Занятие № 6. “Сверточные коды”

1. Основные свойства сверточных кодов, методы задания

Термин «сверточный код» связан с алгебраической моделью этого кода, т.к. выходные символы можно рассматривать как свертку импульсной характеристики кодера с информационной последовательностью поступающей на вход кодера.

Сверточный код – это линейный рекуррентный код. В общем случае он образуется следующим образом. В каждый i-й тактовый момент времени на вход кодирующего устройства поступает k символов сообщения: $b_1 b_2 .. b_k$. Выходные символы $a_1 a_2 ... a_n$ формируются с помощью рекуррентного соотношения из K символов сообщения, поступивших в данный и в предшествующие тактовые моменты времени.

Символы сообщения, из которых формируются выходные символы, хранятся в памяти кодирующего устройства. Величина K называется длиной кодового ограничения. Она показывает, на какое максимальное число выходных символов влияет данный информационный символ, и играет ту же роль, что и длина блочного кода: $K = r + 1$, где r – число ячеек регистра (наибольшая из степеней порождающих многочленов, на которые умножается любая из, входных последовательностей).

В течение этого времени с выхода кодирующего устройства будет считано $m = n(r+1)$ символов. Величину m называют полной длинной кодового ограничения сверточного кода.

Корректирующая способность сверточного кода зависит от свободного расстояния d_{CB}, которое по существу содержит ту же информацию о коде, что и кодовое расстояние для блочных кодов. Оно определяется как минимальный вес пути (минимальное число единиц), начинающегося и заканчивающегося в нулевом состоянии, т.е. d_{CB} определяет минимальное расстояние Хэмминга между кодовыми последовательностями, одна из которых является нулевой.

Сверточный код имеет избыточность $\chi = 1 - \frac{k}{n}$ и обозначается как (n, k).

По аналогии с блочными кодами сверточные можно классифицировать как разделимые и неразделимые.

Сверточный код получается разделимым, если в каждый тактовый момент времени k выходных символов совпадают с символами сообщения. На практике обычно используются неразделимые сверточные коды.

Для неразделимых сверточных кодов каждый набор n выходных символов зависит как от текущего входного набора, так и от предыдущих входных символов. В результате кодирования последовательность становится как бы одной полубесконечной кодовой комбинацией.

При кодировании поток входящих информационных символов разбивается на кадры (такты), содержащих по k символов. Каждым входным k информационным символом соответствует n кодовых символа на выходе кодера, которые последовательно (поочередно) считываются переключателем.

Сверточные коды можно генерировать с помощью линейных регистров сдвига, выполняющих операцию свертки информационной последовательности и импульсной характеристики регистра (порождающей последовательности).

2. Алгоритм кодирования сверточного кода

Кодирующее устройство СК может быть реализовано с помощью сдвигающего регистра и сумматоров по модулю 2. Связь между ячейками сдвигающего регистра и сумматорами по модулю 2 описываются порождающими многочленами, вида:

$$G(x) = g_0 + g_1 x + ... + g_n x^n .$$

Наличие члена $g_i x^i$ (где $i = 0,1,2,...,n$) в порождающем многочлене означает, что $g_i = 1$ и соответствующий разряд регистра сдвига соединен с сумматором. Счет разрядов регистра ведется слева направо.

Рассмотрим кодер сверточного кода $(1/2)$ заданный порождающими полиномами (рис.2.):

$$G_1(x)=1+x+x^2 \text{ и } G_2(x)=1+x^2.$$

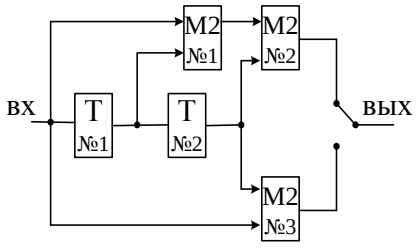

Рис. 2. Кодирующее устройство сверточных кодов

Для кода $(1/2)$ на каждый входной символ образуется 2 выходных символа, которые последовательно во времени через коммутатор подаются в канал.

Выходные символы являются линейными функциями поступающего информационного символа и логического состояния регистра.

Значение каждого двоичного символа на выходе кодера зависит от состояния ячеек регистра сдвига. Эти комбинации называют логическими состояниями регистра кодера и обозначают: $a=00$; $b=10$; $c=01$; $d=11$. Работу кодера удобно описывать при помощи диаграммы состояний регистра (рис.3), представляющей собой направленный граф, вершины которого отождествляются с возможными состояниями триггерных ячеек, а ребра, помеченные стрелками переходы между этими состояниями, с выходными символами. Поступающая на вход информационная последовательность задает конкретную последовательность смены состояний и при этом порождает кодовые символы на выходе кодера.

