А.В. Тютякин

ОСНОВЫ ЭФФЕКТИВНОГО И ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ СООБЩЕНИЙ

ГУПК
Государственный университет

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ - УЧЕБНО-НАУЧНО-
ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС»

А.В. Тютякин

ОСНОВЫ ЭФФЕКТИВНОГО И ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ СООБЩЕНИЙ

Рекомендовано ФГБОУ ВПО «Госуниверситет - УНПК»
для использования в учебном процессе в качестве учебного пособия
для высшего профессионального образования

Орел 2015

УДК 004.627+519.725
ББК 32.965 – 044.3
Т98

Рецензенты:

доктор технических наук, профессор кафедры
«Электроника, вычислительная техника и информационная безопасность»
Федерального государственного бюджетного образовательного
учреждения высшего профессионального образования
«Государственный университет - учебно-научно-
производственный комплекс»
А.И. Суздальцев,

кандидат технических наук, доцент, профессор
Академии Федеральной службы охраны Российской Федерации
А.И. Еременко,

кандидат технических наук, старший преподаватель
Академии Федеральной службы охраны Российской Федерации
А.А. Двилянский

Тютякин, А.В.

Т98 Основы эффективного и помехоустойчивого кодирования сообщений: учебное пособие для высшего профессионального образования / А.В. Тютякин. – Орел: ФГБОУ ВПО «Госуниверситет - УНПК», 2015. – 180 с.

ISBN 978-5-93932-810-4

Учебное пособие состоит из двух разделов, в первом из которых излагаются математические и алгоритмические основы эффективного кодирования (сжатия) сообщений, а во втором – их помехоустойчивого кодирования.

Предназначено студентам высших учебных заведений, обучающимся по направлениям в областях информационных и инфокоммуникационных технологий и информационной безопасности, при изучении дисциплин «Общая теория связи», «Теория кодирования, сжатия и восстановления информации», «Теория информации» и других, подобных им по содержанию.

УДК 004.627+519.725
ББК 32.965 – 044.3

ISBN 978-5-93932-810-4

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

$d_{\min}$ – минимальное кодовое расстояние помехоустойчивого кода
$GF(p)$ – простое конечное поле
$GF(p^m)$ – расширенное конечное поле
$G(x)$ – порождающий полином
$H(X)$ – априорная информационная энтропия
$(\mathrm{mod}\, p)$ – указатель на выполнение математической операции по модулю p
$(\mathrm{mod}\, G(x))$ – указатель на выполнение математической операции по модулю полинома $G(x)$
$RM(r,q)$ – код Рида – Маллера с параметрами r и q [см. выражение (2.24) на с. 121]

АДИКМ – адаптивная дифференциальная импульсно-кодовая модуляция
БЛПК – блочный линейный помехоустойчивый код
БПФ – быстрое преобразование Фурье
БЧХ – код Боуза – Чоудхури – Хоквингема
ДВП – дискретное вейвлет-преобразование
ДИКМ – дифференциальная импульсно-кодовая модуляция
ДКП – дискретное косинусное преобразование
ДПФ – дискретное преобразование Фурье
ИЭ – информационная энтропия
ОДВП – обратное дискретное вейвлет-преобразование
РСК – распределение символов по кодам
СПК – сверточный помехоустойчивый код
ФКС – физический канал связи
ЦПК – циклический помехоустойчивый код
ЦФ – цифровой фильтр
ARQ – Automatic Repetition reQuest (автоматический запрос повторной передачи)
BER – Bit-Error Rate (интенсивность битовых ошибок)
BWT – Burrows – Wheeler Transform (преобразование Барроуза – Уилера)
CTW – Context Tree Weighting (взвешивание с применением контекстного дерева)

HCCC – Hybrid Concatenated Convolutional Coding (гибридное сверточное турбо-кодирование)
LPC – Linear Predictive Coding (кодирование с линейным предсказанием)
LRC – Longitudinal Redundancy Check (продольный контроль четности)
LZW – Lempel – Ziv – Welch
MTF – Move To Front (кодирование способом «Перемещение вперед»)
PCCC – Parallel Concatenated Convolutional Coding (сверточное турбо-кодирование на основе параллельной конкатенации)
PPM – Prediction by Partial Matching (предсказание по частичному совпадению)
RLE – Run – Length Encoding (кодирование длин серий)
SCCC – Serial Concatenated Convolutional Coding (сверточное турбо-кодирование на основе последовательной конкатенации)
TPC – Turbo Product Coding (блочное турбо-кодирование)

ВВЕДЕНИЕ

Основными процедурами кодирования сообщений при передаче и хранении сообщений в современных информационных системах являются:

- *эффективное кодирование* (называемое также *компактным* или *экономным кодированием*, *архивированием*, *сжатием*, *компрессией*) сообщений с целью минимизации времени их передачи по каналам связи и/или занимаемого ими объема на носителе;

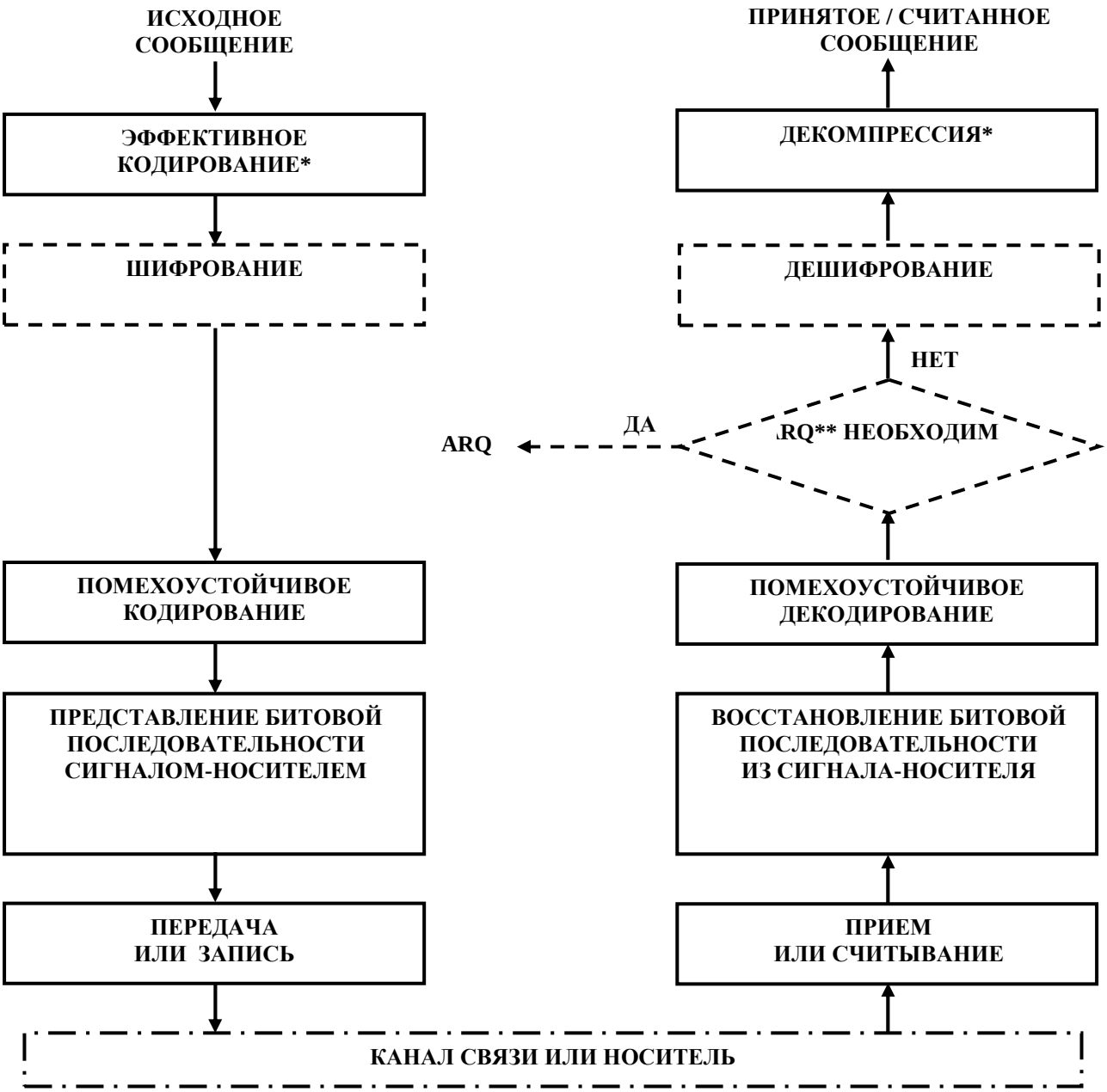
- *помехоустойчивое кодирование* с целью обнаружения и исправления искажений отдельных битов сообщения или их последовательностей, практически неизбежных как при передаче сообщения по каналам связи, так и при их записи на носитель и считывании с него;

- *шифрование* с целью защиты сообщения от несанкционированного доступа.

Процессы передачи сообщения по каналу связи и его приема, а также записи сообщения на носитель и его считывания описываются обобщенной *операционной моделью*, представленной на рис. 1 [1]. Термин «операционная модель» означает описание, обычно в графической форме, совокупности операций, необходимых для реализации некоторой процедуры. При этом часть этих операций могут реализовываться в программной форме, часть – в аппаратной, поэтому для описания их совокупности не применимы ни термин «алгоритм», ни «структурная схема».

Настоящее пособие посвящено основам эффективного и помехоустойчивого кодирования сообщений.

Целью настоящего пособия является формирование у студентов основных профессиональных компетенций в области знания и практического применения современных способов и средств представления и кодирования данных в информационных системах.



*В ряде частных случаев процедуры эффективного кодирования и шифрования могут быть совмещены (в простейшем из подобных случаев компрессия реализуется по алгоритму, требующему знания ключа при декомпрессии);

**ARQ – автоматический запрос повтора передачи или считывания сообщения

Рис. 1. Обобщенная операционная модель кодирования сообщений в информационных системах (пунктирной линией обозначены процедуры, которые могут отсутствовать в ряде частных случаев)

1. ОСНОВЫ ЭФФЕКТИВНОГО КОДИРОВАНИЯ

1.1. Общие положения

Эффективным кодированием (сжатием, компрессией, экономным или *компактным кодированием)* сообщения называют его преобразование в форму, требующую меньшего объема для представления сообщения, чем до преобразования, при возможности однозначного восстановления копии исходного сообщения из результата преобразования, с отклонениями от оригинала, не превышающими некоторые, наперед заданные. Последние, в свою очередь, зависят от характера и назначения сообщения. Восстановление копии исходного сообщения из результата эффективного кодирования называется *декомпрессией.* В зависимости от характера ее отличий от оригинала способы и алгоритмы сжатия разделяются на две группы:

- способы и алгоритмы *сжатия без потерь информации*, обеспечивающие полное (с точностью до бита) соответствие восстановленного при декомпрессии сообщения исходному;

- способы и алгоритмы *сжатия с потерями данных*, для которых характерны отклонения восстановленного сообщения от исходного, характер и значения которых зависят от алгоритма сжатия, а также его *профиля*, т. е. сочетания его параметров и опций.

Сжатие без потерь применяется при компактном кодировании данных, для которых недопустимы искажения даже одного бита. К таковым, в первую очередь, относятся текстовые сообщения (например, при кодировании *ASCII*-кодом искажение одного бита может привести к замене цифры «1» на цифру «9», буквы «*a*» на букву «*b*» и т. п.), а также графическая информация научно-технического назначения. Сжатие с потерями информации используется, в основном, при компактном кодировании мультимедийных данных (звукозаписи, неподвижные изображения и видеоданные), для которых такие потери допустимы, при условии, что они незаметны или малозаметны при восприятии слушателем или, соответственно, зрителем.

Нижний теоретический предел объема сообщения (в битах) после сжатия без потерь задается следующим выражением:

$$V_{\min} = H(X) \times N, \tag{1.1}$$

где N – общее число символов в сообщении, в качестве которых могут служить как буквы, цифры, знаки препинания и служебные символы текстового сообщения, так и, например, значения яркости пикселей полутонового изображения;

$H(X)$ – *априорная информационная энтропия* (ИЭ) (другими словами – изначальная неопределенность) сообщения, бит / символ.

При этом $H(X)$ определяется следующим образом. Для *не Марковских* источников сообщений, характеризуемых независимостью вероятности появления в них каждого из символов их алфавита от предшествующих ему символов, априорная информационная энтропия равна:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \times \log_2[p(x_i)], \tag{1.2}$$

где n – число символов в алфавите сообщения (например, $n = 256$ при *ASCII*-кодировке текстового файла или при представлении яркостей пикселей 8-битовым двоичным кодом);

$p(x_i)$ – вероятность появления i-го символа (например, буквы «*а*» или двоичного значения отсчета яркости, равного 01101100) в сообщении.

В свою очередь, априорная ИЭ *Марковского* источника, характеризуемого зависимостью вероятности появления символа от предшествовавшей ему последовательности символов (*контекста*), описывается выражением

$$H(X) = -\sum_{j=1}^{n^k} p(c_j) \sum_{i=1}^{n} p(x_i / c_j) \times \log_2[p(x_i / c_j)], \tag{1.3}$$

где k – порядок Марковского источника, т. е. длина контекста (в символах);

c_j – j-я последовательность символов длиной k (j-й контекст порядка k);

$p(c_j)$ – вероятность появления j-го контекста (например, сочетания букв «*пр*») в сообщении;

$p(x_i / c_j)$ – вероятность появления символа x_i в контексте c_j (например, буквы «*и*» после сочетания букв «*пр*»).

Априорная ИЭ сообщения тем меньше, чем более оно «однообразно» по составу символов и их сочетаний. Поэтому, например, достижимая степень компрессии научно-технического текста потенциально выше, чем художественного.

Следует отметить, что, с одной стороны, большинство сообщений, встречающихся на практике, являются Марковскими, а с другой – априорная ИЭ Марковского источника при прочих равных условиях существенно ниже, чем не Марковского. Поэтому большинство используемых на практике способов и алгоритмов сжатия рассматривают источники кодируемых сообщений как Марковские. При этом порядок контекста, используемый при сжатии, как правило, выбирается пользователем и может не совпадать с реальным порядком источника. Например, научно-технические и литературные тексты на русском языке реально характеризуются порядком Марковской модели, равным от 4 до 6 (в зависимости от содержания и лексики конкретного текста). Однако, например, алгоритм *PPMd*, реализуемый архиватором *7-Zip*, позволяет пользователю выбрать значение порядка Марковской модели, используемой при сжатии, из диапазона от 2 до 16. Чем ниже указанный порядок по сравнению с реальным, тем меньше время кодирования и декодирования, но тем меньше и степень сжатия. Также необходимо отметить, что ряд алгоритмов сжатия, в том числе вышеупомянутый *PPMd*, использует в процессе кодирования не одну, а ряд контекстных моделей сообщения с порядками от выбранного пользователем максимального до нулевого, т. е. до не Марковской модели (см. далее п. 1.2.5).

При сжатии с потерями может быть достигнут объем сообщения, меньший задаваемого выражением (1.1), причем тем меньший, чем выше потери информации при кодировании. Соотношение между степенью сжатия сообщения и потерями информации определяется конкретным профилем алгоритма кодирования, как правило, выбираемым пользователем в зависимости от конкретных требований к объему сжатого сообщения и качеству его восстановления после декомпрессии.

Классификация способов и алгоритмов эффективного кодирования может осуществляться по различным критериям [2, 3]. В настоящем пособии в качестве основного из них используется назначение тех или иных способов и алгоритмов кодирования, т. е. классы сообщений, для сжатия которых они предназначены.

1.2. Способы и алгоритмы эффективного кодирования дискретных сообщений

Дискретными называют сообщения, алфавит которых по своей сущности представляет собой конечное множество символов (буквы, цифры, знаки препинания, служебные символы и т. п.). Типовыми примерами таких сообщений являются файлы с расширениями *.doc*, *.txt* или *.xls*.

Эффективное кодирование сообщений данного типа должно осуществляться *без потерь информации*, т. е. алгоритмы кодирования и декодирования сообщения должны обеспечивать полное совпадение сообщения, полученного в результате декомпрессии, с исходным. Недопустимы искажения даже одного бита в коде сообщения, так как они могут привести к серьезным искажениям его смысла (см. п. 1.1).

Классификация способов кодирования, на которых базируются основные известные алгоритмы сжатия дискретных сообщений без потерь, представлена на рис. 1.1 [2, 3].

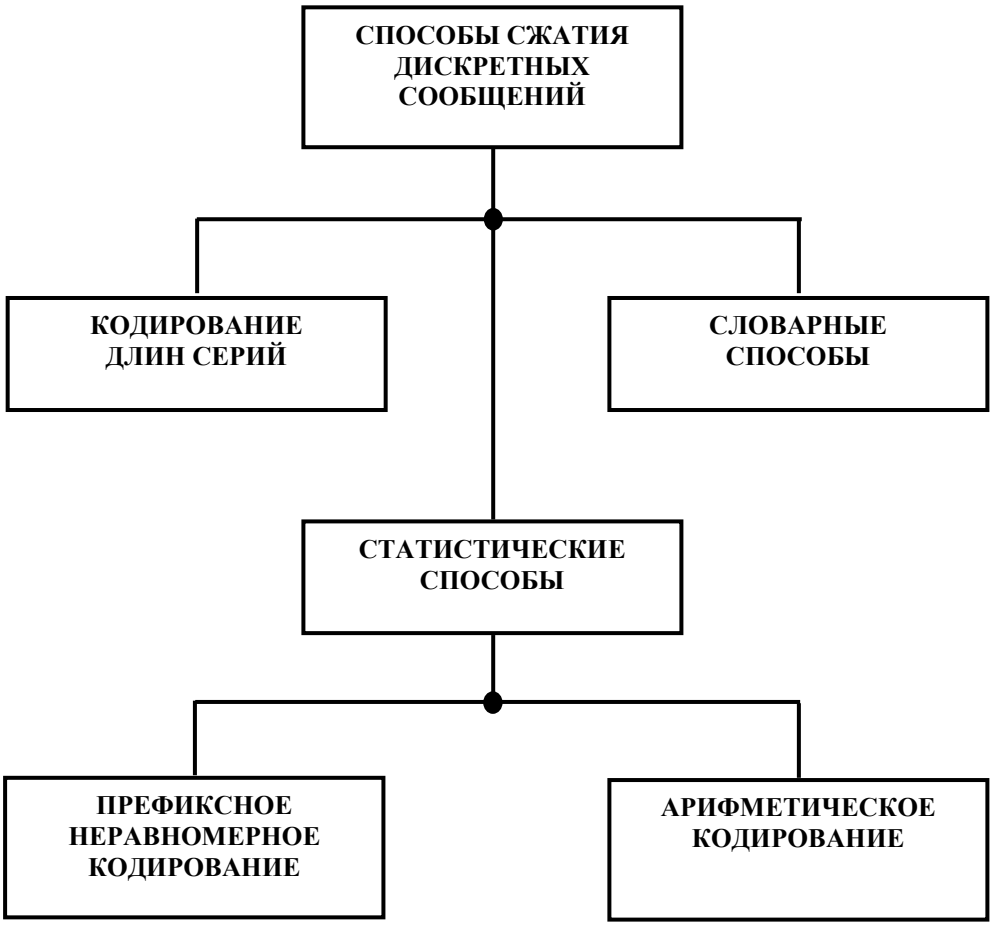


Рис. 1.1. Основные способы эффективного кодирования дискретных сообщений без потерь

Следует отметить, что большинство распространенных программных средств архивирования дискретных сообщений использует сочетание двух или более способов (алгоритмов) кодирования (см. табл. 1.10 в п. 1.2.8).

Рассмотрим сущность указанных способов и типовые алгоритмы их реализации.

1.2.1. Кодирование длин серий

Способ *кодирования длин серий* (*RLE*, *Run – Length Encoding*) наиболее прост по сущности и в реализации. Состоит в замене серии из N повторяющихся символов двумя кодами – повторяющегося символа и числа N. Данный способ эффективен при кодировании сообщений, содержащих длинные последовательности повторяющихся кодов (что характерно для ряда типов графических сообщений, например, для факсимильных). При отсутствии таковых (что имеет место в текстовых сообщениях на естественных языках) он малоэффективен. Однако существуют способы обратимого преобразования сообщения, в результате которого его символы группируются в последовательности, эффективно кодируемые способом *RLE*. Наиболее распространенной разновидностью такого преобразования является *преобразование Барроуза – Уилера* (см. п. 1.2.7).

В целом, для сжатия дискретных сообщений способ *RLE* применяется сравнительно редко и только в комбинации с другими способами преобразования и кодирования. В то же время он относительно широко используется (как правило, также в сочетании с другими способами сжатия), например, в алгоритмах эффективного кодирования изображений (см. п. 1.4).

