Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

О. Р. НИКИТИН П. А. ПОЛУШИН

СОВРЕМЕННЫЕ МЕТОДЫ КОДИРОВАНИЯ ИНФОРМАЦИИ

Учебное пособие

Владимир 2018

УДК 621.391.8
ББК 32.841
Н62

Рецензенты:
Доктор технических наук, профессор
профессор кафедры автоматизированных средств измерения
филиала Военной академии ракетных войск стратегического назначения
имени Петра Великого в г. Серпухове
В. А. Цимбал

Доктор технических наук, доцент
профессор кафедры биомедицинских и электронных средств и технологий
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
В. П. Крылов

Издается по решению редакционно-издательского совета ВлГУ

Никитин, О. Р. Современные методы кодирования информации : учеб. пособие / О. Р. Никитин, П. А. Полушин ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2018. – 80 с. – ISBN 978-5-9984-0857-1.
Н62

В большинстве современных систем передачи цифровых сигналов используются различные методы кодирования, в связи с чем владение научными методами расчета и проектирования систем кодирования крайне важно для специалиста. Рассмотрены основные методы кодирования информации. Предложены задания для выполнения расчетов и тест для самоконтроля.

Предназначено для магистрантов направления подготовки 11.04.01 – Радиотехника.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Табл. 6. Ил. 19. Библиогр.: 20 назв.

УДК 621.391.8
ББК 32.841

ISBN 978-5-9984-0857-1

ВВЕДЕНИЕ

Условия, в которых приходится функционировать радиотехническим системам, усложняются в связи с ростом количества радиоизлучающих средств и обострением социально-политических противоречий. В то же время возрастают требования к качеству передачи информации и ее объемы. Применение современных методов кодирования информации позволяет значительно повысить помехоустойчивость передачи сигналов.

Цели изучения дисциплины «Современные методы кодирования информации»:

1) освоение основных способов повышения помехоустойчивости современных радиотехнических систем при создании радиоэлектронной аппаратуры;

2) формирование практических навыков расчета и проектирования систем кодирования с помощью научных методов.

Данная дисциплина основывается на курсах бакалаврского образования, таких как «Математика», «Физика», «Основы теории связи», и магистерского образования: «История и методология науки и техники (применительно к радиотехнике)» и др. Полученные знания могут быть использованы при подготовке магистерской выпускной квалификационной работы, а также при разработке и проектировании радиоаппаратуры.

1. ОБЗОР ОСНОВНЫХ МЕТОДОВ КОДИРОВАНИЯ

Код – это определенная форма представления сообщения, которая может не зависеть от физической сути сигналов, хотя практически она обычно связана с их физическими параметрами [1 – 4]. Многие сообщения исходно обладают внутренними корреляционными связями, позволяющими устранять часть возможных ошибок, что, например, делает понятной даже сильно зашумленную речь. Однако в общем случае символы исходного информационного сигнала следует считать взаимно независимыми, так как такая последовательность несет наибольший объем информации. Если исходные информационные связи выражены слабо или неизвестны, их затруднительно использовать для повышения помехоустойчивости. В этом случае с помощью кодирования различными методами вводят искусственные связи путем увеличения числа символов. Степень возникающей избыточности определяет исправляющие свойства кода. Известны также коды без избыточности, которые используют заранее известный ограниченный объем передаваемых информационных сообщений, однако они, как правило, узкоспециализированы.

В настоящее время известно большое количество различных кодов [5 – 20]. Основные распространенные виды условно представлены на рис. 1. Различают систематические и несистематические коды. В первых можно в кодовой последовательности выделить исходную информационную последовательность символов. Она может быть непрерывной либо разделенной на фрагменты. Новые избыточные символы по определенному принципу добавляют к исходной последовательности.

В несистематических кодах формируемая в кодере последовательность не содержит неизмененную исходную последовательность. Все символы кодовой последовательности формируются из совокупности исходных символов с использованием определенных

функций или правил, определяющих вид и параметры кода. В двоичных кодах каждый символ содержит один информационный бит, в недвоичных кодах – несколько битов. Различие фактически состоит лишь в усложнении аппаратуры кодирования.

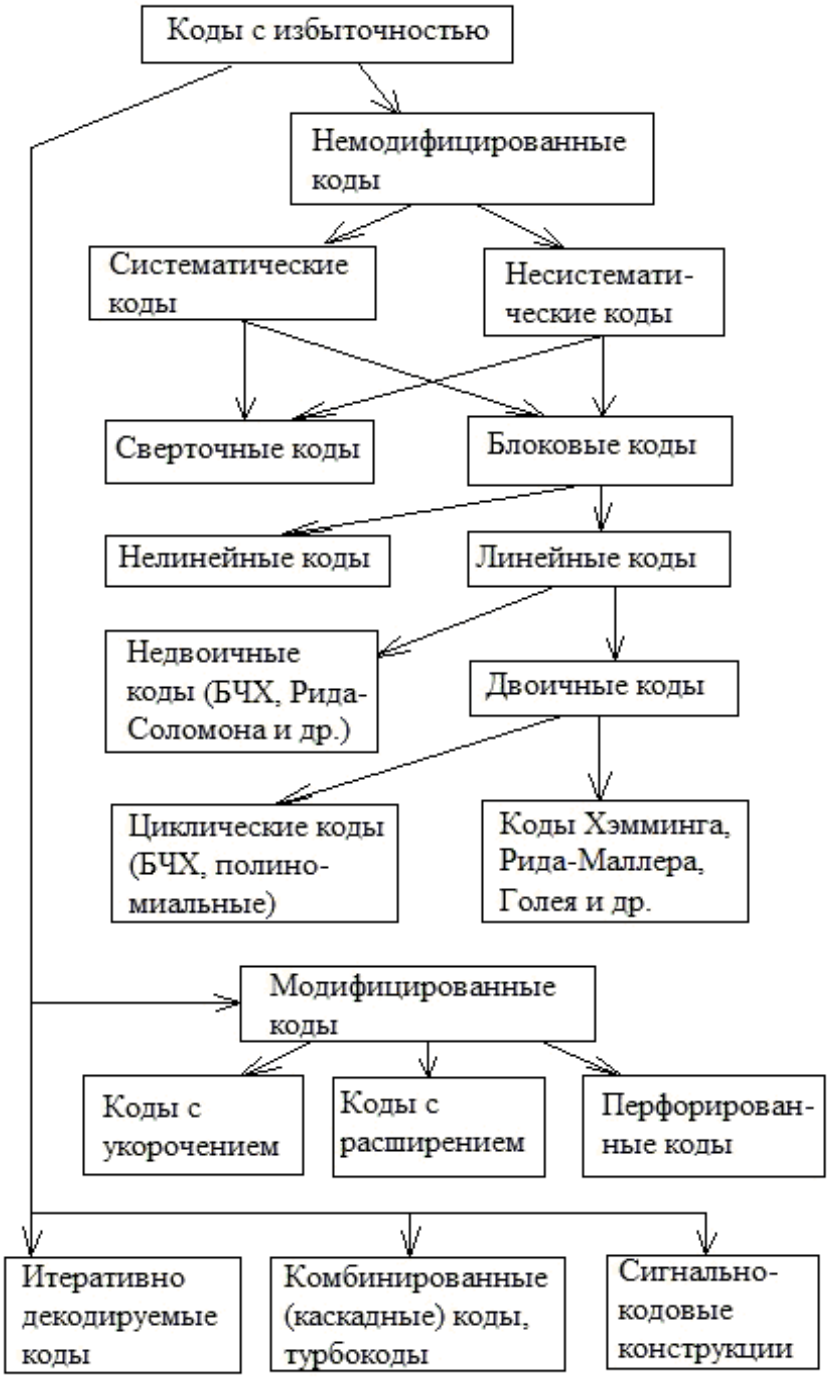


Рис. 1. Классификация кодов

Линейные коды отличаются от нелинейных замкнутостью получаемого множества кодовых слов относительно оператора, с помощью которого реализуется кодирование. Оператор в этом случае линейный, а кодовые слова можно описать в виде векторов некоторого пространства. Линейность кода упрощает его реализацию, и при большой длине кодовых слов применяют только линейные коды. Они образуют обширные классы. Однако нелинейные коды в среднем обладают лучшими характеристиками.

Различные коды удобно разделять на относительно простые (немодифицированные) и модифицированные. Пособие посвящено диагностике кодов первого вида, так как они выступают некоторой основой, на которой строятся коды второго вида.

Рассматриваемые коды в зависимости от основного правила их получения делятся на сверточные и блоковые. Несмотря на их существенные различия, после определенной модификации каждой группе можно придать некоторые свойства другой группы. Рассмотрим подробнее принципы формирования и свойства каждой из этих групп. Поскольку для целей диагностики основное значение имеют операции кодирования, то основное внимание уделено именно им.

2. СВЕРТОЧНЫЕ КОДЫ

Сверточные коды в настоящее время нашли широкое применение [2 – 6]. Иногда их называют непрерывными кодами, потому что в них используется непрерывная (последовательная) обработка символов. Кроме того, они являются линейными кодами. Кодер обладает памятью в том смысле, что каждый выходной кодовый символ зависит не только от текущего входного информационного символа, но и от нескольких предыдущих входных символов.

Избыточность кода определяется соотношением количества информационных и проверочных кодовых символов. Если для передачи k информационных символов требуется n кодовых символов, то кодовая скорость $R = k/n$. Обобщенная структурная схема сверточного кодера приведена на рис. 2.

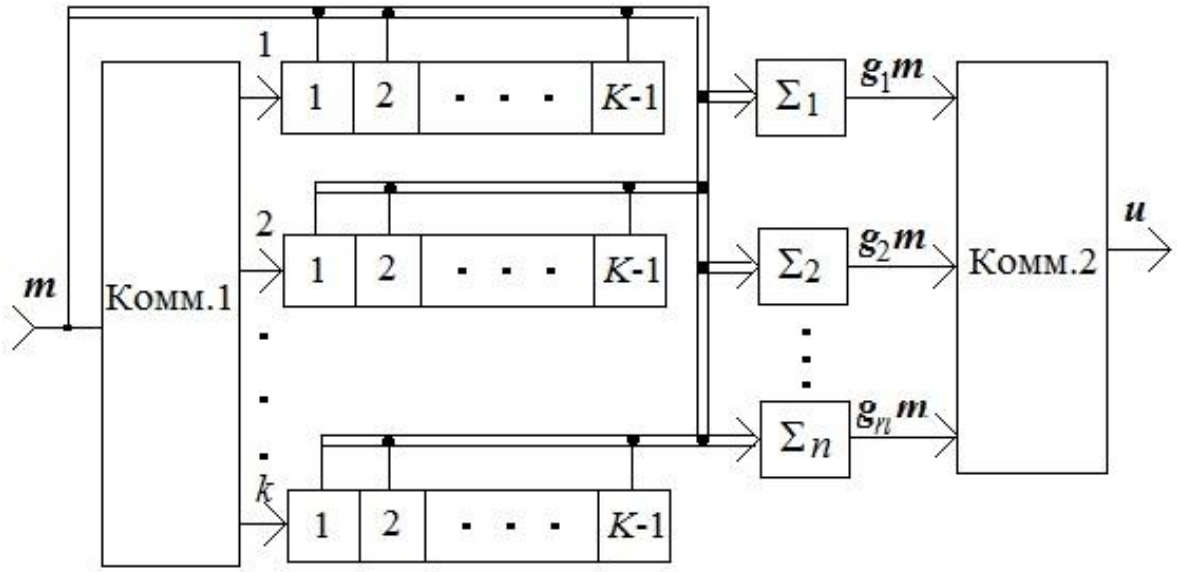


Рис. 2. Обобщенная структура сверточного кодера

На его вход поступает последовательность информационных символов $\boldsymbol{m} = m_1, m_2, \ldots$. На выходе образуется последовательность кодовых символов $\boldsymbol{u} = u_1, u_2, \ldots$. Кодер состоит из k сдвиговых регистров, содержащих по $K - 1$ ячеек. Входной коммутатор (Комм.1) распределяет символы входной последовательности по последовательным входам регистров. При каждом такте содержимое регистров сдвигается в соседнюю ячейку. Сигналы с параллельных выходов регистров подаются на входы n многовходовых сумматоров, $n > k$. В сумматорах над входными сигналами производится логическая операция сложения по модулю 2 («исключающее или»). Входы каждого сумматора подключены к определенному набору ячеек сдвиговых регистров. Эти наборы различаются у всех сумматоров и определяют структуру конкретного используемого кодера.

Выходная кодовая последовательность образуется в результате поочередного подключения с помощью коммутатора (Комм.2) выходных сигналов сумматоров на общий выход кодера. Таким образом, при поступлении k исходных информационных символов образуется n кодовых символов. Варьируя параметры k и n, можно регулировать кодовую скорость R. Однако чаще используются коды со скоростью $1/n$, и скорость регулируется путем применения перфорации (выкалывания).

В формировании каждого кодового символа в текущий момент времени участвует входной символ и K – 1 предыдущих входных символов. Таким образом, кодовое ограничение K определяет память кодовой последовательности. Кодер, приведенный на рис. 2, представляет собой систему с конечным откликом.

Если при k = 1 пропускать через кодер исходную последовательность символов, содержащую только одну единицу, а остальные нули, то формируемая кодовая последовательность будет представлять собой набор импульсных откликов $g_1 \div g_n$ каждого из n сумматоров. Наборы коэффициентов $g_1 \div g_n$ описываются в виде векторов $\boldsymbol{g}$. Размеры векторов составляют максимум K двоичных элементов и полностью определяют структуру кодера. Каждый разряд конкретного двоичного кода соответствует одному отводу регистра, и единица в нем устанавливает факт наличия связи данного разряда регистра с сумматором. Двоичные последовательности $\boldsymbol{g}$ (кодовых генераторов) обычно записывают в десятичной форме или разделяют на группы по три символа и записывают в восьмеричной форме. Также используется запись $\boldsymbol{g}$ в виде полинома (порождающего полинома).

Поскольку выходной сигнал каждого сумматора представляет собой свертку соответствующего порождающего полинома и фрагмента входной информационной последовательности $\boldsymbol{m}$, состоящей из текущего входного символа и K – 1 предыдущих входных символов, то формируемые группы по n кодовых символов, соответствующих одному входному символу, можно записать в векторной форме $\boldsymbol{u_0} = u_1, \ldots, u_n$, где $u_1 = \boldsymbol{g}_1\boldsymbol{m}$, $u_2 = \boldsymbol{g}_2\boldsymbol{m}$, $u_n = \boldsymbol{g}_n\boldsymbol{m}$ (произведением обозначена свертка векторов). Формируемую кодовую последовательность также можно получить с помощью некоторой матрицы **G** (порождающей матрицы), имеющей ленточную структуру.

Исправляющие свойства сверточных кодов с разными наборами порождающих полиномов различаются. Для удобного графического исследования работы кодера и декодера эффективно использование решетчатых диаграмм. После достижения глубины анализа, равной K шагов, структура диаграммы становится постоянной. Решетчатая диаграмма используется при декодировании. Правильно декодированная

кодовая последовательность должна соответствовать на каждом шаге только одному из нескольких разрешенных путей по решетке. Суммарное количество отличий принятой последовательности от разрешенных вариантов, измеренное в определенной метрике, служит указанием для восстановления передаваемого информационного сообщения и освобождения его от ошибок. Для каждого набора исходных данных (кодовой скорости, кодового ограничения) известны наиболее эффективные в смысле исправления ошибок коды, хотя могут использоваться и коды с близкой структурой, несколько уступающие им по эффективности.

Структура систематических сверточных кодеров незначительно отличается от структуры несистематических кодеров, представленной на рис. 2. Различия заключаются в том, что один из входов выходного коммутатора подключен не к выходу одного из сумматоров по модулю 2, а непосредственно к входу кодера. В результате одна из компонент формируемой кодовой последовательности совпадает с входной информационной последовательностью, хотя ее символы следуют не один за другим, а разделены $n - 1$ другими кодовыми символами. Систематические сверточные коды используются редко, так как их характеристики в среднем хуже, чем у соответствующих несистематических кодов.

Кроме требования высокой эффективности исправления ошибок на вид полиномиальных генераторов накладываются и другие ограничения. Кодовые генераторы удобно представлять как полином вида $\boldsymbol{g}(\boldsymbol{X}) = a_0 + a_1 X + a_2 X^2 + \ldots + a_n X^n$, где переменная X обозначает сдвиг по времени на один такт, X^2 – сдвиг по времени на два такта и так далее; коэффициенты $a_0 \div a_n$ могут принимать нулевое или единичное значение. Важное ограничение на вид полиномов связано с возможностью распространения катастрофических ошибок при декодировании, когда конечное число ошибок в кодовых словах вызывает бесконечное число ошибок в декодированных данных.

Возможность появления катастрофических ошибок возникает, если при разложении используемых порождающих полиномов на более простые полиномы-множители у любых двух из них будет присутствовать хотя бы один одинаковый простой полином-множитель.

Известны также рекурсивные сверточные коды, которые обычно бывают систематическими. Они отличаются от нерекурсивных тем, что характеризуются бесконечным импульсным откликом и сумматоры могут находиться не только на выходах, но и на входах регистров. Несистематический кодер и систематический сверточный кодер имеют одно и то же множество кодовых последовательностей, но с другим соответствием между информационными и кодовыми словами.

3. БЛОКОВЫЕ КОДЫ

Рассмотрим сначала двоичные блоковые коды. Разнообразие известных блоковых кодов связано с особенностями кодирования и декодирования, сложностью реализации, быстродействием и помехоустойчивостью [2; 7 – 9]. Однако для целей диагностики большинство методов их получения можно объединить в две связанные между собой группы. В одной из них используются порождающие матрицы, в другой – порождающие полиномы.

Любой блоковый код основан на том, что исходная информационная последовательность символов разбивается на группы, как правило, одинаковой длины k. По определенному правилу на основе значений информационных символов в группе вычисляется последовательность из b проверочных символов и добавляется к исходной информационной последовательности, формируется выходной кодовый блок длиной n.

Если длина информационной части блока невелика, то наиболее простой метод кодирования – использование таблицы соответствия. Она содержит все варианты информационных последовательностей и соответствующие им варианты кодовых блоков. При кодировании на основе каждой новой информационной части для передачи выбирается нужный кодовый блок. Размер таблицы пропорционален 2^k, поэтому при большой длине информационных частей сама таблица и время кодирования могут стать недопустимо большими. В этом случае задачу можно упростить: не хранить в памяти кодовые блоки, соответствующие поступающим информационным группам, и, следовательно, не отыскивать их в таблице, а каждый раз генерировать вновь.

Пусть $\boldsymbol{m}$ – вектор, содержащий k двоичных элементов и составленный из группы исходных информационных символов. Тогда, если имеется произвольный базис из k линейно независимых двоичных векторов, любой информационный вектор можно записать как линейную комбинацию некоторых векторов этого базиса. (При составлении линейной комбинации под умножением понимается операция «и», под сложением – операция «исключающее или».) Таким образом, если имеется набор k линейно независимых двоичных векторов $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_k$, каждый из которых состоит из n элементов, то вектор $\boldsymbol{u}$, описывающий любой кодовый блок, можно записать как $\boldsymbol{u} = m_1\boldsymbol{V}_1 + m_2\boldsymbol{V}_2 + \ldots + m_k\boldsymbol{V}_k$.

Порождающая матрица $\mathbf{G}$ имеет размер $k \times n$ и представляет собой набор векторов $\boldsymbol{V}_1 \div \boldsymbol{V}_k$ как строк:

$$\mathbf{G} = \begin{Vmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \vdots \\ \boldsymbol{V}_k \end{Vmatrix} = \begin{Vmatrix} v_{1,1},\ v_{1,2},\ \ldots,\ v_{1,n} \\ v_{2,1},\ v_{2,2},\ \ldots, v_{2,n} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ v_{k,1},\ v_{k,2},\ \ldots, v_{k,n} \end{Vmatrix},$$

где $v_{i,j}$ – элементы с индексом j вектора $\boldsymbol{V}_i$.

Генерация кодового слова в матричной форме описывается уравнением $\boldsymbol{u} = \boldsymbol{m}\mathbf{G}$. Причем при кодировании в памяти необходимо сохранять только k векторов $\boldsymbol{V}_1 \div \boldsymbol{V}_k$, а не 2^k, как при использовании таблицы соответствия.

Матрицу $\mathbf{G}$ можно представить как состоящую из двух блоков: первый блок $\mathbf{G}_1$ размером $k \times k$, второй блок $\mathbf{G}_2$ размером $k \times (n - k)$, т. е. $\mathbf{G} = \|\mathbf{G}_1 \vdots \mathbf{G}_2\|$. Матрица $\mathbf{G}_1$ определяет вид информационной части кодового слова, матрица $\mathbf{G}_2$ – вид проверочной части кодового слова. При использовании несистематического кодирования вид матрицы $\mathbf{G}_1$ может быть произвольным (при сохранении условия линейной независимости строк). При использовании систематического кодирования $\mathbf{G}_1$ является единичной матрицей, $\mathbf{G}_1 = \mathbf{E}$. Несистематическое кодирование обеспечивает такие же характеристики помехоустойчивости,

как и систематическое, однако в последнем случае процесс кодирования существенно упрощается и ускоряется, поэтому чаще используется систематическое кодирование. К тому же порождающая матрица для несистематического кодирования легко преобразуется в матрицу для систематического кодирования путем умножения на обратную матрицу $\mathbf{G}_1^{-1}$.

После переобозначения части матрицы, отвечающей за формирование проверочной части кодового слова: $\mathbf{G}_2 = \mathbf{P}$, – порождающая матрица приобретает вид $\mathbf{G} = \|\mathbf{E} \vdots \mathbf{P}\|$. При подробном рассмотрении структуры генерируемого кодового слова видно, что каждый проверочный бит имеет свое происхождение и является «индивидуальной» комбинацией каких-либо информационных битов. При этом очевидно, что по сравнению с контролем четности методом дублирования разряда или применением одного бита четности данный метод обеспечивает более широкие возможности для исправления возникающих при передаче ошибок.

Некоторые виды блоковых кодов в силу различных причин используются особенно часто. Рассмотрим некоторые из них.