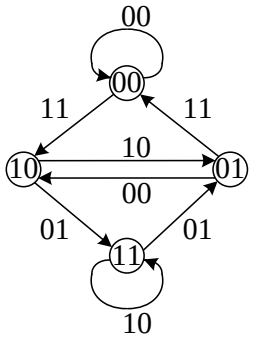

Рис. 3. Диаграмма состояний регистра кодера

Рассматриваемую диаграмму состояний можно развернуть во времени, тогда получим решетчатую диаграмму (рис. 4).

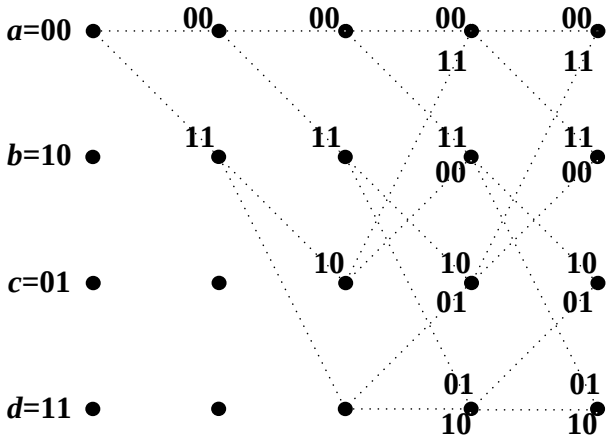

Рис. 4. Решетчатая диаграмма кода **1/2**

Решетчатая диаграмма показывает все разрешенные пути, по которым продвигается кодер при кодировании последовательности во времени, от такта к такту. Начало диаграммы обозначает переходной режим работы кодера (из нулевого состояния), через три такта структура диаграммы становится повторяющейся. Исключая переходной режим, замечаем, что в каждом узле диаграммы сходятся 2 ветви (окончание 2-х путей) и из каждого узла исходит тоже 2 пути. Передаваемой информационной последовательности символов соответствует конкретная последовательность состояний кодера, задающая определенный путь совокупностью узлов на решетчатой диаграмме. Единичному символу сообщения приписываются нижние исходящие ветви, а нулевому – верхние.

Рассмотрим **пример**. Пусть на вход поступает последовательность из информационных символов: 01011001110 .

Покажем работу с помощью таблицы состояний регистра (таб. 1).

Таблица 1.

Состояния регистра кодера:

вход		**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**
сост. ячеек	**00**	**00**	**10**	**01**	**10**	**11**	**01**	**00**	**10**	**11**	**11**	**01**
выход		**00**	**11**	**10**	**00**	**01**	**01**	**11**	**11**	**01**	**10**	**01**

Пример решетчатой диаграммы состояний показан на рис. 5.

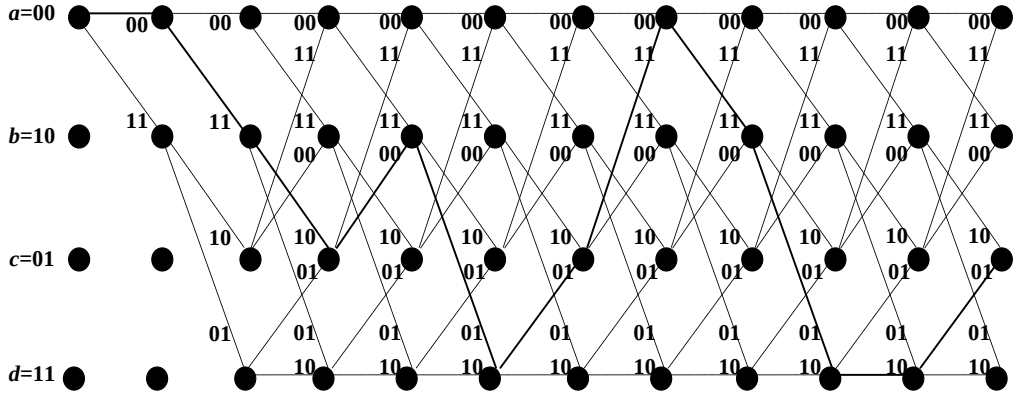

Рис. 5. Решетчатая диаграмма заданной информационной последовательности

Сверточное кодирование применяемое вместе с алгоритмом декодирования Витерби становится все более применимым, т.к. кодеры и декодеры таких кодов проще, чем блочных с той же помехоустойчивостью.

