниже примерах показана возможность построения схем каскадного кодирования при использовании различных типов внутренних кодов с использованием жестких решений.

Пример 2.1. Построим каскадный код с внутренним кодом БЧХ (7,3) и внешним кодом РС (7, 3).

Внутренний код – двоичный код БЧХ (Боуза – Чоудхури – Хоквингема) (7, 3) имеет порождающий многочлен: $g(x) = (1 + x^2 + x^3)(x + 1) = 1 + x + x^2 + x^4$. Порождающая и проверочная матрицы этого кода имеют вид:

$$G_{(7,\,3)} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \qquad H_{(7,\,3)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Минимальное кодовое расстояние данного кода $d_{\min} = 4$. Основная задача внутреннего кода – обнаруживать ошибки в символах внешнего кода Рида – Соломона и формировать стирания на позициях символов кода Рида – Соломона в случае обнаружения ошибок в соответствующих кодовых комбинациях кода БЧХ. В соответствии с $d_{\min}$, двоичный код БЧХ (7, 3) обнаруживает все ошибки кратности до третей включительно и все ошибки старших кратностей, которые по своему виду не совпадают с видом разрешенных кодовых комбинаций. Известно, что все семь ненулевых кодовых комбинаций двоичного кода БЧХ (7, 3) имеют один и тот же вес четыре. Это означает, что данный код обнаруживает 28 из 35 ошибок кратности четыре и все ошибки остальных кратностей, т. е. всего 120 из 127 возможных ошибок.

Внешний код – код Рида – Соломона (РС) (7, 3), порождающий многочлен которого $g(x)$ имеет корни: α^1, α^2, α^3, α^4, являющиеся элементами расширенного двоичного поля $GF(2^3)$.

Для построения поля $GF(2^3)$ необходимо знать примитивный многочлен 3-й степени. Таких многочленов известно два: $x^3 + x + 1$ и $x^3 + x^2 + 1$.

Построим поле по модулю каждого из этих многочленов. Возможные представления элементов поля $GF(2^3)$ приведены в табл. 2.1.

Таблица 2.1.

Представления элементов поля $GF(2^3)$

$\pi_1(\alpha) = \alpha^3 + \alpha + 1$	$\pi_2(\alpha) = \alpha^3 + \alpha^2 + 1$
$\alpha^1 = \alpha = 010$	$\alpha^1 = \alpha = 010$
$\alpha^2 = \alpha^2 = 001$	$\alpha^2 = \alpha^2 = 001$
$\alpha^3 = 1 + \alpha = 110$	$\alpha^3 = 1 + \alpha^2 = 101$
$\alpha^4 = \alpha + \alpha^2 = 011$	$\alpha^4 = 1 + \alpha + \alpha^2 = 111$
$\alpha^5 = 1 + \alpha + \alpha^2 = 111$	$\alpha^5 = 1 + \alpha = 110$
$\alpha^6 = 1 + \alpha^2 = 101$	$\alpha^6 = \alpha + \alpha^2 = 011$
$\alpha^7 = 1 = 100$	$\alpha^7 = 1 = 100$

Получили два различных представления ненулевых элементов $GF(2^3)$. Если дополнить каждое из них нулевым элементом 0 = (000), то получим два представления $GF(2^3)$. Для первого из них первообразным корнем является корень $\pi_1(\alpha)$, а для второго − $\pi_2(\alpha)$. В примере рассмотрим код

Рида – Соломона (7,3) с символами из расширенного двоичного поля $GF(2^3)$, построенного по многочлену $\pi_1(\alpha) = \alpha^3 + \alpha + 1$. Порождающий многочлен внешнего кода РС (7,3) имеет вид: $g(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4) = \alpha^3 + + \alpha x + x^2 + \alpha^3 x^3 + x^4$. Минимальное кодовое расстояние данного кода РС равно $d_{\min} = 5$. Этот код будет использован в качестве внешнего кода. Его основная задача исправлять ошибки и стирания. Данный код способен исправить либо 4 стирания, либо 2 ошибки, либо 1 ошибку и 2 стирания. Результирующий линейный двоичный каскадный код будет иметь обозначение (49,9) и минимальное кодовое расстояние $d_{\min\ рез.} = 5 \times 4 = 20$.

Структура кодовой комбинации создаваемого каскадного кода представлена на рис. 2.3.

Рис. 2.3. Структура кодовой комбинации каскадного кода (49,9)

Пусть передаваемая информация имеет вид: 011111101. В соответствии с параметрами внешнего кода представим эту двоичную последовательность в виде информационной части кодовой комбинации кода РС (7, 3) с коэффициентами, соответствующими элементам используемого поля $GF(23)$: $\alpha 4x6 + \alpha 5x5 + \alpha 6x4$. Кодируемую последовательность в соответствии со структурой кодовой комбинации каскадного кода (49,9) представим в следующем виде.

1	1	0
0	1	0
1	1	0

Далее, в соответствии с известными методами кодирования для кодов РС (эту работу читатель может выполнить самостоятельно [17]) вычисляем избыточные элементы и, тем самым, определяем вид кодовой комбинации внешнего кода. Она имеет следующий вид:

$$F(x) = \alpha^4 + \alpha^6 x + \alpha^2 x^2 + \alpha^5 x^3 + \alpha^6 x^4 + \alpha^5 x^5 + \alpha^4 x^6.$$

В составе формируемой кодовой комбинации каскадного кода полученная комбинация кода РС имеет вид:

0	1	0	1	1	1	0
1	0	0	1	0	1	1
1	1	1	1	1	1	1

Продолжим формирование кодовой комбинации каскадного кода. Двоичная последовательность каждого столбца является информационной частью кодовой комбинации внутреннего кода (7, 3). В соответствии со значением порождающего многочлена определяем вид избыточных элементов для каждой кодовой комбинации внутреннего кода (7, 3) и добавляем их к предыдущей таблице:

0	1	0	1	1	1	0^*
1	0	0	1	0^*	1	1
0	1	1	1	1	1	1
1	1	0	0	1	0	1
1	1	1	0	1	0	0
1	0	1	1	0	1	0
0	0	1	0	0	0	1

Формирование кодовой комбинации каскадного кода завершено.

Дополним условие примера рассмотрением процедуры декодирования рассматриваемого каскадного кода. Предположим, что в сформированной ранее кодовой комбинации каскадного кода (49,9) при передаче по системе связи произошли ошибки в символах, помеченных знаком *.

Декодер внутреннего кода вычисляет синдромы кодовых комбинаций внутреннего кода. В соответствии с видом проверочной матрицы внутреннего

кода они равны: $S_6 = 1101$, $S_5 = 0$, $S_4 = 1110$, $S_3 = 0000$, $S_2 = 0000$, $S_1 = 0000$, $S_0 = 0000$. По значению вычисленных синдромов декодер выносит решение о том, что символы кодовой комбинации внешнего кода при степенях 6 и 4 кодового многочлена внешнего кода стерты.

Декодер внешнего кода выполняет процедуру исправления стираний символов на указанных позициях. Эта процедура выполняется пошагово по методу быстрого декодирования кодов БЧХ.

1 шаг. Стертым символам присваивается значение «0» и принятая комбинация принимает вид: $F(x) = \alpha^4 + \alpha^6 x + \alpha^2 x^2 + \alpha^5 x^3 + \alpha^5 x^5$.

2 шаг. Вычисляются компоненты синдромного многочлена:

$$S_1 = F(x = \alpha) = \alpha^4 + 1 + \alpha^4 + \alpha + \alpha^3 = 0;$$
$$S_2 = F(x = \alpha^2) = \alpha^4 + \alpha + \alpha^6 + \alpha^4 + \alpha = \alpha^6;$$
$$S_3 = F(x = \alpha^3) = \alpha^4 + \alpha^2 + \alpha + 1 + \alpha^6 = \alpha^2;$$
$$S_4 = F(x = \alpha^4) = \alpha^4 + \alpha^3 + \alpha^3 + \alpha^3 + \alpha^4 = \alpha^3.$$

Значит, синдромный многочлен имеет следующий вид:

$$S(x) = \alpha^6 x + \alpha^2 x^2 + \alpha^3 x^3.$$

3 шаг. Вычисляется многочлен локаторов стираний.

$\Gamma(x) = (1 + x\,\alpha^4)\,(1 + x\,\alpha^6) = 1 + \alpha^3 x + \alpha^3 x^2$ и производная многочлена локаторов стираний $\Gamma'(x) = \alpha^3$.

4 шаг. Вычисляются значения стираний.

Коэффициент кодового многочлена при x^4:

$$\Upsilon_4 = \frac{\Omega\left(x = \alpha^{-4}\right)}{\Gamma^1(x)} = \frac{\alpha^2}{\alpha^3} = \alpha^6$$

и коэффициент кодового многочлена при x^6:

$$\Upsilon_6 = \frac{\Omega\left(x = \alpha^{-6}\right)}{\Gamma^1(x)} = \frac{1}{\alpha^3} = \alpha^4.$$

В результате переданная информация поступает к пользователю без искажений.

Пример 2.2. Пусть передаваемая информация имеет тот же вид, что и в предыдущем примере: 011111101. Требуется передать это сообщение с помощью каскадного кодирования с внешним кодом Рида – Соломона и применить в качестве внутреннего кода непрерывный код. Как и в предыдущем примере, для внешнего кодирования выбираем РС-код (7,3)

над полем $GF(2^3)$. Кодовая комбинация внешнего кода $F(x) = \alpha^4 + \alpha^6 x + \alpha^2 x^2 + \alpha^5 x^3 + \alpha^6 x^4 + \alpha^5 x^5 + \alpha^4 x^6$ может быть представлена в виде семи элементов поля $GF(2^3)$: $(\alpha^4\alpha^5\alpha^6\alpha^5\alpha^2\alpha^6\alpha^4)$ или двоичной последовательностью из 21 бита: (011 111 101 111 001 101 011).

В качестве внутреннего кода используем простейший класс непрерывного кода – цепной код. Это систематический код, у которого все сообщения представляются одной кодовой комбинацией, в которой избыточные элементы вставлены между информационными и формируются в соответствии с рекуррентной формулой [2]:

$$R_i = K_i + K_{i-1}. \qquad (2.1)$$

Формирование кодовой комбинации внешнего кода осуществляется кодирующим устройством, схема которого приведена на рис. 2.4.