Циклические коды. Представляют собой класс блоковых кодов, кодирование и декодирование которых основано на использовании полиномиальных представлений, более простых, чем матричные процедуры. Широкое распространение этот класс кодов получил в связи с удобством практической реализации на основе достаточно простой современной цифровой элементной базы. Линейный код называется циклическим, если при любом циклическом сдвиге некоторого кодового слова получается другое кодовое слово, которое может быть получено тем же кодером, но из другой входной информационной группы.

Значения символов кодового слова можно рассматривать как коэффициенты полинома $\boldsymbol{u(X)} = u_0 + u_1 X + u_2 X^2 + \ldots + u_{n-1} X^{n-1}$. Наличие или отсутствие каких-либо членов в полиноме означает наличие нулей в соответствующих разрядах кодового слова. Если $u_{n-1} \neq 0$, то порядок полинома равен $n - 1$.

Циклический код создается с помощью полиномиального генератора $g(X)$. Для заданного циклического кода (n, k) вид полиномиального генератора является единственным:

$$\boldsymbol{g(X)} = g_0 + g_1 X + g_2 X^2 + \ldots + g_b X^b.$$

Предполагается, что коэффициенты g_0 и g_b – ненулевые, $b = n - k$. Каждый полином, описывающий какое-либо кодовое слово, имеет вид $\boldsymbol{u(X) = m(X)g(X)}$, т. е. кодовая последовательность двоичных символов действительно является кодовым словом, сформированным данным кодером, если оно делится без остатка на $\boldsymbol{g(X)}$. Полиномиальный генератор – это один простой множитель или произведение нескольких простых множителей полинома $X^n + 1$. Использование циклического кодирования в описанном виде формирует несистематические коды, т. е. в кодовом слове $\boldsymbol{u(X)}$ нет фрагмента, все символы которого совпадали бы с вектором $\boldsymbol{m(X)} = m_0 + m_1 X + m_2 X^2 + m_3 X^3 + \ldots + m_{k-1} X^{k-1}$.

Однако чаще используется систематическая форма кодов, упрощающая процедуры обработки. При систематическом кодировании вектор $\boldsymbol{m(X)}$ совпадает с фрагментом выходного кодового слова $\boldsymbol{u(X)}$ и используется как его часть. Для этого символы информационного сообщения изначально сдвигаются в сторону бóльших степеней X на $n - k$ позиций с помощью умножения на X^{n-k}. При этом формируется полином

$$X^{n-k}\boldsymbol{m(X)} = m_0 X^{n-k} + m_1 X^{n-k+1} + m_2 X^{n-k+2} + m_3 X^{n-k+3} + \ldots + m_{k-1} X^{n-1}.$$

Далее он делится на выбранный порождающий полином $\boldsymbol{g(X)}$. Результат деления можно записать в виде

$$X^{n-k}\boldsymbol{m(X)} = \boldsymbol{q(X)m(X)} + \boldsymbol{p(X)}, \qquad (1)$$

где $\boldsymbol{q(X)}$ – частное от деления; $\boldsymbol{p(X)}$ – остаток от деления, равный

$$\boldsymbol{p(X)} = p_0 + p_1 X + p_2 X^2 + \ldots + p^{n-k-1} X^{n-k-1}.$$

Остаток от деления суммируется по модулю 2 с обеими частями уравнения (1), в результате получается полином

$$\boldsymbol{u(X)} = X^{n-k}\boldsymbol{m(X)} + \boldsymbol{p(X)} = \boldsymbol{q(X)m(X)}.$$

Полином $\boldsymbol{u(X)}$ – кодовое слово, поскольку делится на $\boldsymbol{g(X)}$, но его фрагмент из k символов, стоящий в области больших степеней, теперь уже совпадает с информационным вектором $\boldsymbol{m(X)}$, а фрагмент, стоящий в области $n - k$ меньших степеней, является проверочной частью кодового слова, т. е.

$$\boldsymbol{u(X)} = p_0 + p_1 X + p_2 X^2 + \ldots + p_{n-k-1} X^{n-k-1} + m_0 X^{n-k} + m_1 X^{n-k+1} + \ldots + m_{k-1} X^{n-1}.$$

Процедура кодирования иллюстрируется структурной схемой (рис. 3). Схема состоит из последовательно включенных элементов памяти $Э_1 \div Э_{n-k-1}$. Фактически она представляет собой регистр сдвига с обратными связями. В момент прихода тактового импульса каждый элемент памяти запоминает значение символа на своем входе и хранит его до поступления следующего тактового импульса. Между элементами памяти находятся сумматоры по модулю 2. На их входы поступают выходные сигналы с элементов памяти и элементов $g_1 \div g_{n-k-1}$. Если коэффициент $g_1 \div g_{n-k-1}$ равен единице, то выходной сигнал ключа (Кл.) подается на соответствующий сумматор, если он равен нулю, то суммирование не производится.

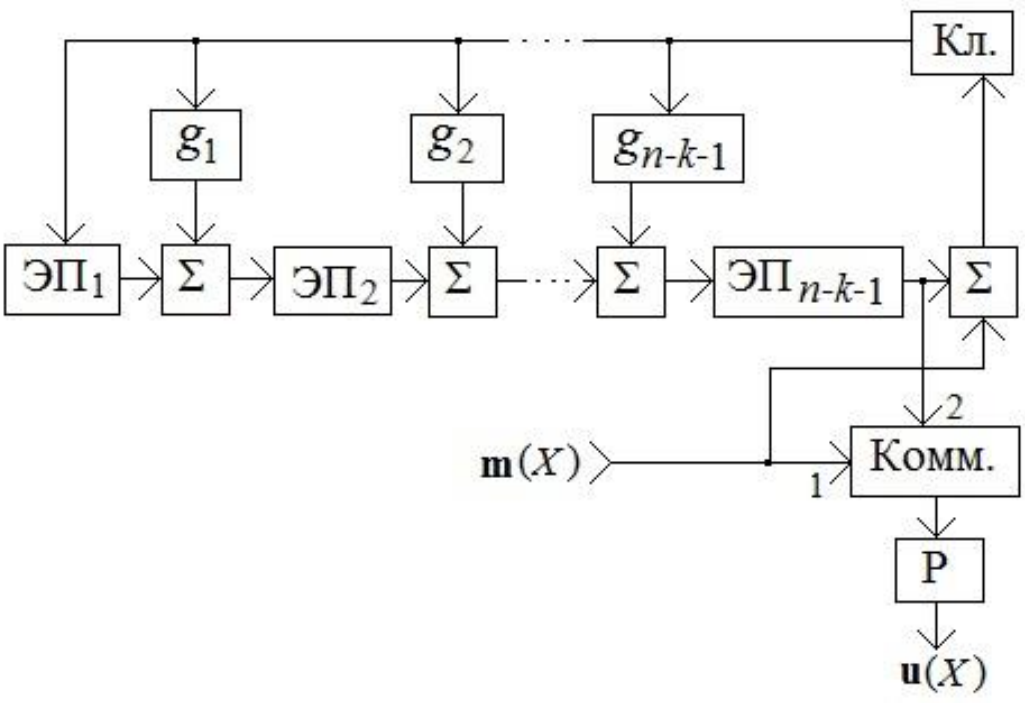


Рис. 3. Процедура блокового кодирования

Схема работает следующим образом. Появление тактовых импульсов совпадает с появлением информационных символов. При первых символах последовательности $\boldsymbol{m(X)}$ ключ закрыт, а коммутатор (Комм.) подключает на выход сигнал со своего первого входа, т. е. непосредственно исходную информационную последовательность. После передачи k-го информационного символа ключ открывается, а коммутатор на свой выход подключает сигнал со второго входа. В следующие $n - k$ такты происходит формирование проверочных символов. После окончания n тактов обработки в выходном регистре (Р) оказываются записаны все сформированные символы кодового слова, которые далее поступают в передатчик. Для повышения скорости работы кодера иногда используют комбинацию табличного метода и регистра с обратными связями.

Коды Хемминга. Это достаточно простой вид блоковых кодов, параметры которых выбираются из следующих условий: $n = 2^m - 1$; $k = 2^m - 1 - m$, где $m = 2, 3, 4, \ldots$.

Они способны исправлять все одиночные ошибки в кодовом слове. Декодирование кодов Хемминга производится достаточно просто. Несмотря на относительно небольшую эффективность по сравнению с другими кодами, коды Хемминга привлекательны тем, что они требуют при заданной длине блока минимальную избыточность для исправления одной ошибки. Кодирование производится обычно с использованием соответствующей матрицы.

Код Голея. Это линейный блоковый код с параметрами $(n, k) = (23, 12)$. Он допускает исправление до трех ошибок в кодовых словах длиной 23 символа. Из-за относительно небольшой длины кодирование и декодирование двоичного кода Голея могут быть выполнены даже с использованием соответствующих таблиц. В этом случае таблица содержит список из $2^{12} = 4096$ кодовых слов, пронумерованных непосредственно двоичной совокупностью соответствующих информационных частей блока. Код Голея имеет циклическую природу и может быть получен с помощью циклического кодирования. Его порождающий полином $\boldsymbol{g(X)} = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11}$.

Степень (скорость) кодирования кода равна 11/23, что не всегда удобно на практике, поэтому часто применяется близкий к нему расширенный код Голея с параметрами (24, 12), получаемый добавлением к кодовому слову еще одного проверочного символа. Его кодовая скорость равна 1/2, и практическая реализация проще. Код может исправлять в кодовом слове кроме всех трехсимвольных ошибок также некоторое число четырехсимвольных ошибок и является существенно более мощным, чем код Хемминга.

Коды Рида – Маллера. Представляют собой линейные блоковые коды. Для кодирования используются соответствующие матричные процедуры, коды с разными наборами параметров. Привлекательность основана на достаточно простой процедуре декодирования, базирующейся на мажоритарной логике, согласно которой решение о значении символов принимается по большинству результатов, полученных из соответствующих проверочных уравнений.

***Коды БЧХ* (*Боуза – Чоудхури – Хоквингема*).** Эти коды можно считать обобщением кодов Хэмминга, с их помощью можно исправлять несколько ошибок в блоке. Они позволяют относительно просто менять параметры кодирования, варьируя длину блока, кодовую скорость и возможности коррекции ошибок. Порождающий полином всегда имеет длину $n - k$. Коды бывают как двоичные, так и недвоичные. В наиболее часто используемых кодах применяются блоки длиной $n = 2^m - 1$, $m = 2, 3, 4, \ldots$. Известны таблицы порождающих полиномов наиболее эффективных кодов. Коды задаются корнями порождающего полинома. Иногда эффективность кодов с большой избыточностью хуже, чем кодов с меньшей избыточностью. Наибольшая эффективность достигается в зависимости от кодовой скорости при фиксированном n ориентировочно в интервале значений R от 1/3 и 3/4 [4; 12 – 14].

Коды Рида – Маллера, БЧХ и Рида – Соломона относятся к полиномиальным кодам, которые отличаются от других кодов тем, что на корни их порождающих полиномов накладываются дополнительные условия.

4. НЕДВОИЧНЫЕ И НЕЛИНЕЙНЫЕ КОДЫ

Недвоичные коды. Требуют более сложной обработки, чем двоичные, однако они более эффективны [2; 4; 19]. В качестве символов недвоичных кодов обычно рассматриваются группы битов. Наиболее часто используются коды Рида – Соломона, являющиеся недвоичными кодами БЧХ.

Коды Рида – Соломона представляют собой множество кодовых слов, компоненты которых равны значениям некоторых определенных полиномов. Обработка значений символов производится в соответствии с правилами вычислений в полях Галуа. Для их описания применяются специальные символы α, обозначающие дополнительные однозначные элементы алгебры конечных полей используемой размерности. Коды позволяют исправлять не только несколько одиночных ошибок, но и пакеты ошибок. Обычно используется систематическая форма кодов. Кодирование при этом осуществляется по схеме, сходной со схемой для двоичных кодов (см. рис. 3).

Элементы памяти запоминают не двоичный символ, содержащий один бит информации, а символ, несущий несколько информационных битов. Вместо двоичных коэффициентов g_i используются коэффициенты, образованные из степеней α, которые в данном случае также представляют несколько битов информации. Сложение, как и умножение, производится по правилам операций в конечных полях.

Коды Рида – Соломона позволяют исправить любой набор из $[(n–k)/2]$ ошибок в слове или любой набор $n – k$ стираний (т. е. определить место в блоке, в котором произошла ошибка без последующего ее исправления). Также имеется возможность одновременной коррекции определенного количества ошибок и стираний. Коды обладают существенными преимуществами перед двоичными кодами при воздействии коротких импульсных помех. Поскольку исправляется не бит, а символ с несколькими битами, то длина помехи может составлять и несколько битов, если она не выходит за длительность символа.

Нелинейные коды. Известны различные виды нелинейных кодов (коды с контрольной суммой, инверсные, коды на основе перестановок, с повторением и без повторения символов и т. д.). Построение нелинейных кодов сложнее, чем линейных, границы возможных значений их параметров обычно связаны сложным образом и более узкие, чем у линейных кодов. В настоящее время такие коды находят ограниченное применение.

5. ВИДЫ СЛОЖНЫХ КОДОВ. МОДИФИЦИРОВАННЫЕ, УКОРОЧЕННЫЕ И РАСШИРЕННЫЕ КОДЫ

Сложные коды получаются в результате различных операций над простыми кодами. Последние создаются с помощью методов сверточного и блокового кодирования [4; 12 – 14]. Рассмотрим коды, модифицированные путем добавления или удаления символов; комбинации, получаемые в результате совместного использования кодов; коды, предназначенные для декодирования путем повторения нескольких последовательных процедур (итеративно декодируемые коды); коды, использующие особенности модуляции сигналов (сигнально-кодовые конструкции).

Модифицированные коды. Строятся на основе некоторого исходного кода: либо к нему добавляют дополнительные символы (расширенный код), либо сокращают часть информационных символов без изменения расстояния между кодовыми символами (укороченный код), либо выбрасывают часть символов (перфорация, или выкалывание). Таким образом, достигается определенная гибкость в получении нужных параметров передачи информации. Например, укорочение эффективно, когда необходимо применить код Хэмминга, но при этом требуется число кодовых символов, не равное 2^m, m – целое.

Укорочение кодов. Происходит путем отбрасывания определенного числа s позиций в информационной части кодового слова (s – глубина укорочения). В слове на передачу информации теперь отводится только $k - s$ символов, но длина проверочной части $n - k$ остается прежней. Укороченное кодовое слово формируется посредством того, что в некоторых фиксированных позициях информационной части исходного (неукороченного) кодового слова устанавливаются нули. Остальные позиции могут принимать произвольные значения в зависимости от информации, передаваемой в этом кодовом слове. Положение нулевых позиций принципиального значения не имеет, однако для практического удобства кодирования и декодирования обычно выбирается s старших позиций. Таким образом, информационная часть кодового слова приобретает вид

$$\boldsymbol{m(X)} = m_0 + m_1X + m_2X^{\,2} + m_3X^{\,3} + \ldots + m_{k-1}X^{\,k-1-s}.$$

Далее кодовое слово систематическим кодером сдвигается на $n - k$ разрядов, делится на порождающий полином, и остаток от деления записывается в младшие пустые разряды. Образуется укороченный линейный код, степень которого не превышает $n - s - 1$. В общем случае такой код не остается циклическим.

Если для кодирования используется порождающая матрица $\mathbf{G} = \|\mathbf{E} \vdots \mathbf{P}\|$, то она также должна быть модифицирована. Для этого из нее удаляют s выбранных столбцов и одновременно s строк, соответствующих ненулевым элементам удаленных столбцов.

Пусть исходную матрицу можно записать в виде

$$\mathbf{G} = \left\| \begin{array}{l} 1, 0, 0, ..., 0;\ g_{1,k+1},\ g_{1,k+2}, ..., g_{1,n} \\ 0, 1, 0, ..., 0;\ g_{2,k+1},\ g_{2,k+2}, ..., g_{2,n} \\ ..... \\ 0, 0, 0, ..., 1;\ g_{k,k+1},\ g_{k,k+2}, ..., g_{k,n} \end{array} \right\| = \left\| \begin{array}{l} 1, 0, 0, ..., 0;\ p_{1,k+1},\ p_{1,k+2}, ..., p_{1,n} \\ 0, 1, 0, ..., 0;\ p_{2,k+1},\ p_{2,k+2}, ..., p_{2,n} \\ ..... \\ 0, 0, 0, ..., 1;\ p_{k,k+1},\ p_{k,k+2}, ..., p_{k,n} \end{array} \right\|.$$

Если при укорочении удаляется s первых столбцов (и, соответственно, s первых строк), то размер новой порождающей матрицы становится равным $(k - s) \times (n - s)$. Единичная информационная матрица становится размером $(k - s) \times (k - s)$. Элементы новой укороченной проверочной матрицы $g_{i,j} = p_{i-s,j-s}$. В результате операции укорочения уменьшается длина информационной части, в то время как число проверочных символов остается прежним. В результате может быть исправлено большее число комбинаций ошибок и помехоустойчивость передачи информации возрастает. Таким образом, укороченные коды более эффективны, чем исходные неукороченные коды.

Важное свойство укороченных кодов состоит в том, что для работы с ними могут быть использованы те же самые кодеры и декодеры, что и для неукороченных кодов. Однако для них обычно не выполняется правило циклического сдвига. При практической реализации слова обычно дополняются нулями в откинутых старших разрядах и используются те же самые алгоритмы кодирования и декодирования. А при передаче сообщений включенные нули не передаются.

Расширение кодов. Означает добавление в кодовое слово проверочных символов при сохранении в нем количества информационных символов. В результате получается код ($n + s$, k). Расширенная порождающая матрица может быть получена из исходной матрицы добавлением к ней соответствующих s столбцов. Ее размер становится равным $k \times (n + s)$. Если вводимые новые элементы обозначить через $q_{i,j}$, то расширенная матрица приобретает вид

$$\mathbf{G} = \left\| \begin{array}{l} 1,\ 0,\ 0,\ ...,\ 0;\ g_{1,k+1},\ g_{1,k+2},...,\ g_{1,n};\ q_{1,n+1},\ ...,\ q_{1,n+s} \\ 0,\ 1,\ 0,\ ...,\ 0;\ g_{2,k+1},\ g_{2,k+2},...,\ g_{2,n};\ q_{2,n+1},\ ...,\ q_{2,n+s} \\ ..... \\ 0,\ 0,\ 0,\ ...,\ 1;\ g_{k,k+1},\ g_{k,k+2},...,\ g_{k,n};\ q_{k,n+1},\ ...,\ q_{k,n+s} \end{array} \right\|.$$

Расширение кодов также повышает помехоустойчивость и изменяет кодовую скорость.

Иногда рассматривается объединенная операция расширения, модифицирующая исходный код и состоящая в добавлении в порождающую матрицу и новых строк, и новых столбцов. Ее размер становится равным $(k + s) \times (n + s)$. При этом ее часть, имеющая вид единичной матрицы, занимает не одну из половин, а меньшую ее часть. Матрица приобретает вид:

$$\mathbf{G} = \left\| \begin{array}{l} 1,\ 0,\ 0,\ ...,\ 0;\ g_{1,k+1},\ g_{1,k+2},\ ...,\ g_{1,n};\ q_{1,n+1},\ ...,\ q_{1,n+s} \\ 0,\ 1,\ 0,\ ...,\ 0;\ g_{2,k+1},\ g_{2,k+2},\ ...,\ g_{2,n};\ q_{2,n+1},\ ...,\ q_{2,n+s} \\ ..... \\ 0,\ 0,\ 0,...,\ 1;\ g_{k,k+1},\ g_{k,k+2},\ ...,\ g_{k,n};\ q_{k,n+1},\ ...,\ q_{k,n+s} \\ q_{k+1,1},\ q_{k+1,2},\ q_{k+1,3},\ ...,\ q_{k+1,n+s} \\ ..... \\ q_{k+s,1},\ q_{k+s,2},\ q_{k+s,3},\ ...,\ q_{k+s,n+s} \end{array} \right\|.$$

Кроме рассмотренных вариантов модификации известны также варианты, когда к множеству всех возможных кодовых слов добавляются новые кодовые слова (операция пополнения) или из этого множества удаляются некоторые кодовые слова (операция выбрасывания). Для осуществления операции пополнения при сохранении

линейности кода в порождающую матрицу добавляются новые линейно независимые строки. Операция выбрасывания в общем случае приводит к нелинейному коду. Для сохранения линейности возможно удалить часть строк исходной порождающей матрицы. Для систематических кодов операции выбрасывания и укорочения совпадают.

6. ПЕРФОРИРОВАННЫЕ КОДЫ

Перфорация применяется в основном для сверточных кодов и состоит в том, что из процесса передачи в канал удаляются некоторые символы с выхода кодера [2; 6 – 9]. Поскольку при этом структура собственно кодера не меняется, то и количество информационных символов не меняется. В результате формируется перфорированный код с более высокой кодовой скоростью.

Правило удаления символов задается обычно матрицей (маской) перфорации P. Она задает правило удаления выходных символов. Обычно процесс перфорации периодический, исходный (неперфорированный) код имеет скорость $R = 1/n$, n – целое. В этом случае количество N_p кодовых символов, через которое процесс повторяется, должно быть кратно n.

Элементы матрицы P равны нулю или единице. Нуль указывает, что соответствующий двоичный символ будет удален, единица – что он будет оставлен. Матрица P имеет размер $n \times N_p$. Если она содержит q нулей, то кодовая скорость после перфорации $R = N_p/(nN_p - q)$.

Перфорированный код формируется следующим образом. После прихода на исходный кодер N_p информационных символов он вырабатывает N_p групп кодовых символов, каждая из которых соответствует одному из исходных символов. Эти группы можно записать в виде векторов размера n. Далее символы каждого из векторов соотносятся с соответствующим столбцом маски перфорации. Если элемент маски равен нулю, то символ с таким номером из вектора удаляется. (После этого размеры векторов становятся неравными.) Всего будет удалено q символов. Далее из оставшихся элементов всех векторов составляется перфорированная последовательность длиной $nN_p - q$. После этого аналогично проводится следующий этап обработки.