1.2.2. Статистические (энтропийные) способы эффективного кодирования: общие положения

Статистические способы, называемые также *энтропийными* или *вероятностными*, характеризуются тем, что коды отдельных символов, их групп или код сообщения в целом формируются в зависимости от статистических характеристик сообщения, в первую очередь – вероятностей появления в нем отдельных символов или последова-

тельностей символов. В зависимости от способа задания / определения указанных вероятностей различают *неадаптивное, полуадаптивное* и *адаптивное* статистическое кодирование.

При мало распространенном на практике *неадаптивном* кодировании вероятности символов или их сочетаний заранее оговариваются протоколом кодирования. В качестве значений указанных вероятностей обычно служат их средние значения по некоторой группе сообщений. Как кодирование, так и декодирование осуществляются по оговариваемым протоколом распределениям вероятностей. Однако использование такого подхода при сжатии *всех* сообщений некоторого класса не обеспечивает высокой эффективности компрессии за счет несоответствия статистических характеристик подавляющего большинства сообщений усредненным характеристикам по соответствующему классу. Например, в среднем, вероятность встречаемости буквы «э» в текстах на русском языке равна 0,003, однако данная вероятность для научно-технических и научно-популярных текстов по электронике, как минимум, на порядок выше. Поэтому для обеспечения коэффициентов энтропийного сжатия, близких к максимальным, необходимо, чтобы используемые в его процессе вероятностные характеристики сообщения максимально соответствовали реальным.

При *полуадаптивном* статистическом кодировании вероятностные характеристики каждого конкретного сообщения определяются перед процедурой кодирования, в простейшем случае – путем предварительного прочтения сообщения перед кодированием, с подсчетом частот появления символов или их сочетаний. Таблица вероятностей, использованных при сжатии, или полученная кодовая таблица при этом включается в архивированное сообщение, а декодирование осуществляется на их основе: декодер восстанавливает кодовую таблицу по таблице вероятностей или непосредственно использует кодовую таблицу, включенную в архив. Однако на практике такой подход существенно увеличивает объем сжатого сообщения и, следовательно, снижает эффективность сжатия. Поэтому он также мало распространен на практике.

При *адаптивном* статистическом кодировании перед процедурой сжатия символам или их сочетаниям присваиваются некоторые начальные значения вероятностей, оговариваемые протоколом кодирования. На их основе строится исходная вероятностная модель сообщения. Затем, по мере считывания и кодирования его символов, вероятностная модель модифицируется по алгоритмам, позволяющим од-

нозначно восстановить действия кодера при декодировании. При декомпрессии декодер восстанавливает процедуру кодирования по его результатам, вследствие чего восстанавливается и исходное сообщение. Данный подход обеспечивает максимальное соответствие вероятностной модели сообщения, используемой при его кодировании, реальной вероятностной модели данного сообщения. Поэтому он наиболее распространен на практике.

Основными известными статистическими способами эффективного кодирования являются *префиксное неравномерное кодирование* и *арифметическое кодирование*.

1.2.3. Префиксное неравномерное кодирование

Префиксное неравномерное кодирование характеризуется следующими особенностями:

- неодинаковой разрядностью кодов, присваиваемых различным символам или группам символов, которая, в общем, тем меньше, чем больше вероятность появления соответствующего символа (группы символов) в сообщении;

- наличием *свойства префикса* у кодов, присваиваемых символам или их группам; данное свойство состоит в том, что ни один из кодов с некоторой разрядностью N не совпадает со старшими N битами любого из кодов с разрядностью, большей или равной N при любом значении N.

Первое из перечисленных свойств позволяет минимизировать объем сжатого сообщения (теоретически – до предела, определяемого его априорной ИЭ) за счет представления чаще встречаемых символов или групп символов кодами меньшей разрядности, чем реже встречаемых. Свойство префикса, в свою очередь, позволяет исключить пробелы между кодами символов (групп символов) сжатого сообщения при сохранении возможности однозначного определения границ между указанными кодами.

Наиболее известными алгоритмами реализации префиксного неравномерного кодирования являются алгоритмы *Шеннона – Фано* и *Хаффмана* [2, 3]. Первый из них не нашел практического применения. Второй достаточно широко используется (в основном – совместно с другими алгоритмами) в современных программных средствах архивирования как дискретных сообщений, так и аудиоданных и графических файлов (см. далее табл. 1.10, 1.11, 1.13 и 1.14).

Простейший *неадаптивный* алгоритм кодирования символов по Хаффману без учета контекста – следующий (кодирование осуществляется начиная с *младшего* бита) [2, 3]:

1. Символы ранжируются в порядке убывания вероятности их появления.

2. Очередному биту символа, оказавшегося последним по результатам ранжирования, присваивается нулевое значение, а очередному биту символа, оказавшегося предпоследним, – единичное. Если вероятности появления этих двух символов одинаковы – порядок присвоения нулевого и единичного значений их очередным битам определяется конкретным протоколом кодирования.

3. Символы, являющиеся последним и предпоследним по результатам ранжирования, объединяются в один, вероятность появления которого принимается равной сумме вероятностей появления объединяемых символов. При этом присваиваемые на последующих этапах кодирования значения очередных битов кода будут *одинаковы* для обоих объединенных символов.

4. Если после объединения очередной пары символов их общее количество остается большим единицы – переход к пункту 1. Если нет – кодирование завершено.

Процесс кодирования по Хаффману в соответствии с вышеприведенным алгоритмом удобно описывать так называемым *кодовым деревом Хаффмана***.** Пример кодирования по Хаффману с помощью такого дерева представлен на рис. 1.2. Кодируемое сообщение – *MEET ME*, кодирование осуществлялось в предположении, что источник сообщения является не Марковским (точнее, тем, что он является Марковским, пренебрегалось).

По результатам кодирования можно сделать следующие выводы:

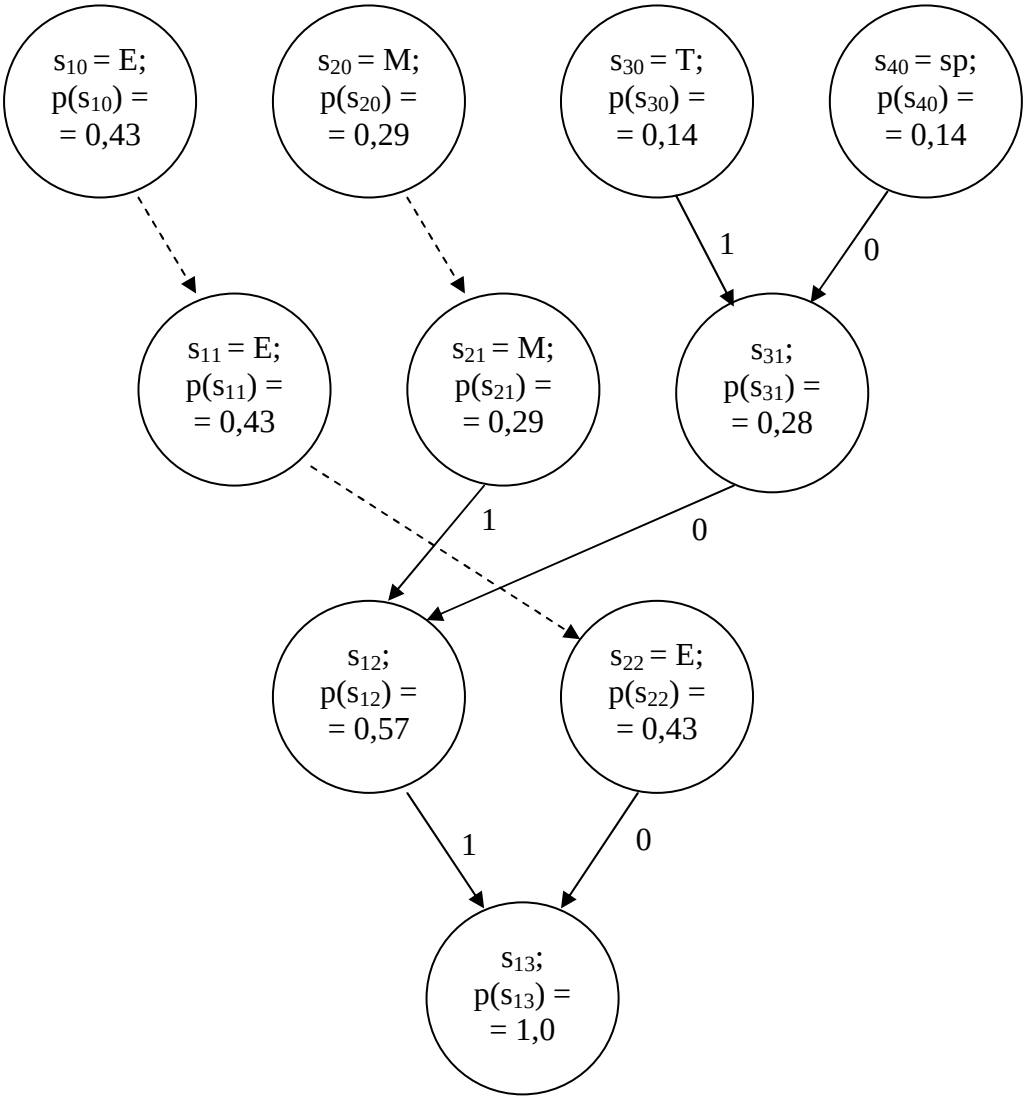
- наиболее часто встречавшийся в сообщении символ – буква «*E*» – закодирован двоичным числом с минимальной разрядностью (1 бит), наименее часто встречавшиеся – буква «*T*» и пробел – двоичными числами с максимальной разрядностью (3 бита);

- полученная в результате кодирования средняя разрядность символа, определяемая по выражению

$$\overline{B} = \sum_{i=1}^{n} B_i \times p(x_i) \qquad (1.4)$$

(где B_i – разрядность кода i-го символа), равна 1,86 бит/символ, что всего на 1 % больше вычисленной по выражению (1.2) априорной ИЭ сообщения *MEET ME*, равной 1,84 бит / символ;

- коды, присвоенные символам, действительно обладают свойством префикса (см. вышеприведенное определение данного свойства).



s_{ij} – i-й по рангу символ на j-м этапе кодирования;

sp – пробел.

Результирующие коды символов:

E – 0;

M – 11;

T – 100;

«Пробел» – 101

Рис. 1.2. Пример кодирования по Хаффману последовательности символов *MEET ME*

На практике применяется и кодирование по Хаффману на основании контекстных Марковских моделей сообщения. Такое кодиро-

вание предполагает независимое формирование кодов символов каждого из контекстов на основании вероятностей их появления в соответствующем контексте. Например, буква «*и*» в контексте «*пр*» (следующая за сочетанием букв «*пр*») может быть представлена двоичным кодом, отличным от ее же кода в контексте «*ан*». Кодирование по Хаффману с использованием контекстных моделей сложнее в реализации, чем использующее не Марковскую модель сообщения, однако позволяет добиться большей (часто существенно большей) степени сжатия за счет меньшей априорной ИЭ Марковского источника по сравнению с не Марковским [см. примечания к выражениям (1.2) и (1.3)].

Декодирование сообщений, сжатых по Хаффману, определяется характером алгоритма кодирования (неадаптивный, полуадаптивный или адаптивный). Общие принципы декодирования при каждой из данных разновидностей кодирования описаны ранее.

Одним из простейших алгоритмов *адаптивного префиксного неравномерного кодирования* (без учета контекста) является следующий [2]:

1. Символам алфавита сообщения, подлежащего сжатию, присваиваются некоторые начальные вероятности их появления, оговариваемые протоколом кодирования.

2. Символы ранжируются в порядке убывания их вероятностей, и им присваиваются префиксные коды, состоящие из префикса фиксированной разрядности, присваиваемого некоторой группе символов, и «тела» с разрядностью, которая тем больше, чем меньше вероятности символов соответствующей группы. Примеры таких кодов представлены в табл. 1.1.

3. По поступлении очередного символа он представляется кодом, присвоенным ему *до данного события*, что необходимо для корректного восстановления процедуры кодирования и, следовательно, исходного сообщения при декодировании. Затем распределение символов по кодам (РСК) модифицируется по таким правилам:

- чем больше число появлений некоторого символа в считанном фрагменте сообщения, тем более «близкий» к вершине кодовой таблицы (и, соответственно, более короткий) код ему присваивается;

- если число появлений некоторых двух и более символов в считанном фрагменте сообщения одинаково, РСК таких символов

формируется в соответствии с их *лексикографическим порядком* (например, в порядке их следования в алфавите или возрастания их *ASCII*-кодов).

Вышеописанный алгоритм поясняется в табл. 1.1.

Таблица 1.1

Пояснение простейшего алгоритма адаптивного префиксного кодирования

Коды		Начальное РСК	1-й символ – x_5		2-й символ – x_{12}		3-й символ – x_5	
Префикс	«Тело»		Выход кодера	РСК	Выход кодера	РСК	Выход кодера	РСК
00	0	x_1	1000	x_5	11011	x_5	000	x_5
00	1	x_2		x_1		x_{12}		x_{12}
01	0	x_3		x_2		x_1		x_1
01	1	x_4		x_3		x_2		x_2
10	00	x_5		x_4		x_3		x_3
10	01	x_6		x_6		x_4		x_4
10	10	x_7		x_7		x_6		x_6
10	11	x_8		x_8		x_7		x_7
11	000	x_9		x_9		x_8		x_8
11	001	x_{10}		x_{10}		x_9		x_9
11	010	x_{11}		x_{11}		x_{10}		x_{10}
11	011	x_{12}		x_{12}		x_{11}		x_{11}
11	100	x_{13}		x_{13}		x_{13}		x_{13}
11	101	x_{14}		x_{14}		x_{14}		x_{14}
11	110	x_{15}		x_{15}		x_{15}		x_{15}
11	111	x_{16}		x_{16}		x_{16}		x_{16}

Декодирование сообщения, сжатого в соответствии с описанным алгоритмом, происходит на основании начального РСК, используемого кодером, которое оговаривается протоколом сжатия и, соответственно, «известно» декодеру. По мере поступления кодов символов сжатого сообщения, за счет того, что каждый из этих кодов формируется *до модификации РСК* (см. п. 3 вышеприведенного алгоритма кодирования), декодер однозначно восстанавливает процесс кодирования и, следовательно, исходное сообщение.

Более полный обзор алгоритмов адаптивного префиксного кодирования представлен, например, в [2].

Префиксное неравномерное кодирование на основе *контекстных Марковских моделей*, в целом, осуществляется по общим принципам статистического эффективного кодирования с контекстным моделированием (см. п. 1.2.5).

1.2.4. Арифметическое эффективное кодирование

Арифметическое эффективное кодирование состоит в представлении последовательности символов или всего сообщения некоторым числом x, в «классических» алгоритмах кодирования – дробным, большим или равным нулю, но меньшим единицы (на практике, однако, применяется и целочисленное арифметическое кодирование). Значение x генерируется в процессе кодирования в зависимости от порядка следования символов в представляемой им последовательности, а также от вероятности появления каждого из данных символов в сообщении.

Простейший *алгоритм* неадаптивного арифметического кодирования последовательности символов без учета контекста следующий [2, 3]:

1. Символы алфавита сообщения ранжируются в порядке убывания вероятности их появления. При одинаковой вероятности появления двух символов они ранжируются в соответствии с их *лексикографическим порядком* (см. приведенное в п. 1.2.3 описание простейшего алгоритма адаптивного префиксного кодирования).

2. Интервал, выделенный для представления числа x, разбивается на N подинтервалов (где N – число символов в алфавите сообщения), границы которых удовлетворяют следующим соотношениям:

$$\left.\begin{aligned} &x_{01} = x_0, \\ &x_{1N} = x_1, \\ &x_{0i} = x_{1i-1}, \\ &\frac{x_{1i} - x_{0i}}{x_1 - x_0} = p_i, \end{aligned}\right\} \qquad (1.5)$$

где x_0 и x_1 – соответственно начальная и конечная точки интервала;

x_{0i} и x_{1i} – соответственно начальная и конечная точки подинтервала, выделенного под символ алфавита с i-м рангом;

p_i – вероятность появления в сообщении символа алфавита с i-м рангом.

3. Считывается очередной (на 1-м шаге кодирования – 1-й и т. д.) символ кодируемой последовательности. Значения границ интервала, выделенного для представления числа x, модифицируются в соответствии с выражениями:

$$\left.\begin{aligned} x_0 &= x_{0i}, \\ x_1 &= x_{1i}, \end{aligned}\right\} \qquad (1.6)$$

где i – ранг считанного символа;

т. е. интервал, выделенный для представления числа x, сужается до пределов, задаваемых числами x_{0i} и x_{1i}. Если считанный символ не является последним в кодируемой последовательности, осуществляется переход к п. 2; в противном случае – переход к п. 4.

4. В качестве результата кодирования x выбирается число, равное нижней границе числового интервала, полученного в результате последовательного выполнения пунктов 2 и 3.

Проиллюстрируем вышеописанный алгоритм конкретным примером. Пусть алфавит источника сообщений включает в себя три символа: «+», «-» и «0» с вероятностями появления в сообщении, равными 0,3; 0,2 и 0,5 соответственно. Необходимо сжать методом арифметического кодирования последовательность **+00-**.

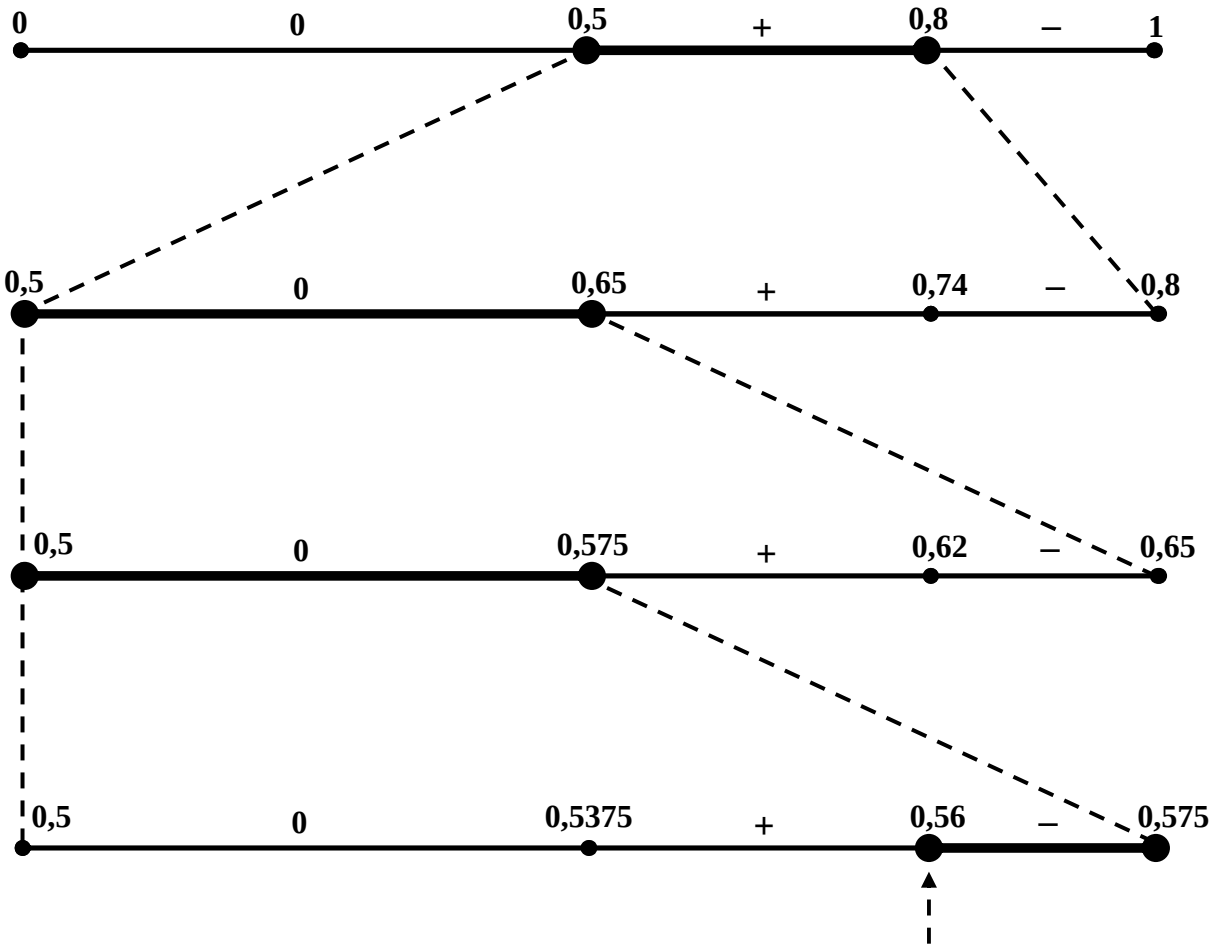
В соответствии с п. 1 вышеприведенного алгоритма символу «0» присваиваем ранг 1, символу «+» – 2, а символу «-» – 3. Процесс кодирования осуществляем последовательным выполнением пунктов 2 и 3 алгоритма (рис. 1.3). Результатом кодирования является число, равное 0,56.