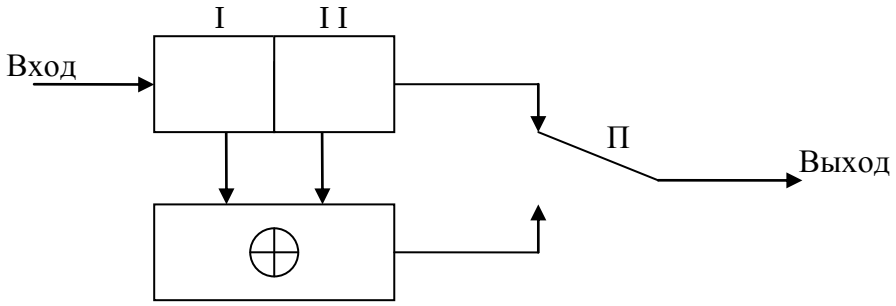

Рис. 2.4. Схема внутреннего кодера

Кодирующее устройство содержит регистр сдвига из двух ячеек, сумматор по модулю 2 и переключатель П. При вводе в кодирующее устройство каждого информационного символа с его выхода снимается два символа: информационный – при верхнем положении переключателя П и проверочный – при нижнем, т. е. скорость выходной закодированной последовательности в два раза выше скорости входной кодируемой информационной последовательности. Каждый проверочный символ участвует только в одной проверке на четность, а каждый информационный – в двух: с предшествующим и последующим проверочным избыточными символами (рис. 2.5).

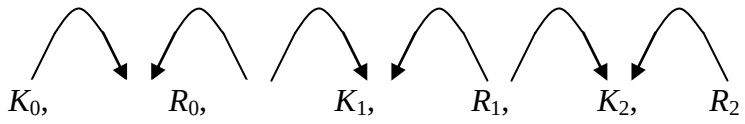

Рис. 2.5. Формирование избыточности в цепном коде

Такое формирование кодовой комбинации дает возможность при декодировании установить, в каком именно разряде принятой последовательности произошла ошибка – в информационном или проверочном. Нарушение одной проверки на четность указывает на искажение проверочного символа, участвующего в этой проверке, а если оно наблюдается дважды, то искажен тот информационный символ, который участвует в двух проверках.

Процедура формирования комбинации внешнего кода для рассматриваемого примера представлена в табл. 2.2. Регистр сдвига кодирующего устройства в исходном состоянии обнулен. В обеих его ячейках записаны нули. В момент записи в ячейку I первого символа кодируемой последовательности в ячейке II всегда будет записан ноль. Он и будет первым символом, считанным из регистра кодера в канал связи (K_0). Этот символ K_0 целесообразно оставить в составе формируемой кодовой комбинации каскадного кода, так как он участвует в формировании первого проверочного элемента R_1 и сам может быть использован как дополнительный проверочный символ. В соответствии с логикой работы схемы кодера после считывания из регистра последнего информационного символа K_{21} вслед за ним в дискретный канал поступает проверочный символ R_{22}, участвующий в защите информационного символа K_{21}. Таким образом, в сформированной кодовой комбинации внутреннего кода содержатся 21 информационный элемент и 23 избыточных, т. е. это комбинация кода (44, 21). Результирующая кодовая комбинация принадлежит каскадному коду (44, 7).

Упрощенная схема декодера для этого кода представлена на рис. 2.6. Декодер каскадного кода с цепным кодом на внутренней ступени состоит из двух последовательно соединенных декодеров – внутреннего и внешнего. Переключатель П на входе внутреннего декодера разделяет информационную последовательность символов принятой комбинации каскадного кода, поступающую с выхода дискретного канала в декодер, на два потока. Положение 1 переключателя П соответствует приему информационных элементов кодовой комбинации внутреннего кода, а положение 2 – приему проверочных элементов этой комбинации. Информационные символы поступают в информационный регистр внутреннего декодера и одновременно в накопитель комбинации кода Рида – Соломона внешнего декодера, а проверочные – в проверочный регистр внутреннего декодера. Информационный регистр совместно с сумматором по модулю 2 формирует последовательность проверочных символов в соответствии со значением принятых информационных символов. На третий вход сумматора от проверочного регистра поступает соответствующий проверочный символ, поступивший из дискретного канала. В случае совпадения сформированного проверочного

символа с принятым из канала на выходе сумматора появляется нулевой символ (нулевой синдром), а в случае несовпадения – единичный символ – синдром ошибки. Значения выработанных синдромов с выхода сумматора записываются в синдромный регистр. Когда в синдромном регистре окажутся записанными все синдромы, вычисленные для совокупности битов, соответствующих символу кодовой комбинации внешнего кода, управляющий регистр осуществляет сброс синдрома из синдромного регистра в логический блок. С выхода логического блока сигнал о наличии или отсутствии ошибок в поступившей во внешний декодер принятого символа комбинации кода Рида – Соломона в виде сопровождающего признака. В соответствии с этим признаком символы комбинации кода Рида – Соломона, которым соответствовали синдромы ошибок, признаются стертыми. Если стирания отсутствуют или число стертых символов не превосходит заданного числа, обусловленного способностью кода Рида – Соломона исправлять стирания, то блок исправления стираний и ошибок осуществляет процедуру декодирования принятой комбинации кода Рида – Соломона. Процедура декодирования кода Рида – Соломона подробно описана в литературе [16, 17]. Один из возможных случаев приведен в примере 2.1.

Рассмотрим процедуру декодирования сформированной выше комбинации каскадного кода (44,7). Работа декодера, представленного на рис. 2.6 и рассмотренная выше процедура декодирования внутреннего кода иллюстрируется табл. 2.3. Здесь переданное сообщение соответствует сформированному сообщению в табл. 2.2. Принятое сообщение содержит две ошибки в тех же символах, в которых они были допущены в примере 2.1. В табл. 2.3 показаны восстановленные в соответствии с процедурой декодирования внутреннего цепного кода проверочные символы, приведены синдромы и моменты формирования признаков стирания. Процедура декодирования комбинации внешнего кода Рида – Соломона для данного примера полностью соответствует рассмотренной в примере 2.1.

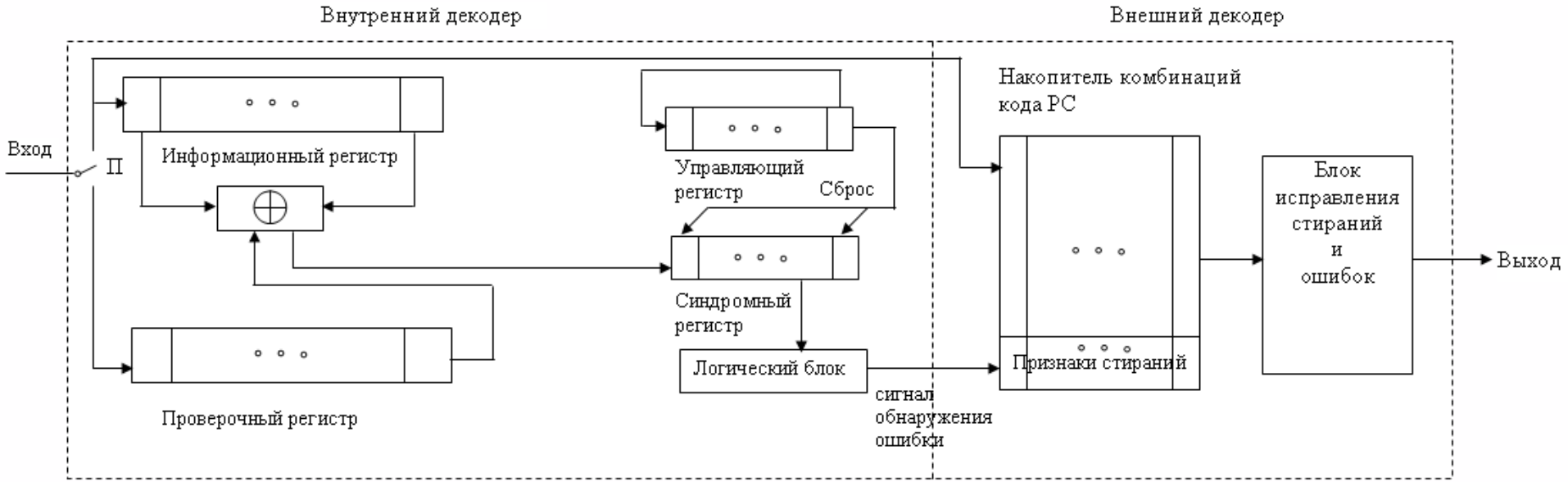

Рис. 2.6. Декодер каскадного кода с цепным кодом на внутренней ступени

Таблица 2.2

Процесс кодирования сообщений

№	0	1	2	3	4	5	6	7	8	9	10	11
Символы на входе кодера $K_{вх}$	-	0	1	1	1	1	1	1	0	1	1	1
Содержимое регистра	00	00	10	11	11	11	11	11	01	10	11	11
Значение $K_{вых}$	-	0	0	1	1	1	1	1	1	0	1	1
Значение $R_{вых}$	-	0	1	0	0	0	0	0	1	1	0	0
Обозначение символов	--	K_0 R_1	K_1 R_2	K_2 R_3	K_3 R_4	K_4 R_5	K_5 R_6	K_6 R_7	K_7 R_8	K_8 R_9	K_9 R_{10}	K_{10} R_{11}

№	12	13	14	15	16	17	18	19	20	21	22
Символы на входе кодера $K_{вх}$	1	0	0	1	1	0	1	0	1	1	-
Содержимое регистра	11	01	00	10	11	01	10	01	10	11	01
Значение $K_{вых}$	1	1	0	0	1	1	0	1	0	1	1
Значение $R_{вых}$	0	1	0	1	0	1	1	1	1	0	1
Обозначение символов	K_{11} R_{12}	K_{12} R_{13}	K_{13} R_{14}	K_{14} R_{15}	K_{15} R_{16}	K_{16} R_{17}	K_{17} R_{18}	K_{18} R_{19}	K_{19} R_{20}	K_{20} R_{21}	K_{21} R_{22}

Таблица 2.3

Процесс декодирования

Переданное сообщение	Обозначение символов	K_0	R_1	K_1	R_2	K_2	R_3	K_3	R_4	K_4	R_5	K_5	R_6	K_6	R_7	K_7	R_8	K_8	R_9	K_9	R_{10}	K_{10}	R_{11}
	Данные	0	0	0	1	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1	0	1	0
Принятое сообщение		0	0	1*	1	1	0	1	0	1	0	1	0	1	0	1	1	1*	1	1	0	1	0
Восстановленные проверочные символы			1*		0*		0		0		0		0		0		0*		0*		0		0
Синдром			1		1		0		0		0		0		0		1		1		0		0
Формирование признака стирания									1						0						1		