При записи структуры перфорированного кода правило получения каждого кодового символа обычно выражается в форме восьмеричного кода, а место перфорированных символов обозначается буквой *X*. Перфорация используется достаточно широко, поскольку для различных вариантов перфорированного кода, полученных из одного и того же исходного кода, можно использовать один и тот же декодер, предназначенный для исходного кода.

Перфорация применяется также и к линейным блоковым кодам. Она состоит в удалении из кодового слова p проверочных символов. Код при этом остается линейным блоковым, количество информационных символов k в кодовом слове не изменяется, а общее количество кодовых символов становится равным $n - p$. Избыточность уменьшается, а скорость кода возрастает.

7. КОМБИНИРОВАННЫЕ И КАСКАДНЫЕ КОДЫ

Комбинированные коды. Известны различные способы комбинирования кодов [10; 15 – 17]. Рассмотрим метод *последовательного соединения* (concatenation – конкатенация). Если используются блоковые коды C_1 и C_2, то их последовательное соединение эквивалентно поочередной передаче кодовых слов, полученных с помощью первого и второго кодов. В результате последовательного соединения $i = 1 \div m$ кодов образуется комбинированный код с параметрами n_0 и k_0:

$$n_0 = \sum_{i=1}^{m} n_i , \quad k_0 = \sum_{i=1}^{m} k_i ,$$

где n_i и k_i – параметры i-го кода. Если $\mathbf{G}_i$ – исходные порождающие матрицы кодов, составляющих последовательное соединение, то порождающая матрица

$$\mathbf{G}_0 = \left\| \begin{array}{l} \mathbf{G}_1,\ 0,\ 0,\ ...,\ 0 \\ 0,\ \mathbf{G}_2,\ 0,\ ...,\ 0 \\ ..... \\ 0,\ 0,\ 0,\ ...,\ \mathbf{G}_m \end{array} \right\| .$$

Величины n_i и k_i у объединяемых кодов часто не совпадают, так как последовательно соединенные коды используют в случае необходимости разные уровни защиты от ошибок. Если же все n_i и k_i одинаковы, то это эквивалентно m-кратной повторной передаче одного кодового слова.

Другой тип комбинированных кодов – *произведения кодов*. В простейшем виде они представляют собой последовательное (повторное) кодирование. При этом выход первого кодера является входом второго кодера и так далее (рис. 4). Несмотря на простоту, с помощью этого метода формируются достаточно помехоустойчивые коды, особенно для низкой кодовой скорости. Если используются два кодера, то код первого из них обычно называется внешним, код второго – внутренним.

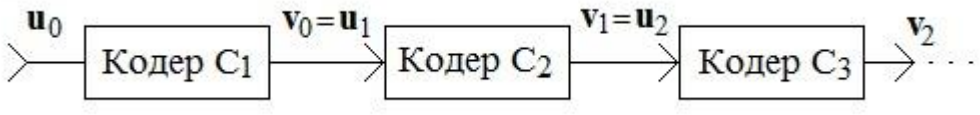


Рис. 4. Реализация комбинированного кодера

Если **G** и **H** – порождающие матрицы двух кодов с размерами $m \times n$ и $p \times r$, то порождающая матрица **Q** произведения этих кодов есть кронекерово произведение этих матриц, т. е.

$$\mathbf{Q} = \begin{Vmatrix} g_{1,1}\mathbf{H},\ g_{1,2}\mathbf{H},\ ...,\ g_{1,n}\mathbf{H} \\ g_{2,1}\mathbf{H},\ g_{2,2}\mathbf{H},\ ...,\ g_{2,n}\mathbf{H} \\ ..... \\ g_{m,1}\mathbf{H},\ g_{m,2}\mathbf{H},\ ...,\ g_{m,n}\mathbf{H} \end{Vmatrix},$$

где $g_{i,j}$ – элементы матрицы **G**.

Матрица **Q** имеет размер $mp \times nr$. Если параметры исходных кодов равны (n_1, k_1) и (n_2, k_2), то при генерации комбинированного кода кодовые слова длиной n_1 символов, уже закодированные внешним кодом, записывают по строкам прямоугольного массива с n_1 столбцами, размещая один символ в один элемент массива. Так заполняют k_2 строк, а в оставшиеся $n_2 - k_2$ строк записывают проверочные символы

внутреннего кода. При этом k_2 элементов каждого столбца считаются информационными символами внутреннего кода. Общее кодовое слово полученного кода передается из массива по столбцам.

Каскадные коды. В них чаще всего используются в качестве внешнего кода недвоичные блоковые коды Рида – Соломона. Символы полученных кодовых слов подвергаются перемежению, в результате чего порядок следования символов изменяется. Далее они кодируются с помощью внутреннего кодера, который может быть как блоковым, так и сверточным. Если внешний и внутренний коды C_1 и C_2 имеют параметры (n_1, k_1) и (n_2, k_2), то сформированный каскадный код имеет параметры (n_1n_2, k_1k_2). Если внутренний код двоичный линейный блоковый, тогда и полученный каскадный код также двоичный линейный блоковый.

Как усложнение каскадных кодов были предложены обобщенные каскадные коды. Они способны исправлять как случайные ошибки, так и случайные пакеты ошибок. Обобщение заключается в том, что вводится разбиение внутреннего кода на подкоды со своей иерархией, а также используется несколько внешних кодов в соответствии с иерархией. При этом если необходимо, то с помощью выбора параметров внутренних и внешних кодов легко получается блоковый или сверточный код с неравной защитой от ошибок. Сообщения, закодированные на верхнем уровне иерархии, имеют усиленную корректирующую способность по сравнению с нижними уровнями.

8. ТУРБОКОДЫ

Турбокоды могут быть отнесены и к комбинированным (составным) кодам, и к итеративно декодируемым кодам, так как имеют особенности обоих классов [17 – 20]. Они образуются путем компоновки нескольких (чаще двух) кодов, создаваемых простыми кодерами (компонентными) и получаемых по-разному из одной и той же информационной последовательности.

Считается, что основной выигрыш при декодировании обусловлен следующим. Обычный декодер и при «жестком», и при «мягком» декодировании выдает «жесткие» двоичные решения. Если же

применен каскадный код и декодирование производится в две ступени, то первый декодер не должен обязательно выдавать «жесткие» решения, так как его выходной результат – не окончательный результат декодирования, он имеет промежуточный характер. Таким образом, он тоже может выдавать «мягкие» решения, что значительно повышает помехоустойчивость. Если общий код скомпонован из двух различных кодов, то при параллельной обработке результат декодирования каждого из кодов поступает в целях коррекции результата декодирования на один из входов декодера другого кода и учитывается им. Причем такой результат выдается в «мягком» виде на основе метрики, построенной с использованием отношения правдоподобия.

Достаточно эффективные коды получаются даже при использовании простых компонентных кодеров. Если такие кодеры осуществляют сверточное кодирование, то более эффективны систематические рекурсивные коды. Они формируются кодерами с обратной связью, имеющими бесконечную импульсную характеристику. На рис. 5 приведен пример структурной схемы турбокодера, содержащего два компонентных рекурсивных сверточных кодера, хотя на их количество ограничений нет.

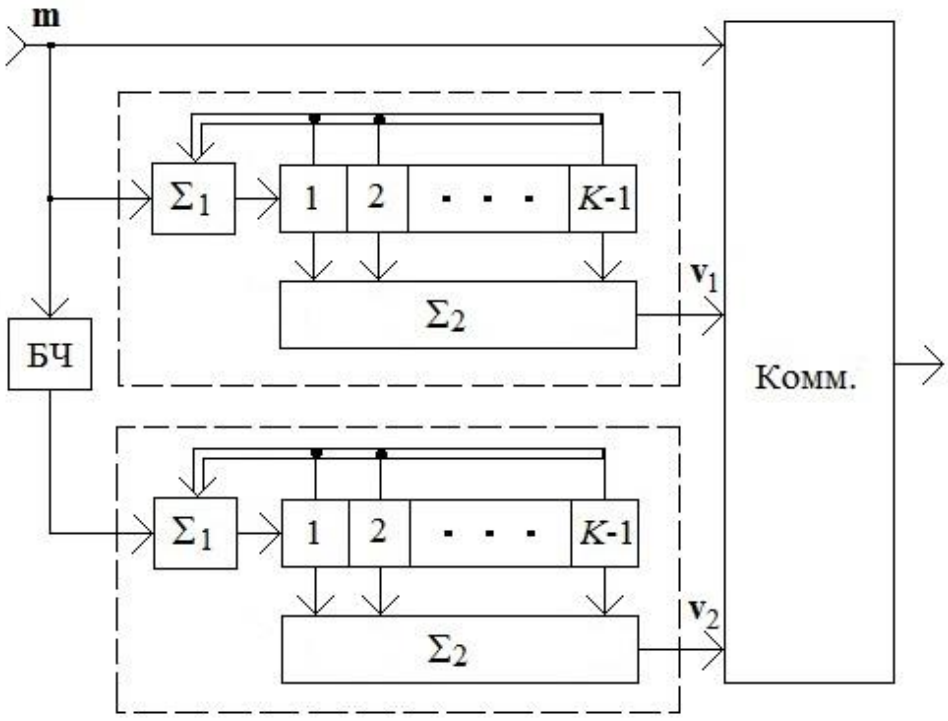


Рис. 5. Структурная схема турбокодера

Входная информационная последовательность $\boldsymbol{m}$ поступает на вход коммутатора (Комм.), а также на первый кодер C_1 и через блок чередования (БЧ) на второй кодер C_2. Оба кодера имеют сдвиговые регистры, содержащие по $K - 1$ ячеек. Обратная связь обеспечивается следующим образом: на входы регистров с помощью сумматоров Σ_1 в каждом такте подается сумма входного символа и символов из всех ячеек регистра. На сумматоры Σ_2, так же как и в нерекурсивном варианте кодера, подаются символы только с некоторых ячеек регистра, определяемых видом используемого порождающего полинома. Коммутатор поочередно подключает на выход символы последовательности $\boldsymbol{m}$ и вырабатываемых сумматорами Σ_2 последовательностей $\boldsymbol{v}_1$ и $\boldsymbol{v}_2$.

Блок чередования производит перемежение символов, изменяя порядок их следования. Последовательности символов с выходов обоих кодеров не совпадают, при этом эффективность кодирования зависит от того, как часто в обеих последовательностях будут встречаться одинаковые фрагменты. Это обусловлено правилом перемежения.

Формируемый турбокод во многом похож на рассмотренное ранее произведение кодов. Различие заключается в том, что если раньше заполнялся весь прямоугольный массив размера $n_1 \times n_2$, то теперь из него удалены $(n_1 - k_1) \times (n_2 - k_2)$ символов, т. е. все проверочные символы одного кода, полученные из проверочных символов другого кода.

9. ИТЕРАТИВНО ДЕКОДИРУЕМЫЕ КОДЫ И МНОГОПОРОГОВОЕ ДЕКОДИРОВАНИЕ

Итеративно декодируемые коды. К ним обычно относят такие коды, в которых многократно используется «мягкое» декодирование и достигаются высокие показатели помехоустойчивости при относительно небольших объемах вычислений [4; 18; 20]. Кроме турбокодов и близких к ним в настоящее время получили распространение коды с низкой плотностью проверки на четность (low-density parity-check code – LDPC) и коды для многопорогового декодирования.

LDPC-коды – это линейные коды, у которых малое число ненулевых элементов в порождающей матрице. Их можно рассматривать как турбокоды, составными кодами которых является совокупность простейших кодов. Во многих исследованиях указывается, что в определенных условиях по эффективности они могут превосходить турбокоды. Регулярный LDPC-код – это линейный код, у которого количество единичных элементов в каждой строке и каждом столбце проверочной матрицы, связанной с порождающей матрицей, много меньше количества символов в кодовом слове (матрица сильно разрежена, ориентировочно содержание единиц может быть порядка 1 % и ниже). Кроме этого почти всегда накладывается ограничение, состоящее в том, чтобы в матрице не было двух строк или столбцов, одновременно содержащих единицы более чем в одной позиции.

В нерегулярных LDPC-кодах распределение единичных элементов в строках и столбцах матрицы выбирается в соответствии с некоторым неравномерным распределением вероятности. Они могут обеспечить несколько бóльшую помехоустойчивость, чем регулярные коды.

Многопороговые методы декодирования. Многопороговое декодирование также основано на итеративных процедурах и может быть применено как для достаточно простых сверточных и блоковых кодеров, так и для сложных каскадных кодеров. Эффективность кодов сравнима с эффективностью LDPC-кодов, но практическая реализация при схожих характеристиках, как правило, проще.

Количество информационных символов в блоковом коде для кодовой скорости $R = 1/2$ равно как минимум $2q + 1$, где q – наибольшая степень порождающего полинома. Иногда для кодовых скоростей $R = 1/n$ с целью уменьшения риска размножения ошибок от кодов вида (n, k) переходят к кодам вида (qn, qk), где q – небольшое целое число. Общая кодовая скорость $R = qk/qn$ при этом сохраняется.

В кодере структура кодового слова формируется следующим образом. Группа символов входного информационного потока первоначально разделяется на q групп, и из них формируется q проверочных групп символов, далее все группы передаются поочередно, таким образом складывается общее кодовое слово. Для формирования каждой

проверочной группы одновременно используются все информационные группы. Пример структуры подобного кодера для $q = 2$ и кодовой скорости $R = 2/4$ приведен на рис. 6.

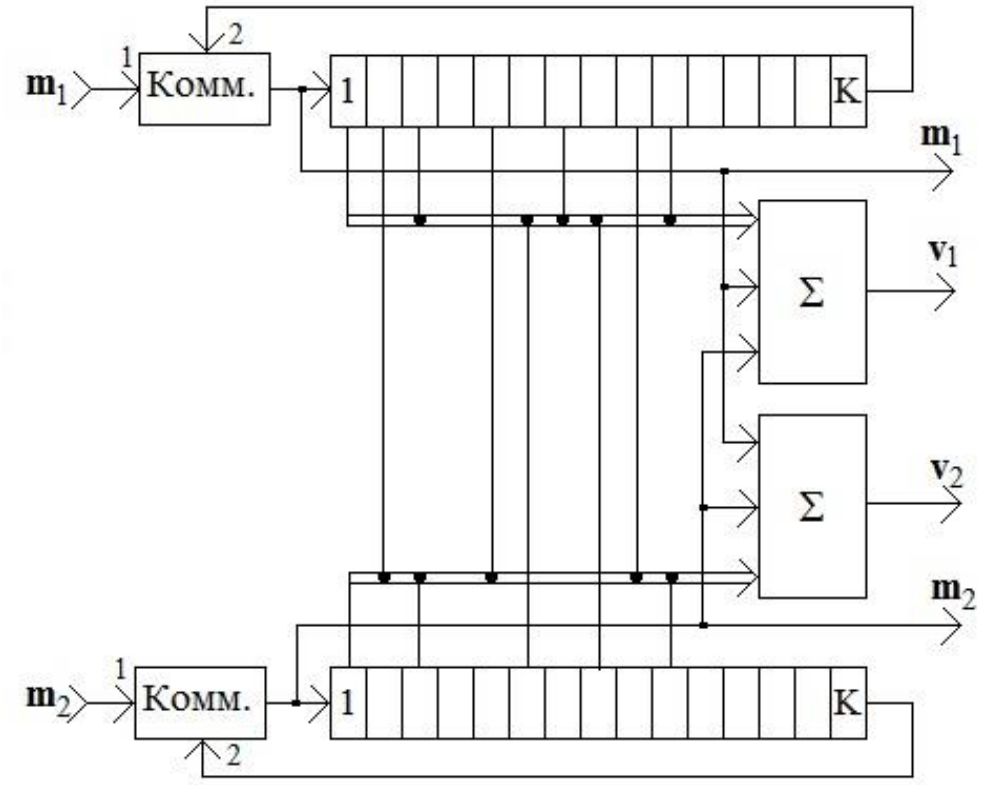


Рис. 6. Схема многопорогового метода кодирования

Информационная группа символов $\boldsymbol{m}$ разбивается на две группы $\boldsymbol{m}_1$ и $\boldsymbol{m}_2$ одинаковой длины K, которые синхронно поступают на входы *1* двух коммутаторов (Комм.). В течение K тактов оба коммутатора соединяют свои первые входы с выходом. При этом K информационных символов подаются на выходы кодера и одновременно заполняют K ячеек сдвиговых регистров. Далее оба коммутатора на выходы подключают свои вторые входы *2*, после этого вычисляются K проверочных символов двух проверочных последовательностей $\boldsymbol{v}_1$ и $\boldsymbol{v}_2$ и подаются на выходы кодера. К входам каждого из сумматоров по модулю 2 (Σ) подключены некоторые ячейки как первого, так и второго регистров. Номера этих ячеек определяются используемыми порождающими полиномами. Таким образом, сформированное кодовое слово образуется из групп символов $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, $\boldsymbol{v}_1$, $\boldsymbol{v}_2$.

10. СИГНАЛЬНО-КОДОВЫЕ КОНСТРУКЦИИ

Для сигнально-кодовых конструкций характерно совместное взаимоувязанное использование модуляции и кодирования (решетчатое кодирование, trellis coded modulation – TCM). Они предназначены для повышения эффективности использования ограниченного частотного диапазона [2; 4]. В случае цифровой модуляции сигнальное множество представляет собой набор сигналов с определенными параметрами (как правило, амплитудно-фазовыми). Каждому сигналу из набора приписывается определенное сочетание нескольких битов кодового слова. Основной результат достигается за счет следующих факторов. Переход от двоичной модуляции к набору сигналов позволяет повысить скорость передачи в той же полосе частот. Однако увеличение набора сигналов снижает помехоустойчивость при ограничении предельной мощности передатчика. В таком случае часть ресурса увеличения скорости передачи можно потратить на введение кодирования, которое компенсирует снижение помехоустойчивости.

Этот же фактор можно использовать и при увеличении количества уже имеющихся символов в наборе. Дополнительные символы позволяют применять тот или иной метод кодирования, что, в свою очередь, повышает помехоустойчивость передачи. Правильный выбор параметров модуляции и кодирования дополнительно повышает помехоустойчивость.

Кроме того, большое значение имеет правильный выбор соотнесения сигналов из используемого набора и вариантов последовательности битов кодового слова. В частности, если набор сигналов представляет собой «созвездие» на фазовой плоскости, то близко расположенным сигнальным точкам присваиваются такие сочетания символов, которые в результате применения кодирования следовать друг за другом не могут. Этот фактор различного уровня вероятности ошибок дополнительно повышает помехоустойчивость.

Формирование передаваемых сигналов в подобных системах передачи информации фактически разбивается на два этапа. На первом этапе производится собственно кодирование сигналов в их логическом

представлении с учетом последующего соответствия используемому набору сигналов. А на втором этапе фрагменты кодированной последовательности символов приписываются передаваемым сигналам в их аналоговом представлении. При этом первый этап использует известные схемы кодеров.

11. ДЕКОДИРОВАНИЕ СВЕРТОЧНЫХ КОДОВ. АЛГОРИТМ ВИТЕРБИ

При декодировании сверточных кодов могут быть использованы различные алгоритмы. Для пояснения методов декодирования первоначально рассмотрим простейший алгоритм кодирования [1 – 4; 13] входного сообщения 110 с помощью сверточного кодера со скоростью 1/2 и $K = 3$ (рис. 7, 8).

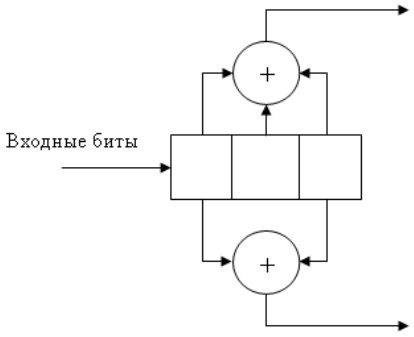


Рис. 7. Сверточный кодер со степенью кодирования 1/2 и $K = 3$

Все символы по очереди поступают на вход регистра сдвига. В момент времени t_1 поступает символ 1. В результате операции суммирования по модулю 2 получается выходное сообщение 11. Аналогично происходит и в моменты времени t_2 и t_3. Далее вводятся два нулевых входных бита для очистки регистра сдвига. В результате на выходе получается кодированная последовательность 11010111.

Любой сверточный код можно представить с помощью так называемого генератора кода, или вектора связи. Генератор кода показывает наличие или отсутствие связи регистра сдвига с сумматором по модулю 2. Если на i-й позиции вектора присутствует символ 1, то соответствующий разряд в регистре сдвига связан с сумматором по модулю 2, а символ 0 указывает на отсутствие такой связи. Например, для кодера, изображенного на рис. 8, генератор кода для верхних и нижних связей выглядит следующим образом:

$g_1 = 111$;

$g_2 = 101$.