Сообщения, сжатые в соответствии с вышеприведенным алгоритмом арифметического кодирования, *декодируются* по следующему алгоритму:

1. Символы алфавита сообщения ранжируются в соответствии с п. 1. алгоритма кодирования (см. выше). При этом алфавит сообщения и вероятности появления в нем каждого из символов должны быть «известны» декодеру. В простейшем случае они оговариваются стандартом (протоколом) кодирования / декодирования.

2. Задаются начальные значения нижней и верхней границ (x_0 и x_1) числового интервала, выделяемого для представления сообщений. При этом интервалы, используемые кодером и декодером, должны совпадать между собой.

3. Числовой интервал, выделяемый для представления сообщений, разбивается на подинтервалы в соответствии с выражениями (1.5).



Кодируемая последовательность символов:

+00-

$p(0) = 0{,}5;\ p(+) = 0{,}3;\ p(-) = 0{,}2.$

Линией большей толщины обозначены интервалы на числовой оси, выделяемые для представления кодируемой последовательности после ввода ее очередного символа

Рис. 1.3. Пояснение процесса арифметического кодирования по «классическому» алгоритму

4. В качестве очередного (на 1-м шаге декодирования – 1-го и т. д.) символа декодированного сообщения выбирается символ, удовлетворяющий условию

$$x_{0i} \le x < x_{1i}, \qquad (1.7)$$

где x – число, представляющее декодируемое сообщение.

5. Если $x = x_{0i}$, то декодированный символ – последний в сообщении, и процесс декодирования завершается. В противном случае границы x_0 и x_1 числового интервала, соответствующего декодируемому сообщению, модифицируются в соответствии с выражениями (1.6), и осуществляется переход к п. 3.

На рис. 1.4 показан процесс декодирования в соответствии с вышеописанным алгоритмом на примере сообщения (числа 0,56), полученного в результате выполненной ранее процедуры кодирования (см. рис. 1.3). В результате декодирования получается исходное сообщение: **+00-.**

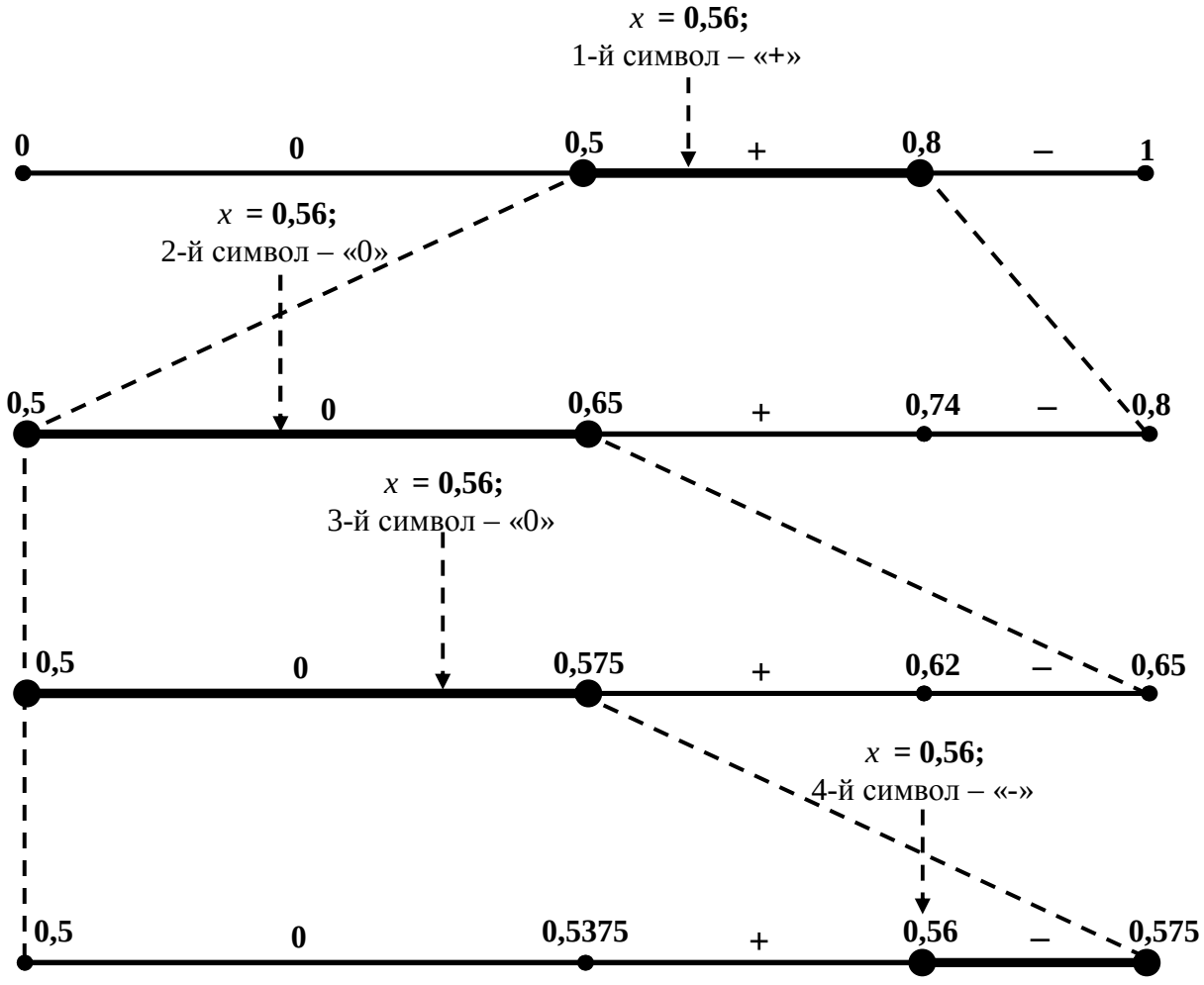


Рис. 1.4. Пояснение процесса декодирования сообщения (числа 0,56), полученного в результате процедуры кодирования, представленной на рис. 1.3

Простейшим алгоритмом *адаптивного* арифметического кодирования [2] (без учета контекста) является аналогичный описываемому выражениями (1.5) и (1.6) и рис. 1.3, за исключением того, что на

каждом j-м шаге кодирования границы подинтервалов, выделяемых для представления каждого из символов, задаются по следующим выражениям:

$$\begin{cases} x_{0ij} = x_{0j} + \left[x_{1j} - x_{0j}\right] \times \dfrac{1}{j+N} \times LCF_j[s_i], \\ x_{1ij} = x_{0j} + \left[x_{1j} - x_{0j}\right] \times \dfrac{1}{j+N} \times HCF_j[s_i], \end{cases} \tag{1.8}$$

где x_{0ij} и x_{1ij} – границы подинтервала, выделяемого для представления символа s_i после считывания j-го символа кодируемого сообщения;

x_{0j} и x_{1j} – значения, присваиваемые x_0 и x_1 (см. выражения (1.5) и комментарии к ним) после считывания j-го символа кодируемого сообщения и равные границам подинтервала, выделенного для его представления после считывания предыдущего, ($j-1$)-го символа;

N – общее число символов в алфавите сообщения;

$LCF_j[s_i]$, $HCF_j[s_i]$ – соответственно нижняя и верхняя *кумулятивные частоты* появления символа s_i в сообщении после считывания его первых j символов, определяемые по выражениям:

$$\begin{cases} LCF_j[s_i] = HCF_j[s_{i-1}], \\ HCF_j[s_i] = LCF_j[s_i] + f_{ij}, \\ LCF_j[s_1] = 0, \end{cases} \tag{1.9}$$

где f_{ij} – число символов s_i в считанном фрагменте сообщения из j символов.

При этом перед началом кодирования (на его 0-м шаге) все значения f_{i0} принимаются равными единице, т. е. все символы полагают равновероятными, вследствие чего отсчет числа считанных символов в процессе кодирования начинается не с нуля, а с количества символов в алфавите сообщения, N (см. выражение (1.8) и пояснения к нему).

Пример адаптивного арифметического кодирования в соответствии с вышеописанным алгоритмом представлен в табл. 1.2 (алфавит кодируемого сообщения при этом состоит из четырех символов).

Таблица 1.2

Пояснение простейшего алгоритма адаптивного арифметического кодирования

j	x_{0j}	x_{1j}	Текущая вероятностная модель сообщения				
			s_i	$LCF_j[s_i]$	$HCF_j[s_i]$	x_{0ij}	x_{1ij}
Начальное состояние кодера							
0	0	1	s_1	0	1	0	0,25
			s_2	1	2	0,25	0,5
			s_3	2	3	0,5	0,75
			s_4	3	4	0,75	1
Состояние кодера после поступления 1-го входного символа (s_3)							
1	0,5	0,75	s_1	0	1	0,5	0,55
			s_2	1	2	0,55	0,6
			s_3	2	4	0,6	0,7
			s_4	4	5	0,7	0,75
Состояние кодера после поступления 2-го входного символа (s_2)							
2	0,55	0,6	s_1	0	1	0,55	0,558
			s_2	1	3	0,558	0,575
			s_3	3	5	0,575	0,592
			s_4	5	6	0,592	0,6
Состояние кодера после поступления 3-го входного символа (s_2)							
3	0,558	0,575	s_1	0	1	0,558	0,560
			s_2	1	4	0,560	0,568
			s_3	4	6	0,568	0,573
			s_4	6	7	0,573	0,575
Состояние кодера после поступления 4-го (последнего) символа (s_4)							
4	**0,573**	0,575	Результат кодирования - **0,573.**				

Декодирование сообщений, сжатых по вышеописанному алгоритму, осуществляется аналогично декодированию при неадаптивном арифметическом сжатии (см. рис. 1.4), за исключением использования выражений (1.8) и (1.9) при определении границ подинтервалов, выделяемых для представления каждого из символов. Процесс декодирования результата сжатия, полученного в предыдущем примере (см. табл. 1.2), поясняет табл. 1.3.

Более полный обзор алгоритмов адаптивного арифметического кодирования представлен, например, в [2].

Следует также отметить, что на практике арифметическое кодирование, как и префиксное неравномерное, часто реализуется на основе контекстных Марковских моделей сообщения (см. п. 1.2.5).

Таблица 1.3

Пояснение процесса декодирования сообщения, полученного в табл. 1.2

Вход декодера	Шаг декодирования	x_0	x_1	Текущая вероятностная модель сообщения					Выход декодера
				s_i	$LCF_j[s_i]$	$HCF_j[s_i]$	x_{0ij}	x_{1ij}	
0,573	1	0	1	s_1	0	1	0	0,25	s_3
				s_2	1	2	0,25	0,5	
				s_3	2	3	**0,5**	**0,75**	
				s_4	3	4	0,75	1	
	2	0,5	0,75	s_1	0	1	0,5	0,55	s_2
				s_2	1	2	**0,55**	**0,6**	
				s_3	2	4	0,6	0,7	
				s_4	4	5	0,7	0,75	
	3	0,55	0,6	s_1	0	1	0,55	0,558	s_2
				s_2	1	3	**0,558**	**0,575**	
				s_3	3	5	0,575	0,592	
				s_4	5	6	0,592	0,6	
	4	0,558	0,575	s_1	0	1	0,558	0,560	s_4 (посл. символ)
				s_2	1	4	0,560	0,568	
				s_3	4	6	0,568	0,573	
				s_4	6	7	**0,573**	**0,575**	

Примечание. Полужирным шрифтом выделены границы подинтервалов, в которые попадает число, представляющее декодируемое сообщение.

В целом, арифметическое кодирование позволяет достичь степени сжатия, более близкой к задаваемому априорной ИЭ теоретическому пределу, чем при префиксном неравномерном кодировании [2]. До недавнего времени основным фактором, сдерживавшим широкое применение арифметического кодирования, была сложность его реализации, обусловленная:

- необходимостью оперирования с дробными двоичными числами с большой разрядностью дробной части, которая в наихудшем случае равна:

$$B_F = \left\rfloor - L_{\max} \log_2 (p_{\min}) \right\lfloor + 1, \qquad (1.10)$$

где $L_{\max}$ – максимальная длина (в символах) подвергаемого кодированию сообщения;

$p_{\min}$ – вероятность появления в сообщении наиболее редко встречаемого символа алфавита;

$\lfloor \bullet \rfloor$ – оператор округления до ближайшего большего целого;

- необходимостью многократного выполнения операций умножения и деления дробных чисел, требующих значительных затрат времени и вычислительных ресурсов.

Однако в настоящее время вышеперечисленные ограничения не являются существенными, благодаря:

- существенному и постоянному повышению вычислительной мощности и производительности средств компьютерной техники;

- применению модифицированных алгоритмов кодирования / декодирования [2] с целочисленным представлением границ интервалов, выделенных для представления результата сжатия, в простейшем случае – с выделением под него диапазона чисел от 0 до $2^{B_F} - 1$ [см. выражение (1.10)], а также с заменой операций умножения и деления на операции сложения/вычитания и сдвига.

В целом, арифметическое компактное кодирование в настоящее время находит все более широкое применение в архиваторах различного назначения.

1.2.5. Общие принципы статистического эффективного кодирования на основе контекстного моделирования

Эффективность сжатия статистическими методами существенно повышается при использовании Марковских контекстных моделей сообщения при сжатии за счет меньшей априорной ИЭ Марковского источника по сравнению с не Марковским [см. примечания к выражениям (1.2) и (1.3)].

Большинство распространенных на практике алгоритмов статистического кодирования являются адаптивными и базируются на оце-

нивании текущей вероятностной модели кодируемого сообщения методом *контекстного моделирования с предсказанием*. Собственно кодирование при этом может осуществляться по алгоритмам как префиксного неравномерного, так и арифметического адаптивного кодирования (которое в настоящее время распространено несколько шире [2]), в целом, аналогичным описанным ранее (см. пп. 1.2.3 и 1.2.4). Кодирование осуществляется *раздельно* для каждого из контекстов.

Типовым примером алгоритмов, использующих контекстное моделирование с предсказанием, являются алгоритмы группы *PPM* (*Prediction by Partial Matching* – предсказание по частичному совпадению) [2, 3]. Данные алгоритмы, отличаясь деталями реализации, в целом основаны на одинаковой по сущности процедуре оценивания вероятностных характеристик сообщения в процессе его сжатия. Эта процедура базируется на использовании набора *контекстных моделей* различных порядков в процессе оценивания вероятностных характеристик сообщения. Под *контекстом* некоторого символа понимается последовательность предшествующих ему символов. Длина данной последовательности называется *порядком контекста*. *Контекстной моделью* некоторого порядка называют распределение вероятностей символов сообщения по встреченным в сообщении контекстам того же порядка.

Процедура оценивания может быть описана следующим обобщенным алгоритмом [3]:

1. Определяется максимальный порядок контекстной модели, используемый в процессе оценивания. Например, алгоритм *PPMd*, реализуемый архиватором *7-Zip*, представляет пользователю возможность выбрать его значение из диапазона от 2 до 16. В ходе кодирования составляются и используются контекстные модели порядков от максимального до 0-го. При этом под контекстной моделью 0-го порядка понимается распределение вероятностей появления символов в сообщении без учета их контекстов (предшествовавших им символов).

2. В алфавит сообщения добавляются по одному *Escape*-символу для каждого из контекстов. Назначение данного символа будет пояснено далее.

3. В ходе сжатия встречаемым в сообщении символам присваиваются вероятности в рамках контекстной модели каждого из поряд-

ков в соответствии с выражениями, зависящими от конкретного алгоритма кодирования. Например, алгоритм *PPMc* использует следующие выражения [4]:

$$p_s(c) = \frac{f_s(c)}{f_c + d(c)}; \tag{1.11}$$

$$p_{esc}(c) = \frac{d(c)}{f_c + d(c)}, \tag{1.12}$$

где $p_s(c)$ и $p_{esc}(c)$ – текущие (полученные к некоторому моменту процедуры кодирования) оценки вероятностей появления соответственно некоторого символа s и *Escape*-символа в некотором контексте c (например, $p_i(th)$ – оценка вероятности появления буквы «i» после сочетания символов «th»);

f_c – зарегистрированное к соответствующему моменту число появлений в сообщении контекста c (например, сочетаний символов «th»);

$d(c)$ – число различных символов, появление которых в контексте c зарегистрировано к соответствующему моменту процедуры кодирования (например, если после сочетания символов «th» до некоторого момента встречались только символы «a», «e» и «i», значение $d(th)$ равно 3).

4. Считывается очередной символ. Если это символ окончания сообщения, выполняется выход из процедуры кодирования. В противном случае порядок используемой для кодирования контекстной модели устанавливается равным максимальному, и осуществляется переход к п. 5.

5. Проверяется, встречался ли уже считанный символ в контексте заданного порядка, замыкающем ранее считанную последовательность. Если встречался – символ поступает на вход энтропийного кодера (арифметического или префиксного неравномерного) символов соответствующего контекста с текущим присвоенным ему значением вероятности в данном контексте. Все вероятности $p_s(c)$ и $p_{esc}(c)$ всех контекстных моделей модифицируются с учетом считанного символа, в соответствии с выражениями, используемыми конкретным алгоритмом для оценивания данных вероятностей, например, с выражениями (1.11) и (1.12). Затем осуществляется переход к п. 4.

В противном случае на вход кодера символов соответствующего контекста поступает *Escape*-символ данного контекста с текущим присвоенным ему значением вероятности. После этого все вероятности $p_s(c)$ и $p_{esc}(c)$ каждого из контекстов анализируемого порядка модифицируются с учетом считанного символа, в соответствии с выражениями для их оценивания, используемыми конкретным алгоритмом. Переход к п. 6.

6. Если порядок контекстной модели равен нулю – переход к п. 7. В противном случае порядок контекстной модели уменьшается на единицу, и осуществляется переход к п. 5.

7. Если считанный символ уже встречался в сообщении – он поступает на вход энтропийного кодера символов контекста 0-го порядка с текущим присвоенным ему значением вероятности в контекстной модели 0-го порядка. Все вероятности $p_s(c)$ и $p_{esc}(c)$ всех контекстных моделей модифицируются с учетом считанного символа, в соответствии с выражениями, используемыми конкретным алгоритмом, например (1.11) и (1.12), и осуществляется переход к п. 4.

В противном случае на вход кодера поступает *Escape*-символ контекстной модели нулевого порядка, а на вход кодера символов контекста 0-го порядка поступает считанный символ, которому присваивается вероятность $1/N$, где N – общее число символов в алфавите сообщения, включая *Escape*. Все вероятности $p_s(c)$ и $p_{esc}(c)$ всех контекстных моделей модифицируются с учетом считанного символа, в соответствии с выражениями для их оценивания, используемыми конкретным алгоритмом, и осуществляется переход к п. 4.

Поясним вышеописанный обобщенный алгоритм небольшим примером [2]. Пусть уже считана и обработана последовательность символов *assanissimassa*. Для упрощения положим, что максимальный порядок контекстной модели, используемой при кодировании, равен 2. Контекстные модели, сформированные по считывании вышеприведенной последовательности символов, представлены в табл. 1.4. При их формировании использованы выражения (1.11) и (1.12).

Возможны четыре варианта сочетания сформированных контекстных моделей и очередного считанного символа.

Вариант № 1. Очередной считанный символ уже встречался ранее в контексте 2-го (максимального из используемых в рассматриваемом примере) порядка, замыкающем ранее считанную последовательность. Данным контекстом является «*sa*», а подобным символом в рассматриваемом примере может быть только символ «*n*». В таком

случае он поступает на вход энтропийного кодера символов контекста «sa» с ранее присвоенной ему вероятностью в контексте «sa», $p_n(sa)$, равной $1/2$ (см. табл. 1.4).

Затем вероятности контекстных моделей всех порядков модифицируются с учетом очередного поступившего символа «n» в соответствии с выражениями (1.11) и (1.12). Например, значение $p_n(sa)$ устанавливается равным $2/3$, а $p_{esc}(sa)$ – $1/3$ и т. д. После этого осуществляется считывание следующего символа сообщения.

Вариант № 2. Очередной считанный символ не встречался ранее в контексте 2-го порядка, замыкающем уже считанную последовательность (в рассматриваемом примере – «sa»), но встречался в замыкающем ее контексте 1-го порядка. В рассматриваемом примере таковым является символ «a», а рассматриваемый вариант будет иметь место, если следующим за последовательностью *assanissimassa* символом является буква «s». В таком случае:

- вначале на вход энтропийного кодера символов контекста «sa» поступает *Escape*-символ данного контекста с ранее присвоенной ему вероятностью $p_{esc}(sa)$, равной $1/2$ (см. табл. 1.4), служащий при декодировании признаком переключения на контекстную модель 1-го порядка;

- затем на вход энтропийного кодера символов контекста «a» поступает символ «s» данного контекста с ранее присвоенной ему вероятностью $p_s(a)$, равной $2/5$ (см. табл. 1.4).