Переданное сообщение	Обозначение символов	K_{11}	R_{12}	K_{12}	R_{13}	K_{13}	R_{14}	K_{14}	R_{15}	K_{15}	R_{16}	K_{16}	R_{17}	K_{17}	R_{18}	K_{18}	R_{19}	K_{19}	R_{20}	K_{20}	R_{21}	K_{21}	R_{22}
	Данные	1	0	1	1	0	0	0	1	1	0	1	1	0	1	1	1	0	1	1	0	1	1
Принятое сообщение		1	0	1	1	0	0	0	1	1	0	1	1	0	1	1	1	0	1	1	0	1	1
Восстановленные проверочные символы			0		1		0		1		0		1		1		1		1		0		1
Синдром			0		0		0		0		0		0		0		0		0		0		0
Формирование признака стирания					0						0						0						0

Пример 2.3. В этом примере рассматривается возможность применения в качестве внутреннего кода непрерывного сверточного кода с декодированием по алгоритму Витерби. (Вопросы построения сверточных кодов, процедур их кодирования и декодирования подробно изложены в разд. 1.) Рассмотрим особенности применения этих кодов в качестве внутренних при каскадном кодировании. При самостоятельном использовании сверточного кода вне системы каскадного кодирования формула алгоритма декодирования Витерби «сложение, сравнение, выбор» позволяет анализировать метрики возможных путей в целях определения выживших на решетчатой диаграмме на временных интервалах, не допускающих перенасыщения метрик. При использовании сверточного кода в системе каскадного кодирования анализ метрик необходимо производить строго на N интервалах, где N – длина кодовой комбинации внешнего кода. Целью этого анализа является определение наличия или отсутствия ошибочных битов на интервале кодовой последовательности, соответствующем символу внешнего кода Рида – Соломона. В связи с этим вся кодовая последовательность сверточного кода должна быть разделена на N участков декодирования (окон декодирования) длины n. Если на интервале окна декодирования определен наилучший путь, то символ кода Рида – Соломона считается принятым верно. Если разности метрик не позволяют выбрать наилучший путь, то соответствующий символ кода Рида – Соломона считается стертым. В конце каждого окна декодирования после принятия решения о наличии или отсутствии ошибок в соответствующем окне декодирования символе кода Рида – Соломона метрики всех путей обнуляются. Упрощенный алгоритм декодирования внутреннего кода в рассматриваемом случае приведен на рис. 2.7.

2.5. Режимы использования каскадных кодов

Возможны различные алгоритмы декодирования внутреннего и внешнего кодов. Внутренний код можно декодировать с исправлением ошибок, с обнаружением ошибок, а также с частичным исправлением ошибок малых кратностей и обнаружением ошибок более высоких кратностей.

В двух последних случаях символы кода Рида – Соломона, соответствующие комбинациям внутреннего кода, в которых обнаружены ошибки, считаются стертыми, т. е. в дальнейшем при декодировании не используются. Содержащаяся в них информация восстанавливается при декодировании внешнего кода.

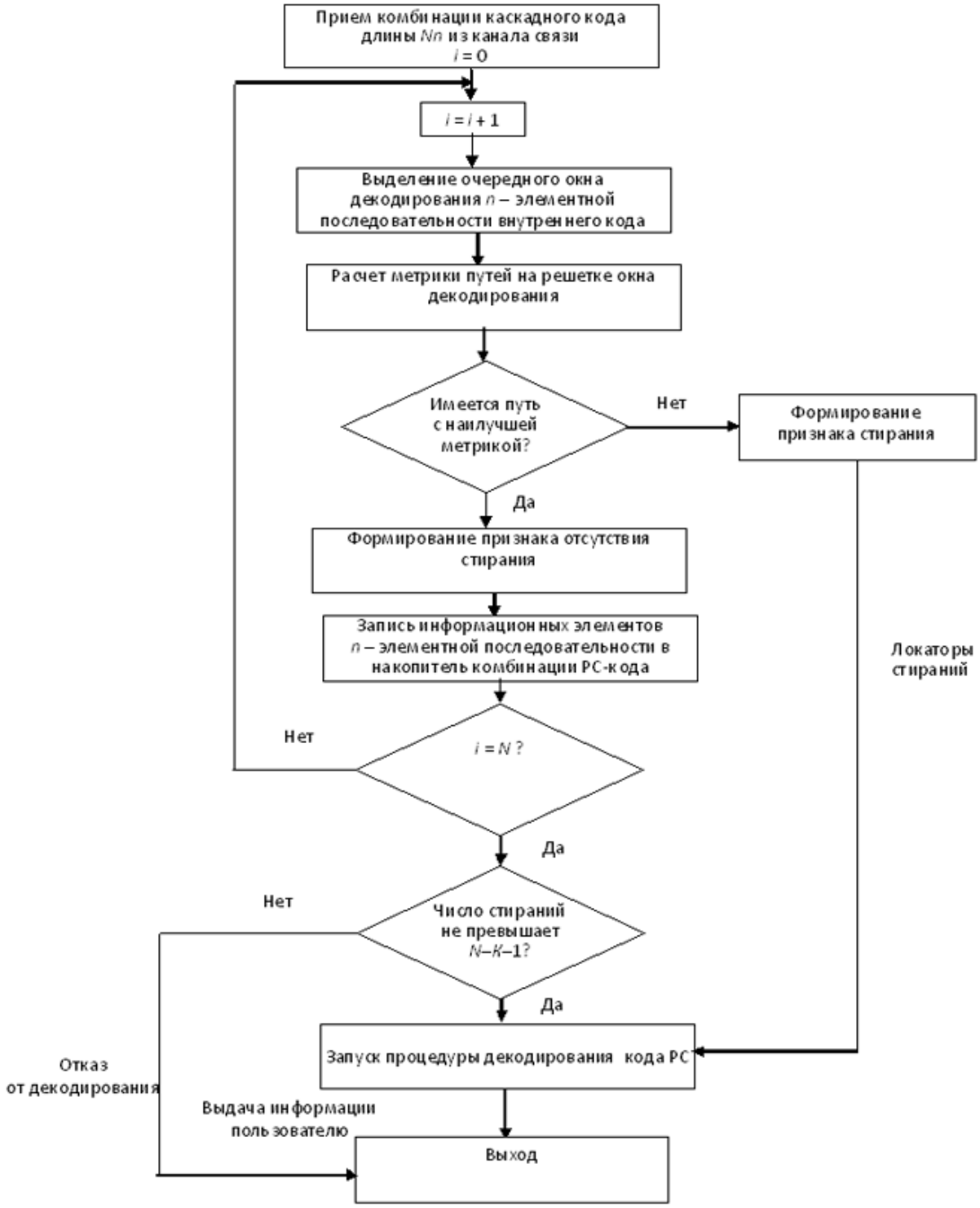

Рис. 2.7. Алгоритм работы декодера каскадного кода со сверточным кодом с декодированием по алгоритму Витерби на внутренней ступени

В соответствии с этим внешний код может использоваться для исправления ошибок, не исправленных внутренним кодом, для исправления стираний либо для совместного исправления стираний и ошибок. Возможно использование внешнего кода в режиме обнаружения ошибок. При этом стертый блок может быть восстановлен путем использования обратной связи.

Таким образом, возможны следующие режимы использования каскадных кодов [15]:

- исправление ошибок внутренним и внешним кодами;
- обнаружение ошибок внутренним и исправление стираний внешним кодами;
- обнаружение ошибок внутренним и исправление стираний и ошибок внешним кодами;
- частичное исправление и обнаружение ошибок внутренним и исправление стираний внешним кодами;
- частичное исправление и обнаружение ошибок внутренним и исправление стираний и ошибок внешним кодами.

Наиболее прост в реализации второй алгоритм. Он требует минимальной избыточности, особенно в случае формирования стираний элементов по оценке надежности их приема, т. е. при применении косвенных методов повышения достоверности. Исправление стираний q-ичным кодом реализуется значительно проще, чем исправление ошибок, поскольку известно их местоположение. Возможности внешнего (N, K) кода Рида – Соломона с минимальным кодовым расстоянием $D = N - K + 1$ позволяют исправление всех стираний кратности до $\Delta = N - K$. Ошибочное декодирование блока при этом происходит в двух случаях: при необнаружении ошибок внутренним кодом и при обнаружении ошибок более чем в Δ комбинациях внутреннего кода. Применение ступенчатой процедуры декодирования каскадных кодов позволяет регулировать введение избыточности в передаваемое сообщение в зависимости от состояния канала связи использованием обратной связи.

3. ТУРБОКОДЫ

В основе современных методов корректирующего кодирования лежит принцип применения различных аспектов теории информации и передачи данных, методов принятия решений, математической алгоритмизации и автоматизированной машинной (компьютерной) обработки. В связи с этим процессы и процедуры кодирования носят сложный комплексный характер, эффективность исполнения которых зависит от множества факторов. Рассматриваемый в данном разделе метод помехоустойчивого кодирования, основанный на применении турбокодов, представляет собой синтезированный способ потенциальной защиты данных, реализующий множество современных парадигм формирования, обработки и передачи информации: параллелизацию вычислений, неравномерное кодирование с переменными кодовыми скоростями, интенсивное скремблирование потока данных, мягкое детектирование передаваемых сигналов, итеративный принцип декодирования и т. д.

Принципы формирования турбокодов в значительной степени зависят от выбора компонуемых составных кодов. В зависимости от свойств внутренних кодов, существенному изменению подвергаются также и принципы предварительной обработки кодируемой информации. В данном разделе будут рассмотрены основные принципы и свойства кодирования сверточных (систематических) турбокодов.

3.1. Основные принципы формирования параллельных сверточных турбокодов

Систематические сверточные коды, применяемые в качестве составных кодов турбокодера, характеризуются объединением конечного набора кодирующих устройств. При конструктивном объединении нескольких сверточных кодеров образуется мультикодирующая система, отличительным признаком которой является поточный метод кодирования и модель системы с множеством входов-выходов. Кратко рассмотрим параметры систематических сверточных кодов, оказывающие важное влияние на свойства формируемого составного сверточного турбокодера.

На рис. 3.1 представлена схема систематического сверточного кодера, характеризующегося степенью кодирования $r = 1/2$, памятью $M_c = 2$ и кодовым ограничением $K = (M_c + 1) = 3$. Для описания всех возможных состояний кодера ($S = M_c^2 = 4$) на рис. 3.2 представлена диаграмма состояний кодера (a) и кодовая решетка (b).