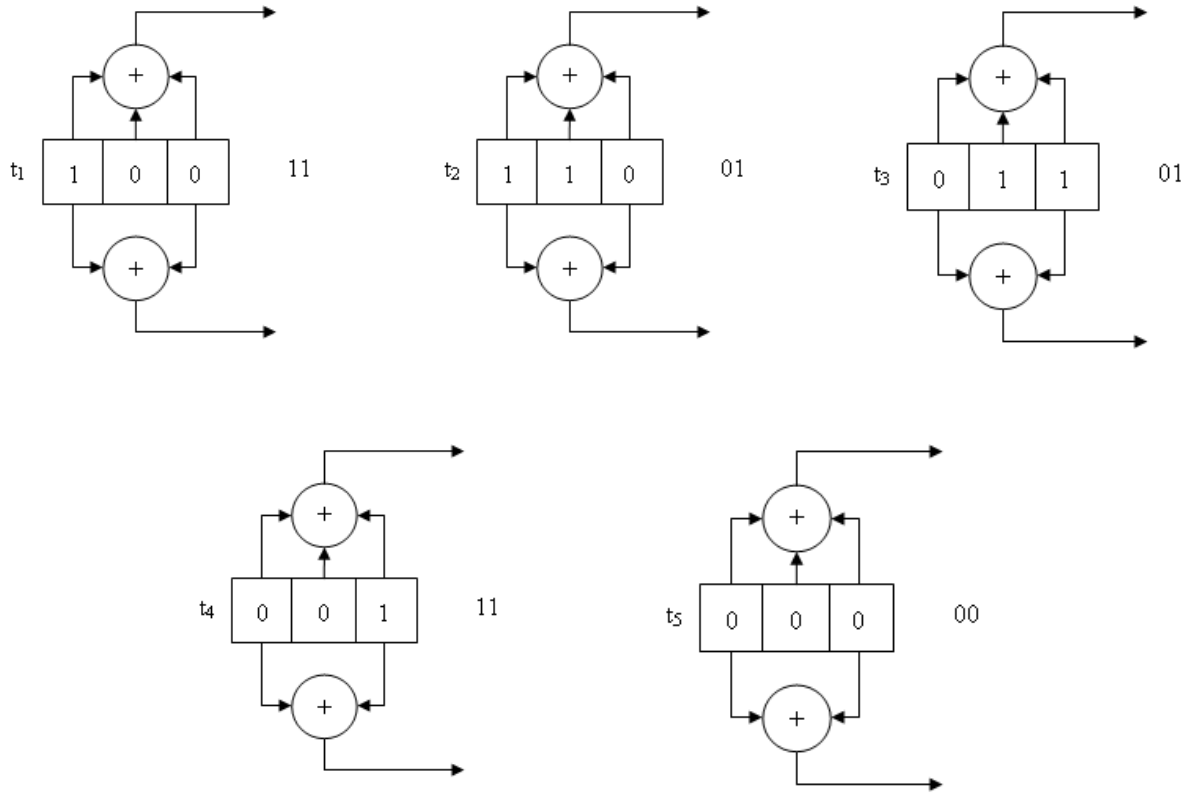


Рис. 8. Процесс кодирования сообщения 110

На практике кодовые генераторы записываются в восьмеричной системе. Рассмотренный выше код имеет вид (7, 5).

Для того чтобы описать сверточный код, необходимо указать кодирующую функцию, т. е. функцию, с помощью которой можно по данной входной последовательности символов определить выходную последовательность. Существует несколько способов задания такой функции: графическая связь, полиномы связи, диаграмма состояний, древовидная и решетчатая диаграммы. Последняя наиболее удобна и наглядна в сравнении со всеми остальными. Рассмотрим принцип построения и структуру решетчатой диаграммы.

На рис. 9 представлена решетчатая диаграмма, соответствующая коду (7, 5). Она показывает все возможные переходы кодера из предыдущего состояния в последующее. Решетка состоит из $2^K - 1$ узлов, где K – длина кодового ограничения. Каждый узел характеризует состояние кодера, т. е. состояние регистра сдвига. Из каждого текущего

состояния кодера можно перейти в одно из двух последующих состояний. При этом одна ветвь соответствует входному нулевому биту, а другая – входной единице. Цифры над переходом обозначают кодовые слова на выходе кодера, сплошная линия – входной нуль, а пунктирная – входную единицу. Начиная с глубины, равной K, решетка имеет периодическую структуру.

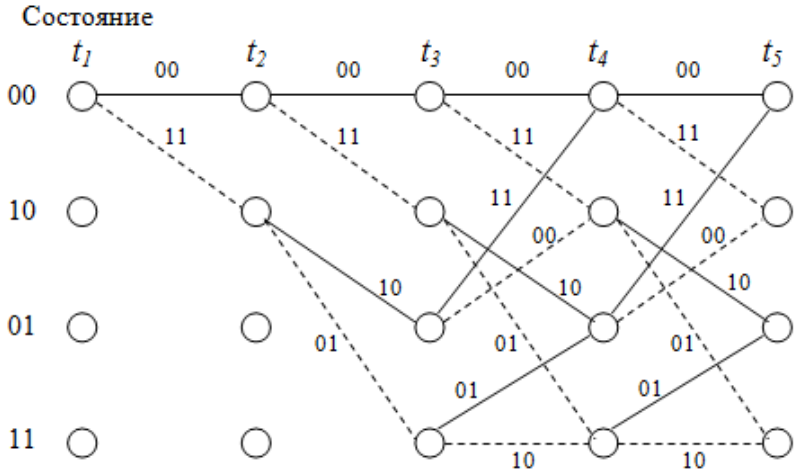


Рис. 9. Решетчатая диаграмма кодера (7, 5)

Если на выходе декодера с некоторой периодичностью проводить выкалывание, или перфорацию, кодовых символов, т. е. не передавать их по каналу связи, то получим перфорированный код. Как известно, наиболее просты в реализации коды со скоростью R=1/2. Но часто бывает необходимо повысить скорость кода без изменения его структуры. Этого можно достигнуть путем перфорации символов на выходе кодера. Так как при этом структура решетки исходного низкоскоростного кода не изменяется, то с помощью одного и того же исходного кода можно получить другие высокоскоростные коды. При этом скорость кода увеличится: R = 2/3. Для обозначения правила удаления выходных символов используется матрица перфорации. В обозначенном выше примере матрица перфорации выглядит следующим образом:

$$P = \begin{pmatrix} 5 & 5 \\ 7 & X \end{pmatrix},$$

где X указывает местоположение вычеркнутого символа. Для примера приведем (табл. 1) различные матрицы перфорации на базе стандартного сверточного кода с исходной скоростью $R = 1/2$ и $K = 7$ (код NASA).

Таблица 1

Матрицы перфорации для различных кодовых скоростей

Скорость кода R	Матрица перфорации
1/2	$(133 \quad 171)$
2/3	$\begin{pmatrix} 133 & 133 \\ 171 & X \end{pmatrix}$
3/4	$\begin{pmatrix} 133 & 133 & X \\ 171 & X & 171 \end{pmatrix}$
4/5	$\begin{pmatrix} 133 & 133 & 133 & 133 \\ 171 & X & X & X \end{pmatrix}$
5/6	$\begin{pmatrix} 133 & 133 & X & 133 & X \\ 171 & X & 171 & X & 171 \end{pmatrix}$
6/7	$\begin{pmatrix} 133 & 133 & 133 & X & 133 & X \\ 171 & X & X & 171 & X & 171 \end{pmatrix}$
7/8	$\begin{pmatrix} 133 & 133 & 133 & 133 & X & 133 & X \\ 171 & X & X & X & 171 & X & 171 \end{pmatrix}$

Рассмотрим решетчатую диаграмму перфорированного сверточного кода $\begin{pmatrix} 5 & 5 \\ 7 & X \end{pmatrix}$ (рис. 10). Она состоит из двух повторяющихся шагов. На первом шаге присутствуют ветви, соответствующие первому столбцу матрицы перфорации. На втором шаге вычеркнутый символ не передается по каналу связи. Таким образом, декодирование сверточных кодов, полученных путем перфорации первоначального кода со скоростью 1/2, возможно декодером, который реализован для этой скорости.

Существует несколько способов декодирования сверточных кодов, среди которых наиболее часто используется *алгоритм декодирования Витерби*. В нем применяется решетчатое представление и сразу

отбрасываются пути, не соответствующие критерию максимального правдоподобия. Работа ведется только с «выжившими» путями в полном соответствии с принципом максимального правдоподобия.

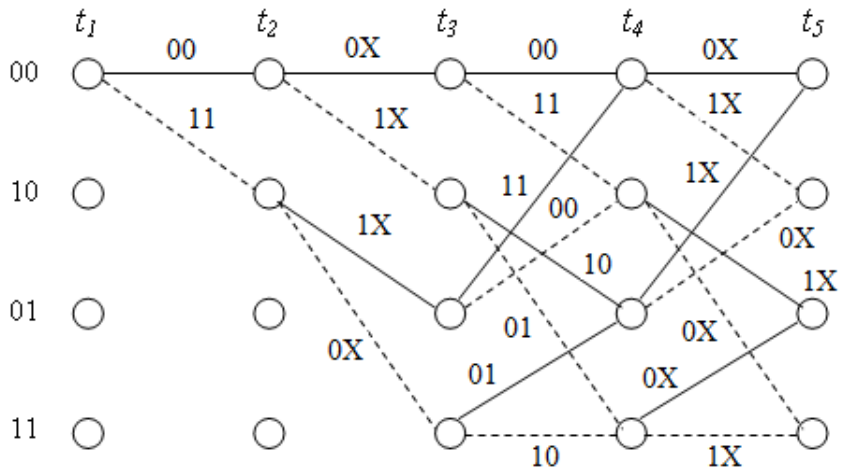


Рис. 10. Решетчатая диаграмма перфорированного кода

Алгоритм декодирования Витерби был разработан в 1967 г. с целью уменьшения объема вычислений по сравнению с алгоритмом последовательного декодирования. В 1969 г. было показано, что данный алгоритм основывается на оценке максимального правдоподобия.

Главное достоинство алгоритма Витерби заключается в том, что в нем не рассматриваются пути, которые согласно принципу максимального правдоподобия не могут быть оптимальными. Алгоритм включает в себя операции вычисления расстояния между принятым сигналом в момент времени t_1 и всеми путями решетки, которые входят в каждое состояние в момент времени t_i. Если в одно состояние входят два пути, то выбирается «выживший» путь с наименьшей метрикой. В результате работы декодер постепенно проходит решетку и исключает наименее вероятные пути.

Рассмотрим работу алгоритма Витерби на примере. В качестве меры расстояния примем метрику Хэмминга. Воспользуемся кодером, изображенным на рис. 7, и соответствующей ему решетчатой диаграммой (см. рис. 9).

Предположим, что мы имеем входную информационную последовательность m = 10010. После кодирования ее сверточным кодером получаем последовательность U = 1110111110, которая передается по

каналу связи. В результате воздействия шума принятая последовательность имеет вид Z = 1110011110, т. е. искажается один символ, а именно пятый бит.

На рис. 11 представлена решетчатая диаграмма, в которой над каждой ветвью обозначено расстояние Хэмминга между принятым кодовым символом и кодовым словом, соответствующим данной ветви.

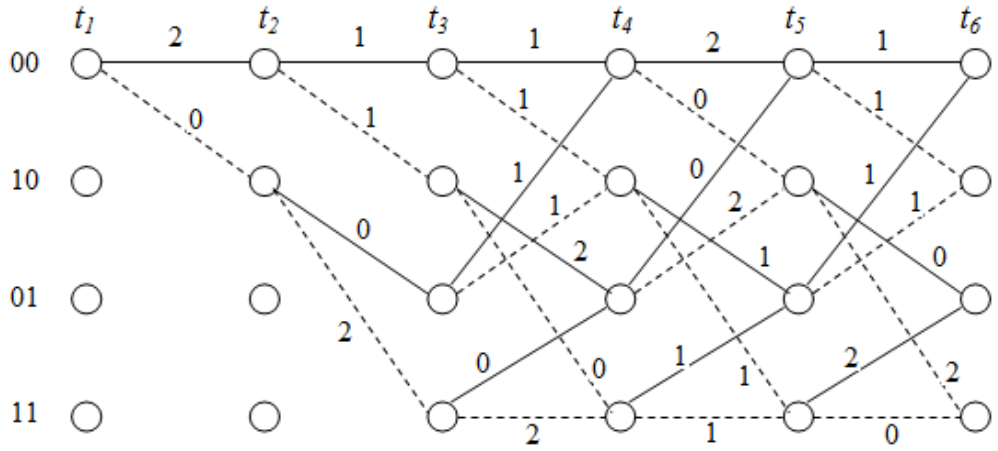
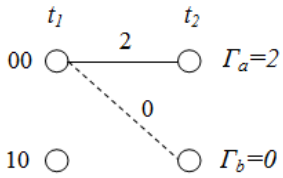


Рис. 11. Решетчатая диаграмма декодера (7, 5)

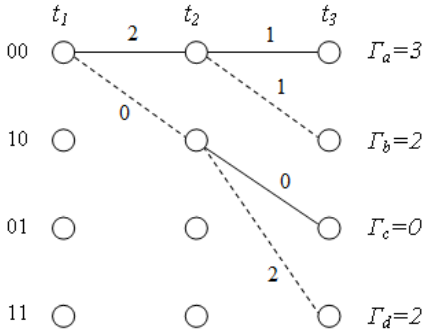
Рассмотрим решетку в момент времени t_1. Переход между состояниями 00→00 приводит к появлению на выходе кодового слова 00, но получено 11, следовательно, Хэммингово расстояние равно двум. Переход между состояниями 00→10 приводит к появлению на выходе кодового слова 11, что полностью совпадает с полученной последовательностью, и, значит, Хэммингово расстояние равно нулю. Таким образом помечается вся решетка в последующие моменты времени.

Теперь рассмотрим детально работу алгоритма декодирования Витерби (рис. 12). Основной смысл алгоритма Витерби заключается в том, что если два пути в решетке сходятся в одной точке, то при поиске оптимального пути исключается путь, имеющий большую суммарную метрику. В случае совпадения суммарных метрик путь выбирается произвольно.

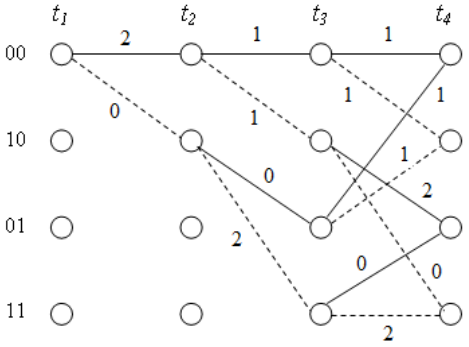
В нашем случае в момент времени t_1 приняты кодовые символы 11. Переходу между состояниями 00→00 соответствует метрика ветви 2, а переходу между состояниями 00→10 – метрика ветви 0 (см. рис. 12, *а*).



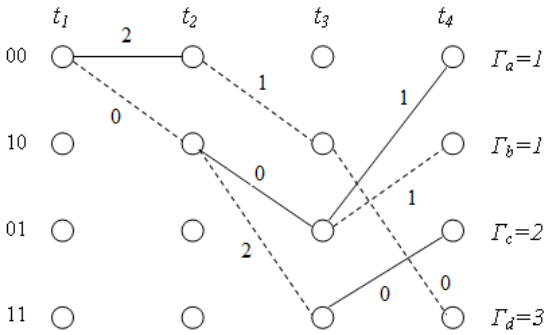
а)



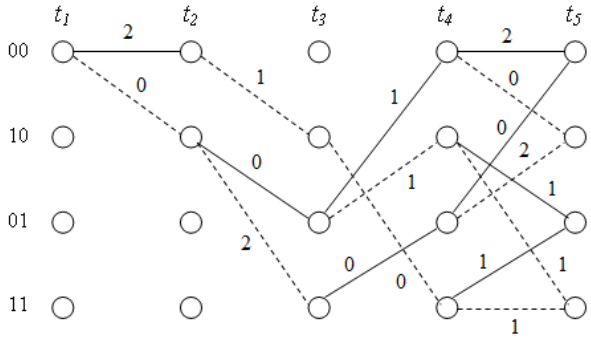
б)



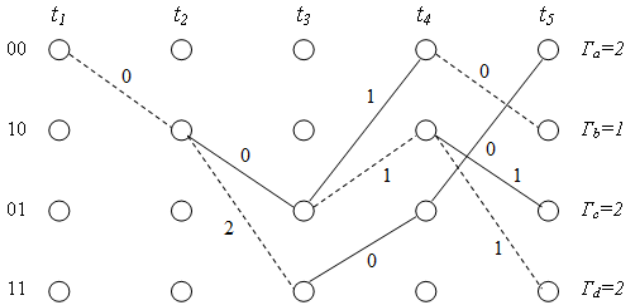
в)



г)



д)



е)

Рис. 12. Пример работы алгоритма Витерби

В следующий момент времени t_2 из каждого предыдущего состояния выходят еще две ветви (рис. 12, *б*). Через Γ_a, Γ_b, Γ_c и Γ_d обозначены суммарные метрики путей. В момент времени t_3 опять происходит разветвление путей (рис. 12, *в*), и в момент времени t_4 имеется по два пути, входящих в каждое состояние. Путь, имеющий наибольшую суммарную метрику, может быть исключен. В результате на рис. 12, *г* показаны «выжившие» пути. Та же самая процедура происходит в момент времени t_5 (рис. 12, *д*). На рис. 12, *е* показаны «выжившие» пути в текущий момент времени. Здесь между моментами времени t_1 и t_2 остался только один «выживший» путь, который называется полной ветвью. При этом декодер, исходя из свойств решетки, приходит к выводу, что сделан переход 00→10, которому соответствует единичный входной бит. Далее на каждом следующем шаге происходит исключение одного из путей, ведущих в каждое состояние. Таким образом, декодер постепенно проходит вглубь решетки и устраняет все пути, кроме одного. Следует заметить, что первый бит декодируется только тогда, когда декодер углубится внутрь решетки на некоторое расстояние. Обычно оно равно 4 – 5 длинам кодового ограничения.

Из анализа решетчатой диаграммы сверточного кода можно сделать вывод, что непересекающихся ячеек 2^{K-2}, каждая из них включает 2^{K-1} возможных перехода. Из этих ячеек и логических элементов, которые корректируют метрики состояний, и состоит декодер. Каждая ячейка реализуется с помощью операций сложения, сравнения и выбора. Поскольку практическая реализация этих операций не составляет большой сложности, то очевидное преимущество алгоритма Витерби – возможность вносить в него некоторые внутренние корректировки применительно к различным внешним условиям.

До появления алгоритма Витерби использовался *алгоритм последовательного декодирования*. Он был предложен Дж. Возенкрафтом и доработан Р. Фано. Главное его достоинство – способность декодировать сверточные коды с большой длиной кодового ограничения ($K > 9$).

Сущность данного алгоритма заключается в том, что происходит обновление метрики только наиболее вероятного пути. Данная метрика рассчитывается как разность между принятым сигналом и гипотезой о переданной последовательности кодовых символов. Если достоверность пути ниже некоторого текущего порога, то декодер последовательно перебирает все другие пути, пока не найдет наиболее правдоподобный путь. Таким образом, данный алгоритм можно сравнить с методом проб и ошибок для поиска правильного пути по решетке.

Главный недостаток алгоритма последовательного декодирования – зависимость числа перебираемых метрик состояний от соотношения сигнал/шум в канале передачи. Из-за такой зависимости требуется большой объем памяти для хранения поступивших последовательностей. Иногда возникают ситуации, когда происходит переполнение буфера памяти. Так как алгоритм последовательного декодирования не является алгоритмом максимального правдоподобия, то при прочих равных условиях он уступает алгоритму Витерби.

Алгоритм декодирования с обратной связью предложен Дж. Хеллером и основан на алгоритме порогового декодирования. Он предназначен для принятия «жестких» решений в двоичном симметричном канале.

Решение об информационном символе на j-м шаге принимается исходя из метрик, вычисленных от шага j до $j + m$ шага, где m – целое положительное число. Вводится параметр – длина упреждения $L = m + 1$. Он определяет количество принятых кодовых символов, которое задается для декодирования информационного бита. Решение в пользу нуля или единицы принимается исходя из минимального расстояния Хэмминга для пути, который начинается на шаге j и кончается на шаге $j + m$, и количества нулей или единиц в исходящих ветвях на шаге j. Часть дерева, которая не связана с решением об информационном символе, отбрасывается, а оставшаяся часть расширяется, и рассматриваются пути от шага $j + 1$ до шага $j + 1 + m$. Данная процедура повторяется на каждом шаге.

Алгоритм с обратной связью имеет меньшую задержку (не более двух длин кодового ограничения) по сравнению с алгоритмом Витерби, но он принимает только «жесткие» решения.

Стек-алгоритм предложен Ф. Йелинеком в 1969 г. Он работает всего с несколькими путями. В верхней части стека располагается путь, имеющий наибольшую метрику. Далее на каждом шаге только головной путь проверяется по разветвлению. В результате образуется 2^K продолжения путей, которые затем вместе с другими путями упорядочиваются согласно значениям их метрик. Все пути, значения метрик которых располагаются ниже некоторого значения метрики главного пути, отбрасываются, и процесс продолжения путей с наибольшими метриками вновь повторяется.

По сравнению с алгоритмом Витерби стек-алгоритм требует меньшего числа сравнений метрик, но необходима большая вычислительная способность для его реализации.

12. ДРУГИЕ АЛГОРИТМЫ ДЕКОДИРОВАНИЯ СВЕРТОЧНЫХ КОДОВ

Последовательное декодирование. Ранее, до того как Э. Витерби предложил оптимальный алгоритм декодирования сверточных кодов, существовали и другие алгоритмы [2; 4 – 6]. Самым первым был алгоритм последовательного декодирования, предложенный Дж. Возенкрафтом и модифицированный Р. Фано. В ходе работы последовательного декодера генерируется гипотеза о переданной последовательности кодовых слов и рассчитывается метрика между этой гипотезой и принятым сигналом. Эта процедура продолжается до тех пор, пока метрика показывает, что выбор гипотезы правдоподобен, в противном случае гипотеза последовательно заменяется, пока не будет найдена наиболее правдоподобная. Поиск при этом происходит методом проб и ошибок. Для «мягкого» или «жесткого» декодирования можно разработать последовательный декодер, но обычно «мягкого» декодирования стараются избегать из-за сложных расчетов и больших требований к памяти.

Рассмотрим ситуацию, когда используется последовательность m = 11011 и она кодирована в последовательность кодовых слов

U = 1101010001. Допустим, что принятая последовательность $\boldsymbol{Z}$ является *правильной* передачей U. У декодера имеется копия кодового дерева, и он может воспользоваться принятой последовательностью $\boldsymbol{Z}$ для прохождения дерева. Декодер начинает с узла дерева в момент t и генерирует оба пути, исходящие из этого узла. Декодер следует пути, который согласуется с полученными n кодовыми символами. На следующем уровне дерева декодер снова генерирует два пути, выходящие из узла, и следует пути, согласующемуся со второй группой символов. Продолжая аналогичным образом, декодер быстро перебирает все дерево.

