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 отечественникам – Л.М. Финку и В.И. Шляпоберскому (1955г.) [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 – Нейфаха А. Э. и Зигангирова К. Ш.

По мере развития и усложнения принципа формирования рекуррентной последовательности для описания свойств создаваемых кодов и способа фрмирования 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 2 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 передаётся 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 её 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 $b \leq 2t$ 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 обобщённом 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 $h_{i,j}$=(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 r_u – 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 К) 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.

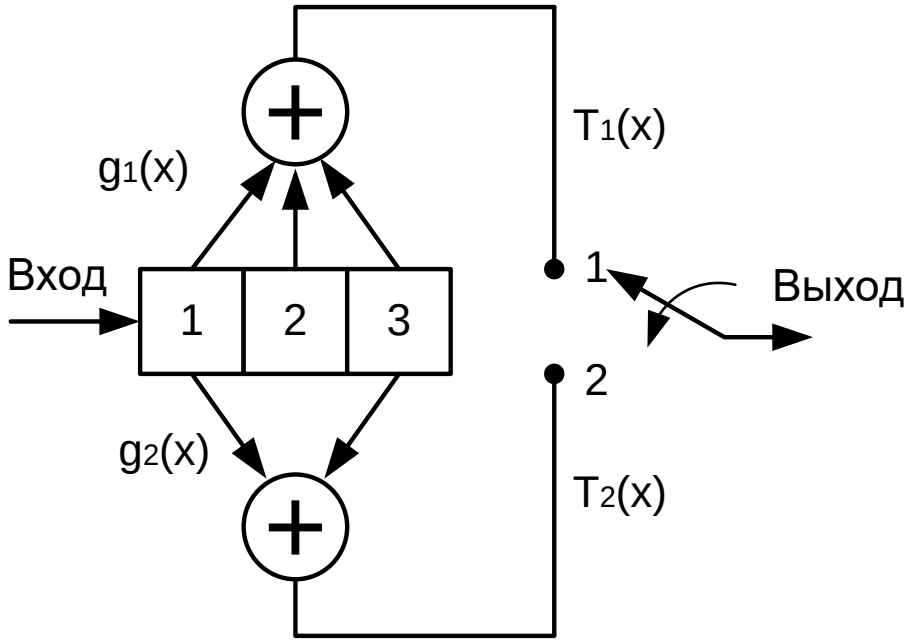

Рис.1.1. Сверточный кодер со скоростью R=1/2 и кодовым ограничением К=3.

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, или 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 примере выходная последовательность, соответствующая входной вида X= 1110000…., будет равна сумме по модулю 2 строк 1, 2 и 3 матрицы G, т.е. Y= 11 01 10 01 11 00 00 …

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.

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.