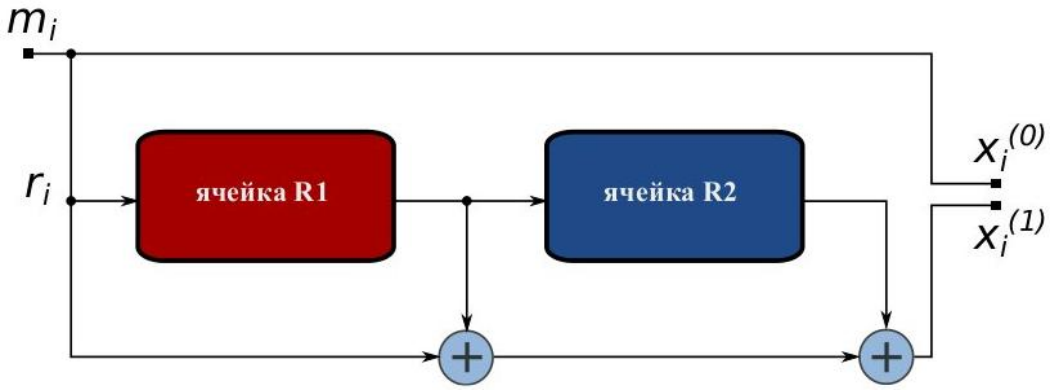

Рис. 3.1. Схема систематического сверточного кодера

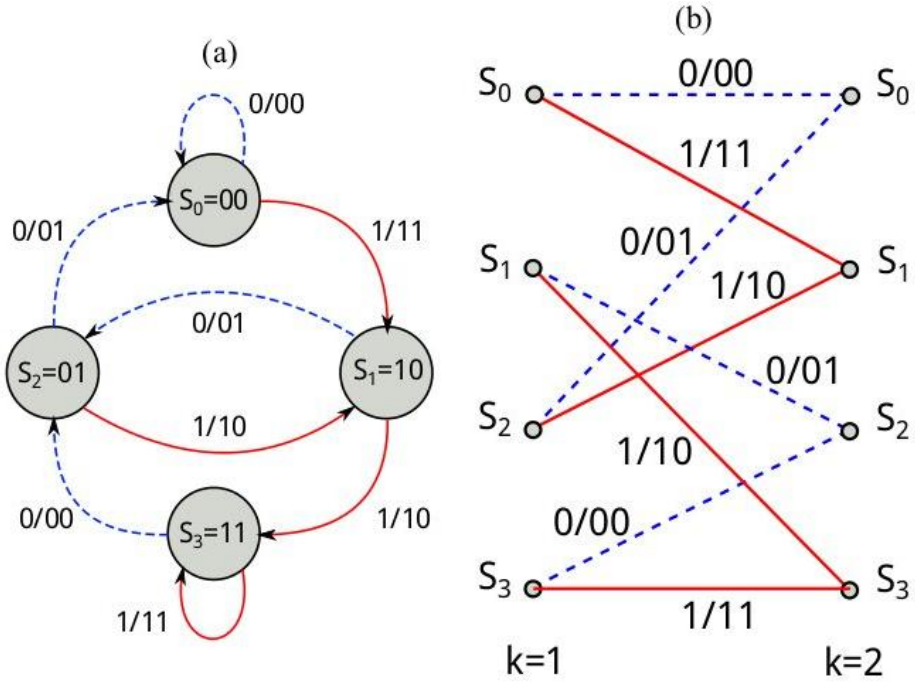

Рис. 3.2. Диаграмма состояний сверточного кодера (a) и кодовая решетка (b)

Структуру связей данного кодера можно представить в виде системы порождающих многочленов для первого ($g^{(0)}$) и второго ($g^{(1)}$) выходов соответственно:

$$g^{(0)} = 1;$$

$$g^{(1)} = 1 + x + x^2.$$

Выход каждой ветви, формируемый в результате выполнения процедуры кодирования, для сверточного кодера на рис. 3.1 представляет собой результат сложения значения входного бита m_i и содержимого ячеек памяти кодера в данном такте.

Как было показано ранее, систематические сверточные турбокоды конструируются по схеме параллельного построения кодирующего устройства. Параллелизация схемы кодирования преследует цели объединения кодовых комбинаций с каждой ветви кодера для увеличения весовых коэффициентов результирующей кодовой комбинации кодера. То есть параллельная схема систематического сверточного турбокодера позволяет снизить вероятность образования на выходе кодера кодовых комбинаций с малым весом.

На рис. 3.3 представлена структура сверточного турбокода. Составляющие кодеры обрабатывают информационные биты параллельно, однако кодирование во втором сверточном кодере предваряется процедурой перемежения информационной последовательности $m_i (i \in (0, \ldots, L-1)$, где L – длина блока). В задачи блока перемежения входит декорреляция и псевдослучайное размещение информационных символов.

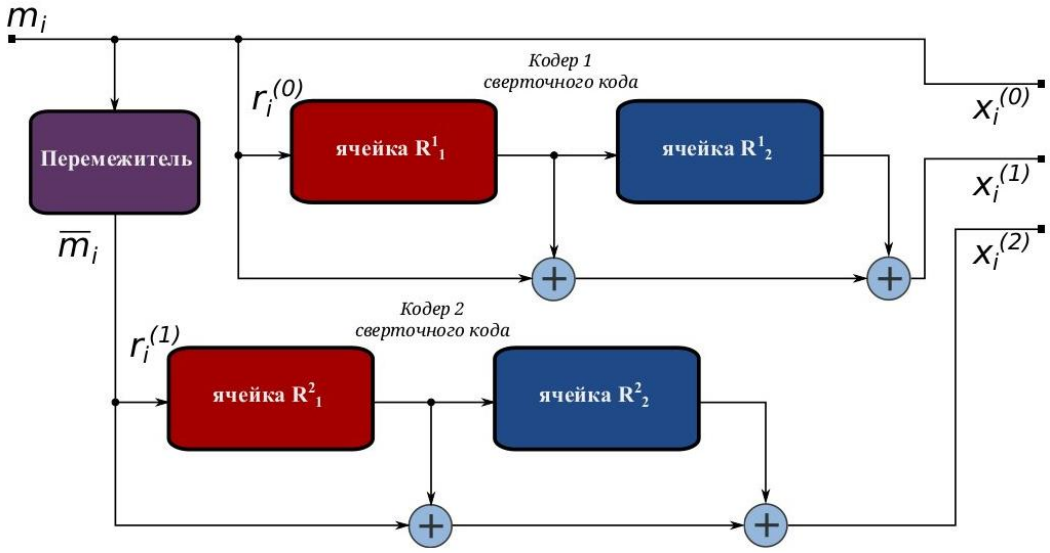

Рис. 3.3. Структура сверточного турбокодер

Составная схема турбокодера, изображенного на рис. 3.3, формализует требования к конкатенируемым кодерам. Все кодирующие блоки должны иметь одинаковую длину кодового ограничения и кодовую скорость. Кодовое слово кодера на рис. 3.3 состоит из информационного бита и двух проверочных бит и имеет вид $x = (x_0^{(0)}, x_0^{(1)}, x_0^{(2)}, \ldots, x_{L-1}^{(0)}, x_{L-1}^{(1)}, x_{L-1}^{(2)})$, что в результате дает кодовую скорость r = 1/3. Для случая большего числа составляющих (внутренних) кодеров турбокода результирующая кодовая скорость может быть найдена с помощью выражения

$$r = \frac{k}{n(N+1)}, \tag{3.1}$$

где N – количество составляющих кодеров, k и n соответственно параметры сверточного кодера.

Рассмотрим метод кодирования данных с помощью систематического сверточного турбокода на примере составного кодера рис. 3.3. Пусть исходная информационная последовательность представляет собой вектор длиной $L = 4$ бита [1 1 0 0]. Условимся, что кодирование начинается в нулевом состоянии, т. е. оба кодера обнулены. Для формирования входной последовательности на втором кодере используется перемежитель размерностью 2 × 2, который читает информационный блок по 2 бита по строкам, а выдает кодеру по столбцам. То есть, содержимое матрицы перемежителя имеет вид:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Следовательно, входные биты, подаваемые на второй кодер, представляют собой вектор [1 0 1 0]. Руководствуясь диаграммой состояний рис. 3.2, а для каждого из составных кодеров представим результаты кодирования в табл. 3.1.

Таблица 3.1

Процесс кодирования

m_i	R_1^1	R_2^1	x_i^0	x_i^1
–	0	0	–	–
$m_{0=1}$	1	0	1	1
$m_{1=1}$	1	1	1	0
$m_{2=0}$	0	1	0	0
$m_{3=0}$	0	0	0	1

Поскольку систематический выход x_i^0 второго (внутреннего) составного сверточного кодера формируется из перемеженной последовательности и может быть восстановлен на принимающей стороне исходя из систематических бит первого кодера (внешнего), биты x_i^0, сформированные вторым кодером, в канал не передаются. Результирующая выходная кодовая комбинация будет иметь вид

$$x = (x_0^{(0)}, x_0^{(1)}, x_0^{(2)}, \ldots, x_3^{(0)}, x_3^{(1)}, x_3^{(2)}) : [1\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1].$$

3.2. Итеративная обработка и декодирование сверточных турбокодов

Распределение вероятностей при передаче сигналов по каналу с аддитивным белым гауссовым шумом (АБГШ) можно представить в виде функций правдоподобия, изображенных на рис. 3.4. Функция $p(x|d = 0)$

выражает распределение случайной непрерывной переменной x, при условии передачи бита данных $d = 0$. Соответственно, функция $p(x|d = 1)$ – распределение x при передаче бита $d = 1$.

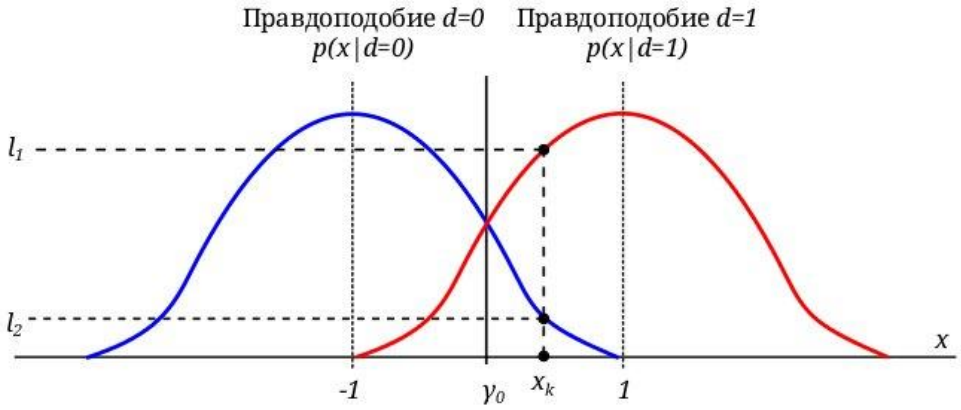

Рис. 3.4. Функция распределения вероятностей принятого сигнала

При детектировании значения сигнала x_k (рис. 3.4), в момент времени k, необходимо определить значение функции правдоподобия для полученного x_k. Расположение x_k вдоль оси x, не позволяет однозначно определить значение передаваемого бита, поскольку отмеченной точке x_k соответствуют два значения правдоподобия – l_1 для $d = 1$ и l_2 для $d = 0$.