3. Алгоритм декодирования сверточного кода по алгоритму Витерби

Сверточные коды можно декодировать различными методами. Различают декодирование с вычислением и без вычисления проверочной последовательности.

Декодирование с вычислением проверочной последовательности применяется только для систематических кодов. По своей сущности оно ничем не отличается от соответствующего метода декодирования блочных кодов. На приемной стороне из принятых информационных символов формируют проверочные символы по тому закону, что и на передающей стороне, которые затем сравнивают с принимаемыми проверочными символами. В результате сравнения образуется проверочная последовательность, которая при отсутствии ошибок состоит из одних нулей. При наличии ошибок на определенных позициях последовательности появляются единичные символы. Закон формирования проверочных символов выбирается таким образом, чтобы по структуре проверочной последовательности можно было определить искаженные символы.

К числу методов декодирования без вычисления проверочной последовательности относятся декодирование по принципу максимума правдоподобия и последовательное декодирование.

Декодирование по принципу максимума правдоподобия сводится к задаче отождествления принятой последовательности с одной из возможных информационных последовательностей. Решение принимается в пользу той кодовой последовательности, которая в меньшем числе позиций отличается от принятой. Метод применим для любого сверточного кода. Однако при больших значениях длины информационной последовательности он практически не реализуем из-за необходимости перебора огромного количества возможных кодовых последовательностей. Существенное упрощение процедуры декодирования по максимуму правдоподобия предложил Витерби.

Идея декодирования состоит в том, что необходимо по принятой кодовой последовательности восстановить на приеме путь кодера по решетке. Принятая на каждом такте работы декодера n–символьная комбинация по выбранной метрике (Хэмминга) сравнивается для каждого узла решетки с возможными последовательностями, определяющими ветви, сходящиеся в этом узле.

Для любого узла сохраняется только один наиболее правдоподобный путь, входящий в этот узел. Такой путь (с меньшим весом) называется «выжившим».

Для построения решетчатой диаграммы необходимо знать следующие характеристики:

вес ветви – равен расстоянию Хэмминга между комбинацией на входе декодера и комбинацией присвоенной данной ветви на решетчатой диаграмме (обозначают цифрами около ветвей): например если поступила кодовая комбинация 00, то для ветви $\overset{\frown}{0}$ вес будет равен 0, а для $\overset{\frown}{1}$ и $\overset{\frown}{0}$ равен 1, естественно для $\overset{\frown}{1}$ вес ветви составит 2;

вес пути – сумма весов ветвей, образующих некоторый путь на решетчатой диаграмме (путь конечной длины заканчивается в определенном состоянии (узле));

вес состояния – равен весу «выжившего» пути, который заканчивается в данном состоянии (обозначают цифрами в кружках), хранится в памяти декодера.

На каждом такте декодирования, согласно алгоритму Витерби в каждом из состояний решетчатой диаграммы производят следующие однотипные операции:

получение веса всех ветвей;

для всех узлов определяются веса входящих путей, как сложение весов предыдущих состояний с весами соответствующих ветвей;

сравнение весов входящих в узел путей и выбор пути с наименьшим весом, величина которого используется как вес состояния на последующем такте декодирования (Если веса сравниваемых путей одинаковы, выбор одного из двух путей производят случайным образом);

выделение пути имеющего наименьшее значение конечного веса состояния;

присвоение значения кодовых символов выходной информационной последовательности, согласно правила:

верхней исходящей из узла ветви присваивается кодовый символ ноль;

нижней – единица.

Пример. Рассмотрим на примере кодовой комбинации полученной при кодировании сверточным кодом $(n, 2)$. Введем преднамеренно в данную комбинацию ошибки, т.е. принятая последовательность будет иметь вид: $000110000101111101100 1$.

Построим решетчатую диаграмма декодирования принятой кодовой комбинации сверточным кодом $(n, 2)$ по алгоритму Витерби (рис. 6).