Допустим теперь, что принятая последовательность $\boldsymbol{Z}$ – *искаженное* кодовое слово $\boldsymbol{U}$. Декодер начинает с узла дерева в момент t_1 и генерирует оба пути, выходящие из этого узла. Если принятые n кодовых символов совпадают с одним из сгенерированных путей, декодер следует этому пути. Если согласования нет, то декодер следует наиболее вероятному пути, но при этом ведет общий подсчет несовпадений между принятыми символами и ответвляющимися словами на пути следования. Если две ветви оказываются равновероятными, то приемник делает произвольный выбор, как и в случае с нулевым входным путем. На каждом уровне дерева декодер генерирует новые ветви и сравнивает их со следующим набором n принятых кодовых символов. Поиск продолжается до тех пор, пока все дерево не будет пройдено по наиболее вероятному пути, при этом составляется счет несовпадений. Если счет несовпадений превышает некоторое число (оно может увеличиваться после прохождения дерева), декодер решает, что он находится на неправильном пути, отбрасывает этот путь и повторяет все снова. Декодер хранит список отброшенных путей, чтобы избегать их при следующем прохождении дерева.

Допустим, устройство кодирует информационную последовательность m = 11011 в последовательность кодовых слов $\boldsymbol{U}$. Предположим, что четвертый и седьмой биты переданной последовательности $\boldsymbol{U}$ приняты с ошибкой. Структуры информационной, переданной и принятой последовательностей приведены в табл. 2.

Таблица 2

Преобразование кода при передаче и приеме

Время / Последовательность	t_1	t_2	t_3	t_4	t_5
Информационная m	1	1	0	1	1
Переданная U	11	01	01	00	01
Принятая Z	11	00	01	10	01

Проследим за траекторией пути декодирования по рис. 13. Допустим, что критерий возврата и повторного прохождения путей – общий счет несогласующихся путей, равный трем. На рис. 13 числа у путей прохождения представляют собой текущие значения счетчика несовпадений. Рассмотрим прохождение дерева по шагам.

1. В момент времени t_1 принимаются символы 11 и сравниваются с ответвляющимися словами, исходящими из первого узла.

2. Наиболее вероятна та ветвь, у которой ответвляющееся слово 11 (соответствующее входной битовой единице или ответвлению вниз), поэтому декодер решает, что входная битовая единица правильно декодирована, и переходит на следующий уровень.

3. В момент времени t_2 на втором уровне декодер принимает символы 00 и сравнивает их с возможными ответвляющимися словами 10 и 01.

4. Здесь нет «хорошего» пути, поэтому декодер произвольно выбирает путь, соответствующий входному битовому нулю (или ответвляющемуся слову 10), и счетчик несовпадений регистрирует «1».

5. В момент времени t_3 декодер принимает символы 01 и сравнивает их на третьем уровне с ответвляющимися словами 11 и 00.

6. Здесь снова ни один из путей не имеет преимуществ. Декодер произвольно выбирает нулевой входной путь (или ответвляющееся слово 11), и счетчик несовпадений показывает «2».

7. В момент времени t_4 декодер принимает символы 10 и сравнивает их на четвертом уровне с ответвляющимися словами 00 и 11.

8. Здесь снова ни один из путей не имеет преимуществ, и декодер произвольно выбирает нулевой входной путь (или ответвляющееся слово 00); счетчик несовпадений регистрирует «3».

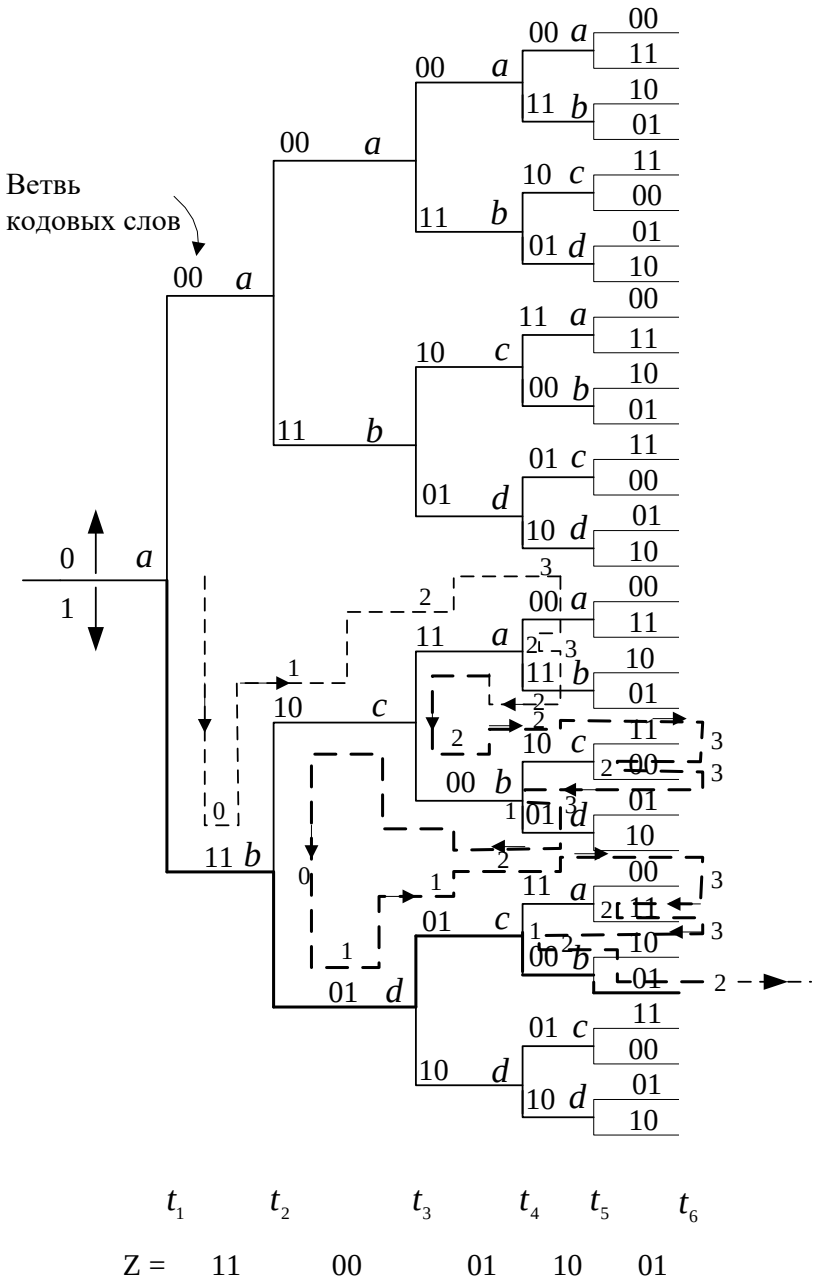


Рис. 13. Схема последовательного декодирования

9. Поскольку счет несовпадений, равный трем, соответствует точке возврата, декодер делает откат и пробует альтернативный путь. Счетчик переустанавливается на два несовпадения.

10. Альтернативный путь на четвертом уровне соответствует пути входной битовой единицы (или ответвляющемуся слову 11). Декодер принимает этот путь, но после сравнения его с принятыми символами 10 несовпадение остается равным единице, и счетчик устанавливается на «3».

11. Счет «3» – критерий точки возврата, поэтому декодер делает откат назад с этого пути и счетчик снова устанавливается на «2». На уровне t_4 все альтернативные пути использованы, поэтому декодер возвращается на узел в момент времени t_3 и переустанавливает счетчик на «1».

12. В узле t_4 декодер сравнивает символы 01, принятые в момент времени t_3, с неиспользованным путем 00. В данном случае число несовпадений равно единице, и счетчик устанавливается на «2».

13. В узле t_4 декодер следует за ответвляющимся словом 10, которое совпадает с принятым в момент времени t_4 кодовым словом 10. Счетчик остается равным двум.

14. В узле t_5 ни один из путей не имеет преимуществ, и декодер, как и определяется правилами, следует верхней ветви. Счетчик устанавливается на три несовпадения.

15. При таком счете декодер делает откат, переустанавливает счетчик на «2» и пробует альтернативный путь в узле t_5. Поскольку другое ответвляющееся слово 00, снова получаем одно несовпадение с принятым в момент времени t_5 кодовым словом 01, и счетчик устанавливается равным трем.

16. Декодер уходит с этого пути, и счетчик переустанавливается на «2». На уровне t_5 все альтернативные пути исправлены, поэтому декодер возвращается в узел в момент времени t_4 и переустанавливает счетчик на «1».

17. Декодер пробует альтернативные пути в узле t_4, метрика которого возрастает до трех, поскольку в ответвляющемся слове имеется несовпадение в двух позициях. В этот момент декодер должен сделать откат всех путей до момента времени t_2, поскольку все пути более высоких уровней уже использованы. Счетчик снова переустановлен на «0».

18. В узле t_2 декодер следует ответвляющемуся слову 01. Поскольку имеются несовпадения в одной позиции с принятыми в момент t_2 кодовыми символами 00, то счетчик устанавливается на «1».

Далее декодер продолжает свои поиски таким же образом. Как видно по рис. 13, финальный путь, счетчик которого не нарушает критерия точки возврата, дает правильно декодированную информационную последовательность 11011. Последовательное декодирование можно понимать как тактику проб и ошибок для поиска правильного пути на кодовом дереве. Поиск осуществляется последовательно; всегда рассматривается только один путь за раз. Если принимается неправильное решение, последующие пути будут ошибочными. Декодер может со временем распознать ошибку, отслеживая метрики пути. Его можно сравнить с путешественником, отыскивающим путь на карте дорог. До тех пор пока путешественник видит, что дорожные ориентиры соответствуют таковым на карте, он продолжает путь. Когда он замечает странные ориентиры (для декодера это увеличение его своеобразной метрики) и приходит к выводу, что находится не на правильном пути, он возвращается к точке, где может узнать ориентиры (для декодера его метрика возвращается в приемлемые рамки). Тогда он пробует альтернативный путь.

Сравнение декодирования по алгоритму Витерби с последовательным декодированием и их ограничения. Главный недостаток декодирования по алгоритму Витерби заключается в том, что с увеличением длины кодового ограничения вероятность появления ошибки экспоненциально убывает, но число кодовых состояний, а значит, сложность декодера, экспоненциально растет. К тому же вычислительная сложность алгоритма Витерби является независимой характеристикой канала (в отличие от «жесткого» и «мягкого» декодирования, которые требуют обычного увеличения объемов вычислений). При методе последовательного декодирования асимптотически достигается та же вероятность появления ошибки, что и при декодировании по принципу максимального правдоподобия, но без поиска всех возможных состояний. Фактически при последовательном декодировании число перебираемых состояний не зависит от длины кодового ограничения, и это позволяет использовать очень большие (K = 41) длины кодового ограничения, что важно при обеспечении низких вероятностей появления ошибок.

Основной недостаток последовательного декодирования состоит в том, что количество перебираемых метрик состояний – случайная величина. Для последовательного декодирования ожидаемое число неудачных гипотез и повторных переборов выражается функцией канального отношения сигнал/шум (signal-to-noise ratio – SNR). При низком SNR приходится перебирать больше гипотез, чем при высоком SNR. Из-за изменчивости вычислительной нагрузки поступившие последовательности необходимо сохранять в буфере памяти. При низком SNR последовательности поступают в буфер до тех пор, пока декодер не сможет найти вероятную гипотезу. Если средняя скорость передачи символов превышает среднюю скорость декодирования, буфер переполняется вне зависимости от его емкости и данные теряются. Обычно, пока идет переполнение, буфер убирает данные без ошибок, в то время как декодер пытается выполнить процедуру восстановления. Порог переполнения буфера существенно зависит от SNR. Отсюда важное техническое требование к последовательному декодеру – обеспечение вероятности переполнения буфера.

Декодирование с обратной связью. Декодер с обратной связью реализует «жесткую» схему принятия решений относительно информационного бита в разряде j, исходя при этом из метрик, полученных из разрядов $j, j + 1, \ldots, j + m$, где m – заранее установленное положительное целое число. Длина упреждения (look-ahead length) $L = m + 1$ определяется как количество принятых кодовых символов, выраженных через соответствующее число входных битов, задействованных для декодирования информационного бита. Решение о том, является ли информационный бит нулем или единицей, принимается в зависимости от того, на какой ветви путь минимального расстояния Хэмминга переходит в окне упреждения (look-ahead window) из разряда J в разряд $j + m$. Поясним это на конкретном примере. Рассмотрим декодер с обратной связью, предназначенный для сверточного кода со степенью кодирования 1/2. На рис. 14 приведена древовидная диаграмма и работа декодера с обратной связью при $L = 3$. При декодировании бита из ветви j декодер содержит пути из ветвей $j, j + 1$ и $j + 2$.

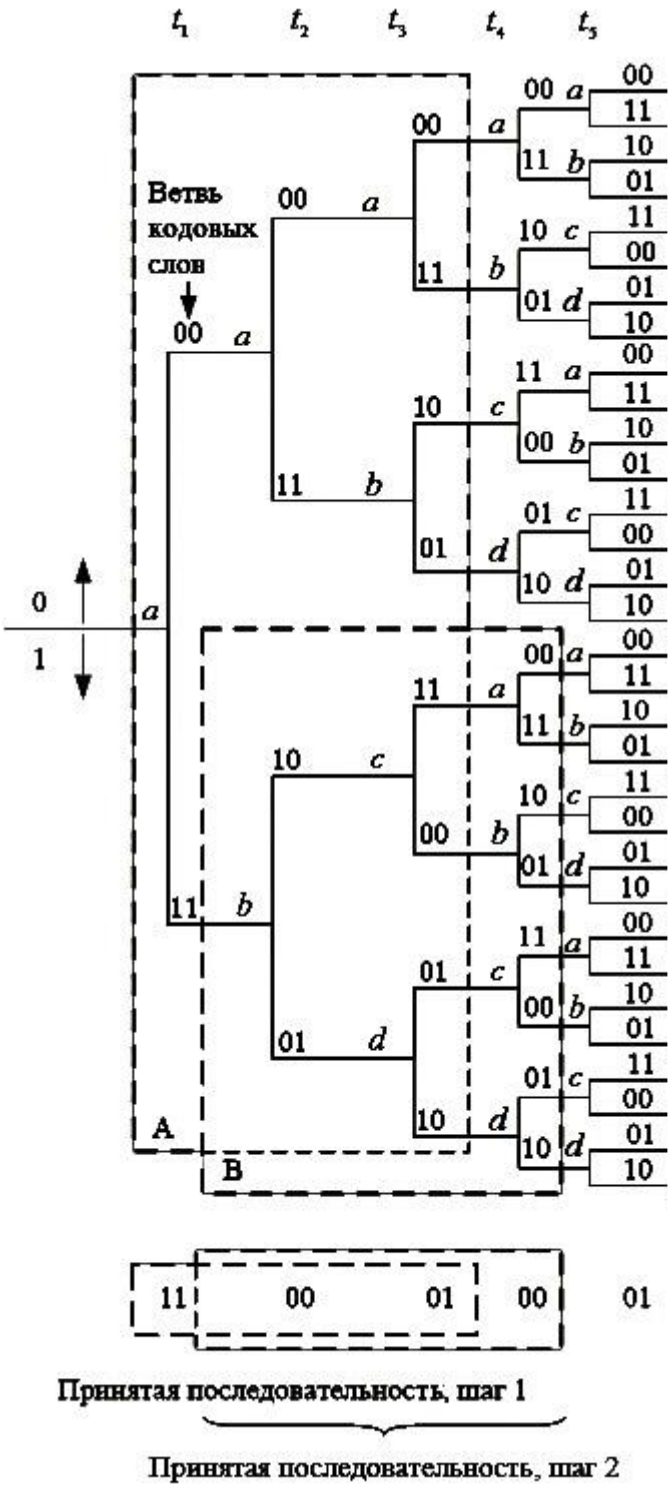


Рис. 14. Пример декодирования
с обратной связью

Начиная с первой ветви, декодер вычисляет 2^L (восемь) совокупных метрик путей расстояния Хэмминга и решает, что бит для первой ветви нулевой, если путь минимального расстояния содержится

в верхней части дерева, и единичный, если путь минимального расстояния находится в нижней части дерева. Пусть принята последовательность Z = 1100010001. Рассмотрим восемь путей от момента времени t_1 до момента времени t_3 в блоке, обозначенном на рис. 14 буквой *A*, и рассчитаем метрики, сравнивая эти восемь путей для первых шести принятых кодовых символов (три ветви вглубь умножить на два символа для ветви). Выписав метрики Хэмминга общих путей (начиная с верхнего пути), видим, что они имеют следующие значения:

метрики верхней части 3, 3, 6, 4
метрики нижней части 2, 2, 1, 3

Наименьшая метрика содержится в нижней части дерева. Следовательно, первый декодированный бит – единица (определяется сдвигом вниз на дереве). Следующий шаг – расширение нижней части дерева («выживший» путь) на один разряд глубже. Здесь снова вычисляется восемь метрик, теперь уже для моментов времени $t_2 \div t_4$. Получив таким образом два декодированных символа, можно сдвинуться на два символа вправо и снова начать расчет метрик путей, но уже для новых шести кодовых символов. Эта процедура видна в блоке, обозначенном на рис. 14 буквой *B*. И снова, проследив метрики верхних и нижних путей, находим следующие значения:

метрики верхней части 2, 4, 3, 3
метрики нижней части 3, 1, 4, 4

Минимальная метрика для ожидаемой переданной последовательности находится в нижней части блока *B*. Следовательно, второй декодируемый бит также является единицей.

Таким образом процедура продолжается до тех пор, пока не будет декодировано все сообщение целиком. Декодер с обратной связью назван так, поскольку найденное решение подается обратно в декодер для определения подмножества кодовых путей, которые будут рассматриваться следующими. В двоичном симметричном канале (BSC) декодер с обратной связью может оказаться почти таким же эффективным, как и декодер, работающий по алгоритму Витерби [7]. Кроме того, он может исправлять все наиболее вероятные ошибочные комбинации, а именно те, которые имеют весовой коэффициент $(d_f - 1)/2$ или менее, где d_f – просвет кода. Важный параметр разработки сверточного декодера с обратной связью – длина упреждения *L*. Увеличение *L* приводит к повышению эффективности кодирования, но при этом растет сложность конструкции декодера.

13. ДЕКОДИРОВАНИЕ БЛОКОВЫХ КОДОВ

Рассмотрим декодирование блоковых циклических кодов – самого распространенного вида используемых блоковых кодов [2 – 4]. При «жестком» декодировании на уровне демодулятора все принимаемые символы считаются взаимно независимыми, поэтому каждый из них отдельно декодируется по принципу максимального правдоподобия. После этого декодер, используя определенные связи между всей совокупностью символов блока, которая была введена при кодировании за счет применения избыточности, исправляет возможные ошибки в информационной части блока.

В случае «мягкого» декодирования реализация принципа максимального правдоподобия тоже потребует для каждого информационного символа выбора его наиболее вероятного значения на основе принимаемой реализации. Однако при этом проверочные символы несут дополнительную информацию о значении информационных символов и могут быть использованы для коррекции тех исходных вероятностей информационных символов, которые ранее определил демодулятор. В результате эти вероятности изменятся и наиболее вероятными могут оказаться другие значения символов.

Таким образом, «мягкое» декодирование можно трактовать как коррекцию исходных вероятностей принимаемых символов, определенных демодулятором, с последующим выбором их наиболее вероятных значений на основе откорректированных вероятностей. Подобный подход позволяет значительно уменьшить требуемый объем вычислений.

Рассмотрим реализацию предлагаемого алгоритма. При «жестком» декодировании для обнаружения и исправления ошибок с использованием циклического кода наиболее общий алгоритм состоит из следующей последовательности операций:

- принятый блок – кодовая комбинация – делится на образующий многочлен $\boldsymbol{g(x)}$. Если остаток от деления $\boldsymbol{R(x)} \neq 0$, то определяется вес остатка w (количество единиц в остатке). Если вес остатка равен числу исправляемых ошибок t ($w \leq t$) или меньше него, то принятую комбинацию можно сложить по модулю 2 с остатком и получить исправленную комбинацию. Однако можно этого и не делать, так как при этом будут откорректированы только проверочные символы, которые далее ценности обычно не представляют. Информационную же часть блока при этом можно считать свободной от ошибок;

- если $w > t$, то производится циклический сдвиг на один символ влево. Полученная после такого сдвига комбинация снова делится на образующий многочлен $\boldsymbol{g(x)}$. Если при этом $w \leq t$, то эту циклически сдвинутую комбинацию складывают по модулю 2 с полученным остатком. Затем результат сложения циклически сдвигают обратно (вправо) на один символ, т. е. восстанавливают исходный вид комбинации. При этом также восстанавливается и исправленная информационная часть;

- если после предыдущего циклического сдвига на один символ и деления на образующий многочлен $\boldsymbol{g(x)}$ сохраняется неравенство $w > t$, производят следующие посимвольные циклические сдвиги влево. При этом после каждого сдвига осуществляется деление сдвинутой комбинации на $\boldsymbol{g(x)}$ и анализируется вес остатка. Когда после какого-либо сдвига начинает выполняться условие $w \leq t$, сдвинутую комбинацию складывают с полученным остатком, после чего производят столько обратных циклических сдвигов вправо, сколько их было сделано влево.

В случае использования недвоичных блоковых кодов декодирование осуществляется с помощью полей Галуа, т. е. алгебры конечных элементов.

Для понимания принципов кодирования и декодирования недвоичных кодов, например кодов Рида – Соломона, рассмотрим понятие конечных полей, известных как *поля Галуа* (Galois fields – GF). Для любого простого числа p существует конечное поле, которое обозначается как GF(p) и содержит p элементов. Понятие GF(p) можно обобщить на поле из p^m элементов, именуемом *полем расширения* GF(p) и обозначаемом GF(p^m), где m – положительное целое число. Заметим, что GF(p^m) содержит в качестве подмножества все элементы GF(p). Символы из поля расширения GF используются при построении кодов Рида – Соломона.