Основываясь на принципах жесткого решения и максимального правдоподобия, приемник должен сравнить значения l_1 и l_2 и вынести решение о принятом бите: $d|_{l_1>l_2} = 1$ или $d|_{l_2>l_1} = 0$. Таким образом, в жесткой схеме принятия решения при расположении переменной x_k правее порога γ_0, принятый сигнал переносит бит $d = 1$, а при расположении x_k левее γ_0, сигнал переносит бит $d = 0$.

При декодировании турбокодов используется алгоритм максимума апостериорной вероятности (MAP – *Maximum Aposteriory Probability*), исполняющий правило минимизации вероятности ошибки и учитывающий значение априорной вероятности данных. Принцип расчета апостериорной вероятности основан на обновлении данных, полученных при детектировании исходного сигнала. Выбор значения переданного бита осуществляется согласно неравенству наибольшей апостериорной вероятности:

$$\begin{cases} d = 1 : P\left(d = 1|x\right) > P\left(d = 0 \mid x\right), \\ d = 0 : P\left(d = 1|x\right) < P\left(d = 0 \mid x\right), \end{cases} \tag{3.2}$$

где $P(d|x)$ – апостериорная вероятность принятого сигнала. Выражение (3.2) можно переписать, используя одну из форм теоремы Байеса и априорную вероятность появления сигнала:

$$\begin{cases} d=1:\ p\left(x\,|\,d=1\right)P(d=1) > p\left(x\,|\,d=0\right)P(d=0), \\ d=0:\ p\left(x|d=1\right)P\left(d=1\right) < p\left(x\,|\,d=0\right)P(d=0). \end{cases} \tag{3.3}$$

Однако на практике для решения задач декодирования используется отношение функций правдоподобия, выраженное в логарифмической форме и ведущее к сокращению машинных вычислений. Сама функция отношения правдоподобий представляет собой дробь от (3.3):

$$\begin{cases} d=1: \dfrac{p\left(x|d=1\right)P\left(d=1\right)}{p(x\,|\,d=0)P(d=0)} > 1, \\ d=0: \dfrac{p\left(x|d=1\right)P\left(d=1\right)}{p(x\,|\,d=0)P(d=0)} < 1. \end{cases} \tag{3.4}$$

Логарифмическое отношение функций правдоподобия (LLR – *Log-Likelyhood Ratio*):

$$L\left(d|x\right) = \ln\left(\frac{p(x\,|\,d=1)}{p\left(x|d\right)=0)}\right) + \ln\left(\frac{P(d=1)}{P(d=0)}\right). \tag{3.5}$$

Первое слагаемое суммы в выражении (3.5) представляет собой логарифмическое отношение функций правдоподобия (LLR) тестовой статистики x, полученное путем измерений x на выходе канала передачи данных при попеременной передаче бит $d = 0$ и $d = 1$. Второе слагаемое представляет собой априорное логарифмическое отношение функций правдоподобия бита данных d. В укороченной форме сумма (3.5) принимает вид:

$$L\left(d|x\right) = L\left(x|d\right) + L(d)\,. \tag{3.6}$$

Выражение (3.6) характеризует данные, полученные детектором сигнала. Далее, оценка детектированных данных должна поступать на декодер помехоустойчивого кода, работающий по алгоритму мягкого декодирования. Мягкий выход декодера формируется из суммы логарифмического отношения функций правдоподобия бита данных d на выходе детектора $L'\left(\hat{d}\right)$ и внешней информации $L_e\left(\hat{d}\right)$, получаемой в результате процедур декодирования:

$$L(\hat{d}) = L'\left(\hat{d}\right) + L_e\left(\hat{d}\right). \tag{3.7}$$

Заключительное выражение для расчета значений мягкого выхода декодера с учетом всей доступной информации до декодирования и собранной информацией в процессе декодирования имеет вид:

$$L\left(\hat{d}\right) = L_c\left(x\right) + L(d) + L_e\left(\hat{d}\right), \tag{3.8}$$

где $L_c(x)$ – канальное измерение величины x, полученное приемником, $L(d)$ – априорное значение данных, имеющееся до декодирования и $L_e\left(\hat{d}\right)$, внешнее значение LLR, получаемое в результате декодирования. Следует отметить, что все три компоненты суммы выражения (3.8) являются статистически независимыми, а операция суммирования определяет вклад каждого значения в заключительное решение. Мягкий выход декодера $L\left(\hat{d}\right)$ является вещественным числом, обеспечивающим в итоге надежность принятия жесткого решения, причем знак логарифма определяет область жесткого решения, а величина значения $L\left(\hat{d}\right)$ устанавливает достоверность данного решения.

При декодировании турбокодов используется некоторое заданное количество итераций обработки данных (Q). В задачу каждой итерации входит получение априорных данных о декодируемой информации, вынесение мягких решений и передача оценки результата декодирования на последующую итерацию. Подобная циклическая схема порождает турбообработку данных, в том смысле, что результаты предыдущей итерации декодирования являются внешней априорной информацией для следующего шага декодирования. За счет цикличной, многоитерационной обработки кодовых слов становится возможным увеличить достоверность обрабатываемой информации и уменьшить вероятность ошибочного декодирования.

На рис. 3.5 изображен принцип итерационной обработки данных турбодекодером с мягкими входами и мягкими выходами (SISO – *Soft Inputs, Soft Outputs*). На первой итерации декодирования, исходя из канальных измерений детектора, формируется массив значений канальных оценок принятых символов $L_c(x_k)$, вычисленных из логарифмического отношения значений l_1 и l_2 для рассматриваемого x_k (рис. 3.4). При инициализации первой итерации априорная информация о декодируемом символе отсутствует, так как декодер 2 не производил декодирования. Таким образом, априорная информация на входе декодера 1 $L_e^2\left(d_k\right)=0.$

Входы $x_k^{\,1}$ и $y_k^{\,1}$ представляют собой детектированные значения сигнала информационного и проверочного бита кодера 1 (рис. 3.3) сформированные по методу мягких оценок. Информационный вход первого декодера $x_k^{\,1}$ также передается на перемежитель для формирования перемеженной последовательности, которая будет использована в качестве информационного входа кодера 2. После выполнения декодирования на выходе декодера 1 может быть сформирован мягкий выход из суммы значений логарифма отношения правдоподобий детектора $L_c(x_k)$, априорной инфор-

мации $L(d_k)$ и собственной информации декодера 1 $L_e^1(d_k)$. Так как на первой итерации $L(d_k) = L_e^2(d_k) = 0$, выход декодера 1 представляет собой только сумму логарифмов канальных оценок и результатов декодирования данного декодера $L_e^1(d_k)$.

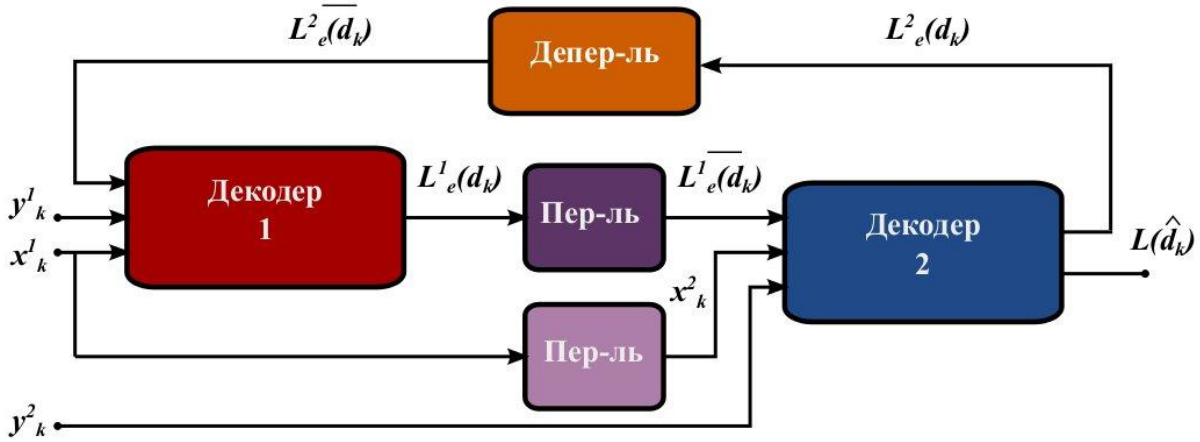

Рис. 3.5. Итерационная обработка данных в формате мягких решений

Условимся, что мягких решений, полученных декодером 1, недостаточно, чтобы вынести решение о декодируемом символе с заданным уровнем надежности. Продолжим декодирование на декодере 2 и используем проверочные биты кодера 2, а также ставшую теперь доступной априорную информацию $L(d_k)$, полученную в ходе декодирования первого набора кодовых слов, т. е. для декодера 2 $L(d_k) = L_e^1(d_k)$. Следует отметить, что результаты декодирования декодера 1 $L_e^1(d_k)$ необходимо подать на перемежитель для правильного соответствия с перемеженными битами систематического входа второго кодера. Таким образом, вход декодера 2 включает в себя перемеженную информационную последовательность x_k^2, проверочные биты второго кодера y_k^2 и априорную информацию $\hat{L}_e^1(d_k)$ декодера 1 для декодируемых x_k^2. В результате декодирования кодовых комбинаций второго кодера, декодер 2 формирует собственные мягкие оценки в виде суммы канальной оценки декодируемого бита $L_c(x_k)$, априорной информации от декодера 1 $L(d_k) = L_e^1(d_k)$, и собственных вычислений $L_e^2(d_k)$.

Используя суммарную оценку, декодер 2 может рассчитать логарифмическое отношение правдоподобия (LLR) для каждого декодируемого бита и, учитывая величину и знак отношения, вынести окончательное жесткое решение.

Если необходимо выполнить заданное количество итераций декодирования, то результаты декодирования декодера 2 $L_e^2(d_k)$ сохраняются и используются в качестве априорной информации на следующей итерации декодирования. Для переупорядочивания порядка следования оценок декодируемых бит значения $L_e^2(d_k)$ поступают на деперемежитель и присваиваются значениям априорной информации для декодера 1. То есть на начало второй итерации декодирования декодер 1 обладает канальными оценками декодируемых символов $L_c(x_k)$ и априорной информацией с декодера 2 (по результатам первой итерации), т. е. $L(d_k) = L_e^2(d_k)$. После данной процедуры начинается очередная итерация декодирования. На последующих итерациях происходит уточнение информации и, наконец, на последней заданной итерации декодирования выносится окончательное жесткое решение. Как уже было сказано ранее, мягкий выход декодера представляет собой вещественное число, указывающее на достоверность принятия жесткого решения, а знак LLR определяет область жесткого решения. Необходимо подчеркнуть, что основная парадигма итеративного декодирования заключается в непрерывном обмене результатами декодирования между составляющими декодерами. Результаты декодирования по завершению второй полной итерации представляют собой уточненные оценки, скорректированные за счет наличия априорных данных, полученных с первой итерации и т. д.