После этого вероятности контекстных моделей всех порядков модифицируются с учетом очередного поступившего символа «s», в соответствии с выражениями (1.11) и (1.12).

Вариант № 3. Очередной считанный символ ранее не встречался ни в контексте 2-го порядка, ни в контексте 1-го порядка, замыкающих ранее считанную последовательность (в рассматриваемом примере – ни после сочетания букв «sa», ни после буквы «a»), однако уже встречался в других контекстах. В рассматриваемом примере данный вариант будет иметь место, если следующим за последовательностью *assanissimassa* символом является, например, буква «m». В таком случае:

- вначале на вход энтропийного кодера символов контекста «sa» поступает *Escape*-символ данного контекста с ранее присвоенной ему вероятностью $p_{esc}(sa)$, равной $1/2$ (см. табл. 1.4), служащий при декодировании признаком переключения на контекстную модель 1-го порядка;

Таблица 1.4

Пояснения к процессу формирования контекстных моделей по алгоритмам группы PPM

Контекстная модель 2-го порядка						Контекстная модель 1-го порядка						Контекстная модель 0-го порядка					
c	f_c	$d(c)$	s	$f_s(c)$	$p_s(c)$	c	f_c	$d(c)$	s	$f_s(c)$	$p_s(c)$	c	f_c	$d(c)$	s	$f_s(c)$	$p_s(c)$
as	2	1	*s*	2	2 / 3	*a*	3*	2	*s*	2	2 / 5	-	14**	5	*a*	4	4 / 19
			esc	-	1 / 3				*n*	1	1 / 5				*s*	6	6 / 19
ss	3	2	*a*	2	2 / 5				*esc*	-	2 / 5				*n*	1	1 / 19
			i	1	1 / 5	*s*	6	3	*s*	3	3 / 9				*i*	2	2 / 19
			esc	-	2 / 5				*a*	2	2 / 9				*m*	1	1 / 19
sa	1*	1	*n*	1	1 / 2				*i*	1	1 / 9				*esc*	-	5 / 19
			esc	-	1 / 2				*esc*	-	3 / 9						
an	1	1	*i*	1	1 / 2	*n*	1	1	*i*	1	1 / 2						
			esc	-	1 / 2				*esc*	-	1 / 2						
ni	1	1	*s*	1	1 / 2	*i*	2	2	*s*	1	1 / 4						
			esc	-	1 / 2				*m*	1	1 / 4						
is	1	1	*s*	1	1 / 2				*esc*	-	2 / 4						
			esc	-	1 / 2	*m*	1	1	*a*	1	1 / 2						
si	1	1	*m*	1	1 / 2					-	1 / 2						
			esc	-	1 / 2												
im	1	1	*a*	1	1 / 2												
			esc	-	1 / 2												
ma	1	1	*s*	1	1 / 2												
			esc	-	1 / 2												

*В число f_c не включаются контексты, после которых пока не зарегистрирован какой-либо символ, например, контексты «*sa*» и «*a*», расположенные в конце последовательности *assanissimassa*.

**В контекстной модели 0-го порядка в качестве f_c служит общее число символов в уже обработанном фрагменте сообщения.

- затем на вход энтропийного кодера символов контекста «a» поступает *Escape*-символ данного контекста с ранее присвоенной ему вероятностью $p_{esc}(a)$, равной $2/5$ (см. табл. 1.4), служащий при декодировании признаком переключения на контекстную модель 0-го порядка;

- после этого на вход энтропийного кодера символов контекста 0-го порядка поступает символ «m» с ранее присвоенной ему в данном контексте вероятностью, равной $1/19$ (см. табл. 1.4).

Затем вероятности контекстных моделей всех порядков модифицируются с учетом очередного поступившего символа «m», в соответствии с выражениями (1.11) и (1.12).

Вариант № 4. Очередной считанный символ еще не встречался в сообщении, например, данным символом является буква «d». В таком случае:

- на входы энтропийных кодеров контекстов «sa», «a» и 0-го контекста последовательно поступают *Escape*-символы данных контекстов с ранее присвоенными им вероятностями, равными $1/2$, $2/5$ и $5/19$ соответственно (см. табл. 1.4); последовательность данных символов служит при декодировании признаком того, что далее следует ранее не встречавшийся в сообщении символ;

- затем на вход энтропийного кодера 0-го контекста поступает символ «d» с присвоенной ему вероятностью $1/N$, где N – общее число символов в алфавите сообщения, включая *Escape* (например, $N = 257$ при *ASCII* -кодировании).

После этого вероятности контекстных моделей всех порядков модифицируются с учетом очередного поступившего символа «d», в соответствии с выражениями (1.11) и (1.12).

При вышеописанной процедуре кодирования на этапе декодирования могут быть однозначно восстановлены все операции кодера.

Следует отметить, что существуют достаточно разнообразные варианты реализации вышеописанного обобщенного алгоритма группы *PPM*. В частности, в ряде данных вариантов после передачи очередного символа на вход энтропийного кодера обновляются контекстные модели не всех порядков, а только бóльших или равных тому, к которому принадлежит контекст символа, переданного на вход кодера. При этом, например, в ранее рассмотренном варианте № 2 были бы обновлены только контекстные модели 1-го и 2-го порядков.

Также конкретные алгоритмы могут различаться между собой выражениями для оценки вероятностей $p_s(c)$ и $p_{esc}(c)$ и рядом других особенностей реализации.

Способ *PPM* является не единственно возможным способом энтропийного сжатия с контекстным моделированием. В частности, перспективным считается способ *CTW* (*Context Tree Weighting* – взвешивание с применением контекстного дерева) [2, 4]. Данный способ также основан на контекстном моделировании с предсказанием. Однако, в отличие от *PPM*, он рассматривает кодируемое сообщение не как поток *символов*, а как поток *битов*, т. е. предполагается, что алфавит данного сообщения состоит всего из двух символов – «0» и «1». При этом входной битовый поток кодера может представлять собой как текст, так и фонограмму, изображение и пр., т. е. способ *CTW*, в отличие от *PPM*, пригоден для сжатия практически любых типов сообщений.

Кодирование битового потока по способу *CTW*, как правило, осуществляется арифметическим кодером. В качестве текущей длины числового интервала, представляющего уже считанный фрагмент кодируемого сообщения (см. представленное ранее описание процесса арифметического кодирования), служит оценка *взвешенной вероятности* уже считанной последовательности битов, откуда слово *Weighting* (взвешивание) в названии рассматриваемого способа. Взвешенной данная вероятность называется потому, что при ее оценивании учитывается «вклад» в нее контекстных моделей порядков от 0-го до некоторого максимального, выбираемого пользователем или определяемого конкретным протоколом сжатия. Оценивание указанной вероятности осуществляется на основе множества контекстных моделей, описываемого *контекстным деревом* (см. название рассматриваемого способа). Пример контекстного дерева при максимальном выбранном порядке контекста, равном 3, представлен на рис. 1.5, где:

- z и o – общее число нулевых и единичных битов в уже считанном фрагменте сообщения;

- $P_w[z, o]$ – оценка взвешенной вероятности уже считанной последовательности битов, поступающая на вход арифметического кодера;

- $z(c)$ и $o(c)$ – общее число соответственно нулевых и единичных значений бита, следующего за контекстом c (например, после битовой последовательности 000), встретившихся в уже считанном фрагменте кодируемого сообщения;

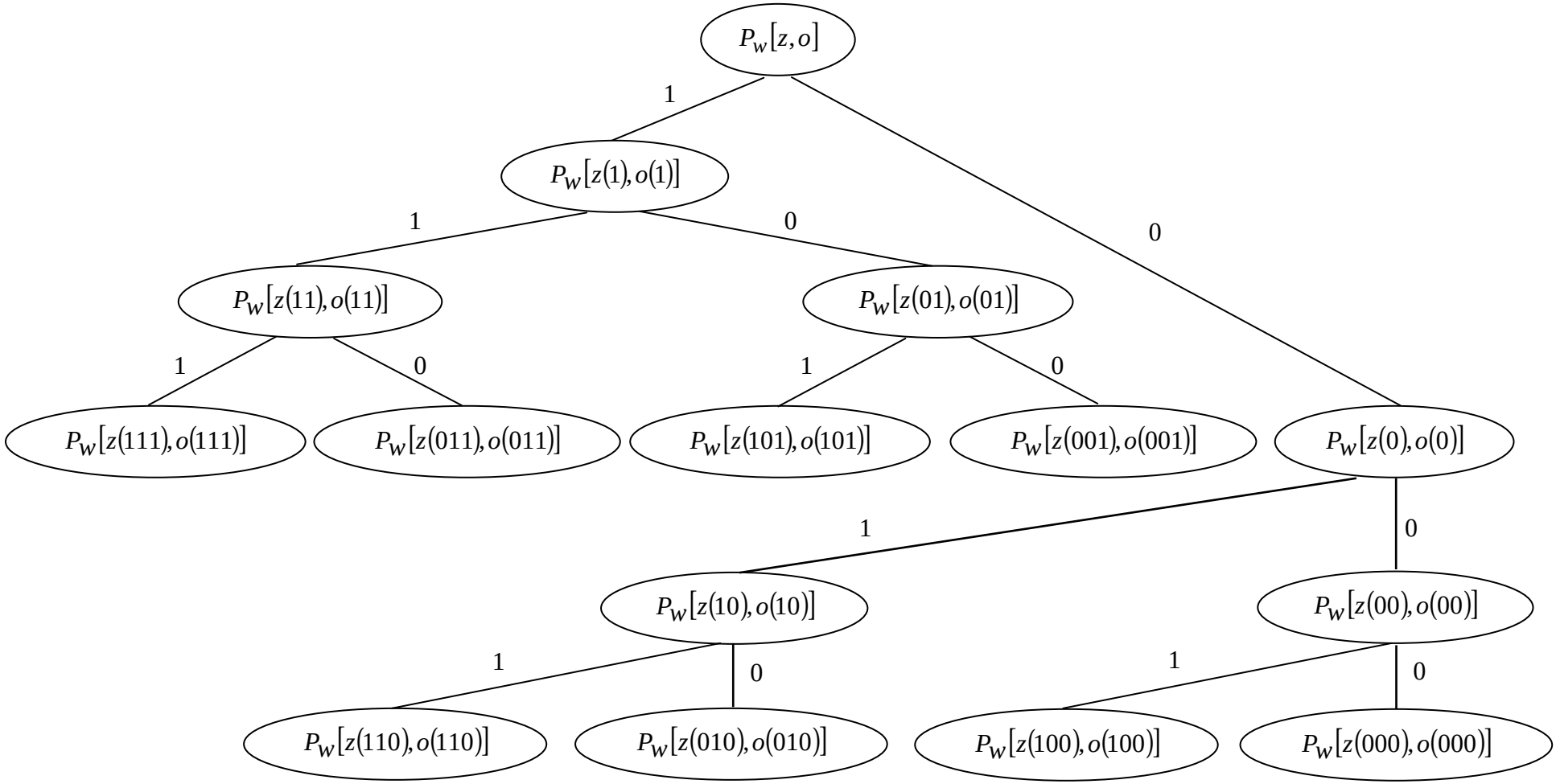


Рис. 1.5. Пример контекстного дерева, используемого способом *CTW*

- $P_w[z(c), o(c)]$ – оценка взвешенной вероятности того, что бит, следующий после контекста c, z раз окажется равным нулю и o раз – единице; данная оценка вычисляется на основании уже считанного фрагмента сообщения по следующему выражению:

$$P_w[z(c_j), o(c_j)] = \begin{cases} \dfrac{P_e[z(c_j), o(c_j)] + P_{w0}[z(c_{j+1}), o(c_{j+1})] \times P_{w1}[z(c_{j+1}), o(c_{j+1})]}{2} & \text{при } j < j_{\max}, \\ P_e[z(c_j), o(c_j)] \text{ при } j = j_{\max}, \end{cases} \quad (1.13)$$

где j – порядок контекста, для которого выполняется оценивание;

$j_{\max}$ – выбранное максимальное значение порядка контекстных моделей;

$P_e[z(c_j), o(c_j)]$ – оценка совместного появления в двоичной последовательности $z(c_j)$ нулевых и $o(c_j)$ единичных битов, как правило, вычисляемая *по формуле Кричевского – Трофимова*, которая в общем случае при оценивании вероятности совместного появления в двоичной последовательности z нулевых и o единичных битов имеет следующий вид:

$$P_e[z, o] = \frac{\left[\prod_{i=1}^{z}(i-0{,}5)\right] \times \left[\prod_{k=1}^{o}(k-0{,}5)\right]}{\left[\prod_{m=1}^{z+o} m\right]}, \quad (1.14)$$

причем значение $P_e[0, 0]$ считается равным единице;

$P_{w0}[z(c_{j+1}), o(c_{j+1})]$ и $P_{w1}[z(c_{j+1}), o(c_{j+1})]$ – оценки взвешенной вероятности того, что бит, следующий после контекстов $(j+1)$-го порядка $0c_j$ и $1c_j$, z раз окажется равным нулю и o раз – единице; применительно к контекстному дереву (см. рис. 1.5) данные оценки представляют собой веса листовых узлов, для которых узел, соответствующий контексту c_j, является корневым.

Например:

$P_w[z(11), o(11)] = \dfrac{P_e[z(11), o(11)] + P_w[z(011), o(011)] \times P_w[z(111), o(111)]}{2}$ (см. рис. 1.5).

Поясним процесс построения контекстного дерева небольшим примером. Пусть кодером считана битовая последователь-

ность 100100110; значение $j_{\max}$ выбрано равным 3 (см. рис. 1.5). Тогда:

$z(011) = 1$; $o(011) = 0$; $P_w[z(011), o(011)] = P_e[1, 0] = 0{,}5$ [см. выражения (1.13) и (1.14)];

$z(111) = o(111) = 0$; $P_w[z(111), o(111)] = P_e[0, 0] = 1$ [см. примечание к выражению (1.14)];

$z(001) = o(001) = 1$; $P_w[z(001), o(001)] = P_e[1, 1] = 0{,}125$;

$z(101) = o(101) = 0$; $P_w[z(101), o(101)] = P_e[0, 0] = 1$;

$z(11) = 1$; $o(11) = 0$; $P_e[z(11), o(11)] = P_e[1, 0] = 0{,}5$;

$$P_w[z(11), o(11)] = \frac{P_e[z(11), o(11)] + P_w[z(011), o(011)] \times P_w[z(111), o(111)]}{2} =$$

$$= \frac{0{,}5 + 0{,}5 \times 1}{2} = = 0{,}5;$$

$z(01) = o(01) = 1$; $P_e[z(01), o(01)] = P_e[1, 1] = 0{,}125$;

$$P_w[z(01), o(01)] = \frac{P_e[z(01), o(01)] + P_w[z(001), o(001)] \times P_w[z(101), o(101)]}{2} =$$

$$= \frac{0{,}125 + 0{,}125 \times 1}{2} = 0{,}125;$$

$z(1) = 3$; $o(1) = 1$; $P_e[z(1), o(1)] = P_e[3, 1] \approx 0{,}039$;

$$P_w[z(1), o(1)] = \frac{P_e[z(1), o(1)] + P_w[z(01), o(01)] \times P_w[z(11), o(11)]}{2} \approx$$

$$\approx \frac{0{,}039 + 0{,}125 \times 0{,}5}{2} \approx 0{,}05075$$ и т. п.

Собственно *кодирование* при сжатии на основе способа *CTW*, как указано ранее, осуществляется обычно арифметическим кодером. Один из возможных алгоритмов кодирования следующий [4]:

1. Первые $j_{\max}$ битов кодируемого сообщения непосредственно выдаются на выход декодера.

2. После выдачи указанных битов строится контекстное дерево, соответствующее выбранному значению $j_{\max}$ (см. рис. 1.5). По выражениям (1.13) и (1.14) определяются веса его узлов, соответствующие количеству и сочетаниям нулей и единиц в переданной на выход кодера битовой комбинации разрядностью $j_{\max}$.

3. Начальные значения границ x_0 и x_1 интервала, выделяемого для представления результата кодирования (см. выражения (1.5) и пояснения к ним) устанавливаются равными нулю и единице соответственно.

4. Считывается очередной бит кодируемого сообщения. Веса всех узлов контекстного дерева пересчитываются по выражениям (1.13) и (1.14), после чего значения границ x_0 и x_1 модифицируются по следующим правилам:

- если очередной бит равен нулю, x_0 не изменяется, а x_1 присваивается значение $x_0 + P_w[z,o]$;

- если очередной бит равен единице, x_1 не изменяется, а x_0 присваивается значение $x_1 - P_w[z,o]$.

5. Если считанный бит является последним в кодируемом сообщении, конечному результату арифметического кодирования присваивается значение x_0, и процесс кодирования завершается. В противном случае – переход к п. 4.

Проиллюстрируем данный алгоритм примером. Пусть на вход кодера поступила битовая комбинация 11001. Для простоты примера примем максимальную длину контекста $j_{\max}$ равной единице. Контекстное дерево при этом приобретает вид, представленный на рис. 1.6. Процесс кодирования показан в табл. 1.5.

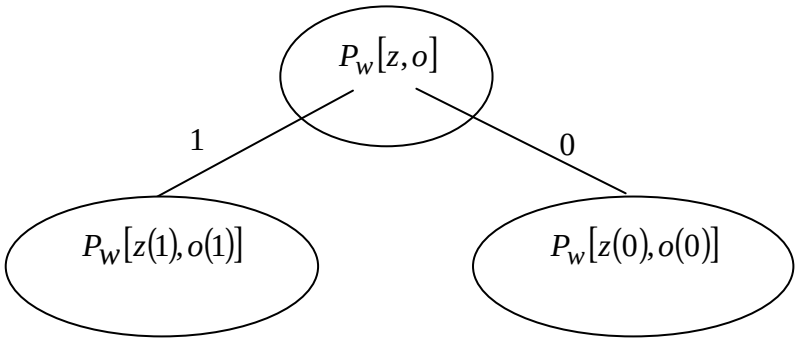


Рис. 1.6. Контекстное дерево, используемое в процессе кодирования (см. табл. 1.5)

Декодирование сообщений, сжатых по вышеописанному алгоритму, осуществляется следующим образом:

1. Считывается входящая в сжатое сообщение последовательность из первых $j_{\max}$ битов исходного сообщения (в табл. 1.5 в каче-

стве такой последовательности выступает бит «1»). На основании указанной последовательности строится контекстное дерево, и определяются веса его узлов по выражениям (1.13) и (1.14).

Таблица 1.5

Пояснение процесса кодирования битовой последовательности 11001 способом CTW на основе контекстного дерева (рис. 1.6)

Шаг кодирования	Входной бит	$z(1)$	$o(1)$	$z(0)$	$o(0)$	$P_w[z,o]$	x_0	x_1
0	1	0	0	0	0	0,75	0	1
1	1	0	1	0	0	0,4375	0,5625	1
2	0	1	1	0	0	0,09375	0,5625	0,65625
3	0	1	1	1	0	0,04297	0,5625	0,60547
4	1	1	1	1	1	0,01367	**0,5918**	0,60547
Результат арифметического кодирования: **0,5918.** Выход кодера: последовательность из бита «**1**» и числа **0,5918**.								

2. Начальные значения границ x_0 и x_1 устанавливаются равными нулю и единице соответственно.

3. Производятся две пробные модификации весов всех узлов контекстного дерева по выражениям (1.13) и (1.14): одна – в предположении, что очередной бит исходного сообщения имеет нулевое значение, другая – что значение данного бита является единичным. По результатам данных модификаций, в соответствии с п. 4 алгоритма кодирования, определяются значения, которые были бы присвоены границам x_0 и x_1 при нулевом и единичном значении очередного бита.

4. Если число, являющееся результатом кодирования исходного сообщения, совпадает со значением, которое было бы присвоено границе x_0 при равенстве очередного бита нулю, очередному биту результата декодирования присваивается нулевое значение, делается вывод, что этот бит – последний в сообщении, и декодирование завершается.

5. Если число, являющееся результатом кодирования исходного сообщения, совпадает со значением, которое было бы присвоено границе x_0 при равенстве очередного бита единице, очередному биту результата декодирования присваивается единичное значение, делается вывод, что этот бит – последний в сообщении, и декодирование завершается.

6. Если число, являющееся результатом кодирования исходного сообщения, попадает в интервал, ограниченный значениями, которые были бы присвоены границам x_0 и x_1 при равенстве очередного бита нулю, очередному биту результата декодирования присваивается нулевое значение, в противном случае – единичное. Весам узлов контекстного дерева и границам x_0 и x_1 присваиваются значения, соответствующие очередному биту результата декодирования, после чего осуществляется переход к п. 3.

Процесс декодирования по данному алгоритму представлен в табл. 1.6 на примере декодирования сообщения, полученного в предыдущем примере (см. табл. 1.5 и рис. 1.6).