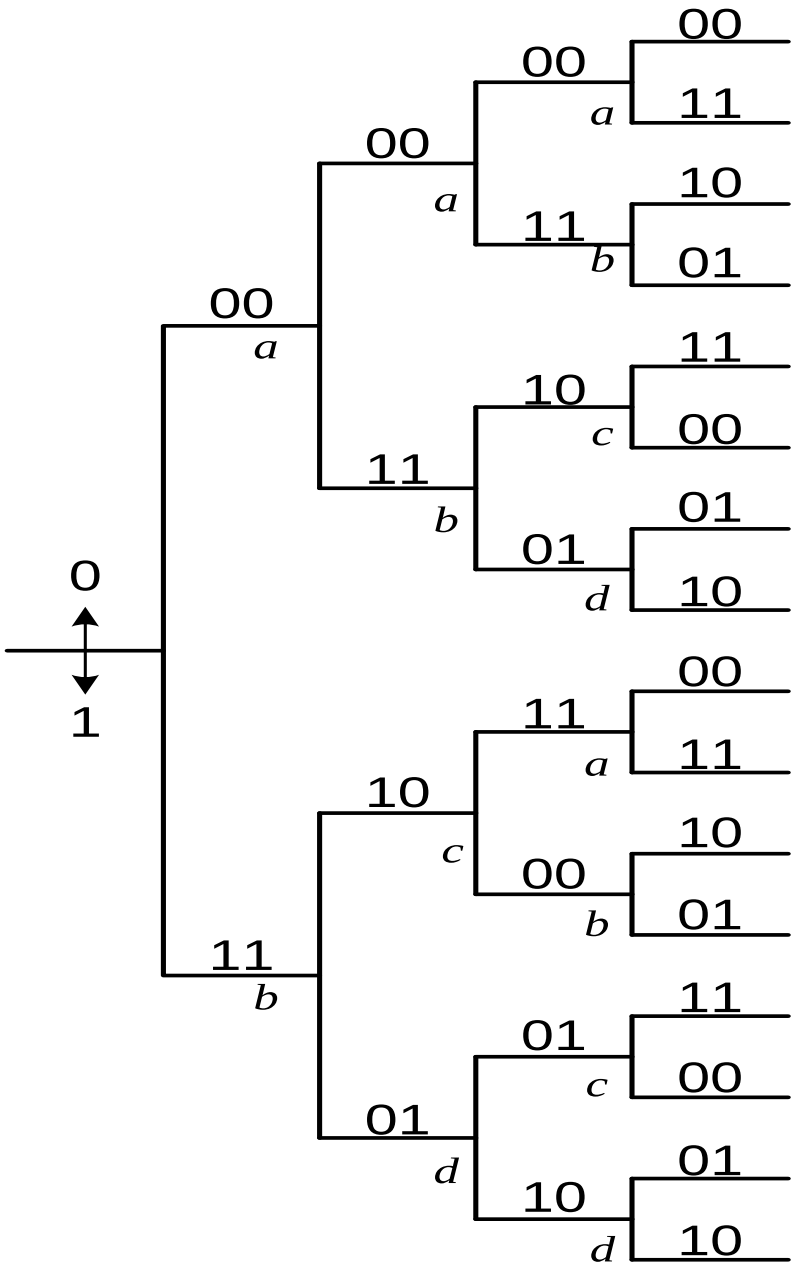

Рис. 1.2. Древовидная диаграмма сверточного кода для кодера на рис. 1.1 со скоростью кода R=1/2 и длиной кодового ограничения K=3.

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 выше было показано, для сверточного кода с $R=1/n$ и длиной регистра сдвига из K ячеек памяти n выходных символов в i-ый 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 длину кодового ограничения определяют 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-ом 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-ом 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 ранее было показано, 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 в нем повторяющиеся узлы и исходящие из них пути объединить. В этом случае вместо разрастающегося вширь кодового дерева мы получим другую компактную структуру, которую Форни назвал решетчатой (рис. 1.3).