Благодаря «разгонному» циклу декодирования, на каждом из этапов которого происходит уточнение (корректировка) значений декодируемых символов, рассматриваемый метод помехоустойчивого кодирования приобрел название «турбодекодирования», поскольку алгоритмика и цели турбодекодирования во многом схожи с процессами, происходящими в блоке турбореактивного двигателя.

Как указано ранее, применение сверточных турбокодов и их декодирование проводится по принципам минимизации вероятности ошибки и учета значений априорной вероятности данных. Благодаря формированию каждым составным декодером собственных апостериорных оценок (вероятностей) значений информационных бит осуществляется многопороговая схема принятия решения о наиболее вероятном значении декодируемого символа, основанная на вычислении ряда метрик вероятностей перехода по решетчатой диаграмме сверточного кода.

Кратко рассмотрим процедуры вероятностного декодирования сверточных турбокодов и особое внимание уделим численному примеру декодирования, позволяющему наглядно продемонстрировать эффективность метода итеративного мягкого декодирования составных кодов.

3.3. Декодирование турбокодов по алгоритму максимума апостериорной вероятности MAP

Необходимо отметить, что в методе расчета и определения метрик алгоритма MAP широко используется принцип прямой и обратной рекурсии. Это вызывает некоторое осложнение понимания теории и методов вероятностного декодирования турбокода, однако позволяет достаточно эффективно строить и реализовывать алгоритмы для применения машинных и компьютерных вычислений. В основу главной задачи, решаемой с помощью алгоритма MAP, входит вычисление апостериорных вероятностей всех возможных переходов принятого кодового символа по решетке сверточного декодера. В соответствии с выражением (3.5), расчет начинается с логарифмического отношения функций правдоподобия апостериорных вероятностей того, что декодируемый бит равен $d_k = 1$ или $d_k = 0$:

$$L(d_k) = \ln\left(\frac{\sum_m p_k^{1,m}}{\sum_m p_k^{0,m}}\right), \tag{3.9}$$

где $p_k^{i,m}$ – вероятность того, что $d_k = [0 : 1]$ в состоянии кодера $S_k = m$ при принятой кодовой последовательности C. Применяя теорему Байеса можно представить $p_k^{i,m}$ в полном виде:

$$p_k^{i,m} = \frac{P\left(C_1^{k-1}|_{d_k=i,S_k=m,C_k^N}\right)P\left(C_{k+1}^N|_{d_k=i,S_k=m,C_k,}\right)P\left(d_k=i,S_k=m,C_k\right)}{P\left(R_1^N\right)}, \tag{3.10}$$

где C_1^N – значение принятой последовательности бит $\left(C_k^N = \left(C_k, C_{k+1}^N\right)\right)$. Каждый из членов совокупного вероятностного уравнения (3.10) представляет собой метрику состояний прямого $\left(\alpha_k^m\right)$, обратного прохода $\left(\beta_k^m\right)$ по кодовой решетке и метрику ветвей $\left(\delta_k^{i,m}\right)$ кодовой решетки сверточного кода. Области значения метрик MAP кодовой решетки изображены на рис. 3.6.

Используя выражение логарифмического отношения функций правдоподобия, уравнение для вычисления мягкого выхода на итерации декодирования имеет вид:

$$L_e(d_k) = \ln\left(\frac{\sum_m \alpha_k^m \delta_k^{1,m} \beta_{k+1}^{(1,m)}}{\sum_m \alpha_k^m \delta_k^{0,m} \beta_{k+1}^{(1,m)}}\right), \tag{3.11}$$

где делимое представляет собой функцию правдоподобия бита $d_k = 1$, а делитель функцию правдоподобия бита $d_k = 0$. Как было сказано ранее,

мягкий выход представляет собой вещественное число, устанавливающее надежность данного решения, а знак логарифма определяет область принятия жесткого решения.

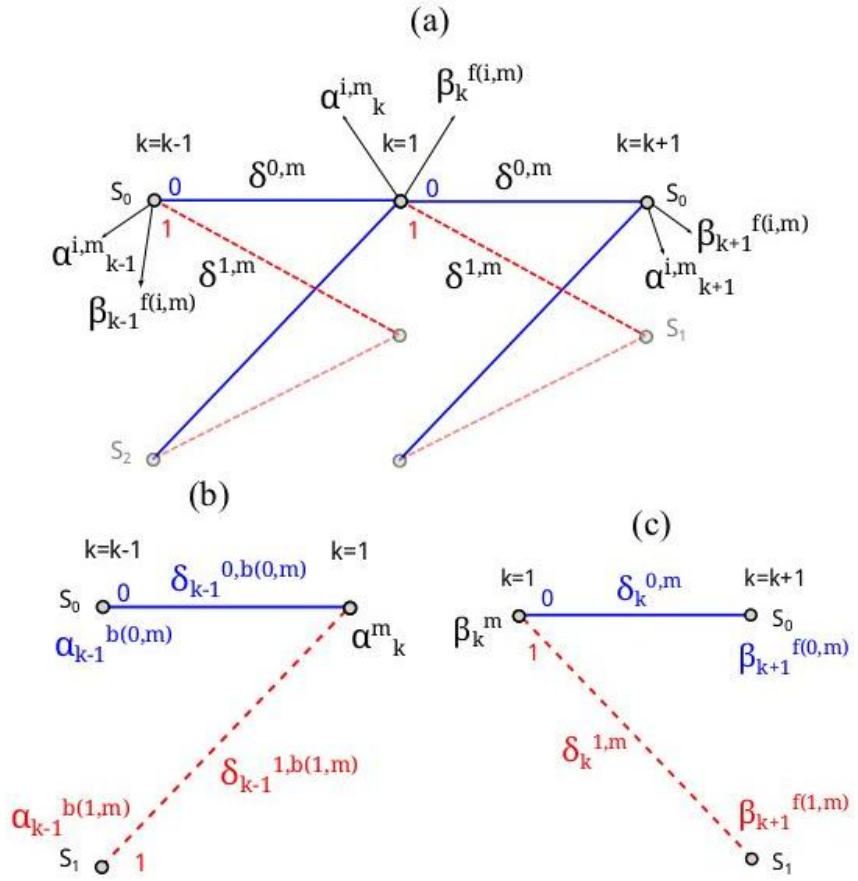

Рис. 3.6. Фрагменты решетки
для расчета метрик состояний и ветвей

Рассмотрим применение методов мягкого итеративного декодирования с использованием алгоритма MAP на примере сверточного турбодекодера. Пусть в качестве исходной информационной последовательности выступает двоичный вектор [1 1 0 0], используемый в примере кодирования (подразд. 3.2). Тогда результирующая выходная кодовая комбинация имеет вид: [1 1 1 1 0 1 0 0 0 0 1 1].

Для упрощения понимания примера условимся, что передача информации между приемником и передатчиком осуществляется посредством биполярных импульсов [+1; −1], потенциально кодирующих биты 1 и 0 соответственно. Сформированная модулятором передатчика последовательность принимает вид: [+1 +1 +1 +1 −1 +1 −1 −1 −1 −1 +1 +1].

Выделенный приемником сигнал $s(t)$ является суммой исходного дискретного сообщения $s(t)$ и помехи в виде АБГШ $n(t)$:

$$r(t) = s_i(t) + n(t). \qquad (3.12)$$

Шум канала передачи данных имеет нулевое среднее и дисперсию $\sigma^2 = 1$. Априорные вероятности принятия бита, равного 1 или 0, одинаковы и равны $\pi^1 = \pi^1 = 1/2$.

Пусть в результате воздействия помехи исходный биполярный сигнал примет значения: [1,2 0,9 0,7 1,1 0,5 1,5 0,3 – 0,2 – 0,8 – 0,6 0,5 0,9]. Перед непосредственным декодированием необходимо определить принадлежность каждого бита принятой кодовой последовательности к составному кодеру сверточного турбокода. Исходя из общей кодовой скорости рассматриваемого в примере кода $r = 1/3$, кодовое слово состоит из одного систематического бита (информационного) и двух проверочных бит, с первого и второго кодера соответственно. В табл. 3.2 представлена принадлежность бит к кодерам из принятого сигнального вектора. При параллельной схеме декодирования вначале будут декодироваться систематические и проверочные биты первого кодера, а затем систематические и проверочные биты второго кодера и т. д.

Таблица 3.2

Порядок следования значений кодовых слов составного турбокодера

Назначение	Уровень			
Информационный бит (x_k^1)	1,2	1,1	0,3	–0,6
Проверочный бит (x_k^1)	0,9	0,5	–0,2	0,5
Проверочный бит (x_k^2)	0,7	1,5	–0,8	0,9

В соответствии с алгоритмом проведения декодирования MAP, процесс декодирования начинается с расчета первого сегмента ($k = 1$) кодовой решетки декодера. Исходя из условий, рассчитаем метрику ветвей $\delta_k^{i,m}$ на каждом участке k кодовой решетки:

$$\delta_k^{i,m} = \frac{1}{2} e^{x_k u_k^i + y_k v_k^{i,m}}, \tag{3.13}$$

где x_k и y_k представляют собой канальные значения информационного и соответствующего ему проверочного бита, а u_k^i и $v_k^{i,m}$ обозначают уровни выходного сигнала, формируемые кодером, исходя из значений выходных бит ([0 : −1] и [1 : +1]).

Основываясь на решетке переходов кодера, метрики ветвей для первого перехода равны:

$$\begin{aligned} \delta_{k=1}^{0,m=S_0} &= \frac{1}{2} \mathrm{e}^{\left[1,2(-1)+0,9(-1)\right]} = 0,061; \\ \delta_{k=1}^{1,m=S_0} &= \frac{1}{2} \mathrm{e}^{\left[1,2(+1)+0,9(+1)\right]} = 4,083; \end{aligned} \tag{3.14}$$

$$\delta_{k=1}^{0,m=S_1} = \frac{1}{2}\mathrm{e}^{\left[1{,}2(-1)+0{,}9(+1)\right]} = 0{,}370;$$

$$\delta_{k=1}^{1,m=S_1} = \frac{1}{2}\mathrm{e}^{\left[1{,}2(+1)+0{,}9(-1)\right]} = 0{,}674;$$

$$\delta_{k=1}^{0,m=S_2} = \frac{1}{2}\mathrm{e}^{\left[1{,}2(-1)+0{,}9(+1)\right]} = 0{,}370; \qquad (3.14)$$

$$\delta_{k=1}^{1,m=S_2} = \frac{1}{2}\mathrm{e}^{\left[1{,}2(+1)+0{,}9(-1)\right]} = 0{,}674;$$

$$\delta_{k=1}^{1,m=S_3} = \frac{1}{2}\mathrm{e}^{\left[1{,}2(+1)+0{,}9(+1)\right]} = 4{,}083.$$

Полученные значения изображены на первом переходе ($k = 1$) кодовой решетки на рис. 3.7, а. Аналогичным образом рассчитываются метрики ветвей для последующих переходов $k = 2 \ldots 5$. Рассчитанные значения метрик ветвей декодера для каждого перехода обозначены на соответствующих переходах рис. 3.7.