Таблица 1.6

*Пояснение процесса декодирования сообщения, состоящего из бита «**1**» и числа **0,5918** (табл. 1.5)*

Шаг декодирования	Предполагаемое значение входного бита	$z(1)$	$o(1)$	$z(0)$	$o(0)$	$P_w[z,o]$	x_0	x_1	Выход декодера
0	1*	0	0	0	0	0,75	0	1	1
1	0	1	0	0	0	0,4375	0	0,3125	1
	1	0	1	0	0	0,3125	**0,5625**	**1**	
2	0	1	1	0	0	0,09375	**0,5625**	**0,65625**	0
	1	0	2	0	0	0,34375	0,65625	1	
3	0	1	1	1	0	0,04297	**0,5625**	**0,60547**	0
	1	1	1	0	1	0,05078	0,60547	0,65625	
4	0	1	1	2	0	0,02930	0,5625	0,5918	1
	1	1	1	1	1	0,01367	**0,5918**	**0,60547**	(посл. бит)

*Значение известно, так как непосредственно входит в общий результат сжатия, подаваемый на вход декодера.

Примечание. Полужирным шрифтом выделены границы подинтервалов, в которые попадает число, представляющее декодируемое сообщение.

Кроме вышеописанных, существуют и другие алгоритмы кодирования / декодирования способом *CTW* [3, 4].

Бóльшая часть специалистов считает способ *CTW* потенциально наиболее эффективным для большинства приложений [2, 3]. Однако

до настоящего времени еще не накоплен достаточный опыт практического использования и исследования данного способа. Поэтому окончательные выводы об его эффективности и рациональных областях его применения следует ожидать в ближайшем будущем.

Кроме ранее рассмотренных способов и алгоритмов адаптивного энтропийного сжатия, известны и другие, менее распространенные на практике [3, 4].

Коэффициент энтропийного сжатия при прочих равных условиях тем выше, чем:

- «однообразнее» состав символов и сочетаний символов сообщения, т. е. чем меньше его энтропия (что характерно для всех способов и алгоритмов сжатия);
- больше объем сообщения, так как коэффициент сжатия начала сообщения незначителен, поскольку при его считывании происходит только формирование контекстных моделей, которые используются при сжатии последующих фрагментов.

1.2.6. Словарные способы эффективного кодирования

Словарные способы эффективного кодирования [2, 3], в целом, состоят в формировании собственного *словаря* (фактически – кодовой таблицы) каждого из сжимаемых сообщений. Словарь включает в себя не только коды стандартных символов (букв, цифр и т. п.), но и коды, присваиваемые последовательностям таких символов, встречающимся в сжимаемом сообщении. Подобные последовательности при этом получают статус нестандартных символов сообщения. Разрядность кодов, присваиваемых как стандартным, так и нестандартным символам, является фиксированной. Например, алгоритм *LZW* использует 12-битовое представление символов. Словарь (кодовая таблица) сжимаемого сообщения формируется в процессе кодирования. Все алгоритмы словарного кодирования и декодирования строятся таким образом, что сформированный в процессе кодирования словарь однозначно восстанавливается при декодировании, без необходимости его включения в сжатое сообщение.

Теоретические основы словарных способов эффективного кодирования были разработаны израильскими математиками А. Лемпелем и Я. Зивом. Поэтому наиболее распространенные на практике алгоритмы сжатия, реализующие данные способы, известны под названи-

ем *алгоритмов Лемпеля – Зива*, или *алгоритмов группы LZ*. Существует несколько их разновидностей, из которых наиболее распространенными являются алгоритмы *LZW (Lempel – Ziv – Welch), BTLZ (British Telecom Lempel – Ziv)* и *LZ77*.

Общая методика сжатия по алгоритмам группы *LZ* состоит в следующем. Перед началом кодирования формируется словарь стандартных символов сообщения (называемых также *символами входного алфавита*) и определяется набор кодов нестандартных символов, зарезервированных для присвоения встречающимся в сообщении сочетаниям стандартных. Затем начинается последовательный просмотр сообщения, и каждое сочетание из двух и более стандартных символов (называемое *фразой*) при первой его встрече также заносится в словарь, и ему присваивается определенный символ из числа выделенных нестандартных. При повторных встречах фразы в сообщении она заменяется соответствующим ей нестандартным символом из словаря. При корректной стратегии составления последнего нет необходимости в его передаче вместе со сжатым сообщением, так как он может быть однозначно восстановлен на приемной стороне непосредственно из полученного сообщения (алгоритмы группы *LZ* обладают свойством *самовосстановления*).

Ниже рассмотрены процедуры кодирования и расшифровки сообщений в соответствии с простейшим алгоритмом группы *LZ – LZW*.

Кодирование по алгоритму *LZW* реализуется следующим образом [2]:

1. Инициализируется словарь сообщения: определяются коды стандартных символов, а также набор кодов, зарезервированных для присвоения нестандартным символам. Их указатель устанавливается на начальный элемент данного набора.

2. Определяется начальное значение *кода текущей фразы, w*, равное коду первого символа сжимаемого сообщения.

3. Считывается *код очередного стандартного символа сжимаемого сообщения, K*. Если это код конца сообщения, последнему символу сжатого сообщения присваивается код *w*, и на этом процесс кодирования завершается. Если нет – осуществляется переход к п. 4.

4. Если код фразы *wK* уже имеется в словаре – он присваивается переменной *w*, и выполняется переход к п. 3. Если же нет – переход к п. 5.

5. Фраза *wK* заносится в словарь, т. е. ей присваивается код, соответствующий текущему состоянию указателя нестандартных символов. Затем данный указатель устанавливается на следующий по порядку элемент набора кодов нестандартных символов. Очередному символу сжатого сообщения присваивается код *w*; переменной *w* присваивается значение *K*, и осуществляется переход к п. 3.

Поясним вышеприведенный алгоритм кодирования небольшим примером. Пусть необходимо осуществить сжатие сообщения:

HEY, HEY, HEY!!!

При этом стандартные символы должны представляться в *ASCII*-кодировке, а нестандартные символы, вводимые в словарь в процессе сжатия, – кодами от *080h* до *0FFh* включительно (здесь и далее «*h*» указывает на то, что код является шестнадцатеричным). Разрядность как стандартных, так и нестандартных символов устанавливаем фиксированной и равной 12 битам.

Процесс кодирования показан в табл. 1.7. Сжатое сообщение представляет собой следующую кодовую последовательность:

048h 045h 059h 02Ch 020h 080h 082h 084h 081h 021h 089h

Она содержит 11 12-битовых символов, в то время как при отсутствии сжатия исходное сообщение состояло бы из 16 8-битовых *ASCII*-кодов. Коэффициент сжатия, таким образом, равен 0,97. Это обусловлено небольшим объемом сообщения. Чем больше объем сообщения и «повторяемость» его лексики, тем выше коэффициент сжатия.

Декомпрессия сообщения, сжатого в соответствии с алгоритмом *LZW*, производится в таком порядке:

1. Инициализируется словарь декодируемого сообщения, аналогично п. 1 алгоритма кодирования (при этом для корректного декодирования коды стандартных символов и набор кодов, выделенных для нестандартных символов, должны быть полностью идентичны соответствующим наборам кодов, применявшимся при сжатии сообщения).

2. Считывается первый код сжатого сообщения (это всегда код стандартного символа). Он присваивается переменной «*предыдущий код*», которая обозначается *P*. Значение *P* присваивается первому коду расшифровываемого сообщения.

3. Считывается *очередной код сжатого сообщения, K***.** Если это код окончания сообщения, процесс расшифровки завершается. Если нет – переход к п. 4.

Таблица 1.7

Пояснение процесса кодирования по алгоритму LZW

Указатель нестандартных символов	*K*	*w*	*wK*	Фраза *wK* есть в словаре ?	Добавление в словарь	Выход кодера
080h	48h (H)	048h (H)			-	-
080h	45h (E)	048h (H)	HE	Нет	Cd(HE) = = 080h	048h (H)
081h	59h (Y)	045h (E)	EY	Нет	Cd(EY) = = 081h	045h (E)
082h	2Ch (,)	059h (Y)	Y,	Нет	Cd(Y,) = 082h	059h (Y)
083h	20h (sp)	02Ch (,)	,sp	Нет	Cd(,sp) = 083h	02Ch (,)
084h	48h (H)	020h (sp)	spH	Нет	Cd(spH) = = 084h	020h (sp)
085h	45h (E)	048h (H)	HE	Есть	-	-
085h	59h (Y)	080h (HE)	HEY	Нет	Cd(HEY) = = 085h	080h (HE)
086h	2Ch (,)	059h (Y)	Y,	Есть	-	-
086h	20h (sp)	082h (Y,)	Y,sp	Нет	Cd(Y,sp) = = 086h	082h (Y,)
087h	48h (H)	020h (sp)	spH	Есть	-	-
087h	45h (E)	084h (spH)	spHE	Нет	Cd(spHE) = = 087h	084h (spH)
088h	59h (Y)	045h (E)	EY	Есть	-	-
088h	21h (!)	081h (EY)	EY!	Нет	Cd(EY!) = = 088h	081h (EY)
089h	21h (!)	021h (!!)	!!	Нет	Cd(!!) = = 089h	021h (!)
08Ah	21h (!)	021h (!)	!!	Есть	-	-
08Ah	Конец сообщения	089h (!!)	-		-	089h (!!)

Примечания:
sp – пробел;
Cd(•) – код соответствующей последовательности символов

4. Если K – код нестандартного символа, осуществляется переход к п. 5. Если нет – очередному коду расшифровываемого сообщения присваивается значение K; фраза PK заносится в словарь, и ей присваивается код, соответствующий текущему состоянию указателя нестандартных символов. Затем выполняется переход к п. 7.

5. Если позиция словаря, соответствующая коду K, еще не заполнена (код K еще не присвоен какой-либо фразе), осуществляется переход к п. 6. В противном случае в расшифровываемое сообщение заносится последовательность кодов стандартных символов, соответствующая (по словарю) коду K. Фраза Pf_K (где f_K – первый из символов данной последовательности) заносится в текущую позицию словаря, и выполняется переход к п. 7.

6. В расшифровываемое сообщение заносится последовательность кодов стандартных символов, соответствующая фразе Pf_P, где f_P – первый из последовательности стандартных символов, соответствующей коду P. Фраза Pf_P заносится в текущую позицию словаря. После этого осуществляется переход к п. 7.

7. Указатель нестандартных символов устанавливается на следующий по порядку элемент набора их кодов, переменной P присваивается значение K, и выполняется переход к п. 3.

Данный алгоритм поясняется табл. 1.8, в которой представлен процесс декомпрессии последовательности, полученной в предыдущем примере.

Общие принципы кодирования, реализуемые другими алгоритмами группы *LZ* [2, 3], в целом аналогичны принципу кодирования по вышеописанному алгоритму *LZW*.

Из вышеизложенного нетрудно заметить, что эффективность сжатия словарными способами тем выше, чем больше повторяющихся сочетаний символов оно содержит и чем больше длина данных последовательностей. Известны способы обратимого преобразования сообщения, позволяющие повысить эффективность его сжатия, в том числе словарного, например, *преобразование Барроуза – Уилера* (см. п. 1.2.7).

Следует отметить, что на практике часто применяется сочетание словарного кодирования с последующим статистическим (префиксным неравномерным или арифметическим) кодированием символов словаря (см. табл. 1.10 на с. 52).

Таблица 1.8

Пояснение процесса декомпрессии по алгоритму LZW

Указатель нестандартных символов	*K*	*P*	f_K	f_P	Символ *K* стандартный?	Символ *K* есть в словаре?	Добавление в словарь	Выход декодера
080h	048h (H)	048h (H)	H	H	Да	Да	-	48h (H)
080h	045h (E)	048h (H)	E	H	Да	Да	Cd(HE) = = 080h	45h (E)
081h	059h (Y)	045h (E)	Y	E	Да	Да	Cd(EY) = = 081h	59h (Y)
082h	02Ch (,)	059h (Y)	,	Y	Да	Да	Cd(Y,) = = 082h	2Ch (,)
083h	020h (sp)	02Ch (,)	sp	,	Да	Да	Cd(,sp) = = 083h	20h (sp)
084h	080h (HE)	020h (sp)	H	sp	Нет	Да	Cd(spH) = = 084h	48h 45h (HE)
085h	082h (Y,)	080h (HE)	Y	H	Нет	Да	Cd(HEY) = = 085h	59h 2Ch (Y,)
086h	084h (spH)	082h (Y,)	sp	Y	Нет	Да	Cd(Y,sp) = = 086h	20h 48h (spH)
087h	081h (EY)	084h (spH)	E	sp	Нет	Да	Cd(spHE) = = 087h	45h 59h (EY)
088h	021h (!)	081h (EY)	!	E	Да	Да	Cd(EY!) = = 088h	21h (!)
089h	089h (!!)	021h (!)	!	!	Нет	Нет	089h = = Cd(Pf_P) = = Cd(!!)	21h 21h (!!)
08Ah	Конец сообщения							

1.2.7. Преобразование сообщения с целью снижения его энтропии

Как следует из приведенных в пп. 1.2.3 – 1.2.6 описаний алгоритмов сжатия, его эффективность тем выше, чем более «регулярный»

характер имеет подвергаемое кодированию сообщение, например, чем большее количество повторяющихся последовательностей символов в нем содержится. Поэтому одним из распространенных методов повышения эффективности сжатия дискретных сообщений является преобразование подвергаемой сжатию последовательности символов в другую последовательность, обладающую большей «регулярностью», а следовательно, меньшей информационной энтропией. После сжатия полученное в результате такого преобразования сообщение будет обладать меньшим объемом, чем исходное сообщение после сжатия по тому же алгоритму с тем же профилем.

Следовательно, снижение энтропии сообщения перед сжатием позволяет повысить коэффициент последнего, причем независимо от того, каким способом и по какому алгоритму оно осуществляется. Естественно, преобразование сообщения перед сжатием должно быть *обратимым*, т. е. должен существовать алгоритм однозначного восстановления исходного, подвергнутого преобразованию сообщения по результату декомпрессии.

Типичным и одним из наиболее распространенных на практике представителем данной разновидности преобразований является *преобразование Барроуза – Уилера* (*Burrows – Wheeler Transform – BWT*) [3, 4]. Его сущность состоит в обратимом переупорядочивании символов сжимаемого сообщения, после которого данное сообщение представляет собой последовательность из n цепочек по m_i одинаковых символов в каждой, где n – число символов в алфавите сообщения, а m_i – количество раз, которое i-й символ алфавита встречается в сообщении. Например, последовательность символов *banana* посредством *BWT* преобразуется в последовательность *nnbaaa* (рис. 1.7). Полученная в результате преобразования последовательность обладает потенциально лучшей «сжимаемостью», чем исходная. Естественно, эффективность применения *BWT* тем выше, чем больше объем (в символах) сообщения, подвергаемого данному преобразованию.

Одним из распространенных алгоритмов *прямого BWT* является следующий (см. рис. 1.7) [3]:

1. Строится матрица размером $N \times N$, где N – суммарное число символов в сообщении. В первую строку матрицы записывается исходное сообщение, а в качестве каждой последующей строки служит предыдущая, циклически сдвинутая на один символ влево.

2. Осуществляется перестановка строк матрицы в соответствии с алфавитным порядком их элементов (в общем случае – в порядке возрастания числовых кодов строк).

3. В качестве результата преобразования служат:

- последний столбец матрицы, полученной в результате выполнения п. 2;

- номер строки данной матрицы, содержащей исходное сообщение.

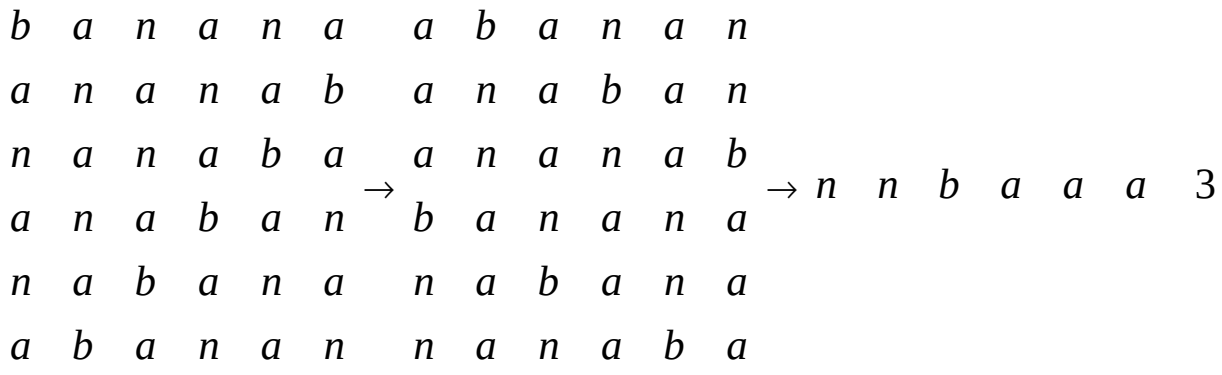


Рис. 1.7. Пояснение алгоритма прямого *BWT*

Полученный результат преобразования подвергается компрессии. При этом результат декомпрессии, представляющий собой результат прямого *BWT* исходного сообщения, подвергается *обратному BWT* с целью его восстановления. При реализации прямого *BWT* по вышеприведенному алгоритму обратное *BWT* осуществляется следующим образом (рис. 1.8):

1. Строится матрица размером $N \times N$. В ее N-й столбец записывается последовательность символов, полученная в результате прямого *BWT* исходного сообщения, а в 1-й столбец – та же последовательность в алфавитном порядке. Остальные элементы матрицы оставляются незаполненными.

2. Все строки матрицы циклически сдвигаются на один символ вправо.

3. Строки матрицы переставляются в соответствии с алфавитным порядком их элементов, после чего в N-й столбец записывается последовательность символов, полученная в результате прямого *BWT* исходного сообщения.

4. Пункты 2 и 3 повторяются до тех пор, пока не окажутся заполненными все позиции в матрице.

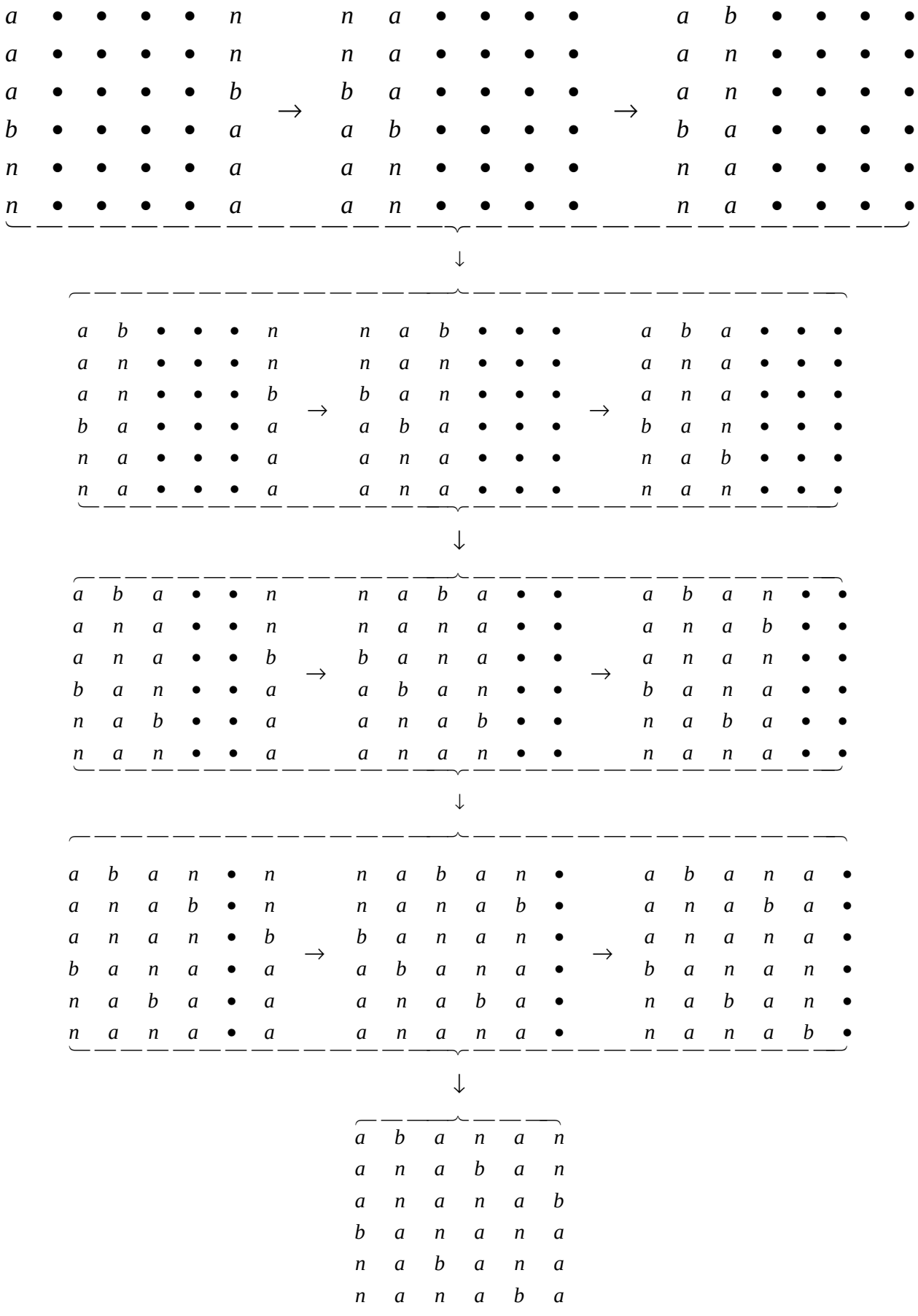


Рис. 1.8. Пояснение алгоритма обратного *BWT*

5. По окончании процедуры заполнения матрицы ее строка, номер которой входит в состав результатов прямого *BWT* (см. рис. 1.7), совпадает с исходным сообщением и, следовательно, является результатом обратного *BWT*.