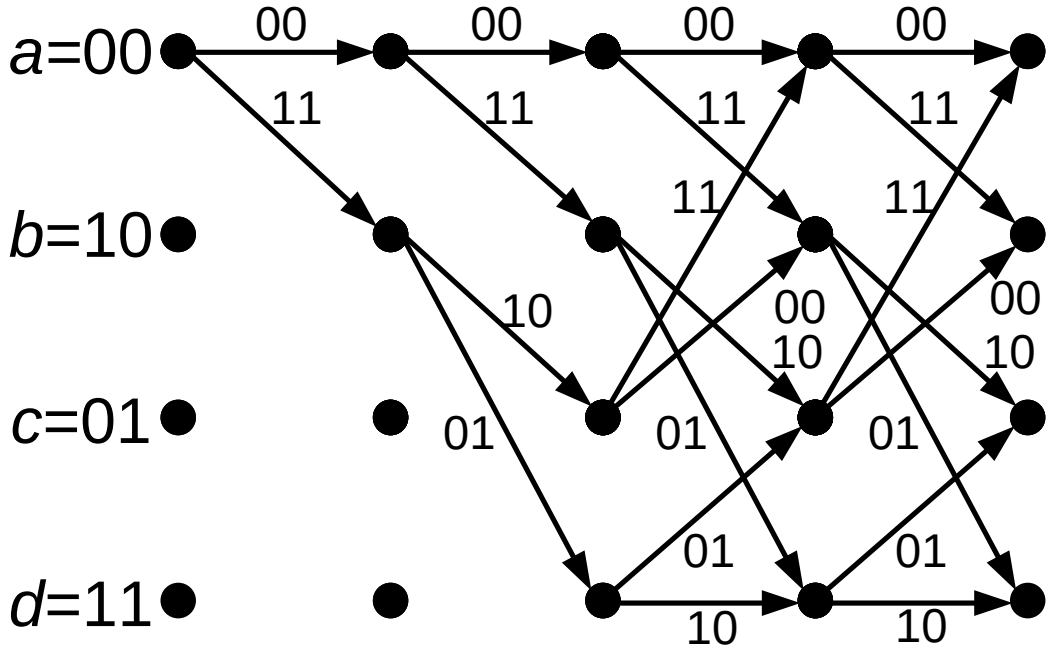

Рис. 1.3. Решетка кодера для сверточного кода со схемой на рис. 1.1 (R=1/2, K=3).

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 $d_{св}$. Следовательно, минимальный вес $w_{\min}$ линейного сверточного кода будет равен свободному расстоянию $d_{св}$.

Таким образом, в режиме обнаружения ошибок свободное расстояние $d_{св}$ 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 $t = \left\lfloor \frac{d_{св} - 1}{2} \right\rfloor$ включительно, $\lfloor\ \rfloor$ – целая часть дроби.

Минимальное расстояние Хемминга, т.е. свободное расстояние $d_{св}$, найдем его как расстояние между двумя различными, но сливающимися путями. Для этого воспользуемся решеткой на рис. 1.3, соответствующей сверточному коду рис. 1.1. Пусть один из двух путей идет через узлы *a* и будет полностью нулевым, т.е. ему будет соответствовать также 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*, затем снова сольется с узлом *a*. При этом расходящимся в узле *a* и снова сходящимся в том же узле *a* участкам будут соответствовать

верхний (00 00 00) для входных символов 000 и нижний путь (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, или c, или d. Зону между участками со свободным расстоянием иногда называют минимальным просветом.

Таким образом, мы построили сверточный код со свободным расстоянием $d_{св}$=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, о чем более подробно будет сказано ниже.

Поиск хороших сверточных кодов (с наибольшим $d_{св}$ 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, показанного 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 или1) на графе для наглядности стоят рядом с ветвью. Два символа у каждой ветви на диаграмме состояний представляют выходные биты.