На следующем этапе алгоритма декодирования MAP необходимо произвести расчет метрик прямого прохода по решетке декодера. Выражение для расчета метрики прямого прохода имеет вид:

$$\alpha_{k+1}^{m} = \sum_{j=0}^{1} \delta_k^{j,b(j,m)} \alpha_k^{b(j,m)} \qquad (3.15)$$

Вследствие того, что кодирование информационной последовательности начинается в исходном S_0 (обнуленном состоянии), метрика прямого прохода на первом шаге ($k = 1$) для S_0 равна 1, а для всех остальных состояний нулю. Далее для расчета метрики прямого прохода в последующих состояниях необходимо учитывать значение метрик ветвей, входящих в рассматриваемый узел решетки, и значение метрики прямого прохода на предыдущем шаге. Основываясь на выражении (3.15), решетке декодера и значениях метрик ветвей $\delta_k^{i,m}$, полученных на предыдущем этапе декодирования, метрики прямого прохода рассчитаны и нанесены на решетку рис. 3.7.

Так как метрики прямого прохода для состояний S_1, S_2 и S_3 ($k = 1$) равны нулю, то для момента времени $k = 2$ имеют значения только ветви, исходящие из состояния S_0 ($k = 1$), поскольку остальные члены суммы будут равны нулю. Аналогично для момента времени $k = 3$ метрики прямого прохода учитывают значения метрик ветвей, входящих в заданный узел, и значения метрик прямого прохода для $k = 2$. На рис. 3.8 изображена решетка декодера и рассчитанные значения метрик прямого прохода для каждого узла решетки.

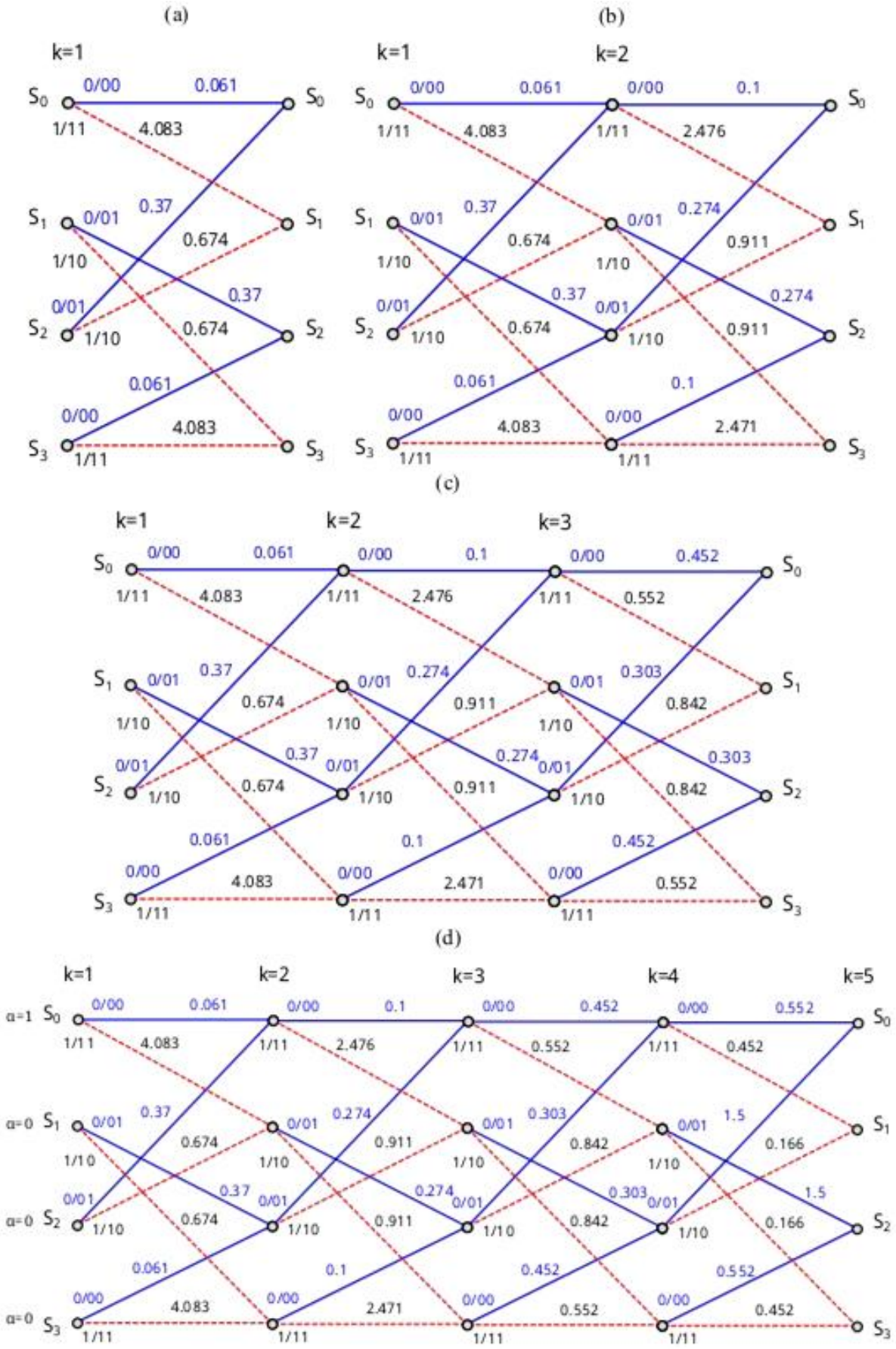

Рис. 3.7. Метрики ветвей решетки сверточного декодера D_1

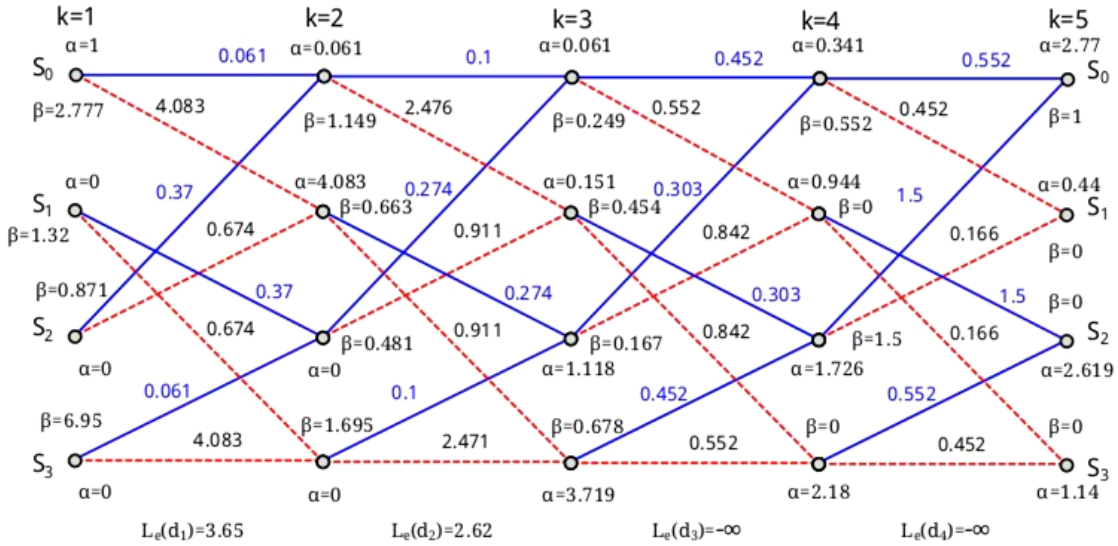

Рис. 3.8. Метрики прямого и обратного прохода декодера D_1

Используя значения метрик прямого прохода α_k^m, становится возможным вычислить метрики обратного прохода β_k^m по решетке декодера:

$$\beta_k^m = \sum_{j=0}^{1} \delta_k^{j,m} \beta_{k+1}^{f(j,m)}. \qquad (3.16)$$

Процесс вычисления метрик обратного прохода начинается с конца (k = 5) кодовой решетки (рис. 3.8). Согласно диаграмме состояний (рис. 3.2) первого кодера составного турбокодера, изображенного на рис. 3.3, процесс кодирования исходной последовательности завершается в обнуленном состоянии кодера. Исходя из данного обстоятельства, метрика обратного прохода $\beta_{k=5}^{m=S_0}$ в момент времени k = 5 для состояния S_0 равна 1, а для всех остальных состояний кодовой решетки – нулю. Необходимо подчеркнуть, что, хотя в данном примере не использовалось принудительное завершение (обнуление) кодеров составного турбокодера, подобранный в примере информационный вектор привел к обнулению внешнего кодера. Данное положение несколько упрощает декодирование для внешнего декодера. Однако внутренний кодер не перешел в нулевое состояние по окончанию процедур кодирования, что необходимо учитывать далее при декодировании кодовых слов внутреннего кодера. На рис. 3.8 изображена решетка декодера с рассчитанными значениями метрик обратного прохода для каждого узла решетки.

Обладая значениями трех метрик для каждого состояния решетки декодера, можно произвести расчет логарифмического отношения функций правдоподобия (d_k) по декодируемым символам, согласно выражению (3.11). Рассчитаем отношение правдоподобия для первого участка кодовой решет-

ки $k = 1$, учитывая значения метрик ветвей, метрик прямого прохода для данных узлов и значения метрик обратного направления для узлов $k = 1$:

$$L_e(d_1) = 3{,}65;$$
$$L_e(d_2) = 2{,}62;$$
$$L_e(d_3) = -\infty;$$
$$L_e(d_4) = -\infty.$$

Полученные значения отмечены на соответствующих участках кодовой решетки (рис. 3.8). Необходимо отметить, что при машинном вычислении, наличие бесконечных величин, полученных для бит d_3 и d_4, в принципе не требует дальнейшей обработки, поскольку задает максимально вместимое отрицательное число системы, на которой проводится вычисление. При машинном компьютерном расчете, например в распространенных прикладных пакетах инженерных систем расчета, значение $-\infty$ представляет собой число $-9 \cdot 10^{-19}$, которое в данном случае однозначно определяет результат декодирования. Однако в целях демонстрации и анализа методов итеративного декодирования продолжим процедуры обработки для увеличения надежности результатов декодирования для бит d_1 и d_2.