Сопоставляя вышеприведенные алгоритмы прямого и обратного *BWT* (см. рис. 1.7 и 1.8), нетрудно заметить, что в процессе обратного *BWT*, по существу, восстанавливаются операции, выполненные во время процедуры прямого *BWT*.

Последовательность, полученная в результате прямого *BWT*, как указано ранее, подвергается собственно компактному кодированию. В принципе, оно может осуществляться такими из вышеописанных способов, как кодирование длин серий, префиксное неравномерное и арифметическое кодирование, словарное сжатие, или их сочетаниями. На практике преобразование Барроуза – Уилера часто сопровождается последующим кодированием по способу *MTF* (*Move To Front* – Перемещение вперед), которое, с одной стороны, обладает свойствами сжатия само по себе, а с другой – позволяет повысить эффективность вышеперечисленных способов кодирования.

Кодирование способом *MTF* реализуется по следующему обобщенному алгоритму (табл. 1.9) [4]:

1. Каждому символу алфавита сообщения присваивается определенная позиция в алфавите (в рассматриваемом примере для простоты предполагается, что алфавит сообщения состоит только из строчных латинских букв).

2. Считывается очередной символ сообщения. Ему присваивается числовой код, равный номеру его текущей позиции. После этого считанному символу присваивается позиция первого символа алфавита, откуда происходит название способа – «Перемещение вперед». Остальные символы занимают последующие позиции в соответствии с ранее произведенными изменениями позиций символов.

3. Пункт 2 повторяется до считывания последнего символа сообщения включительно.

Из табл. 1.9 нетрудно заметить, что чем чаще встречается символ, тем меньшее значение числового кода ему присваивается. Это позволяет обеспечить сжатие само по себе при префиксном неравномерном кодировании номеров позиций и/или повысить эффективность последующего сжатия результатов *MTF*-кодирования.

Таблица 1.9

Пояснения принципа кодирования способом MTF на примере последовательности nnbaaa, полученной в результате прямого BWT слова banana (рис. 1.7)

Шаг	**Символ**	**Код**	**Позиции символов**																								
			0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
0	**-**	**-**	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
1	***n***	**13**	*n*	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
2	***n***	**0**	*n*	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
3	***b***	**2**	*b*	*n*	*a*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
4	***a***	**2**	*a*	*b*	*n*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
5	***a***	**0**	*a*	*b*	*n*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
6	***a***	**0**	*a*	*b*	*n*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*

Кроме вышеописанных, известны и другие, в настоящее время менее распространенные на практике способы преобразования сообщений с целью снижения их априорной ИЭ [3, 4].

1.2.8. Обзор применяемых на практике алгоритмов сжатия дискретных сообщений

Краткий обзор распространенных в настоящее время алгоритмов эффективного кодирования дискретных сообщений представлен в табл. 1.10 [2 – 4].

Таблица 1.10

Распространенные алгоритмы сжатия дискретных сообщений

Алгоритм	Применяемые способы кодирования дискретных сообщений	Архиваторы, использующие алгоритм
PPM	Статистические. Определение статистики сообщения методом контекстного моделирования с предсказанием. Кодирование символов – префиксное неравномерное Хаффмана или арифметическое (в зависимости от версии)	WinZip, RAR, 7-Zip, HA
LZMA	Словарное кодирование по модифицированному алгоритму LZ77	WinZip, 7-Zip
BZip2	Префиксное неравномерное кодирование Хаффмана с предварительными, последовательно выполняемыми процедурами преобразований Барроуза – Уилера и MTF	
Deflate	Комбинация словарного кодирования по алгоритму LZ77 и префиксного неравномерного кодирования Хаффмана	7-Zip, GZip

1.3. Способы и алгоритмы эффективного кодирования аудиоданных

Звуковые сообщения относятся к аналоговым: их алфавит представляет собой теоретически бесконечное множество символов, в качестве которых выступают уровни звукового сигнала. Однако на практике число таких уровней конечно из-за представления сигнала

в виде последовательности цифровых отсчетов и, как следствие, его квантования по уровню. В отличие от, например, текстовых сообщений непосредственное применение статистических или словарных методов кодирования звуковых сообщений малоэффективно из-за отсутствия собственной смысловой нагрузки у отдельных символов (например, один и тот же код цифрового отсчета звукового сигнала может нести различную информацию в зависимости от кодов соседних отсчетов).

Следовательно, эффективное кодирование звуковых сообщений требует *преобразования их алфавита в некоторый другой*, символы которого обладают собственным однозначным смыслом. Классификация основных способов преобразования звуковых сообщений при сжатии представлена на рис. 1.9 [2].

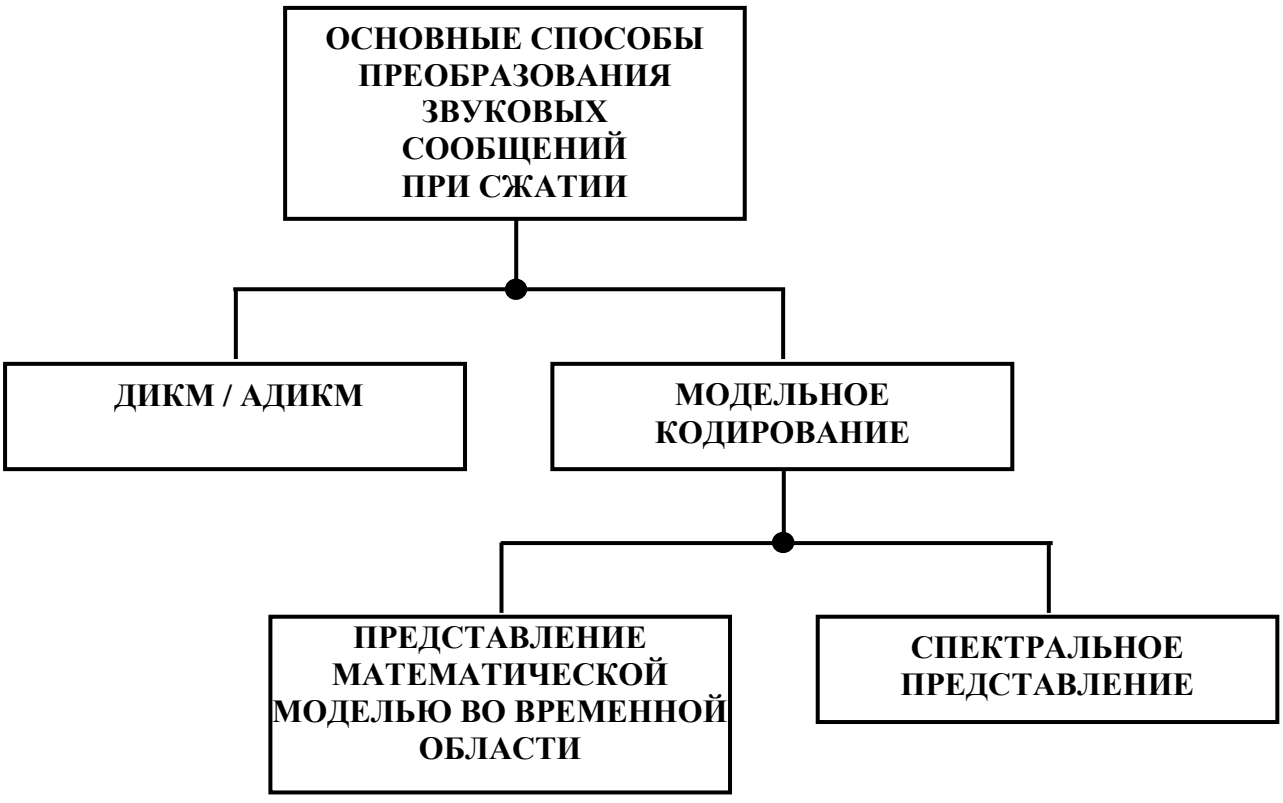


Рис. 1.9. Основные способы преобразования звуковых сообщений при их эффективном кодировании

Указанные преобразования сами по себе существенно снижают объем сообщения. Поэтому ряд протоколов и алгоритмов эффективного кодирования звука предполагает применение только какого-либо из данных преобразований или их сочетаний. Однако на практи-

ке чаще используется последующее кодирование результатов преобразования каким-либо из методов эффективного кодирования дискретных сообщений (см. рис. 1.1) или их сочетанием.

Необходимо также отметить, что компактное кодирование звуковых сообщений, как правило, допустимо осуществлять *с потерями информации* при условии, что данные потери незаметны или малозаметны для слуха. Например, человеческий слух обладает значительно меньшей чувствительностью к изменениям уровня спектральных составляющих в частотном диапазоне 50 – 100 Гц, чем в диапазоне 2 – 3 кГц [2]. Следовательно, амплитуды спектральных компонент первого из указанных диапазонов допустимо представлять двоичными числами меньшей разрядности, чем второго. Потери информации вносятся на этапе преобразования (см. рис. 1.9), а их степень определяется конкретным профилем протокола и/или алгоритма преобразования, задаваемым пользователем в зависимости от конкретных требований к качеству звукового сообщения после декодирования.

Дифференциальная импульсно-кодовая модуляция (ДИКМ – DPCM) в простейшем случае состоит в представлении звукового сигнала в виде последовательности чисел вида [2]:

$$\Delta[iT] = x[iT] - x[(i-1)T], \qquad (1.15)$$

где T – период дискретизации;

$x[iT]$ и $x[(i-1)T]$ – текущий и предыдущий цифровые отсчеты звукового сигнала.

Представление вида (1.15) обладает следующими преимуществами:

- символы, представляющие собой разности отсчетов сигнала, несут более определенную информацию, чем одиночные символы, и обладают большей повторяемостью; поэтому сообщение, состоящее из таких символов, значительно эффективнее сжимается, например, методом *RLE* или префиксным неравномерным кодированием;

- разности между отсчетами, как правило, меньше по значению и, следовательно, могут представляться двоичными числами меньшей разрядности, чем сами отсчеты.

Для повышения эффективности компрессии на практике достаточно широко применяется *адаптивная* ДИКМ (*АДИКМ – ADPCM*), базирующаяся на двух основных принципах или (чаще) на их сочетаниях:

- представление сигнала не последовательностью разностей текущего и предыдущего отсчетов (см. выражение (1.15)), а последовательностью разностей следующего вида [2]:

$$\Delta_a[iT] = x[iT] - x_p[iT], \qquad (1.16)$$

где $x_p[iT]$ – «предсказанное» значение текущего отсчета, оцениваемое кодером (модулятором) как некоторая функция $F_p\{x[(i-1)T],\ldots,x[(i-m)T]\}$ от m предыдущих отсчетов, число которых, а также вид функции $F_p\{\bullet\}$ оговариваются протоколом кодирования/декодирования; при декодировании отсчеты $x[iT]$ восстанавливаются как сумма значений $\Delta_a[iT]$ и $F_p\{\bullet\}$;

- адаптация разрядности двоичных чисел, представляющих разность вида (1.15) или (1.16), к ее значению; в наиболее «продвинутых» алгоритмах – индивидуальный выбор значения данной разрядности равным $]\log_2\{\Delta[iT]\}[$ или, соответственно, $]\log_2\{\Delta_a[iT]\}[$ для каждого конкретного значения разности.

Как и ДИКМ, АДИКМ не только снижает объем звукового сообщения сама по себе, но и способствует эффективности последующего сжатия результатов модуляции методами, указанными на рис. 1.1, в том числе их сочетаниями.

В целом, степень сжатия при использовании ДИКМ тем выше, чем более «плавной» и «предсказуемой» является форма сигнала. Следует также отметить, что ДИКМ позволяет сжимать аудиоданные *без потерь***.**

В целом, собственно ДИКМ (АДИКМ) обеспечивает относительно невысокую степень сжатия (порядка нескольких единиц). Типовой пример использования АДИКМ – кодирование речи в системах бесшнуровой телефонии стандарта *DECT*.

Модельное кодирование состоит в замене последовательности отсчетов каждого из фрагментов звукового сигнала его математической моделью, вид которой определяется методом и конкретным алгоритмом кодирования, а длительность фрагментов – профилем алгоритма. Результатами кодирования служат параметры математической модели, на основании которых при декодировании восстанавливается исходный сигнал.

Наиболее распространенными методами математического моделирования звукового сигнала, применяемыми при его компактном кодировании, являются [2]:

- *кодирование с линейным предсказанием (Linear Predictive Coding – LPC)* во временной области;

- представление сигнала в виде *суммы спектральных компонент* (гармоник или вейвлет-составляющих).

Кодирование звукового сигнала с линейным предсказанием основывается на его представлении как результата фильтрации некоторого элементарного сигнала, периодического (тонального) или шумового, некоторым рекурсивным цифровым фильтром (ЦФ). При этом сигнал описывается математической моделью, которая в наиболее простом случае имеет следующий вид [2]:

$$\begin{cases} x[iT] = u[iT] + \sum\limits_{k=1}^{K} a_{kj} \times x[(i-k)T], \\ u[iT] = G_j \times \left\{ V_j \times v[iT] + \overline{V_j} \times n[iT] \right\}, \end{cases} \tag{1.17}$$

где j – номер фрагмента звукового сигнала;

$x[\bullet]$ – его отсчеты;

$u[\bullet]$, $v[\bullet]$, $n[\bullet]$ – отсчеты входного сигнала ЦФ, его периодической (тональной, например импульсной) и шумовой составляющих соответственно;

V_j – параметр, равный нулю или единице, который определяет преобладающую компоненту звука (периодическую при $V_j = 1$ или шумовую при $V_j = 0$) в пределах j-го фрагмента сигнала;

G_j – параметр, задающий интенсивность звука в пределах j-го фрагмента;

a_{kj} – коэффициенты ЦФ, определяемые как решения следующего матричного уравнения:

$$\begin{pmatrix} R[0] & R[1] & R[2] & \bullet\bullet\bullet & R[K-2] & R[K-1] \\ R[1] & R[0] & R[1] & \bullet\bullet\bullet & R[K-3] & R[K-2] \\ R[2] & R[1] & R[0] & \bullet\bullet\bullet & R[K-4] & R[K-3] \\ R[3] & R[2] & R[1] & \bullet\bullet\bullet & R[K-5] & R\{K-4\} \\ \bullet & \bullet & \bullet & \bullet\bullet\bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet\bullet\bullet & \bullet & \bullet \\ R[K-2] & R[K-3] & R[K-4] & \bullet\bullet\bullet & R[0] & R[1] \\ R[K-1] & R[K-2] & R[K-3] & \bullet\bullet\bullet & R[1] & R[0] \end{pmatrix} \times \begin{pmatrix} a_{1j} \\ a_{2j} \\ a_{3j} \\ a_{4j} \\ \bullet \\ \bullet \\ a_{(K-1)j} \\ a_{Kj} \end{pmatrix} = \begin{pmatrix} -R[1] \\ -R[2] \\ -R[3] \\ -R[4] \\ \bullet \\ \bullet \\ -R[K-1] \\ -R[K] \end{pmatrix}, \tag{1.18}$$

где $R[k]$ – значение автокорреляционной функции (АКФ) последовательности отсчетов $x[\bullet]$, равное:

$$R[k] = \sum_{i=0}^{N-k} x[iT] \times x[(i+k)T], \qquad (1.19)$$

где N – общее число отсчетов в пределах фрагмента, равное отношению его длительности к периоду дискретизации.

При этом число K коэффициентов модели вида (1.17) называется *порядком* модели или порядком ЦФ. Чем он выше, тем точнее модель представляет сигнал, но, с другой стороны, тем бóльшие вычислительные и временные ресурсы требуются для кодирования сигнала. Значение K определяется конкретным протоколом кодирования. Также последний устанавливает вид периодической составляющей входного сигнала ЦФ и его шумовой компоненты (большинство протоколов кодирования с линейным предсказанием оговаривает использование в ее качестве отсчетов «белого» шума). В свою очередь, в качестве периодической составляющей часто используется импульсная последовательность. Данный тип периодической составляющей известен под аббревиатурой *RPE* (*Regular Pulse Excitation*).

На модели вида (1.17) основан следующий алгоритм *кодирования* с линейным предсказанием звукового сигнала:

1. Сигнал разбивается на фрагменты, в пределах которых параметры a_{kj}, V_j и G_j модели вида (1.17) могут считаться постоянными. На практике длительность данных фрагментов обычно выбирается равной 20 мс [2].

2. Получают N отсчетов звукового сигнала в пределах текущего фрагмента.

3. На основании выражений (1.18) и (1.19) определяются коэффициенты a_{kj} модели вида (1.17).

4. По последовательности отсчетов сигнала в пределах текущего фрагмента устанавливается, является ли он периодическим. Если он периодичен, то определяется его период, T_{sj}, а параметр V_j принимается равным единице. В противном случае данный параметр принимается равным нулю.

5. Находится параметр G_j как решение уравнения вида (1.17) относительно G_j при определенных в пп. 3 и 4 коэффициентах a_{kj}, значениях T_{sj} и V_j, а также при оговоренных протоколом кодирования видах тональной и шумовой составляющих входного сигнала ЦФ.

6. Формируется результат кодирования j-го фрагмента сигнала как массив его параметров $\{a_{1j},\ldots,a_{Kj}, G_j, V_j, T_{sj}\}$.

7. Если текущий фрагмент не последний, значение j увеличивается на единицу, и осуществляется переход к п. 2. В противном случае – завершение процедуры кодирования.

Декодирование результатов кодирования с линейным предсказанием, основанного на модели вида (1.17), реализуется по следующему обобщенному алгоритму (при этом в качестве входных данных декодера выступают параметры $\{a_{1j},\ldots,a_{Kj}, G_j, V_j, T_{sj}\}$ для каждого из фрагментов сигнала):

1. На основании параметров G_j, T_{sj} и V_j сигнала, а также оговоренных протоколом кодирования видов тональной и шумовой составляющих входного сигнала ЦФ восстанавливаются отсчеты $u[\bullet]$ данного сигнала в пределах j-го фрагмента.

2. На основании отсчетов $u[\bullet]$, а также коэффициентов a_{kj} по выражению (1.17) восстанавливаются отсчеты $\hat{x}[\bullet]$ звукового сигнала в пределах j-го фрагмента. В общем случае они отличаются от исходных отсчетов $x[\bullet]$, т. е. процессы кодирования и декодирования вносят *потери данных***.** Однако на практике всегда может быть обеспечен приемлемый уровень данных потерь, не искажающий смысл информации, передаваемой сигналом.

3. Если текущий фрагмент не последний, значение j увеличивается на единицу, и выполняется переход к п. 1. В противном случае – завершение процедуры декодирования.

Кроме вышеописанных, существуют более сложные алгоритмы кодирования и декодирования с линейным предсказанием, обеспечивающие более качественное восстановление сигнала [2]. В частности, на практике применяются способы кодирования, в которых в качестве отсчетов входного сигнала $u[\bullet]$ ЦФ используются кодовые комбинации, генерируемые по специальным таблицам (кодовым книгам). Данные способы известны под аббревиатурой *CELP* (*Code-Exited Linear Prediction* – линейное предсказание с кодовым возбуждением). К ним относится применяемый в ряде стандартов мобильной связи способ *ACELP* (*Algebraic CELP*) [2].

Основным достоинством кодирования звукового сигнала с линейным предсказанием является относительная простота процедур

кодирования и декодирования по сравнению со спектральными методами. Основной недостаток состоит в относительно низком качестве восстановления звука при декодировании. Поэтому данный способ кодирования используется при жестких требованиях к временным и вычислительным затратам на процедуры кодирования/декодирования при одновременном отсутствии необходимости в высоком качестве восстановления сигнала при декодировании. К таким областям применения, в первую очередь, относится кодирование речи в системах связи (в том числе сотовой [2]). Также данный метод в сочетании со спектральным моделированием и АДИКМ используется для кодирования речи стандартом *MPEG-4 Audio* [3].