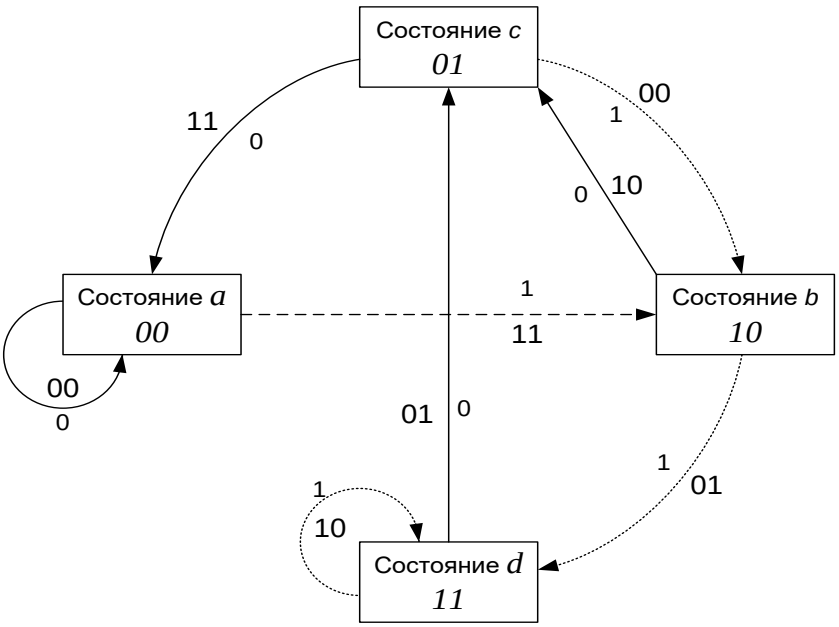

Рис. 1.4. Диаграмма состояний для сверточного кода.

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.

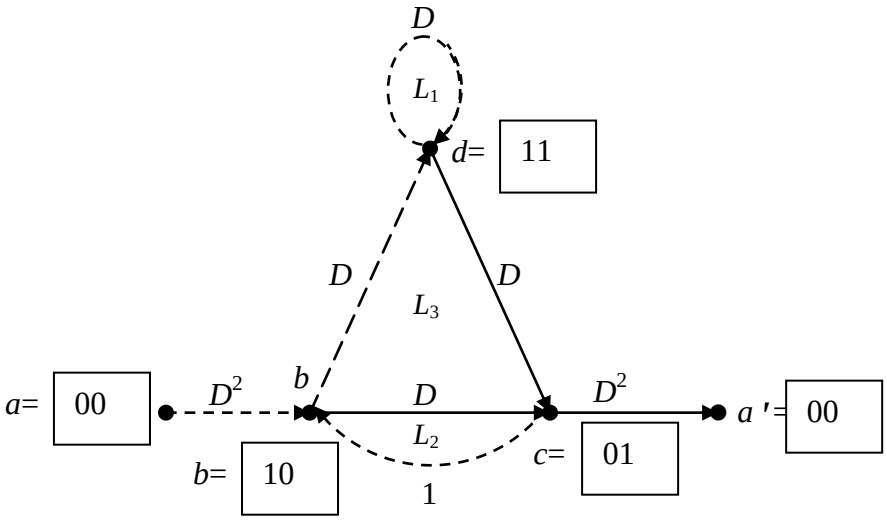

Рис. 1.5. Диаграмма состояний (на ребрах при D^i степень i указывает на вес выходной пары бит).

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:

$$T_{(a\to a')} = \frac{\sum_i P_i \prod_j (1-L_j)}{\prod_{m=1}^{k}(1-L_m)} = \frac{\sum_i P_i \prod_j (1-L_j)}{1-\sum_{m=1}^{k} L_m + (\sum_{i,j} L_i L_j)^* - (\sum_{i,j,z} L_i L_j L_z)^* + \ldots}, \qquad (1.3)$$

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

L_j – передача j-ой 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 путём 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) условно в фигурных скобках 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 впервые 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 последовательностей, 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) надо еще 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.

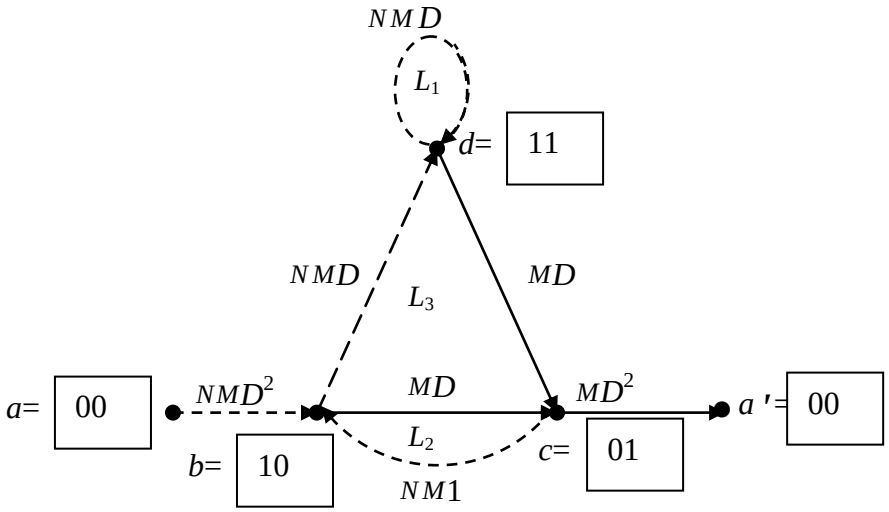

Рис. 1.6. Диаграмма состояний (на ребрах при D^i степень i указывает на вес выходной пары бит).