Вновь обратимся к схеме декодера на рис. 3.5. На данном этапе декодирование кодовых слов внешнего декодера D_1 завершено, и результаты декодирования можно передать на внутренний декодер D_2 в формате априорной информации. Следует особенно подчеркнуть необходимость перемежения априорной информации для соответствия исходной (перемеженной) информационной последовательности внутреннего кодера D_2.

Алгоритм вычисления метрик ветвей декодера D_2, полностью аналогичен рассмотренному выше методу расчета метрик ветвей декодера D_1. Результаты отмечены на соответствующих переходах кодовой решетки на рис. 3.9.

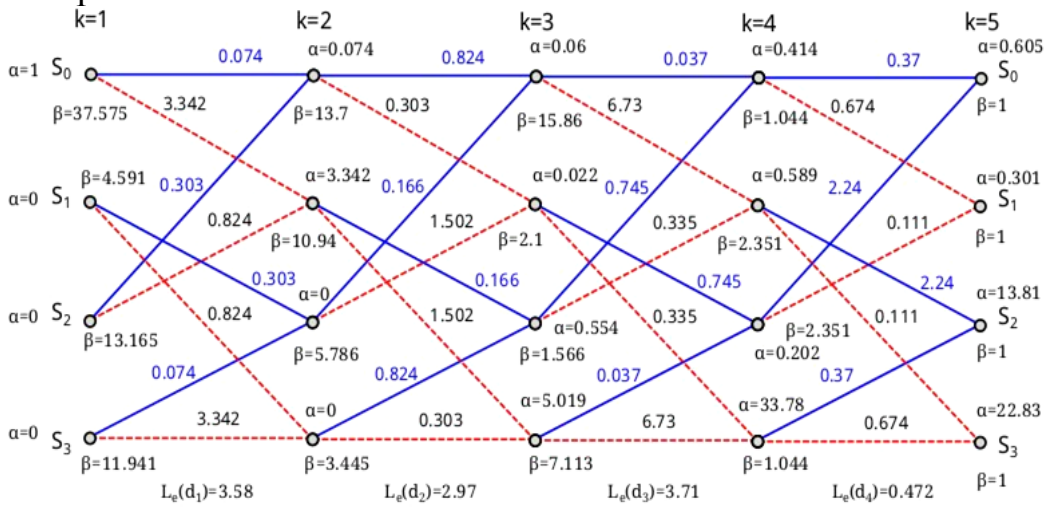

Рис. 3.9. Метрики ветвей и состояний декодера D_2

Используя массив метрик ветвей, как и в предыдущем случае для декодера D_1, вычисляем метрики состояний прямого прохода по кодовой решетке. Значения метрик состояний при прямом проходе для декодера D_2 отмечены на узлах кодовой решетки.

Как было сказано ранее, при кодировании информационной последовательности не происходило принудительного обнуления составных кодеров. В связи с этим приемник не имеет возможности вычислить точное значение обратной метрики состояний для декодера D_2 в конечный момент времени (k = 5). Исходя из данного условия, предполагается, что все конечные состояния кодера равновероятны, т. е. с одинаковой вероятностью кодер может находиться в любом из 2^{Mc} состояний. В зависимости от реализаций обратная метрика для конечного момента времени (в данном случае k = 5) может быть приравнена к прямой метрике состояний в конечный момент времени, т. е. $\beta_N^m = \alpha_n^m$. Однако существует и более простой способ определения обратной метрики состояний, который при равновероятности состояний кодера определяет обратную метрику для конечного момента времени, равной 1, т. е. $\beta_N^m = 1$.

Необходимо отметить, что оба указанных метода вносят определенную погрешность в результат декодирования, что требует разработки быстрого общего обнуления внутренних кодеров составных турбокодеров. При условии, что оба кодера турбокодера были бы обнулены и закончили кодирование в состоянии S_0, определение значений метрик состояний при обратном проходе по кодовой решетке не составило бы труда, поскольку декодеру априори известно конечное состояние (S_0) любого составного кодера (D_1, D_2). Используя предположение, что метрики состояний для k = 5 при обратном проходе по кодовой решетке равны 1, результаты расчета для всей решетки отмечены на рис. 3.9.

Наконец, рассчитаем отношение правдоподобия для всех участков кодовой решетки, учитывая значения метрик ветвей, метрик прямого прохода для данных узлов и значения метрик обратного направления для узлов k = 1:

$$\begin{aligned} &L_e(d_1) = 3{,}58;\ \ L_e(d_2) = 2{,}97; \\ &L_e(d_3) = 3{,}77;\ \ L_e(d_4) = 0{,}472. \end{aligned} \tag{3.17}$$

После декодирования кодовых слов декодера D_2 существует два сценария развития дальнейших действий. В первом случае полученные приращения надежности $L_e(d_k)$ могут быть переданы далее на новую итерацию декодирования внешнему декодеру в виде априорной информации $L(d_i)$. Будет повторен весь рассмотренный цикл декодирования для обоих декодеров. Наконец, по завершению заданного количества (Q) итераций декодирования, декодеру турбокода необходимо вынести окончательное жесткое решение.

Предположим, что завершение декодирования наступает в конце первой итерации ($Q = 1$), после декодирования кодовых слов декодерами D_1 и D_2. В этом случае для расчета мягкого выхода итерации декодирования необходимо воспользоваться заключительным выражением (3.8). Для вычисления мягкого выхода турбодекодера следует учесть вклад априорной информации от внешнего декодера D_1 и вклад собственных результатов декодирования декодера D_2:

$$\begin{aligned} &L\left(\hat{d}_1\right) = 9,63; \quad L\left(\hat{d}_2\right) = 9,43; \\ &L\left(\hat{d}_3\right) = -\infty; \quad L\left(\hat{d}_4\right) = -\infty. \end{aligned} \tag{3.18}$$

Основываясь на принципе принятия жестких решений, согласно выражению (3.4), результат декодирования, принятого кодового вектора после выполнения полной первой итерации декодирования, равен [1 1 0 0] и полностью совпадает с исходным информационным вектором, введенным в кодер, что свидетельствует об отсутствии ошибок декодирования.

Рассмотренный способ вероятностного декодирования сверточного турбокода, основанный на алгоритме MAP, помимо достоинств обладает одним существенным недостатком в случае его использовании для большого числа декодируемых кодовых векторов. С увеличением количества декодируемой информации возрастает число операций умножения и деления над метриками кодовой решетки для каждого бита. Среднее количество операций умножения для вычисления значений метрик состояний для обоих направлений прохода и метрик ветвей зависит от размеров памяти кодера и с увеличением ее объема также возрастает. В связи с конвейерной обработкой данных также возрастает и время декодирования, что может оказать существенное влияние на быстродействие телекоммуникационного приложения в целом. С другой стороны, для снижения вычислительной нагрузки на систему обработки и передачи данных при декодировании сверточного турбокода могут быть использованы методы приближенного вычисления метрик решетки кодера, позволяющие ускорить процесс вычисления (декодирования) за счет частичного снижения точности результата декодирования.

Список литературы

1. *Хагельбергер, Д. В*. Рекуррентные коды, легко реализуемые и исправляющие пачки ошибок / Д. В. Хагельбергер // Bell System Technical Journal. – 1959. – Vol. 37. – № 46.

2. *Бородин, Л. Ф.* Введение в теорию помехоустойчивого кодирования / Л. Ф. Бородин. – М. : Советское радио, 1968. – 408 с.

3. *Возенкрафт, Дж*. Последовательное декодирование / Дж. Возенкрафт, Б. Рейффен. – М. : ИЛ, 1963.

4. *Возенкрафт, Дж.* Теоретические основы техники связи / Дж. Возенкрафт, И. Джекобс ; пер. с англ. – М. : Мир, 1969. – 640 с.

5. *Viterbi, A. J*. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm / A. J. Viterbi. – IEEE Trans. Inform. Teory. – 1967. – Vol.IT-13. – P. 260–269.

6. *Витерби, А. Д*. Принципы цифровой связи и кодирования / А. Д. Витерби, Дж. Омура ; пер. с англ. – М. : Радио и связь, 1982. – 536 с.

7. *Кларк, Дж*. Кодирование с исправлением ошибок в системах цифровой связи / Дж. Кларк, Дж. Кейн ; пер. с англ. – М. : Радио и связь, 1987. – 391 с.

8. *Морелос-Сарагоса, Р*. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Р. Морелос-Сарагоса ; пер. с англ. – М. : Техносфера, 2006. – 319 с.

9. *Мэзон, С.* Электронные цепи, сигналы и системы / С. Мэзон. – М. : Изд-во ИЛ, 1963.

10. А. с. № 145911 ; Кл. 21а, 22/04 с приоритетом от 14.02.1955. Способ повышения устойчивости радиотелеграфной связи по двухканальным радиолиниям (ДЧТ) / В. И. Шляпоберский, Л. М. Финк.

11. *Шляпоберский, В. И.* Основы техники передачи дискретных сообщений / В. И. Шляпоберский – М. : Связь, 1973. – 480 с.

12. *Вернер, М*. Основы кодирования : учеб. пособие для вузов : Сер. «Мир программирования» / Вернер, М. ; пер. с нем. Д. К. Зигангирова. – М. : Техносфера, 2004. – 288 с.

13. *Варгаузин, В. А*. Методы повышения энергетической и спектральной эффективности цифровой радиосвязи / В. А. Варгаузин, И. А. Цикин. – СПб. : БХВ-Петербург, 2013. – 352 с.

14. *Шлома, А. М*. Новые алгоритмы формирования и обработки сигналов в системах подвижной связи / А. М. Шлома, М. Г. Бакулин ; под ред. профессора А. М. Шломы. – М. : Горячая линия-Телеком, 2008. – 344 с.

15. *Небаев, И. А*. Компьютерное моделирование системы кодирования параллельным сверточным турбокодом для повышения уровня достоверности передачи данных в непрерывном канале/ И. А. Небаев // Вестник компьютерных и информационных технологий. – 2013. – № 8. – С. 41–45.

Когновицкий Олег Станиславович
Охорзин Виктор Михайлович
Небаев Игорь Алексеевич

ТЕОРИЯ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

Часть 2

СВЕРТОЧНЫЕ КОДЫ. ТУРБОКОДЫ

Учебное пособие

Редактор *С. Д. Щербакова*
Компьютерная верстка *Н. А. Ефремовой*

План издания 2015 г., п. 25

Подписано к печати 03.11.2015
Объем 4,0 усл.-печ. л. Тираж 30 экз. Заказ 587
Редакционно-издательский отдел СПбГУТ
191186 СПб., наб. р. Мойки, 61

Отпечатано в СПбГУТ