На первом такте работы декодера веса состояний (записанные в кружках) имеют нулевое значение, определим веса ветвей для каждого из узлов. Двум ветвям, входящим во второй узел $a = 00$, присвоены кодовые слова (00) – верхней, и (11) – нижней, следовательно веса ветвей будут соответственно составлять (0) для верхней, и (2) для нижней. Из двух путей входящих в данный узел, «выжившим» окажется верхний, т.к. его вес будет равен (0), т.е. является минимальным.

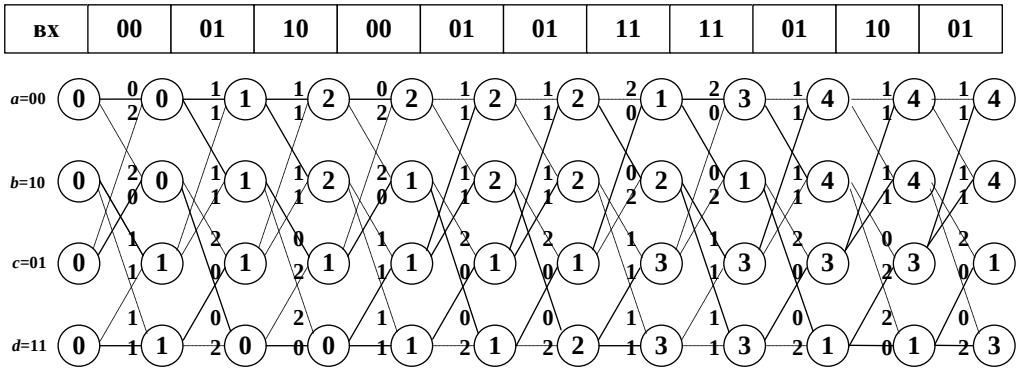

Рис. 6. Решетчатая диаграмма декодирования принятой последовательности с помощью алгоритма Витерби

Рассмотрим этот процесс, для узла $c = 01$. Ветвям, входящим в данный узел, присвоены кодовые слова: (01) – верхней, и (10) – нижней, веса ветвей составляют (1) как для верхней, так и для нижней. Веса сравниваемых путей одинаковы, выбор одного из двух путей можно произвести случайным образом, например, выберем верхний.

На каждом такте в результате сравнения половина возможных путей отбрасывается и в дальнейшем не используется. Другая половина образует продолжение путей для следующего такта декодирования и т.д.

Декодер прослеживает по кодовой решетке путь, имеющий минимальное Хэммингово расстояние от пути, который генерирует кодер.

В итоге после одиннадцатого такта из четырех весов состояний наименьшим оказался путь вошедший в узел $c = 01$. Выделим данный путь пройдя по диаграмме в обратном направлении, т.к. в каждый узел из двух входящих путей нами оставлялся один, то и обратный путь окажется однозначно определяемым.

Окончательное принятие решения проследим на видоизмененной решеточной диаграмме (рис. 7):

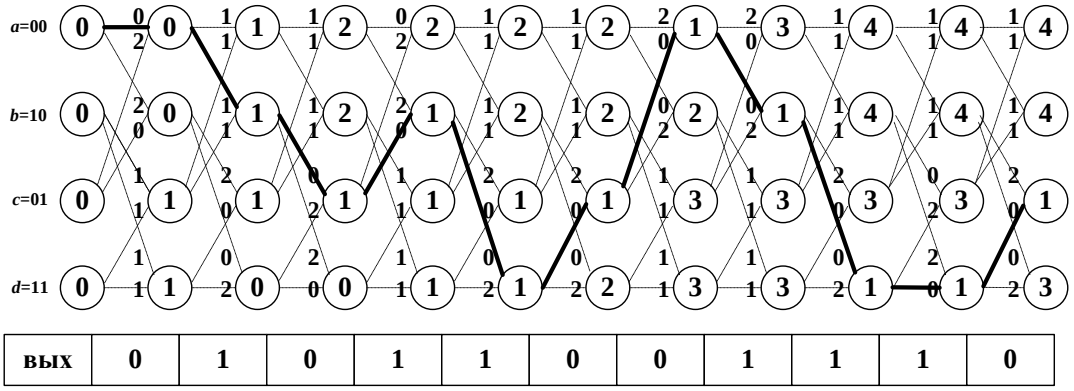

Рис. 7. Решетчатая диаграмма декодирования исправляемой последовательности

В рассмотренном случае принятая последовательность содержала один ошибочный символ, а код имел свободное расстояние $d_{CB} = 5$, что позволило ему исправить эту ошибку. Если принятая последовательность содержит три и более ошибок, то данный код не способен их исправить.