Как видно из этого рисунка, у рёбер, 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:

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

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 впервые 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 $d_{св}$=5. Декодировав такой путь, будет получена соответствующая входная информационная 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 нулевая последовательность может быть декодирована 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)

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

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

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], который 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.

Рассмотрим, в качестве примера, декодирование на основе алгоритма Витерби для сверточного кода с R=1/2 и кодовым ограничением K=3, схема кодера которого и решетчатая диаграмма 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).

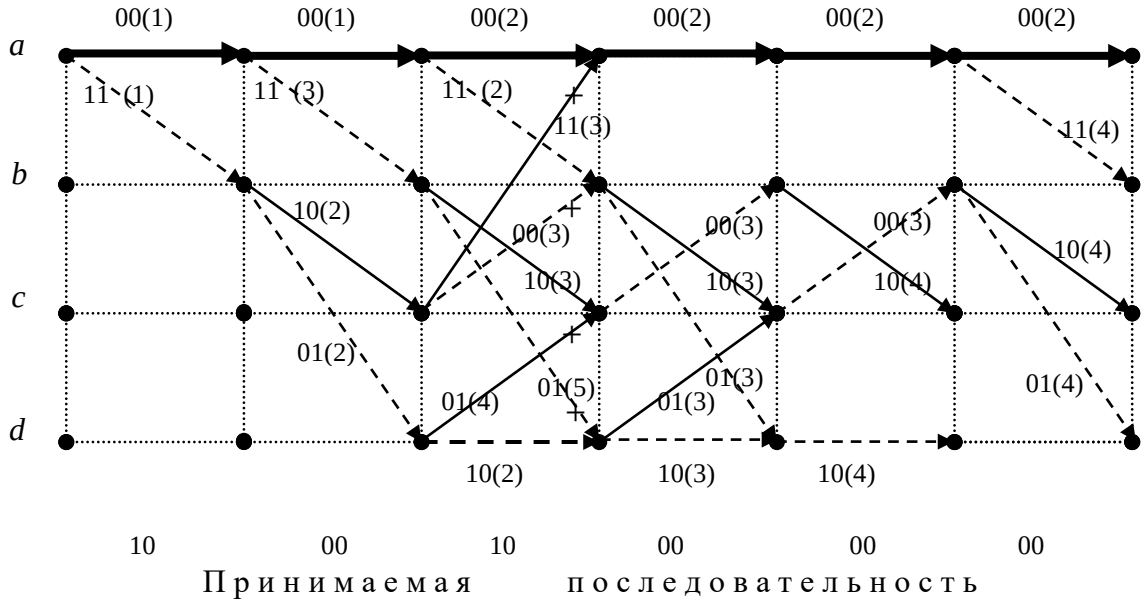

Рис. 1.7. Пример декодирования исправляемой комбинации ошибок по алгоритму Витерби

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. Сохраняется из двух 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-ой пар (00 00 00) для упрощения диаграммы отбрасываемые пути не показаны. Как видим, на каждом их 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-ом уровне в узлы приходят по два пути с одинаковыми метриками. В этом случае равновероятно выбирается один из них. На 6-ом уровне имеется один выживший путь с метрикой 2, а остальные три – метрику 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.