В целом, кодирование с линейным предсказанием обеспечивает степень сжатия речевых сигналов порядка нескольких десятков. Однако чем выше степень сжатия, тем ниже качество восстановленного сигнала.

Моделирование звукового сигнала *суммой спектральных компонент* требует значительно бóльших временных и вычислительных затрат, чем *LPC*, но обеспечивает и более высокие степень сжатия и качество восстановления сигнала. Обобщенная математическая модель некоторого j-го фрагмента сигнала при его спектральном представлении выглядит следующим образом [2]:

$$x[iT] = \sum_{k=1}^{K_j} C_{jk} \times F_{jk}[iT], \qquad (1.20)$$

где $F_{jk}[iT]$ – функции, принадлежащие к некоторому *ортогональному* базису; наиболее распространенными из таких базисов являются гармонические функции (косинусные и синусные) и различные типы вейвлет-функций;

C_{jk} – коэффициенты, получаемые в результате разложения сигнала по функциям соответствующего базиса на j-м интервале времени, реализуемого методами преобразования Фурье, косинусного или вейвлет-преобразования (при использовании в качестве базисных функций синусоид и косинусоид, только косинусоид или вейвлет-функций соответственно).

При этом конкретный вид базиса определяется стандартом (протоколом) кодирования. Число (K_j) и параметры функций $F_{jk}[iT]$, представляющих различные фрагменты сигнала, в общем случае раз-

личны. Под *спектральными компонентами* сигнала при этом понимаются слагаемые суммы (1.20). Коэффициенты C_{jk}, по существу, являются *амплитудами* данных компонент.

Обобщенный алгоритм эффективного кодирования звукового сигнала на основе спектрального представления таков [2]:

1. Разделение сигнала на фрагменты, в пределах которых его спектральный состав (состав функций $F_{jk}[iT]$ и значения коэффициентов C_{jk}) может считаться неизменным с точностью, определяемой требуемым качеством восстановления звука при декодировании.

2. Разложение каждого из фрагментов сигнала на функции базиса, оговариваемого стандартом (протоколом) кодирования.

3. Анализ спектрального состава сигнала в пределах фрагмента и выявление компонент, несущественных при требуемом качестве восстановления.

4. Формирование результата кодирования каждого из фрагментов как набора параметров спектральных компонент сигнала в пределах фрагмента, т. е. значений коэффициентов C_{jk} и параметров функций $F_{jk}[iT]$ (например, частот гармонических составляющих). При этом несущественные спектральные компоненты (см. п. 3) не включаются в результат представления или их коэффициенты C_{jk} кодируются двоичными числами меньшей разрядности, чем аналогичные коэффициенты основных спектральных компонент.

5. Массив чисел, полученный в результате выполнения п. 4, в свою очередь, подвергается компактному кодированию одним из методов, указанных на рис. 1.1, или их сочетанием.

Декодирование аудиоданных, сжатых в соответствии с вышеприведенным алгоритмом, осуществляется последовательным выполнением процедур декодирования массива чисел, описывающего спектральные компоненты сигнала, и восстановления отсчетов сигнала по выражениям вида (1.20). Строго говоря, восстановленные отсчеты не совпадают с исходными, т. е., как и при кодировании с линейным предсказанием, имеют место *потери данных*. Последние тем не менее всегда могут быть сведены к уровню, приемлемому с точки зрения требуемого в каждом конкретном случае качества восстановления.

Типовым практическим примером стандарта эффективного кодирования аудиоданных на основе их спектрального представления является *MP3* (рис. 1.10) [2].

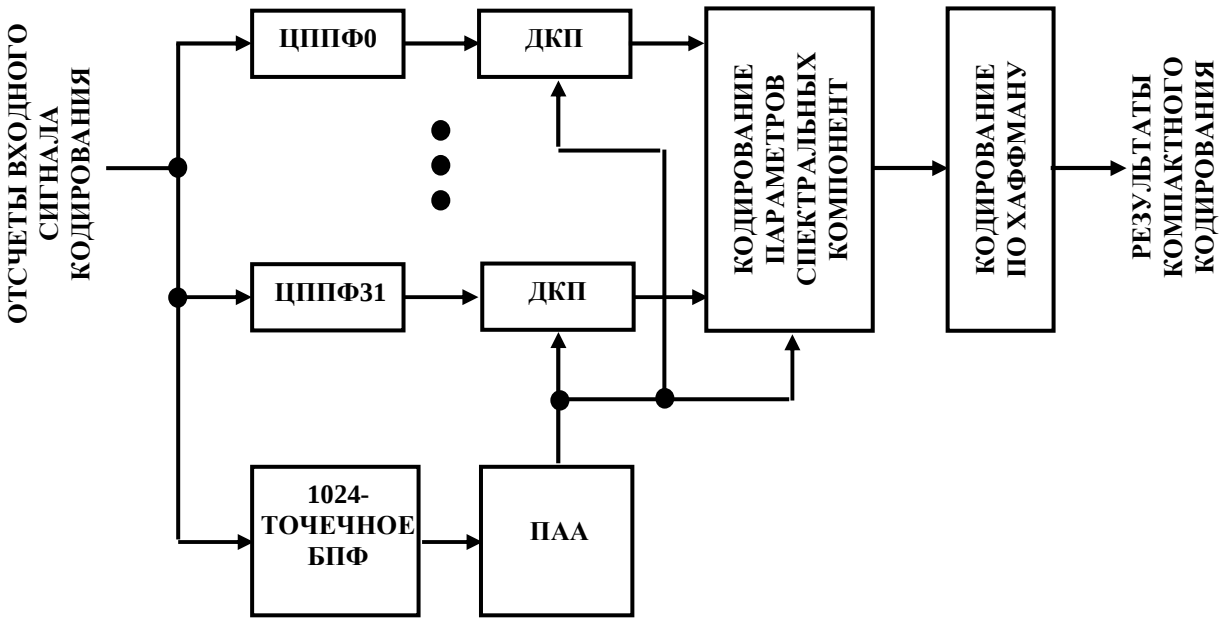


Рис. 1.10. Упрощенная операционная модель эффективного кодирования по протоколу *MP3:*

ЦППФ – цифровые полосно-пропускающие фильтры;

БПФ – быстрое преобразование Фурье;

ДКП – дискретное косинусное преобразование;

ПАА – психоакустический анализ

Звуковой сигнал разбивается на равные по продолжительности фрагменты, результаты кодирования каждого из которых оформляются в отдельный фрейм (кадр). В пределах каждого из фрагментов отсчеты звукового сигнала подвергаются фильтрации набором («банком») из 32 цифровых полосно-пропускающих фильтров, разбивающих частотный диапазон указанного сигнала на 32 поддиапазона. На выходе каждого из фильтров формируется последовательность отсчетов спектральных компонент сигнала, лежащих в соответствующем частотном поддиапазоне.

Последовательность отсчетов выходного сигнала каждого из фильтров, в свою очередь, представляется как сумма вида (1.20) с косинусоидами в качестве базисных функций $F_{jk}[iT]$. Коэффициенты C_{jk} при этом представляют собой произведения их амплитуд

на косинусы их начальных фаз и определяются методом дискретного косинусного преобразования (ДКП), описываемого выражением

$$C_{jk} = \sqrt{\frac{c_k}{N}} \times \sum_{i=0}^{N-1} x[jNT + iT] \times \cos\left[\frac{\pi k(2i+1)}{2N}\right];$$

$$k = 0,\ldots,N-1;\ c_k = \begin{cases} 1 \text{ при } k = 0; \\ 2 \text{ при } k \neq 0, \end{cases} \qquad (1.21)$$

где N – число отсчетов сигнала в пределах фрагмента.

Представление сигнала результатами ДКП, полученными по выражениям вида (1.21), *не позволяет производить полноценный анализ* его спектрального состава, который возможен, например, по результатам дискретного преобразования Фурье (ДПФ). Однако данное представление позволяет *восстановить сигнал* с любой заданной точностью методом обратного ДКП, описываемого следующим выражением [2]:

$$x[jNT + iT] = \sqrt{\frac{c_k}{N}} \times \sum_{k=0}^{K-1} C_{jk} \times \cos\left[\frac{\pi k(2i+1)}{2N}\right];\ i = 0,\ldots,N-1, \qquad (1.22)$$

где K – число спектральных компонент, необходимых для восстановления сигнала с заданной точностью,

при отсутствии необходимости оперировать комплексными числами, что имеет место при прямом и обратном ДПФ.

Кроме ДКП, каждый из фрагментов сигнала подвергается быстрому преобразованию Фурье (БПФ) во всем диапазоне информативных частот. По результатам БПФ оценивается *психоакустическая модель* сигнала, т. е. особенности его спектрального состава с точки зрения слухового восприятия, которые учитываются при кодировании результатов ДКП. Основными из данных особенностей являются следующие:

- распределение амплитуд спектральных компонент по диапазонам частот и по фрагментам сигнала; на основании данного распределения выбираются разрядности представления указанных амплитуд для различных частотных диапазонов и фрагментов сигнала;

- наличие двух или более близких по частоте спектральных компонент с амплитудами одинакового порядка; такие компоненты слабо различимы на слух, поэтому при кодировании они заменяются одной, усредненной по частоте и амплитуде;

- наличие двух или более спектральных компонент, близких по частоте к некоторой гармонике, существенно превышающей их по амплитуде; такие компоненты слабо воспринимаются на слух и поэтому исключаются из результатов кодирования;

- наличие двух или более спектральных компонент, следующих во времени непосредственно за некоторой гармоникой, существенно превышающей их по амплитуде; такие компоненты также слабо воспринимаются при прослушивании и поэтому исключаются из результатов кодирования.

Кроме кодирования параметров спектральных компонент, результаты психоакустического анализа сигнала используются также для управления параметрами ДКП (см. рис. 1.10).

Массив параметров спектральных компонент фрагмента сигнала, полученный в результате ДКП и обработанный в соответствии с психоакустической моделью сигнала, подвергается префиксному неравномерному кодированию по Хаффману. Его результаты оформляются в кадр (фрейм), являющийся конечным результатом кодирования.

Декодирование результатов сжатия по стандарту *MP3* осуществляется последовательным выполнением процедур декодирования по Хаффману и восстановления сигнала методом обратного ДКП по параметрам его спектральных компонент. При этом, с целью обеспечения непрерывности восстанавливаемого сигнала, при восстановлении отсчетов некоторого фрагмента используются данные не только кадра, соответствующего этому фрагменту, но также предыдущего и последующего кадров.

Основывающиеся на результатах психоакустического анализа исключение ряда спектральных компонент из результатов кодирования и/или снижение точности представления параметров ряда компонент являются источниками потерь информации при кодировании. Данные потери тем больше, чем больше степень сжатия. Однако существующие средства кодирования по стандарту *MP3* (как и по другим стандартам сжатия аудиоданных), как правило, предоставляют пользователю возможность выбора *профиля* кодирования из нескольких, различающихся между собой степенью сжатия и потерями информации.

При прочих равных условиях степень сжатия на основе спектрального представления сигнала тем выше, чем меньше спектральных составляющих он содержит и чем меньше изменяются их пара-

метры во времени. Наивысшая степень сжатия может быть достигнута для одиночной ноты, наименьшая – для шумоподобного звукового сигнала (например, для записи рок-концерта).

Следует отметить, что, в принципе, на основе спектрального представления возможно и кодирование *без потерь*.

В табл. 1.11 приведены наиболее распространенные форматы представления аудиоданных при компактном кодировании, а также характеристики используемых данными форматами стандартов/протоколов сжатия [2, 3].

Таблица 1.11

Распространенные форматы представления сжатых аудиоданных

Формат	Протокол сжатия	Методы кодирования, используемые протоколом
1	2	3
*.mp3	MP3	Сжатие с потерями на основе спектрального представления методом ДКП с последующим кодированием параметров спектральных компонент по Хаффману
*.ogg	Vorbis	Сжатие с потерями на основе спектрального представления методом ДКП с последующим энтропийным кодированием параметров спектральных компонент. Кодирование шумовых составляющих отдельно от остальных
*.m4a, *.aac	Advanced Audio Coding	Сжатие с потерями на основе спектрального представления методом ДКП с различной длиной окна для различных частотных диапазонов и опциональным линейным предсказанием
*.ac3	Dolby Digital (ATSC A/52)	Сжатие с потерями на основе спектрального представления методом ДКП с взвешиванием отсчетов, различной длиной окна для различных частотных диапазонов и разностным кодированием результатов ДКП
*.wma	Windows Media Audio	Сжатие с потерями или без потерь (в зависимости от профиля) на основе спектрального представления методом ДКП с последующим кодированием параметров спектральных компонент по Хаффману
*.amr, *.3ga	Adaptive Multi-Rate (AMR or AMR-NB or GSM-AMR) audio codec	Сжатие с потерями модельным кодированием во времени способом ACELP (линейное предсказание с алгебраическим кодовым возбуждением) и применением дополнительных технологий прерывистой передачи, детектирования голосовой активности и генерирования комфортного шума. Протокол ориентирован преимущественно на сжатие речи

Окончание табл. 1.11

1	2	3
*.wav	GSM 6.10 WAV	Моделирование во времени способом долговременного предсказания с регулярным импульсным возбуждением (RPE-LTP)
*.flac	Free Lossless Audio Codec	Сжатие без потерь способом моделирования сигнала во времени полиномом или линейным предсказанием с энтропийным кодированием ошибок моделирования

1.4. Способы и алгоритмы эффективного кодирования изображений

Наиболее универсальной формой представления изображения является *растровая* [2, 3]. При этом *монохромное* изображение описывается одним двумерным массивом чисел, каждый из элементов I_{xy} которого прямо пропорционален интенсивности пикселя с координатами *x* и *y*. В свою очередь, монохромные изображения подразделяются на *бинарные*, интенсивности пикселей которых могут принимать только два значения (0 и 1) и *полутоновые*, интенсивности которых характеризуются числом уровней, бóльшим двух. *Цветное* изображение в общем случае описывается тремя или четырьмя двумерными массивами чисел, элементы каждого из которых прямо пропорциональны интенсивности определенной цветовой составляющей пикселя с соответствующими координатами. При этом цветовые составляющие принадлежат к одной из следующих *ортогональных* систем цветов [3]:

- *RGB* (красный, зеленый, синий), используемой в мониторах и телевизорах;
- *CMYK* **(**голубой, пурпурный, желтый, черный), применяемой в цветной печати.

Ортогональность данных систем состоит в том, что ни один из входящих в них цветов не может быть получен смешиванием остальных цветов системы. Вследствие этого любой существующий в природе цвет может быть получен как сумма цветовых составляющих ортогональной системы в определенных пропорциях. При этом на практике цветные изображения часто представляются только одним массивом двоичных кодов, в каждом из элементов которого выделяются по три или, соответственно, по четыре поля для кодирования интенсивности каждой из цветовых составляющих.

Основные отличия изображений от звуковых сообщений с точки зрения эффективного кодирования состоят в том, что [2, 3]:

- для массивов чисел, описывающих изображения, характерна значительно меньшая априорная ИЭ, чем для последовательностей отсчетов звукового сигнала, за счет частой встречаемости групп пикселей с одинаковыми параметрами, большей «предсказуемости» параметров пикселя по параметрам соседних пикселей, чем отсчета звукового сигнала по соседним отсчетам и т. п.;

- существуют как разновидности изображений, допускающих сжатие с потерями, малозаметными для человеческого глаза (например, фотографии из периодических изданий), так и категории изображений, не допускающих каких-либо потерь информации при сжатии (например, изображения в системах медицинской диагностики).

Указанные отличия обусловливают следующие особенности сжатия изображений [2, 3]:

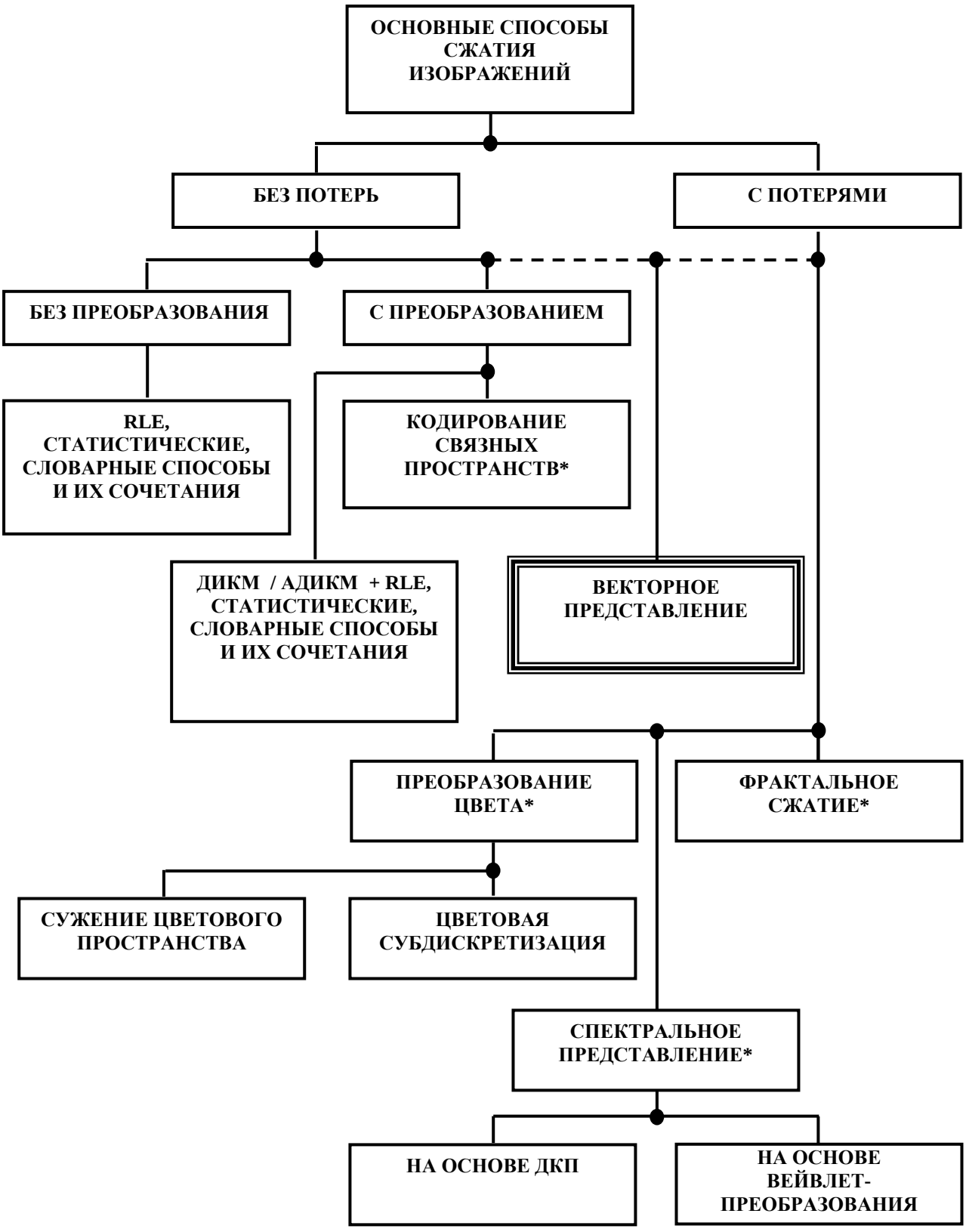
• оно относительно эффективно (по сравнению с аудиоданными) может быть выполнено способами, применяемыми для сжатия дискретных сообщений (см. рис. 1.1), без предварительных преобразований; однако широко используется и сочетание данных способов с предварительным преобразованием массива чисел, описывающего изображение;

• примерно одинаково распространены протоколы/алгоритмы сжатия изображений как без потерь, так и с потерями.

Классификация основных известных способов сжатия изображений представлена на рис. 1.11 [2, 3]. На практике часто применяются сочетания данных способов с целью повышения эффективности сжатия. Типовым примером такого сочетания являются протоколы группы *JPEG* (см. с. 73).

Сжатие *без потерь и без предварительного преобразования* массивов чисел, представляющих изображение, осуществляется теми же способами, что и эффективное кодирование дискретных сообщений (см. рис. 1.1).

Способы сжатия изображений *без потерь с предварительной ДИКМ или АДИКМ* аналогичны способам сжатия аудиоданных, описываемым выражениями (1.15) или (1.16) соответственно, за исключением того, что кодирование осуществляется не в одномерной, а в двумерной системе координат. Одним из простых примеров алгоритма сжатия изображений, использующего ДИКМ (конкретно – АДИКМ), является алгоритм *Lossless JPEG* (*JPEG* без потерь) [2, 3], предусматриваемый протоколом *JPEG*. *Lossless JPEG* не следует путать с *JPEG-LS*,



* На практике часто применяется дополнение кодирования данными способами последующим сжатием его результатов различными сочетаниями *ДИКМ*, *RLE*, статистических и словарных способов

Рис. 1.11. Основные способы сжатия изображений

также основывающимся на АДИКМ, но отличающимся от него более сложным, но в то же время более эффективным алгоритмом кодирования.