Двоичное поле GF(2) является подполем поля расширения GF(2^m), точно так же как поле вещественных чисел – это подполе поля комплексных чисел. Кроме чисел 0 и 1 в поле расширения существуют дополнительные однозначные элементы, которые представлены новым символом α. Каждый ненулевой элемент в GF(2^m) можно

представить как степень α. Бесконечное множество элементов F образуется из стартового множества $\{0, 1, \alpha\}$ и дополнительных элементов путем последовательного умножения последней записи на α:

$$F = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^j, \ldots\} = \{0, \alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^j, \ldots\}.$$

Для вычисления из последовательности F конечного множества элементов GF(2^m) на последовательность F нужно наложить ограничения: она может содержать только 2^m элемента и должна быть замкнута относительно операции умножения. Условие замыкания множества элементов поля по отношению к операции умножения имеет вид неприводимого полинома $\alpha^{(2^m-1)} + 1 = 0,$ или, что то же самое, $\alpha^{(2^m-1)} + 1 = \alpha^0.$

С помощью полиномиального ограничения любой элемент со степенью, большей или равной $2^m - 1$, можно следующим образом понизить до элемента со степенью, меньшей $2^m - 1$:

$$\alpha^{(2^m+n)} = \alpha^{(2^m-1)}\alpha^{n+1} = \alpha^{n+1}.$$

Последнее уравнение можно использовать для формирования конечной последовательности *F** из бесконечной последовательности *F:*

$$F^* = \{0, 1, \alpha, \alpha^2, ..., \alpha^{2^m-2}, \alpha^{2^m-1}, \alpha^{2^m}, ...\} = \{0, \alpha^0, \alpha^1, \alpha^2, ..., \alpha^{2^m-2}\}. \quad (2)$$

Из уравнения (2) видно, что элементы конечного поля GF(2^m) обозначены следующим выражением:

$$\mathrm{GF}(2^m) = \{0, \alpha^0, \alpha^1, \alpha^2, ..., \alpha^{2^m-2}\}. \quad (3)$$

Операция сложения в поле расширения GF(2^m). Каждый из 2^m элементов конечного поля GF(2^m) можно представить как отдельный полином степени m – 1 или меньше. Степень полинома – это степень члена максимального порядка. Обозначим каждый ненулевой элемент GF(2^m) полиномом $\boldsymbol{\alpha}_i(\boldsymbol{X})$, в котором последние m коэффициентов $\boldsymbol{\alpha}_i(\boldsymbol{X})$ нулевые. Для $i = 0, 1, 2, \ldots, 2^m - 2$

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i(\boldsymbol{X}) = \alpha_{i,0} + \alpha_{i,1}X + \alpha_{i,2}X^2 + ... + \alpha_{i,m-1}X^{m-1}. \quad (4)$$

Рассмотрим случай при m = 3. Конечное поле обозначается GF(2^3). В табл. 3 показано отображение семи элементов $\{a_i\}$ и нулевого элемента в слагаемые базисных элементов $\left\{X^0,\ X^1,\ X^2\right\}$, описываемых уравнением (4). Из уравнения $\alpha^0 = \alpha^7$ следует, что в этом поле имеется семь ненулевых элементов или всего восемь элементов.

Таблица 3

Связь элементов поля с образующими элементами

Образующие элементы / Элементы поля	X^0	X^1	X^2
0	0	0	0
α^0	1	0	0
α^1	0	1	0
α^2	0	0	1
α^3	1	1	0
α^4	0	1	1
α^5	1	1	1
α^6	1	0	1
α^7	1	0	0

Каждая строка в табл. 3 содержит последовательность двоичных величин, представляющих коэффициенты $\alpha_{i,0}$, $\alpha_{i,1}$ и $\alpha_{i,2}$ из уравнения (4). Одно из преимуществ использования элементов $\left\{\alpha^i\right\}$ поля расширения вместо двоичных элементов – это компактность записи, что удобно при математическом описании процессов недвоичного кодирования и декодирования. Сложение двух элементов конечного поля, следовательно, есть суммирование по модулю 2 всех коэффициентов при элементах с одинаковыми степенями:

$$\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_j = (\alpha_{i,0} + \alpha_{j,0}) + (\alpha_{i,1} + \alpha_{j,1})X + ... + (\alpha_{i,m-1} + \alpha_{j,m-1})X^{m-1}. \quad (5)$$

Описание конечного поля с помощью примитивного полинома. Примитивные полиномы определяют конечные поля GF(2^m), которые нужны для описания кодов Рида – Соломона. Следующее утверждение – необходимое и достаточное условие примитивности

полинома. Неприводимый полином ***f*(*X*)** порядка *m* примитивный, если наименьшее положительное целое число *n*, для которого $X^n + 1$ делится на ***f*(*X*)**, равно $2^m - 1$. Заметим, что неприводимый полином – это такой полином, который нельзя представить в виде произведения полиномов меньшего порядка; делимость *A* на *B* означает, что *A* делится на *B* с нулевым остатком и ненулевым частным. Обычно полином записывают в порядке возрастания степеней. Иногда удобнее обратный формат записи (например, при выполнении полиномиального деления).

Поле расширения GF(2³). Рассмотрим пример, в котором задействованы примитивный полином и конечное поле, которое он определяет. В табл. 4 содержатся примеры некоторых примитивных полиномов.

Таблица 4

Примитивные полиномы для различных параметров кода

m	Примитивный полином	*m*	Примитивный полином
3	$1 + X + X^3$	14	$1 + X + X^6 + X^{10} + X^{14}$
4	$1 + X + X^4$	15	$1 + X + X^{15}$
5	$1 + X^2 + X^5$	16	$1 + X + X^3 + X^{12} + X^{16}$
6	$1 + X + X^6$	17	$1 + X^3 + X^{17}$
7	$1 + X^3 + X^7$	18	$1 + X^7 + X^{18}$
8	$1 + X^2 + X^3 + X^4 + X^8$	19	$1 + X + X^2 + X^5 + X^{19}$
9	$1 + X^4 + X^9$	20	$1 + X^3 + X^{20}$
10	$1 + X^3 + X^{10}$	21	$1 + X^2 + X^{21}$
11	$1 + X^2 + X^{11}$	22	$1 + X + X^{22}$
12	$1 + X + X^4 + X^6 + X^{12}$	23	$1 + X^5 + X^{23}$
13	$1 + X + X^3 + X^4 + X^{13}$	24	$1 + X + X^2 + X^7 + X^{24}$

Выберем первый из указанных в табл. 4 полиномов, $\boldsymbol{f(X)} = 1 + X + X^3$, который определяет конечное поле GF(2^m), где степень полинома $m = 3$. Таким образом, в поле, определяемом полиномом ***f*(*X*)**, имеется $2^m = 2^3 = 8$ элементов. Поиск корней полинома ***f*(*X*)** – это поиск таких значений *X*, при которых $\boldsymbol{f(X)} = 0$. Привычные нам двоичные элементы 0 и 1 не подходят полиному $\boldsymbol{f(X)} = 1 + X + X^3$ (они не являются корнями), поскольку $\boldsymbol{f(1)} = 1$ и $\boldsymbol{f(0)} = 1$ (в рамках операции

по модулю 2). Кроме того, основная теорема алгебры утверждает, что полином порядка *m* должен иметь в точности *m* корней. Следовательно, в этом примере $\boldsymbol{f(X)}=1$ должно иметь три корня. Возникает определенная проблема, поскольку три корня не лежат в том же конечном поле, что и коэффициенты ***f*(*X*)**. А если они находятся где-то еще, то наверняка в поле расширения $\mathrm{GF}(2^3)$. Пусть α – элемент поля расширения – определяется как корень полинома ***f*(*X*)**. Следовательно, можно записать следующее:

$$\left.\begin{aligned} \boldsymbol{f(\alpha)} &= 0, \\ 1+\alpha+\alpha^3 &= 0, \\ \alpha^3 &= -1-\alpha. \end{aligned}\right\} \tag{6}$$

Поскольку при операциях над двоичным полем +1 = –1, то α^3 можно представить следующим образом:

$$\alpha^3 = 1+\alpha \tag{7}$$

Таким образом, α^3 представляется в виде взвешенной суммы всех α – членов более низкого порядка. Фактически так можно представить все степени α. Например, рассмотрим

$$\alpha^4 = \alpha\cdot\alpha^3 = \alpha(1+\alpha) = \alpha+\alpha^3. \tag{8}$$

А теперь другой случай:

$$\alpha^5 = \alpha\cdot\alpha^4 = \alpha(\alpha+\alpha^2) = \alpha^2+\alpha^3. \tag{9}$$

Из уравнений (7) и (9) получаем

$$\alpha^5 = 1+\alpha+\alpha^2. \tag{10}$$

Из уравнений (9) и (10) получаем

$$\alpha^6 = \alpha\cdot\alpha^5 = \alpha(1+\alpha+\alpha^2) = \alpha+\alpha^2+\alpha^3 = 1+\alpha^2 \tag{11}$$

А теперь из уравнения (11) вычисляем

$$\alpha^7 = \alpha\cdot\alpha^6 = \alpha(1+\alpha^2) = \alpha+\alpha^3 = 1+\alpha^0. \tag{12}$$

Заметим, что $\alpha^7 = \alpha^0$ и, следовательно, восемь элементов конечного поля $\mathrm{GF}(2^3)$ следующие:

$$\left\{0,\ \alpha^0,\ \alpha^1,\ \alpha^2,\ \alpha^3,\ \alpha^4,\ \alpha^5,\ \alpha^6\right\}. \tag{13}$$

На данном конечном поле GF(2^3) можно определить две арифметические операции – сложение и умножение. В табл. 5 показана операция сложения, а в табл. 6 – операция умножения, но только для ненулевых элементов. Правила суммирования следуют из уравнений (6) и (7), и их можно рассчитать путем сложения (по модулю 2) соответствующих коэффициентов из базисных элементов. Правила умножения, указанные в табл. 6, следуют из обычной процедуры, в которой произведение элементов поля вычисляют путем сложения их показателей степеней по модулю $2^m - 1$ или для данного случая по модулю 7.

Таблица 5

Таблица сложения для поля GF(2^3)

	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	0	α^3	α^6	α^1	α^5	α^4	α^1
α^1	α^3	0	α^4	α^0	α^2	α^6	α^5
α^2	α^4	α^4	0	α^5	α^1	α^3	α^0
α^3	α^1	α^0	α^5	0	α^6	α^2	α^4
α^4	α^5	α^2	α^1	α^4	0	α^0	α^3
α^5	α^4	α^6	α^3	α^2	α^0	0	α^1
α^6	α^2	α^5	α^0	α^4	α^3	α^1	0

Таблица 6

Таблица умножения для поля GF(2^3)

	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^1	α^1	α^2	α^3	α^4	α^5	α^6	α^0
α^2	α^2	α^3	α^4	α^5	α^6	α^0	α^1
α^3	α^3	α^4	α^5	α^6	α^0	α^1	α^2
α^4	α^4	α^5	α^6	α^0	α^1	α^2	α^3
α^5	α^5	α^6	α^0	α^1	α^2	α^3	α^4
α^6	α^6	α^0	α^1	α^2	α^3	α^4	α^5

Существует еще один, чрезвычайно простой, способ проверки, является ли полином примитивным. У неприводимого примитивного полинома по крайней мере хотя бы один из корней должен быть примитивным элементом. *Примитивным элементом* называется такой элемент поля, который при возведении в более высокие степени даст ненулевые элементы поля. Поскольку данное поле конечное, количество таких элементов также конечно.

Кодирование Рида – Соломона. Ранее была представлена наиболее распространенная форма кодов Рида – Соломона через параметры n, k, t и некоторое положительное число $m > 2$. Приведем это уравнение:

$$(n,\ k) = (2^m - 1, 2^m - 1 - 2t). \qquad (14)$$

Здесь $n - k = 2t$ – число контрольных символов, а t – количество ошибочных битов в символе, которое может исправить код. Генерирующий полином для кода Рида – Соломона имеет вид

$$\boldsymbol{g(X)} g_0 + g_1 X + g_2 X + \cdots + g_{2t-1} X^{2t-1} + X^{2t}. \qquad (15)$$

Степень полиномиального генератора равна числу контролируемых символов. Коды Рида – Соломона представляют собой подмножество кодов БЧХ, поэтому для них характерна связь между степенью полиномиального генератора и числом контрольных символов, как и в кодах БЧХ. Поскольку полиномиальный генератор имеет порядок $2t$, должны быть в точности $2t$ последовательные степени α, которые являются корнями полинома. Обозначим корни $\boldsymbol{g(X)}$ как $\alpha, \alpha^2, \ldots, \alpha^{2t}$. Нет необходимости начинать формирование $\boldsymbol{g(X)}$ именно с корня α, это можно сделать с помощью любой степени α. Возьмем, к примеру, код (7, 3) с возможностью коррекции двухсимвольных ошибок. Выразим полиномиальный генератор через $2t = n - k = 4$ корня следующим образом:

$$\begin{aligned}
\boldsymbol{g(X)} &= (X - \alpha)(X - \alpha^2)(X - \alpha^3)(X - \alpha^4) = \\
&= (X^2 - (\alpha + \alpha^2)X + \alpha^3)(X^2 - (\alpha^3 + \alpha^4)X + \alpha^7) = \\
&= (X^2 - \alpha^4 X + \alpha^3)(X^2 - \alpha^6 X + \alpha^0) = \\
&= X^4 - (\alpha^4 + \alpha^6)X^3 + (\alpha^3 + \alpha^{10} + \alpha^0)X^2 - (\alpha^4 + \alpha^9)X + \alpha^3 = \\
&= X^4 - \alpha^3 X^3 + \alpha^0 X^2 - \alpha^1 X + \alpha^3.
\end{aligned}$$

Кодирование в систематической форме. Так как код Рида – Соломона циклический, кодирование в систематической форме аналогично процедуре двоичного кодирования. Осуществим сдвиг полинома сообщения $\boldsymbol{m(X)}$ в крайние правые k разряды регистра кодового слова и прибавим полином четности $\boldsymbol{p(X)}$ в крайние левые $n - k$ разряды. Умножим $\boldsymbol{m(X)}$ на X^{n-k} таким образом, чтобы $\boldsymbol{m(X)}$ оказалось сдвинутым вправо на $n - k$ позиций. Далее делим $X^{n-k}\boldsymbol{m(X)}$ на полиномиальный генератор $\boldsymbol{g(X)}$, что можно записать следующим образом:

$$X^{n-k}\boldsymbol{m(X)} = \boldsymbol{q(X)g(X)} + \boldsymbol{p(X)}. \qquad (16)$$

Здесь $\boldsymbol{q(X)}$ и $\boldsymbol{p(X)}$ – это частное и остаток от полиномиального деления. Как и в случае двоичного кодирования, остаток четный. Уравнение (16) можно представить по-другому:

$$\boldsymbol{p(X)} = X^{n-k}\boldsymbol{m(X)} \text{ по модулю } \boldsymbol{g(X)}. \qquad (17)$$

Результирующий полином кодового слова $\boldsymbol{U(X)}$ можно переписать так:

$$\boldsymbol{U(X)} = \boldsymbol{p(X)} + X^{n-k}\boldsymbol{m(X)}. \qquad (18)$$

Продемонстрируем шаги, подразумеваемые уравнениями (17) и (18), закодировав сообщение из трех символов

$$\underset{\alpha^1}{010}\ \underset{\alpha^3}{110}\ \underset{\alpha^5}{111}$$

с помощью кода (7, 3), генератор которого определяется уравнением (16). Сначала умножаем (сдвиг вверх) полином сообщения $\alpha^1 + \alpha^3 X + \alpha^5 X^2$ на $X^{n-k} = X^4$, что дает $\alpha^1 X + \alpha^3 X^5 + \alpha^5 X^6$. Далее делим такой сдвинутый вверх полином сообщения на полиномиальный генератор из уравнения (16): $\alpha^3 + \alpha^1 X + \alpha^0 X^2 + \alpha^3 X^3 + X^4$. Полиномиальное деление недвоичных коэффициентов – это еще более длительная процедура, чем ее двоичный аналог, поскольку операции сложения (вычитания) и умножения (деления) выполняются согласно табл. 5 и 6.

Исходя из уравнения (18), полином кодового слова можно записать следующим образом:

$$\boldsymbol{U(X)} = \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6.$$

Декодирование Рида – Соломона. Тестовое сообщение кодируется в систематической форме с помощью кода (7, 3), что дает в результате полином кодового слова, описываемый приведенным уравнением. Допустим, что в ходе передачи это кодовое слово подверглось искажению: два символа были приняты с ошибкой. (Такое количество ошибок соответствует максимальной способности кода к коррекции ошибок.) При использовании семисимвольного кодового слова ошибочную комбинацию можно представить в полиномиальной форме следующим образом:

$$\boldsymbol{e(X)} = \sum_{n=0}^{6} e_n X^n. \qquad (19)$$

Пусть двухсимвольная ошибка будет такой, что

$$\boldsymbol{e(X)} = 0 + 0X + 0X^2 + \alpha^2 X^3 + \alpha^5 X^4 + 0X^5 + 0X^6 =$$
$$= (000) + (000)X + (000)X^2 + (001)X^3 + (111)X^4 + (000)X^5 + (000)X^6. \quad (20)$$

Другими словами, контрольный символ искажен битовой ошибкой (представленной как α^2), а символ сообщения – трехбитовой ошибкой (представленной как α^5). В данном случае принятый полином поврежденного кодового слова $\boldsymbol{r(X)}$ представляется в виде суммы полинома переданного кодового слова и полинома ошибочной комбинации:

$$\boldsymbol{r(X)} = \boldsymbol{U(X)} + \boldsymbol{e(X)}. \qquad (21)$$

Далее суммируются $\boldsymbol{U(X)}$ и $\boldsymbol{e(X)}$ из уравнения

$$\boldsymbol{r(X)} = (100) + (001)X + (011)X^2 + (100)X^3 + (101)X^4 + (110)X^5 + (111)X^6 =$$
$$= \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^0 X^3 + \alpha^6 X^4 + \alpha^3 X^5 + \alpha^5 X^6. \qquad (22)$$

В данном примере исправления двухсимвольной ошибки имеется четыре неизвестных: два относятся к расположению ошибки, а два касаются ошибочных значений. Отметим важное различие между недвоичным декодированием $\boldsymbol{r(X)}$ (уравнение [22]) и двоичным. При двоичном декодировании декодеру нужно знать лишь расположение ошибки. Если известно, где находится ошибка, бит нужно поменять с 1 на 0 или наоборот. В случае недвоичных символов необходимо не только узнать расположение ошибки, но и определить правильное значение символа, находящегося на этой позиции. Поскольку в данном примере у нас имеется четыре неизвестных, нужны четыре уравнения, чтобы найти их.

Вычисление синдрома. Синдром – это результат проверки четности, выполняемой над $\boldsymbol{r}$, чтобы определить, принадлежит ли $\boldsymbol{r}$ набору кодовых слов. Если $\boldsymbol{r}$ – член набора, то синдром S имеет значение, равное нулю. Любое ненулевое значение S указывает на наличие ошибок. Точно так же как и в двоичном случае, синдром S состоит из $n - k$ символов $\{S_i\}$ $(i = 1, \ldots, n-k)$. Таким образом, для кода (7, 3) имеется по четыре символа в каждом векторе синдрома; их значения можно рассчитать из принятого полинома $\boldsymbol{r}(\boldsymbol{X})$. Заметим, как облегчаются вычисления благодаря самой структуре кода, определяемой уравнением (18):

$$\boldsymbol{U}(\boldsymbol{X}) = \boldsymbol{m}(\boldsymbol{X})\boldsymbol{g}(\boldsymbol{X}).$$

Из этой структуры можно видеть, что каждый правильный полином кодового слова $\boldsymbol{U}(\boldsymbol{X})$ кратен полиномиальному генератору $\boldsymbol{g}(\boldsymbol{X})$. Следовательно, корни $\boldsymbol{g}(\boldsymbol{X})$ также должны быть корнями $\boldsymbol{U}(\boldsymbol{X})$. Поскольку $\boldsymbol{r}(\boldsymbol{X}) = \boldsymbol{U}(\boldsymbol{X}) + \boldsymbol{e}(\boldsymbol{X})$, то $\boldsymbol{r}(\boldsymbol{X})$, вычисляемый с каждым корнем $\boldsymbol{g}(\boldsymbol{X})$, должен давать нуль, только если $\boldsymbol{r}(\boldsymbol{X})$ – правильное кодовое слово. Любые ошибки приведут в итоге к ненулевому результату в одном (или более) случае. Символы синдрома вычисляют следующим образом:

$$\boldsymbol{S}_i = \boldsymbol{r}(\boldsymbol{X})|_{X=\alpha^i} = \boldsymbol{r}(\boldsymbol{\alpha}^i), \quad i = 1, \ldots, n-k. \tag{23}$$

Здесь, как показано в уравнении (23), $\boldsymbol{r}(\boldsymbol{X})$ содержит двухсимвольные ошибки. Если $\boldsymbol{r}(\boldsymbol{X})$ окажется правильным кодовым словом, то все символы синдрома $\boldsymbol{S}_i$ будут равны нулю. В данном примере четыре символа синдрома находят следующим образом:

$$\begin{aligned}\boldsymbol{S}_1 = \boldsymbol{r}(\alpha) &= \alpha^0 + \alpha^3 + \alpha^6 + \alpha^3 + \alpha^{10} + \alpha^8 + \alpha^{11} = \\ &= \alpha^0 + \alpha^3 + \alpha^6 + \alpha^3 + \alpha^2 + \alpha^1 + \alpha^4 = \alpha^3;\end{aligned} \tag{24}$$

$$\begin{aligned}\boldsymbol{S}_2 = \boldsymbol{r}(\alpha^2) &= \alpha^0 + \alpha^4 + \alpha^8 + \alpha^6 + \alpha^{14} + \alpha^{13} + \alpha^{17} = \\ &= \alpha^0 + \alpha^4 + \alpha^1 + \alpha^6 + \alpha^0 + \alpha^6 + \alpha^3 = \alpha^5;\end{aligned} \tag{25}$$

$$\begin{aligned}\boldsymbol{S}_3 = \boldsymbol{r}(\alpha^3) &= \alpha^0 + \alpha^5 + \alpha^{10} + \alpha^9 + \alpha^{18} + \alpha^{18} + \alpha^{23} = \\ &= \alpha^0 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha^4 + \alpha^4 + \alpha^2 = \alpha^6;\end{aligned} \tag{26}$$

$$\begin{aligned}\boldsymbol{S}_4 = \boldsymbol{r}(\alpha^4) &= \alpha^0 + \alpha^6 + \alpha^{12} + \alpha^{12} + \alpha^{22} + \alpha^{23} + \alpha^{29} = \\ &= \alpha^0 + \alpha^6 + \alpha^5 + \alpha^5 + \alpha^1 + \alpha^2 + \alpha^1 = 0.\end{aligned} \tag{27}$$

Результат подтверждает, что принятое кодовое слово содержит ошибку (введенную нами), поскольку $\boldsymbol{S} \neq 0$.