Таким образом, двигаясь по решетчатой диаграмме, декодер запоминает на каждом уровне в каждом состоянии только по одному выжившему пути (всего 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 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, в который 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 это все 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 $g_{11}(x)= g_{12}(x)=1+x$; $g_{13}(x)=1$, для нижних ячеек $Я_{21}$ и $Я_{22}$ соответственно будут $g_{21}(x)= 0$; $g_{22}(x)=x$; $g_{23}(x)=1+x$.

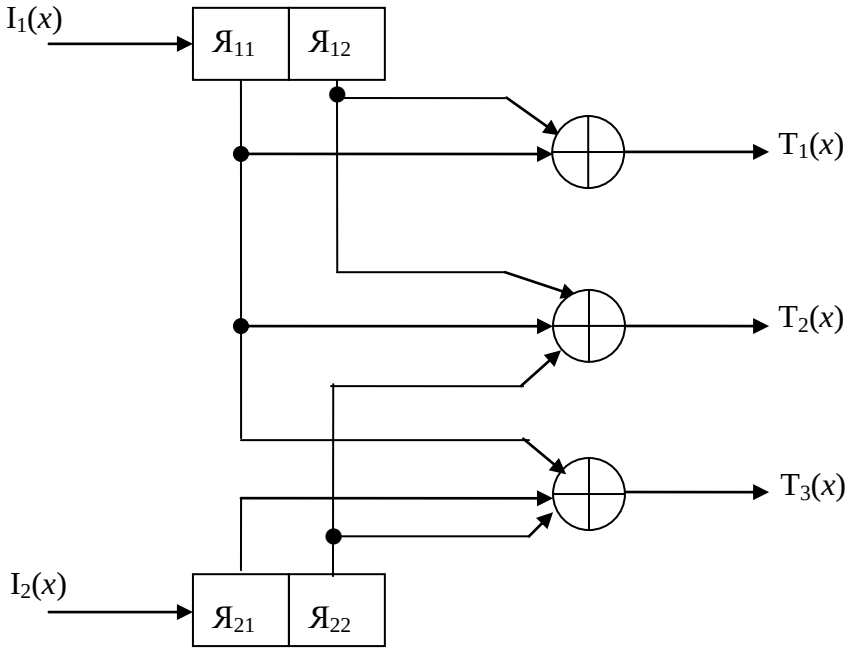

Рис. 1.8. Схема сверточного кодера с R=2/3 и длиной кодового ограничения К=4.

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 $G(x)$:

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

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:

$$\begin{bmatrix} T_1(x) & T_2(x) & T_3(x) \end{bmatrix}=\begin{bmatrix} (1+x^2) & x \end{bmatrix}\begin{bmatrix} 1+x & 1+x & 1 \\ 0 & x & 1+x \end{bmatrix}=\begin{bmatrix} (1+x+x^2+x^3) & (1+x+x^3) & (1+x) \end{bmatrix}.$$

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.

$$\{T_1T_2T_3\} = \begin{bmatrix} I_1 & I_2 \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} I_1 & I_2 \end{bmatrix} \begin{bmatrix} 111 & 110 & 000 & 000 & 000 & \dots \\ & 111 & 110 & 000 & 000 & \dots \\ & & 111 & 110 & 000 & \dots \\ & & & 111 & 110 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 001 & 011 & 000 & 000 & 000 & \dots \\ & 001 & 011 & 000 & 000 & \dots \\ & & 001 & 011 & 000 & \dots \\ & & & 001 & 011 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

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 …).