Повторная проверка значений синдрома. Для рассматриваемого кода (7, 3) ошибочная комбинация известна, поскольку мы выбрали ее заранее. Все элементы класса смежности (строка) нормальной матрицы имеют один и тот же синдром. Путем вычисления полинома ошибок $\boldsymbol{e(X)}$ со значениями корней $\boldsymbol{g(X)}$ нужно показать, что это свойство справедливо и для кода Рида – Соломона. Это должно дать те же значения синдрома, что и вычисление $\boldsymbol{r}(X)$ со значениями корней $\boldsymbol{g(X)}$:

$$\boldsymbol{S}_i = \boldsymbol{r(X)}|_{X=\alpha^i} = \boldsymbol{r(\alpha^i)}, \quad i = 1, 2, \ldots, n-k;$$

$$\boldsymbol{S}_i = \left[\boldsymbol{U(X)} + \boldsymbol{e(X)}\right]|_{X=\alpha^i} = \boldsymbol{U(\alpha^i)} + \boldsymbol{e(\alpha^i)};$$

$$\boldsymbol{S}_i = \boldsymbol{r(\alpha^i)} = \boldsymbol{U(\alpha^i)} + \boldsymbol{e(\alpha^i)} = 0 + \boldsymbol{e(\alpha^i)}.$$

Локализация ошибки. Допустим, в кодовом слове имеется v ошибок, расположенных на позициях X^{j_1}, X^{j_2}, …, X^{j_v}. Тогда полином ошибок

$$\boldsymbol{e(X)} = e_{j_1} X^{j_1} + e_{j_2} X^{j_2} + \ldots + e_{j_v} X^{j_v}. \tag{28}$$

Индексы 1, 2, ..., v обозначают 1, 2, ..., v-ю ошибки, а индекс j – расположение ошибки. Для коррекции искаженного кодового слова нужно определить каждое значение ошибки e_{j_l} и ее расположение X^{j_l}, где $l = 1, 2, \ldots, v$. Обозначим номер локатора ошибки как $\beta_l = \alpha^{j_t}$. Далее вычислим $n - k = 2t$ символа синдрома, подставляя α_i в принятый полином при $i = 1, 2, \ldots, 2t$:

$$\begin{cases} \boldsymbol{S}_1 = \boldsymbol{r(\alpha)} = e_{j_1}\beta_1 + e_{j_2}\beta_2 + \cdots + e_{j_v}\beta_v, \\ \boldsymbol{S}_2 = \boldsymbol{r(\alpha^2)} = e_{j_1}\beta_1^2 + e_{j_2}\beta_2^2 + \ldots + e_{j_v}\beta_v^2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \boldsymbol{S}_{2t} = \boldsymbol{r(\alpha^{2t})} = e_{j_1}\beta_1^{2t} + e_{j_2}\beta_2^{2t} + \cdots + e_{j_v}\beta_v^{2t}. \end{cases} \tag{29}$$

Имеется $2t$ неизвестных (t значений ошибок и t расположений) и система $2t$ уравнений. Впрочем, эту систему $2t$ уравнений нельзя решить обычным путем, поскольку уравнения в ней нелинейны (некоторые неизвестные входят в уравнение в степени). Методика, позволяющая решить эту систему уравнений, называется алгоритмом декодирования Рида – Соломона.

Если вычислен ненулевой вектор синдрома (один или более его символов не равны нулю), это означает, что была принята ошибка. Далее нужно узнать расположение ошибки (или ошибок). Полином локатора ошибок можно определить следующим образом:

$$\boldsymbol{\sigma(X)} = (1+\beta_1 X)(1+\beta_2 X)\ldots(1+\beta_v X) = \\ = 1+\sigma_1 X + \sigma_2 X^2 + \ldots + \sigma_v X^v. \qquad (30)$$

Корнями $\boldsymbol{\sigma(X)}$ будут $1/\beta_1$, $1/\beta_2$, …, $1/\beta_v$. Величины, обратные корням $\boldsymbol{\sigma(X)}$, представляют номера расположений ошибочной комбинации $\boldsymbol{e(X)}$. Можно составить из синдромов матрицу, в которой первые t синдромов используются для предсказания следующего синдрома:

$$\begin{bmatrix} \boldsymbol{S}_1 & \boldsymbol{S}_2 & \boldsymbol{S}_3 & \ldots & \boldsymbol{S}_{t-1} & \boldsymbol{S}_t \\ \boldsymbol{S}_2 & \boldsymbol{S}_3 & \boldsymbol{S}_4 & \ldots & \boldsymbol{S}_t & \boldsymbol{S}_{t+1} \\ \ldots. & \ldots & \ldots & \ldots & \ldots & \ldots \\ \boldsymbol{S}_{t-1} & \boldsymbol{S}_t & \boldsymbol{S}_{t+1} & \ldots & \boldsymbol{S}_{2t-3} & \boldsymbol{S}_{2t-2} \\ \boldsymbol{S}_t & \boldsymbol{S}_{t+1} & \boldsymbol{S}_{t+2} & \ldots & \boldsymbol{S}_{2t-2} & \boldsymbol{S}_{2t-1} \end{bmatrix} \begin{bmatrix} \sigma_t \\ \sigma_{t-1} \\ \ldots \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{S}_{t+1} \\ -\boldsymbol{S}_{t+s} \\ \ldots \\ -\boldsymbol{S}_{2t-1} \\ -\boldsymbol{S}_{2t} \end{bmatrix}. \qquad (31)$$

Для кода (7, 3) с коррекцией двухсимвольных ошибок матрица имеет размерность 2×2, и модель записывается следующим образом:

$$\begin{bmatrix} \boldsymbol{S}_1 & \boldsymbol{S}_2 \\ \boldsymbol{S}_3 & \boldsymbol{S}_4 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_3 \\ \boldsymbol{S}_4 \end{bmatrix}, \qquad (32)$$

$$\begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^5 & \alpha^6 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^6 \\ 0 \end{bmatrix}. \qquad (33)$$

Чтобы найти коэффициенты σ_1 и σ_2 полинома локатора ошибок $\boldsymbol{\sigma(X)}$, сначала необходимо вычислить обратную матрицу для уравнения (33). Обратная матрица для матрицы $[A]$ определяется следующим образом:

$$\mathrm{Inv}[A] = \frac{\mathrm{cofactor}[A]}{\det[A]}.$$

Следовательно,

$$\det\begin{bmatrix}\alpha^3 & \alpha^5\\ \alpha^5 & \alpha^6\end{bmatrix} = \alpha^3\alpha^6 - \alpha^5\alpha^5 = \alpha^9 + \alpha^{10} = \alpha^2 + \alpha^3 = \alpha^5; \qquad (34)$$

$$\text{cofactor}\begin{bmatrix}\alpha^3 & \alpha^5\\ \alpha^5 & \alpha^6\end{bmatrix} = \begin{bmatrix}\alpha^6 & \alpha^5\\ \alpha^5 & \alpha^3\end{bmatrix}; \qquad (35)$$

$$\text{Inv}\begin{bmatrix}\alpha^3 & \alpha^5\\ \alpha^5 & \alpha^6\end{bmatrix} = \frac{\begin{bmatrix}\alpha^6 & \alpha^5\\ \alpha^5 & \alpha^3\end{bmatrix}}{\alpha^5} = \alpha^{-5}\begin{bmatrix}\alpha^6 & \alpha^5\\ \alpha^5 & \alpha^3\end{bmatrix} =$$

$$= \alpha^2\begin{bmatrix}\alpha^6 & \alpha^5\\ \alpha^5 & \alpha^3\end{bmatrix} = \begin{bmatrix}\alpha^8 & \alpha^7\\ \alpha^7 & \alpha^5\end{bmatrix} = \begin{bmatrix}\alpha^1 & \alpha^0\\ \alpha^0 & \alpha^5\end{bmatrix}. \qquad (36)$$

Проверка надежности. Если обратная матрица вычислена правильно, то произведение исходной и обратной матрицы должно дать единичную матрицу:

$$\begin{bmatrix}\alpha^3 & \alpha^5\\ \alpha^5 & \alpha^6\end{bmatrix}\begin{bmatrix}\alpha^1 & \alpha^0\\ \alpha^0 & \alpha^5\end{bmatrix} = \begin{bmatrix}\alpha^4 + \alpha^5 & \alpha^3 + \alpha^{10}\\ \alpha^6 + \alpha^6 & \alpha^5 + \alpha^{11}\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}. \qquad (37)$$

С помощью уравнения (33) начнем поиск положений ошибок с вычисления коэффициентов полинома локатора ошибок $\boldsymbol{\sigma(X)}$, как показано далее.

$$\begin{bmatrix}\sigma_2\\ \sigma_1\end{bmatrix} = \begin{bmatrix}\alpha^1 & \alpha^0\\ \alpha^0 & \alpha^5\end{bmatrix}\begin{bmatrix}\alpha^6\\ 0\end{bmatrix} = \begin{bmatrix}\alpha^7\\ \alpha^6\end{bmatrix} = \begin{bmatrix}\alpha^0\\ \alpha^6\end{bmatrix}. \qquad (38)$$

Однако

$$\boldsymbol{\sigma(X)} = \alpha^0 + \sigma_1 X + \sigma_2 X^2 = \alpha^0 + \alpha^6 X + \alpha^0 X^2. \qquad (39)$$

Корни $\boldsymbol{\sigma(X)}$ – обратные числа к положениям ошибок. После того как эти корни найдены, становится известно расположение ошибок. Вообще, корни $\boldsymbol{\sigma(X)}$ могут быть одним или несколькими элементами поля. Определим эти корни путем полной проверки полинома $\boldsymbol{\sigma(X)}$ со всеми элементами поля, как будет показано ниже. Любой

элемент X, который дает $\boldsymbol{\sigma(X)} = 0$, является корнем, что позволяет нам определить расположение ошибки.

$$\boldsymbol{\sigma}(\alpha^0) = \alpha^0 + \alpha^6 + \alpha^0 = \alpha^6 \neq 0,$$
$$\boldsymbol{\sigma}(\alpha^1) = \alpha^2 + \alpha^7 + \alpha^0 = \alpha^2 \neq 0,$$
$$\boldsymbol{\sigma}(\alpha^2) = \alpha^4 + \alpha^8 + \alpha^0 = \alpha^6 \neq 0,$$
$$\boldsymbol{\sigma}(\alpha^3) = \alpha^6 + \alpha^9 + \alpha^0 = 0 \Rightarrow \textit{ОШИБКА},$$
$$\boldsymbol{\sigma}(\alpha^4) = \alpha^8 + \alpha^{10} + \alpha^0 = 0 \Rightarrow \textit{ОШИБКА},$$
$$\boldsymbol{\sigma}(\alpha^5) = \alpha^{10} + \alpha^{11} + \alpha^0 = \alpha^2 \neq 0,$$
$$\boldsymbol{\sigma}(\alpha^6) = \alpha^{12} + \alpha^{12} + \alpha^0 = \alpha^0 \neq 0.$$

Как видно из уравнения (30), расположение ошибок – обратная величина к корням полинома. Отсюда $\boldsymbol{\sigma}(\alpha^3) = 0$ означает, что один корень получается при $1/\beta_l = \alpha^3$. Отсюда $\beta_l = 1/\alpha^3 = \alpha^4$. Аналогично $\boldsymbol{\sigma}(\alpha^4) = 0$ означает, что другой корень появляется при $1/\beta_{l'} = 1/\alpha^4 = \alpha^3$, где (в данном примере) l и l' обозначают 1-ю и 2-ю ошибки. Поскольку речь идет о двухсимвольных ошибках, полином ошибок можно записать следующим образом:

$$\mathbf{e}(\boldsymbol{X}) = e_{j_1} X^{j_1} + e_{j_2} X^{j_2}. \qquad (40)$$

Здесь были найдены две ошибки на позициях α^3 и α^4. Заметим, что индексация номеров расположения ошибок сугубо произвольная. Итак, в этом примере величины $\beta_l = \alpha^{j_l}$ обозначены как $\beta_l = \alpha^{j_1} = \alpha^3$ и $\beta_2 = \alpha^{j_2} = \alpha^4$.

Значения ошибок. Обозначим ошибки e_{j_l}, где индекс j указывает на расположение ошибки, а индекс l – на саму ошибку. Поскольку каждое значение ошибки связано с конкретным месторасположением, систему обозначений можно упростить, приняв e_{j_l} за e_l. Теперь, приготовившись к нахождению значений ошибок e_1 и e_2, связанных с позициями $\beta_1 = \alpha^3$ и $\beta_2 = \alpha^4$, можно использовать любое из четырех синдромных уравнений. Выразим из уравнения (29) $\boldsymbol{S}_1$ и $\boldsymbol{S}_2$:

$$\left.\begin{aligned} \boldsymbol{S}_1 &= \boldsymbol{r}(\alpha) = e_1\beta_1 + e_2\beta_2, \\ \boldsymbol{S}_2 &= \boldsymbol{r}(\alpha^2) = e_1\beta_1^2 + e_2\beta_2^2. \end{aligned}\right\} \tag{41}$$

Эти уравнения можно переписать в матричной форме следующим образом:

$$\begin{bmatrix} \beta_1 & \beta_2 \\ \beta_1^2 & \beta_2^2 \end{bmatrix}\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_1 \\ \boldsymbol{S}_2 \end{bmatrix}, \tag{42}$$

$$\begin{bmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^8 \end{bmatrix}\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \alpha^3 \\ \alpha^5 \end{bmatrix}. \tag{43}$$

Чтобы найти значения ошибок e_1 и e_2, нужно выполнить поиск обратной матрицы для уравнения (43):

$$\mathrm{Inv}\begin{bmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^1 \end{bmatrix} = \frac{\begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix}}{\alpha^3\alpha^1 - \alpha^6\alpha^4} = \frac{\begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix}}{\alpha^4 + \alpha^3} =$$

$$= \alpha^{-6}\begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix} = \alpha^1\begin{bmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha^5 \\ \alpha^7 & \alpha^4 \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha^5 \\ \alpha^0 & \alpha^4 \end{bmatrix}. \tag{44}$$

Теперь можно найти из уравнения (43) значения ошибок:

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha^5 \\ \alpha^0 & \alpha^4 \end{bmatrix}\begin{bmatrix} \alpha^3 \\ \alpha^5 \end{bmatrix} = \begin{bmatrix} \alpha^5 + \alpha^{10} \\ \alpha^3 + \alpha^9 \end{bmatrix} = \begin{bmatrix} \alpha^5 + \alpha^3 \\ \alpha^3 + \alpha^2 \end{bmatrix}\begin{bmatrix} \alpha^2 \\ \alpha^5 \end{bmatrix}. \tag{45}$$

Исправление принятого полинома с помощью найденного полинома ошибок. Из уравнений (40) и (41) находим полином ошибок:

$$\hat{\boldsymbol{e}}(\boldsymbol{X}) = e_1 X^{j_1} + e_2 X^{j_2} = \alpha^2 X^3 + \alpha^5 X^4. \tag{46}$$

Показанный алгоритм восстанавливает принятый полином, выдавая в итоге предполагаемое переданное кодовое слово и в конечном счете декодированное сообщение.

$$\hat{\boldsymbol{U}}(\boldsymbol{X}) = \boldsymbol{r}(\boldsymbol{X}) + \hat{\boldsymbol{e}}(\boldsymbol{X}) = \boldsymbol{U}(\boldsymbol{X}) + \boldsymbol{e}(\boldsymbol{X}) + \hat{\boldsymbol{e}}(\boldsymbol{X}). \tag{47}$$

$$\left.\begin{aligned}
&\boldsymbol{r(X)} = (100) + (001)X + (011)X^2 + (100)X^3 + (101)X^4 + (110)X^5 + (111)X^6,\\
&\hat{\boldsymbol{e}}\boldsymbol{(X)} = (000) + (000)X + (000)X^2 + (001)X^3 + (111)X^4 + (000)X^5 + (000)X^6,\\
&\hat{\boldsymbol{U}}\boldsymbol{(X)} = (100) + (001)X + (011)X^2 + (101)X^3 + (010)X^4 + (110)X^5 + (111)X^6 =\\
&= \alpha^0 + \alpha^2 X + \alpha^4 X^2 + \alpha^6 X^3 + \alpha^1 X^4 + \alpha^3 X^5 + \alpha^5 X^6.
\end{aligned}\right\} \quad (48)$$

Поскольку символы сообщения содержатся в трех крайних правых символах, декодированным будет сообщение

$$\underset{\alpha^4}{010}\ \underset{\alpha^3}{110}\ \underset{\alpha^5}{111}.$$

Оно в точности соответствует выбранному для этого примера сообщению.

14. КОДЫ С ЧЕРЕДОВАНИЕМ

Ранее подразумевалось, что у канала отсутствует память, поскольку рассматривались коды, которые должны были противостоять случайным независимым ошибкам [1 – 7]. Канал с памятью – это такой канал, в котором проявляется взаимная зависимость ухудшений передачи сигнала. Канал, в котором происходит замирание сигнала вследствие его многолучевого распространения (сигнал поступает на приемник по двум или более путям различной длины), есть пример канала с памятью. Следствие замирания – различная фаза сигналов, и в итоге суммарный сигнал оказывается искаженным. Замирание сигнала характерно для каналов мобильной беспроводной связи, так же как для ионосферных и тропосферных каналов. В некоторых каналах (например, телефонные каналы или каналы с возникающими импульсными помехами) имеются коммутационные и другие импульсные помехи. Все эти ухудшения коррелируют во времени и в результате дают статистическую взаимную зависимость успешно переданных сигналов. Иными словами, искажения вызывают не отдельные изолированные ошибки, а пакет ошибок.

Если канал имеет память, то ошибки не являются независимыми, одиночными и случайно распределенными. Большинство блочных и сверточных кодов разрабатывается для борьбы с независимыми

случайными ошибками. Влияние канала с памятью на кодированный таким образом сигнал приведет к ухудшению достоверности передачи. Существуют схемы кодирования для каналов с памятью, но наибольшую проблему в этом кодировании представляет расчет точных моделей сильно нестационарных статистик таких каналов. Подход, при котором требуется знать только объем памяти канала, а не его точное статистическое описание, использует временное разнесение, или чередование, битов.

Чередование битов кодированного сообщения перед передачей и обратная операция после приема приводят к рассеиванию пакета ошибок во времени. Таким образом, они становятся для декодера случайно распределенными. Поскольку в реальной ситуации память канала уменьшается с временным разделением, идея, лежащая в основе метода чередования битов, заключается в разнесении символов кодовых слов во времени. Получаемые промежутки времени точно так же заполняются символами других кодовых слов. Разнесение символов во времени эффективно превращает канал с памятью в канал без памяти и, следовательно, позволяет использовать коды с коррекцией случайных ошибок в канале с импульсными помехами.

Устройство чередования смешивает кодовые символы в промежутке нескольких длин блоков (для блочных кодов) или нескольких длин кодового ограничения (для сверточных кодов). Требуемый промежуток определяется длительностью пакета. Подробности структуры битового перераспределения должны быть заданы в приемнике, чтобы он мог выполнить восстановление порядка битов перед декодированием. Идея чередования битов используется во всех блочных и сверточных кодах. Обычно применяют два типа устройств чередования – блочные и сверточные.

Блочное чередование. Блочное устройство чередования принимает кодированные символы блоками от кодера, переставляет их, а затем передает измененные символы на модулятор. Как правило, перестановка блоков завершается заполнением столбцов матрицы M строками и N столбцами ($M \times N$) кодированной последовательности. После того как матрица полностью заполнена, символы подаются на

модулятор (по одной строке), а затем передаются по каналу. В приемнике устройство восстановления выполняет обратные операции: оно принимает символы из демодулятора, восстанавливает исходный порядок битов и передает их на декодер. Символы поступают в массив устройства восстановления в виде строк и заменяются столбцами. Выходная последовательность, предназначенная для передатчика, состоит из кодовых символов, которые построчно удалены из массива, как показано на рис. 15.

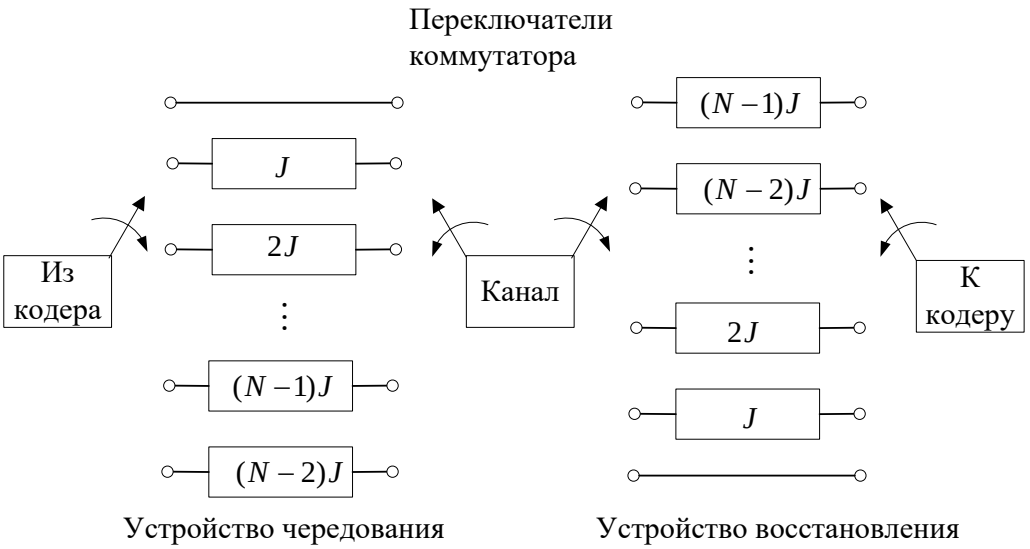


Рис. 15. Схема чередования символов

Ниже перечислены наиболее важные характеристики описываемого блочного устройства.

1. Пакет, который содержит меньше N последовательных канальных символов, дает на выходе устройства восстановления исходного порядка символов ошибки, разнесенные между собой по крайней мере на M символов.

2. Пакет из bN ошибок, где $b > 1$, дает на выходе устройства восстановления пакет, который содержит не меньше $\lfloor b \rfloor$ символьных ошибок. Каждый из пакетов ошибок отделен от другого не меньше чем на $M - \lceil b \rceil$ символов. Запись $\lfloor b \rfloor$ указывает на операцию округления

некоторого числа b до ближайшего меньшего целого числа, запись $\lceil b \rceil$ – операцию округления некоторого числа b до ближайшего большего целого числа.

3. Периодическая последовательность одиночных ошибок, разделенных N символами, дает на выходе устройства восстановления одиночные пакеты ошибок длиной M.

4. Прямая задержка между устройствами чередования и восстановления равна приблизительно длительности $2MN$ символов. Точнее, перед тем как начать передачу, нужно заполнить лишь $M(N-1)+1$ ячеек памяти (как только будет внесен первый символ последнего столбца массива $M \times N$). Соответствующее время нужно приемнику, чтобы начать декодирование. Значит, минимальная прямая задержка составляет длительность $2MN - 2M + 2$ символов без учета задержки на передачу по каналу.

5. Необходимая память составляет MN символов для каждого объекта (устройств чередования и восстановления исходного порядка). Однако массив $M \times N$ нужно заполнить (по большей части) до того, как он будет считан. Для каждого объекта нужно предусмотреть память для $2MN$ символов, чтобы опорожнить массив $M \times N$, пока другой наполняется, и наоборот.

Сверточное чередование. Сверточные устройства чередования рассматриваются в работах [9; 10].

Кодовые символы последовательно подаются в блок из N регистров; каждый последующий регистр может хранить на J символов больше, чем предыдущий. Нулевой регистр не предназначен для хранения (символ сразу же передается). С каждым новым кодовым символом коммутатор переключается на новый регистр, и кодовый символ подается на него до тех пор, пока наиболее старый кодовый символ в регистре не будет передан на модулятор/передатчик. После $N-1$ регистра коммутатор возвращается к нулевому регистру и повторяет все снова. После приема операции повторяются в обратном порядке. Вход и выход устройств чередования и восстановления должны быть синхронизированы.

ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

1. Сконструировать код с проверкой на четность с параметрами (*n, k)*, который определяет все модели, содержащие 1, 3, 5 и 7 ошибочных битов. Найти значения *n* и *k* и определить вероятность невыявленной ошибки в блоке, если вероятность ошибки в канальном символе равна 10^{-2}.

2. Определить вероятность ошибки в сообщении для 12-битовой последовательности данных, кодированной линейным блочным кодом (24, 12). Считать, что код может исправлять одно- и двухбитовые модели ошибки; модели более чем с двумя ошибками не подлежат исправлению; вероятность ошибки в канальном символе равна 10^{-3}.

3. Пусть используется линейный блочный код (127, 92), который может исправлять трехбитовые ошибки. Для этого случая определить:

а) чему равна вероятность ошибки в сообщении для некодированного блока из 92 битов, если вероятность ошибки в канальном символе равна 10^{-3};

б) чему равна вероятность ошибки для сообщения, кодированного блочным кодом (127, 92), если вероятность ошибки в канальном символе равна 10^{-3}.

4. Рассчитать уменьшение вероятности ошибки в сообщении, кодированном линейным блочным кодом (24, 12) с коррекцией двухбитовых ошибок, по сравнению с некодированной передачей. Пусть используется когерентная модуляция BPSK и принятое отношение $E_b/N_0 = 10$ дБ.

5. Дан линейный блочный код (24, 12) с возможностью исправления двухбитовых ошибок. Пусть используется модуляция BPSK, а принятое отношение $E_b/N_0 = 14$ дБ.

а) Определить, дает ли такой код какое-либо уменьшение вероятности ошибки в сообщении? Если да, то насколько? Если нет, то почему?

б) Повторите исследования при $E_b/N_0 = 10$ дБ.

6. При передаче цифровых сигналов применяется кодер типа «лучший из пяти» для некоторых цифровых каналов данных. В такой схеме все биты данных повторяются пять раз. В приемнике выполняется мажоритарное декодирование сообщения. Если вероятность ошибки в некодированном бите составляет 10^{-3} и используется кодирование «лучший из пяти», чему равна вероятность ошибки в декодированном бите?

7. Дана матрица генератора кода (7, 4) следующего вида:

$$\mathbf{G} = \begin{vmatrix} 1111000 \\ 1010100 \\ 0110010 \\ 1100001 \end{vmatrix}.$$

а) Определить все кодовые слова и проверочную матрицу **H**.

б) Рассчитать синдром для принятого вектора 1101101. Определить, правильно ли принят этот вектор?

в) Каковы возможности кода при обнаружении и исправлении ошибок?

8. Дан линейный блочный код, контрольные уравнения которого имеют следующий вид:

$p_1 = m_1 + m_2 + m_4$,

$p_2 = m_1 + m_3 + m_4$,

$p_3 = m_1 + m_2 + m_3$,

$p_4 = m_2 + m_3 + m_4$.

Здесь p_i – разряды информационного сообщения, m_i – контрольные разряды.

а) Найти для этого кода матрицу генератора и проверочную матрицу.

б) Определить, сколько ошибок может исправить этот код.

в) Выяснить, является ли вектор 10101010 кодовым словом.

г) Выяснить, является ли вектор 01011100 кодовым словом.

9. Код БЧХ (63, 36) может исправить пять ошибок. Девять блоков кода (7, 4) могут исправить девять ошибок. Оба кода имеют одинаковую степень кодирования. При этом код (7, 4) может исправить больше ошибок.

а) Является ли код (7, 4) более мощным?

б) Сравнить оба кода, когда наблюдается пять случайных ошибок в 63 битах.

10. Нарисовать диаграмму состояний, древовидную и решетчатую диаграммы для сверточного кодера, который описывается блочной диаграммой, показанной на рис. 16.

11. Рассмотреть сверточный кодер, показанный на рис. 17.

а) Записать векторы и полиномы связи для этого кодера.

б) Нарисовать диаграмму состояний, древовидную и решетчатую диаграммы.

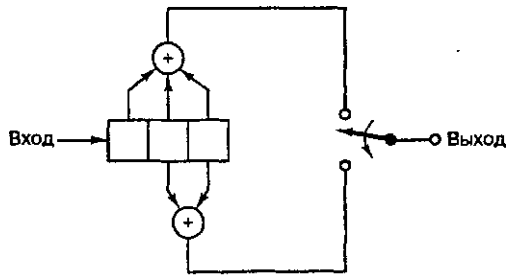


Рис. 16. Сверточный кодер (для задания 10)

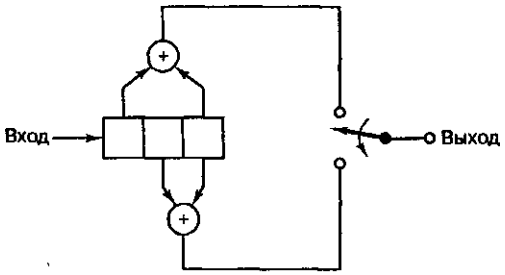


Рис. 17. Сверточный кодер (для задания 11)

12. Какова импульсная характеристика в задании 11? С помощью этой характеристики определить выходную последовательность, если на вход подается последовательность 101. Проверить полученный ответ с помощью полиномиальных генераторов.

13. Имеется ли в кодере (см. рис. 17) возможность для накопления катастрофической ошибки? Проиллюстрировать ответ с помощью примера.

14. Пусть кодовые слова в схеме кодирования имеют вид a = = 000000, b = 101010, c = 010101, d = 111111. Определить, каким будет декодированный символ в случае, если по двоичному симметричному каналу принимается последовательность 111010 и при этом осуществляется декодирование по принципу максимального правдоподобия.

15. Пусть в двоичном симметричном канале (binary symmetric channel – BSC) используется кодер со степенью кодирования 1/2 и $K = 3$. Код (7, 5). Начальное состояние кодера 00. На выходе канала BSC принимается последовательность $\boldsymbol{Z}$ (1100001011, остальные все нули).

а) Найти на решетчатой диаграмме (см. рис. 9) максимально правдоподобный путь и определить первые пять декодированных информационных битов. При наличии двух сливающихся путей выбирать верхнюю ветвь.

б) Определить канальные биты в последовательности $\boldsymbol{Z}$, которые подверглись искажению в ходе передачи.

16. Пусть используется сигнал BPSK с когерентным детектированием, кодируемый с помощью кодера (7, 5). Найти верхнюю границу вероятности появления битовой ошибки P_B, если номинальное значение E_b/N_0 равно 6 дБ. (Предполагается «жесткое» декодирование.) Сравнить значение P_B с некодированным случаем и определить выигрыш в отношении сигнал/шум.

17. С помощью последовательного декодирования изобразить путь вдоль древовидной диаграммы, если принята последовательность 0111000111. Критерий отката – три несовпадения.

18. Пусть дан сверточный кодер со степенью кодирования 2/3, показанный на рис. 18. За один такт в кодер подается два бита (k); три бита (n) подается на выход кодера. Имеется четыре разряда регистра (K), и длина кодового ограничения равна $K - 2$ в единицах двухбитовых байтов. Состояние кодера определяется как содержимое $K - 1$ крайних правых разрядов k-кортежа. Нарисовать диаграмму состояний, древовидную и решетчатую диаграммы.

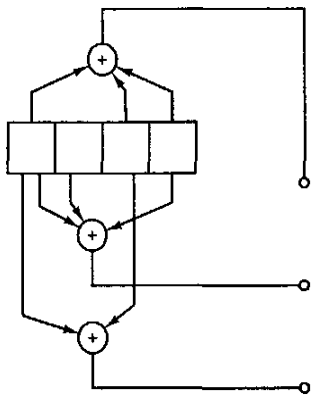


Рис. 18. Сверточный кодер (для задания 18)

19. Определить, какой из следующих полиномов примитивный (можно применить способ LFSR):

а) $1 + X^2 + X^3$;

б) $1 + X + X^2 + X^3$;

в) $1 + X^2 + X^4$;

г) $1 + X^3 + X^4$;

д) $1 + X + X^2 + X^3 + X^4$;

е) $1 + X + X^5$;

ж) $1 + X^2 + X^5$;

з) $1 + X^3 + X^5$;

и) $1 + X^4 + X^5$.

20. Определить множество элементов $\{0, \alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{2m-2}\}$ через образующие элементы конечного поля $GF(2^m)$ при $m = 4$.

а) Для конечного поля составить таблицу сложения.

б) Для конечного поля построить таблицу умножения.

в) Найти полиномиальный генератор для кода (31, 27).

г) Кодировать сообщение {96 нулей, затем 110010001111} (крайний правый бит является первым) систематическим кодом (7, 3). Определить, почему сообщение построено с таким большим количеством нулей в начале?

21. С помощью полиномиального генератора для кода (7, 3) кодировать сообщение 010110111 (крайний правый бит является первым) в систематической форме. Для нахождения полинома контроля четности использовать полиномиальное деление. Представить итоговое кодовое слово в двоичной и полиномиальной формах.

22. Применить регистр LFSR для кодирования сообщения {6, 5, 1} (крайний правый бит является первым) с помощью кода (7, 3) в систематической форме. Представить итоговое кодовое слово в двоичной форме. Проверить результат кодирования путем вычисления полинома кодового слова со значениями корней полиномиального генератора кода (7, 3).

23. Пусть кодовое слово, найденное в задаче 22, искажается в ходе передачи, в результате чего крайние правые шесть битов оказались инвертированы. Найти значения всех синдромов путем вычисления полинома поврежденного кодового слова со значениями корней

полиномиального генератора $\boldsymbol{g(X)}$. Проверить, можно ли значения синдромов найти путем вычисления полинома ошибок $\boldsymbol{e(X)}$ со значениями корней генератора $\boldsymbol{g(X)}$.

24. Последовательность 1011011000101100 подается на вход блочного устройства чередования размером 4 × 4. Какой будет выходная последовательность?

25. По каждому из следующих условий разработать устройство чередования для системы связи, действующей в канале с импульсными помехами со скоростью передачи 19 200 кодовых символов в секунду.

а) Как правило, пакет шума длится 250 мс. Системным кодом является код БЧХ (127, 36) при $d_{\min} = 31$. Прямая задержка не превышает 5 с.

б) Как правило, пакет шума длится 20 мс. Системным кодом является сверточный код (127, 36) с обратной связью и степенью кодирования 1/2, способный корректировать в среднем три символа в последовательности из 21 символа. Прямая задержка не превышает 160 мс.

26. На рис. 19 показан рекурсивный систематический сверточный (recursive systematic convolutional – RSC) кодер со степенью кодирования 2/3, $K = 3$. Используется формат памяти в виде однобитовых блоков задержек. Составить таблицу, которая определяет все возможные переходы в цепи. С ее помощью изобразить участок решетки. Найти выходное кодовое слово для информационной последовательности 1100110011. В течение каждого такта данные поступают в цепь в виде пары значений $\{d_{2k-1}, d_{2k}\}$, а каждое выходное кодовое слово $\{d_{2k-1}, d_{2k}, v_k\}$ образуется из пары битов данных и одного контрольного бита v_k.

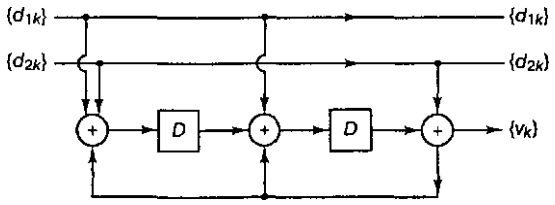


Рис. 19. Сверточный кодер (для задания 26)

ТЕСТ ДЛЯ САМОКОНТРОЛЯ

1. За счет каких факторов при кодировании различными методами возникает выигрыш в помехоустойчивости передачи сигналов:

- введение различных видов модуляции;
- введение избыточности символов;
- сужение занимаемой полосы частот?

2. Основное отличие сверточных кодов от кодов прочих видов заключается в следующем:

- каждый кодовый символ получается на основе нескольких предыдущих информационных символов;
- каждый кодовый символ получается на основе одного специально выбранного предыдущего информационного символа;
- каждый кодовый символ получается умножением очередного информационного символа на предыдущий кодовый символ.

3. Основное отличие блоковых кодов от кодов прочих видов заключается:

- в поочередной группировке информационных и кодовых символов;
- формировании на приемной стороне блоков из принятых символов;
- вычислении кодовых символов на основе блоков информационных символов.

4. Основное различие систематических и несистематических кодов в сверточных и блоковых алгоритмах кодирования заключается в следующем:

- кодовые символы формируются систематически или бессистемно;
- в формируемой кодовой последовательности можно или нельзя выделить информационные символы;
- виды кодов образуют или не образуют строгую систему для их классификации.

5. Основное различие двоичных и недвоичных блоковых кодов заключается в следующем:

- каждый символ принимает значения из двоичной (троичной, четвертичной и т. д.) системы счисления;
- для недвоичных кодов необходимо использовать специальные виды модуляции;
- для формирования кодов требуется использовать алгебру Буля или алгебру Галуа.

6. Какой самый простой путь построения блоковых кодов:

- с помощью таблиц соответствия;
- порождающих полиномов;
- порождающих матриц?

7. В чем различия кода Хемминга и кодов БЧХ:

- в принципах формирования;
- сложности получения;
- использовании различной элементной базы?

8. В чем состоит назначение модификации кодов и выигрыш от ее применения:

- удешевление применяемой элементной базы;
- увеличение помехоустойчивости;
- изменение кодовой скорости?

9. Каковы особенности алгоритма перфорации кодов:

- удаление части кодовых символов после кодирования;
- удаление части информационных символов до кодирования;
- удаление части принятых символов на приемной стороне;
- передача различных кодовых символов с разной мощностью?

10. Как образуется структура каскадных кодов:

- путем последовательных операций с применением нескольких видов кодирования;
- путем параллельных операций с применением нескольких видов кодирования;
- на различных интервалах радиолинии применяются различные виды кодирования?

11. Каковы достоинства итеративно декодируемых кодов:
- повышение помехоустойчивости;
- реализация «мягкого» декодирования;
- применение однотипной элементной базы?
12. Каковы недостатки итеративно декодируемых кодов:
- замедление операции декодирования;
- возможные срывы алгоритмов декодирования;
- увеличение объема и массы декодеров?
13. Каковы особенности сигнально-кодовых конструкций:
- упрощение операции кодирования;
- комплексное осуществление операций кодирования и модуляции;
- упрощение передачи кодированных сигналов;
- упрощение приема кодированных сигналов?
14. Каковы основные принципы декодирования сверточных кодов:
- вычисление и сравнение метрик различных путей по кодовой решетке;
- обратное развертывание кодов;
- отбрасывание ненужных проверочных символов?
15. Каковы основные принципы декодирования блоковых кодов:
- разбиение принятых блоков в одномерную последовательность;
- вычисление синдромов;
- преобразование блокового кода в сверточный код;
- запрос на повторную передачу поврежденных символов?

ЗАКЛЮЧЕНИЕ

В пособии рассмотрены основные типы применяемых в настоящее время кодов. Приведены сравнительные характеристики их эффективности и особенности формирования. Даны рекомендации по использованию простых кодов и наиболее перспективных, таких как турбокоды и LDPC-коды. Рассмотрены особенности методов многопорогового кодирования и сигнально-кодовых конструкций.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Прокис, Дж.* Цифровая связь / Дж. Прокис. – М. : Радио и связь, 2000. – 800 с. – ISBN 5-256-01434-X.

2. *Скляр, Б.* Цифровая связь. Теоретические основы и практическое применение / Б. Скляр. – М. : Вильямс, 2003. – 1104 с. – ISBN 5-8459-0386-6.

3. *Полушин, П. А.* Избыточность сигналов в радиосвязи / П. А. Полушин, А. Г. Самойлов. – М. : Радиотехника, 2007. – 256 с. – ISBN 5-88070-121-2.

4. *Морелос-Сарагоса, Р.* Искусство помехоустойчивого кодирования : методы, алгоритмы, применение / Р. Морелос-Сарагоса. – М. : Техносфера, 2006. – 320 с. – ISBN 5-94836-035-0.

5. *Берлекэмп, Э. Р.* Алгебраическая теория кодирования / Э. Р. Берлекэмп. – М. : Мир, 1971. – 477 с.

6. *Блейхут, Р.* Теория и практика кодов, контролирующих ошибки / Р. Блейхут. – М. : Мир, 1986. – 576 с.

7. *Витерби, А. Д.* Принципы цифровой связи и кодирования / А. Д. Витерби, Дж. К. Омура. – М. : Радио и связь, 1982. – 536 с.

8. *Питерсон, У.* Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон ; под ред. Р. Л. Добрушина и С. И. Самойленко. – М. : Мир, 1976. – 593 с.

9. Теория кодирования / Т. Касами [и др.] ; под. ред. С. И. Гельфанда и Б. С. Цыбакова. – М. : Мир, 1978. – 576 с.

10. *Форни, Д.* Каскадные коды / Д. Форни ; под ред. С. И. Самойленко. – М. : Мир, 1970. – 207 с.

11. *Шеннон, К.* Работы по теории информации и кибернетике / К. Шеннон. – М. : Изд-во иностр. лит., 1963. – 829 с.

12. *Кларк, Дж., мл.* Кодирование с исправлением ошибок в системах цифровой связи / Дж. Кларк, мл., Дж. Кейн ; под ред. Б. С. Цыбакова. – М. : Радио и связь, 1987. – 392 с.

13. *Мак-Вильямс, Ф. Дж.* Теория кодов, исправляющих ошибки / Ф. Дж. Мак-Вильямс, Н. Дж. А. Слоэн. – М. : Связь, 1979. – 744 с.

14. *Марков, А. А.* Введение в теорию кодирования / А. А. Марков. – М. : Наука, 1982. – 201 с.

15. *Блох, Э. Л.* Линейные каскадные коды / Э. Л. Блох, В. В. Зяблов. – М. : Наука, 1982. – 229 с.

16. *Блох, Э. Л.* Обобщенные каскадные коды / Э. Л. Блох, В. В. Зяблов. – М. : Связь, 1976. – 240 с.

17. *Мэсси, Дж.* Пороговое декодирование / Дж. Мэсси. – М. : Мир, 1966. – 207 с.

18. *Берлекэмп, Э. Р.* Техника кодирования с исправлением ошибок / Э. Р. Берлекэмп // Труды Института инженеров по электронике и радиотехнике. – 1980. – Т. 68, № 5. – С. 24 – 58.

19. *Золотарев, В. В.* Помехоустойчивое кодирование / В. В. Золотарев, Г. В. Овечкин. – М. : Горячая линия – Телеком, 2004. – 126 с. – ISBN 5-93517-169-4.

20. *Квашенников, В. В.* Адаптивное помехоустойчивое кодирование в технике связи / В. В. Квашенников, А. Д. Кухарев. – Калуга : Изд-во науч. лит. Н. Ф. Бочкаревой, 2007. – 147 с. – ISBN 5-89552-234-3.

ОГЛАВЛЕНИЕ

Учебное издание

НИКИТИН Олег Рафаилович
ПОЛУШИН Петр Алексеевич

СОВРЕМЕННЫЕ МЕТОДЫ КОДИРОВАНИЯ ИНФОРМАЦИИ

Учебное пособие

Редактор Т. В. Евстюничева

Технический редактор А. В. Родина

Корректор О. В. Балашова

Компьютерная верстка Л. В. Макаровой

Выпускающий редактор А. А. Амирсейидова

Подписано в печать 02.10.18.
Формат 60x84/16. Усл. печ. л. 4,65. Тираж 50 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.