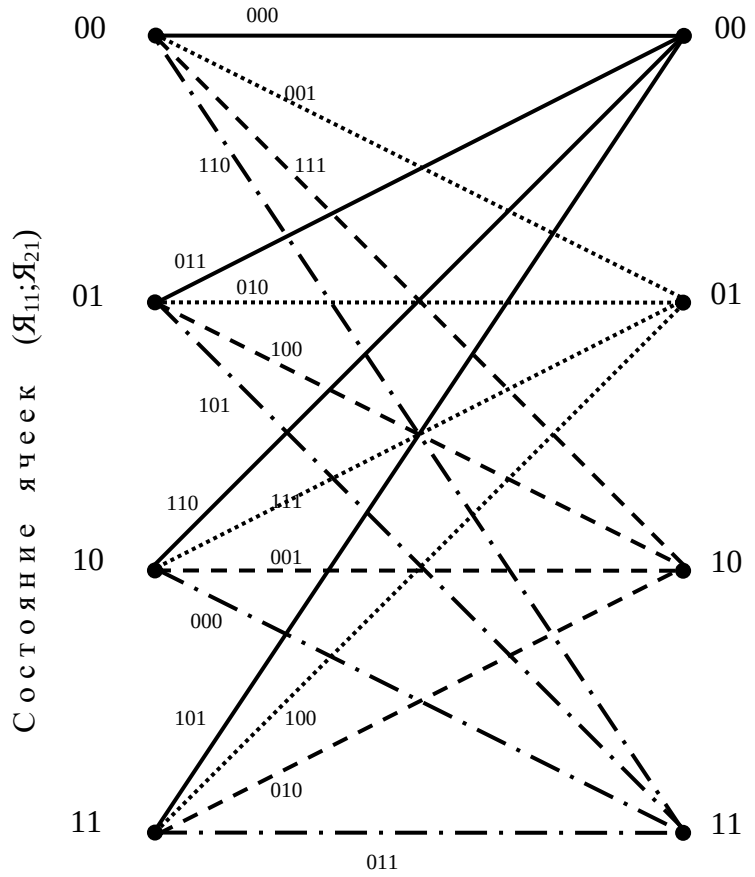

Рис. 1.9. Решетчатая структура сверточного кода с R=2/3 и К=4.
(на диаграмме парам различных входных символов ($I_1(x)$; $I_2(x)$) соответствуют различные линии:

——— для пары (00); ····· для пары (01);
– – – – для пары (10); — · — · — · для пары (11)

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 (выживший). 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 $(T_1^i, T_2^i, T_1^{i+1}, T_2^{i+1})$ каждый второй символ второй пары будет удален, т.е. на выходе кодера появятся только три символа $(T_1^i, T_2^i, T_1^{i+1})$. Здесь нижние индексы при T указывают на символы на выходах первого и, соответственно, второго сумматоров кодера, а верхний индекс i указывает на i-ый тактовый момент работы кодера. Таким образом, выходная последовательность кодера на рис. 1.1

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

будет преобразуется в перфорированную последовательность

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

Положительным свойством полученного перфорированного сверточного кода со скоростью R=2/3 и K=3 является то, что его декодер, реализующий алгоритм Витерби, остаётся 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*) на приемной стороне на удаленных позициях специальных символов стираний. 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 (таблица 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)	$\begin{bmatrix}1\\1\end{bmatrix}$	(T_1^i, T_2^i)	10
2/3	$\begin{bmatrix}1 & 0\\1 & 1\end{bmatrix}$	$(T_1^i, T_2^i, T_2^{i+1})$	6
3/4	$\begin{bmatrix}1 & 0 & 1\\1 & 1 & 0\end{bmatrix}$	$(T_1^i, T_2^i, T_2^{i+1}, T_1^{i+2})$	5
5/6	$\begin{bmatrix}1 & 0 & 1 & 0 & 1\\1 & 1 & 0 & 1 & 0\end{bmatrix}$	$(T_1^i, T_2^i, T_2^{i+1}, T_1^{i+2}, T_2^{i+3}, T_1^{i+4})$	4
7/8	$\begin{bmatrix}1 & 0 & 0 & 0 & 1 & 0 & 1\\1 & 1 & 1 & 1 & 0 & 1 & 0\end{bmatrix}$	$(T_1^i, T_2^i, T_2^{i+1}, T_2^{i+2}, T_2^{i+3}, T_1^{i+4}, T_2^{i+5}, T_1^{i+6})$	3

В таблице 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].