Кодирование по алгоритму *Lossless JPEG* выполняется следующим образом [2]. Изображение представляется в виде двумерного массива чисел, каждое из которых равно разности между предсказанным и действительным значением интенсивности пикселя, описываемого соответствующим элементом массива. Интенсивность *x* некоторого пикселя предсказывается на основе интенсивностей трех соседних пикселей (рис. 1.12).

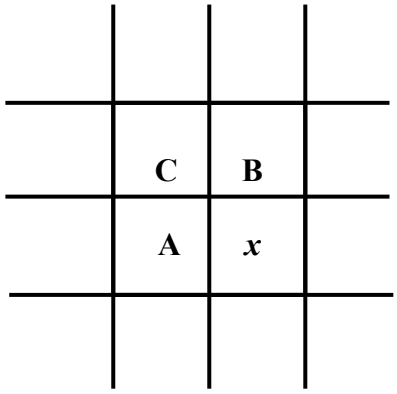


Рис. 1.12. Пояснения к процессу предсказания по алгоритму *Lossless JPEG*

Профиль алгоритма, определяющий функцию предсказания (табл. 1.12), выбирается пользователем и указывается в файле сжатого изображения для обеспечения корректного декодирования, с использованием той же функции предсказания, что и при кодировании. Интенсивности трех пикселей, расположенных в левом верхнем углу изображения, не подвергаются кодированию, указываются в файле непосредственно и служат исходными данными при декодировании. Полученный массив разностей затем подвергается кодированию по Хаффману или арифметическому кодированию (в зависимости от конкретной опции алгоритма).

Декодирование изображения, сжатого по алгоритму *Lossless JPEG*, осуществляется последовательным выполнением процедур:

- восстановления массива разностей между реальными и предсказанными интенсивностями пикселей декодированием по Хаффману или арифметическим декодированием;

- восстановления интенсивностей пикселей путем суммирования разностей, полученных в результате выполнения предыдущего этапа, с интенсивностями, предсказанными по тем же выражениям, которые применялись при кодировании (профиль предсказания указывается в сжатом файле).

Таблица 1.12

Функции предсказания при различных профилях кодирования по алгоритму Lossless JPEG

Профиль	Функция предсказания
1	$x = A$
2	$x = B$
3	$x = C$
4	$x = A + B - C$
5	$x = A + \frac{B - C}{2}$
6	$x = B + \frac{A - C}{2}$
7	$x = \frac{A + B}{2}$

Алгоритм *Lossless JPEG* не отличается высокими коэффициентами сжатия (не более 2). Поэтому в настоящее время более распространен протокол сжатия изображений без потерь *JPEG-LS* [2]. Он также основывается на АДИКМ с предсказанием, однако обеспечивает при прочих равных условиях более высокие коэффициенты сжатия, чем *Lossless JPEG*, за счет таких особенностей, как:

- адаптация алгоритма предсказания к изображению, т. е. применение не одной и той же функции предсказания на всех участках изображения, а ее автоматический выбор в зависимости от особенностей участка (структуры, градиента интенсивности и т. п.);

- автоматический переход к *RLE*-кодированию на однородных участках изображения.

Более подробное описание протокола *JPEG-LS* представлено, например, в [2].

Сжатие, основанное на *кодировании связных пространств*, состоит в представлении изображения в виде набора независимо кодируемых фигур – связных пространств.

Связное пространство, по определению, представляет собой топологическое пространство, которое невозможно разбить на два непустых непересекающихся открытых подмножества. Иными словами, связное пространство – это сплошная фигура на плоскости. В противном случае пространство называется *несвязным* (рис. 1.13).

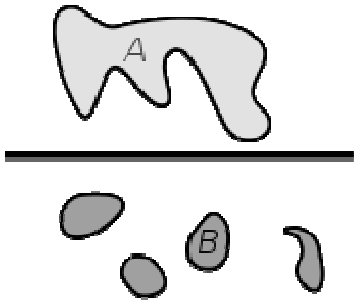


Рис. 1.13. Примеры связного (*A*)
и несвязного (*B*) пространств

Представление связных пространств при сжатии, как правило, осуществляется так называемыми *цепными кодами* – последовательностями символов, описывающими контур пространства и его атрибуты (интенсивность пикселей, формирующих пространство и т. п.) [3]. В свою очередь, описание контура состоит из координат начальной точки его обхода и последовательности символов, описывающих траекторию движения по контуру (например, последовательность приращений координат при движении по контуру). Представление изображений в виде совокупности таких описаний намного компактнее, чем их представление в растровом виде. Кроме того, описывающая контур последовательность приращений координат может быть эффективно сжата методами *RLE*, статистическими или словарными.

Сжатие изображений с потерями осуществляется на основе одного из следующих принципов или их сочетаний (см. рис. 1.11):

- преобразование цвета, состоящее в устранении из файла изображения избыточной информации о цвете его элементов, не влияющей или мало влияющей на зрительное восприятие изображения;

- представление сжатого изображения в виде пространственного спектра интенсивностей его пикселей с устранением или кодированием с пониженной точностью спектральных компонент, слабо влияющих на восприятие изображения, и с восстановлением изображения из спектра при декодировании посредством преобразования, обратного примененному при спектральном анализе;

- представление изображения в виде алгоритма его построения на основе многократных преобразований некоторой геометрической фигуры (*фрактала*), обладающей свойством *самоподобия*.

Известны два основных способа *преобразования цвета* при сжатии изображений (рис. 1.11) [2, 3]:

- сужение цветового пространства;
- цветовая субдискретизация.

Сужение цветового пространства состоит в уменьшении числа цветов палитры изображения до минимально достаточного для восприятия при выбранном пользователем уровне качества (например, с 256 до 16 цветов). При этом, естественно, снижается разрядность чисел, представляющих интенсивности цветовых составляющих пикселей и, как следствие, объем файла изображения.

Цветовая субдискретизация заключается в представлении информации о цветности изображения с меньшим разрешением, чем о яркости. Данный подход основан на том, что человеческое зрение более чувствительно к перепадам яркости, чем цвета. Цветовая субдискретизация реализуется следующим образом:

1. Интенсивности цветовых составляющих каждого из пикселей изображения преобразуются в интенсивности *яркостной* и *цветоразностных* составляющих пикселя, несущие информацию соответственно о его яркости и цвете. Например, в системе *RGB* яркостная и цветоразностные составляющие связаны с цветовыми составляющими следующими соотношениями:

$$\left.\begin{aligned} Y &= 0{,}299\times R + 0{,}587\times G + 0{,}114\times B;\\ C_R &= 0{,}713\times(R-Y);\\ C_B &= 0{,}564\times(B-Y), \end{aligned}\right\} \qquad (1.23)$$

где R, G, B – интенсивности соответственно красной, зеленой и синей составляющих цвета пикселя;

Y, C_R, C_B – интенсивности его яркостной и цветоразностных составляющих.

Описываемая соотношениями (1.23) система представления цвета известна под аббревиатурой YC_RC_B или YUV.

Составляющие R, G, B однозначно восстанавливаются из значений Y, C_R, C_B по следующим выражениям:

$$\left.\begin{aligned} R &= Y + 1{,}402\times C_R;\\ G &= Y - 0{,}714\times C_R - 0{,}344\times C_B;\\ B &= Y + 1{,}772\times C_B. \end{aligned}\right\} \qquad (1.24)$$

2. Описание изображения в ортогональной системе цветов преобразуется в его описание в системе из яркостной и цветоразностных составляющих, например в системе YC_RC_B, по следующим обобщенным правилам:

- интенсивность яркостной составляющей указывается индивидуально для каждого из пикселей;

- каждой группе пикселей, содержащей a столбцов и b строк, назначаются одинаковые значения интенсивностей цветоразностных составляющих.

Таким образом, цветоразностные составляющие представляются с в a раз меньшим разрешением по горизонтали и в b раз меньшим – по вертикали, чем яркостная. За счет указанных выше особенностей человеческого зрения это существенно не сказывается на качестве восприятия. При этом массив яркостных составляющих имеет ту же размерность, что и массив пикселей, а массивы цветоразностных – в a раз меньше столбцов и в b раз меньше строк, за счет чего достигается уменьшение объема файла изображения.

При декодировании цветовые составляющие восстанавливаются в соответствии с выражениями (1.24).

Конкретные значения a и b определяются стандартом (протоколом) кодирования. Например, формат сжатия видеоданных *MPEG-2* предусматривает снижение разрешения по горизонтали в 2 раза ($a = 2$) при полном разрешении по вертикали ($b = 1$).

Как правило, цветовая субдискретизация используется в сочетании с другими методами сжатия, например спектральными.

Большинство алгоритмов сжатия, основанных на *спектральном представлении* изображения, использует *косинусоиды* или *вейвлет-функции* в качестве базиса [2, 3]. Соответственно, декомпозиция изображения на указанные базисные функции осуществляется методом *пространственного ДКП* или *пространственного дискретного вейвлет-преобразования (ДВП)*. Как указано ранее, базовый принцип сжатия состоит в удалении или кодировании с пониженной точностью спектральных компонент, слабо влияющих на восприятие изображения. Как правило, данный прием сочетается с предваритель-

ной цветовой субдискретизацией, а также с кодированием параметров спектра одним из способов, указанных на рис. 1.1, или их сочетаниями.

Показательным примером алгоритма сжатия, основанного на пространственном ДКП, является *JPEG*. Он включает в себя следующие этапы [2, 3]:

1. Представление изображения в системе YC_RC_B в соответствии с выражениями (1.23).

2. Субдискретизация цветоразностных составляющих со значениями a и b (см. выше), равными 2.

3. Разбиение матриц интенсивностей яркостной и цветоразностных составляющих на неперекрывающиеся подматрицы размерностью 8×8.

4. Применение к каждой из указанных подматриц процедуры двумерного ДКП, аналогичного по сущности описываемому выражением (1.21) одномерному ДКП и реализуемого в соответствии со следующими базовыми выражениями:

$$\begin{cases} G_c(i,j) = c_i c_j \sum\limits_{k=0}^{N-1}\sum\limits_{m=0}^{N-1} I_c(k,m) \times \cos\left[\dfrac{\pi i(2k+1)}{2N}\right] \times \cos\left[\dfrac{\pi j(2m+1)}{2N}\right]; \\ i, j = 0, \ldots, N-1; \\ c_i, c_j = 1/\sqrt{2} \quad \text{при} \quad i, j = 0; \\ c_i, c_j = 1 \quad \text{при} \quad i, j \neq 0, \end{cases} \tag{1.25}$$

где N – число элементов (пикселей) подматрицы по горизонтали и по вертикали;

$I_c(k,m)$ – интенсивность c-й составляющей (яркостной или цветоразностной) пикселя, соответствующего элементу подматрицы, находящемуся на пересечении ее k-й строки и m-го столбца;

$G_c(i,j)$ – коэффициенты разложения c-й составляющей соответствующего фрагмента изображения на пространственные спектральные компоненты (рис. 1.14) [2]. При этом каждый из коэффициентов соответствует компоненте, приведенной на пересечении i-й строки и j-го столбца.

Чем выше индекс i или j коэффициента $G_c(i, j)$, тем более высокочастотной и менее информативной для восприятия спектральной компоненте он соответствует (рис. 1.14).

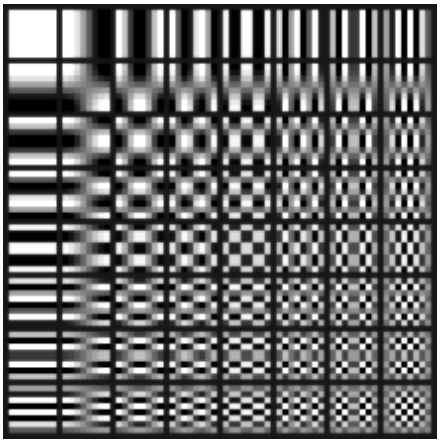


Рис. 1.14. Общий вид пространственных спектральных компонент ДКП фрагмента изображения размером 8×8 пикселей

5. Квантование коэффициентов $G_c(i, j)$ ДКП в соответствии с выражением

$$B_c(i, j) = round\left[\frac{G_c(i, j)}{Q_{ij}}\right], \qquad (1.26)$$

где $B_c(i, j)$ – целочисленное представление коэффициента $G_c(i, j)$;

$round[\bullet]$ – оператор округления;

Q_{ij} – элементы *матрицы квантования*, значения которых зависят от профиля алгоритма и определяют качество изображения при восстановлении. Матрица квантования, соответствующая базовому профилю *JPEG*, приведена ниже [2]:

$$Q = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}.$$

Из данной матрицы, из рис. 1.14 и из выражения (1.26) нетрудно заметить, что, чем более высокочастотной компоненте соответствует коэффициент $B_c(i, j)$, тем с меньшей точностью он представляется.

Например, при следующей матрице коэффициентов ДКП:

$$G = \begin{pmatrix} -415{,}38 & -30{,}19 & -61{,}20 & 27{,}24 & 56{,}13 & -20{,}10 & -2{,}39 & 0{,}46 \\ 4{,}47 & -21{,}86 & -60{,}76 & 10{,}25 & 13{,}15 & -7{,}09 & -8{,}54 & 4{,}88 \\ -46{,}83 & 7{,}37 & 77{,}13 & -24{,}56 & -28{,}91 & 9{,}93 & 5{,}42 & -5{,}65 \\ -48{,}53 & 12{,}07 & 34{,}10 & -14{,}76 & -10{,}24 & 6{,}30 & 1{,}83 & 1{,}95 \\ 12{,}12 & -6{,}55 & -13{,}20 & -3{,}95 & -1{,}88 & 1{,}75 & -2{,}79 & 3{,}14 \\ -7{,}73 & 2{,}91 & 2{,}38 & -5{,}94 & -2{,}38 & 0{,}94 & 4{,}30 & 1{,}85 \\ -1{,}03 & 0{,}18 & 0{,}42 & -2{,}42 & -0.88 & -3{,}02 & 4{,}12 & -0{,}66 \\ -0{,}17 & 0{,}14 & -1{,}07 & -4{,}19 & -1{,}17 & -0{,}10 & 0{,}50 & 1{,}68 \end{pmatrix}$$

и вышеприведенной матрице квантования его результаты выглядят следующим образом:

$$B = \begin{pmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

т. е. большинство результатов квантования оказываются нулевыми. Большое количество нулевых коэффициентов в матрицах ДКП, в свою очередь, существенно снижает объем сжатого файла.

6. Из элементов матрицы B результатов квантования формируется 64-элементный массив в соответствии с представленной траекторией упорядочивания [2, 3] (рис. 1.15), повышающей эффективность последующего кодирования элементов указанного массива методом *RLE*.

7. Полученный в результате упорядочивания массив кодируется методом *RLE*.

8. Результаты *RLE*-кодирования подвергаются кодированию по Хаффману.

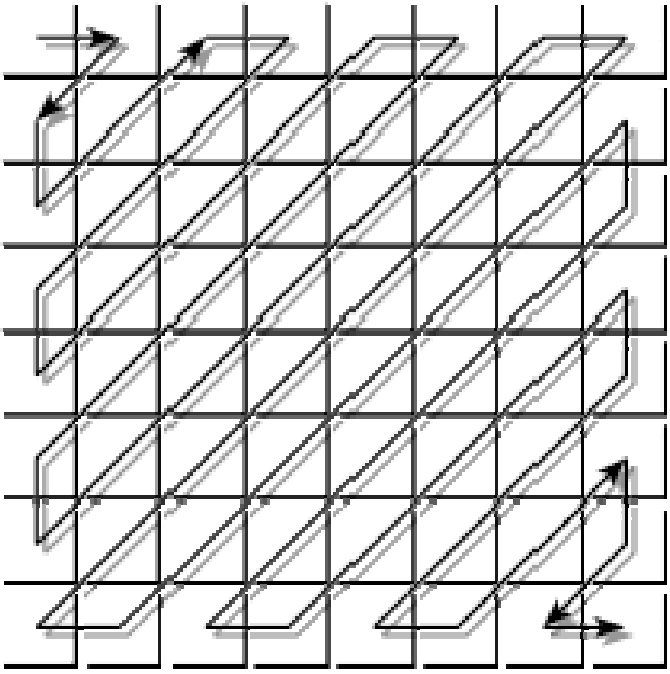


Рис. 1.15. Траектория упорядочивания результатов квантования коэффициентов ДКП в соответствии с алгоритмом *JPEG*

Декодирование реализуется выполнением в обратном порядке процедур, также обратных вышеперечисленным. При этом обратное двумерное ДКП реализуется в соответствии со следующими базовыми выражениями [ср. с выражениями (1.25)] [2, 3]:

$$\begin{cases} \hat{I}_c(k,m) = c_i c_j \sum\limits_{i=0}^{N-1} \sum\limits_{j=0}^{N-1} B_c(i,j) \times \cos\left[\dfrac{\pi i(2k+1)}{2N}\right] \times \cos\left[\dfrac{\pi j(2m+1)}{2N}\right]; \\ i, j = 0, \ldots, N-1; \\ c_i, c_j = 1/\sqrt{2} \quad \text{при} \quad i, j = 0; \\ c_i, c_j = 1 \quad \text{при} \quad i, j \neq 0, \end{cases} \tag{1.27}$$

где $\hat{I}_c(k,m)$ – восстановленное значение интенсивности $I_c(k,m)$, как правило, не совпадающее с нею из-за потерь информации при квантовании в соответствии с выражением (1.26), т. е. по существу, из-за менее точного представления или игнорирования малоинформативных спектральных компонент.

Типовым примером алгоритма сжатия, основанного на *ДВП*, является *JPEG 2000*. Он включает в себя следующие основные процедуры [2, 3]:

1. Представление подлежащего сжатию изображения в системе YC_RC_B, описываемой выражениями (1.23) и (1.24) (при сжатии с потерями) или в модифицированной системе YUV (при сжатии как с потерями, так и без них). Преобразование $RGB \to YUV$ реализуется по выражениям:

$$\left.\begin{aligned} Y &= \lfloor (R + 2\times G + B)/4 \rfloor; \\ U &= R - G; \\ V &= B - G, \end{aligned}\right\} \qquad (1.28)$$

где $\lfloor \bullet \rfloor$ – оператор округления до ближайшего меньшего или равного целого,

а обратное преобразование – следующим образом:

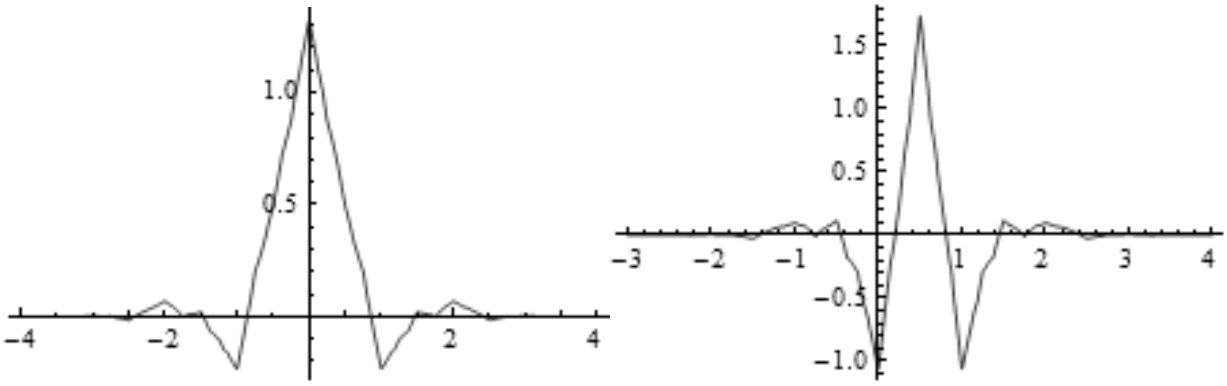
$$\left.\begin{aligned} G &= Y - \lfloor (U + V)/4 \rfloor; \\ R &= U + G; \\ B &= V + G. \end{aligned}\right\} \qquad (1.29)$$

Представление в модифицированной системе YUV, описываемое выражениями (1.28) и (1.29), характеризуется меньшими погрешностями округления, чем в системе YC_RC_B, и поэтому применяется при сжатии как с потерями, так и без потерь.

2. После представления изображения в соответствующей цветовой системе (YC_RC_B или YUV) оно разбивается на фрагменты квадратной формы с размером, который определяется выбранным пользователем профилем сжатия (в том числе возможно использование только одного фрагмента, в качестве которого выступает все изображение целиком).

3. Каждая из составляющих системы YC_RC_B или YUV каждого из фрагментов подвергается *ДВП*. Последнее, в отличие от ДПФ или ДКП, состоит в декомпозиции фрагмента не на гармонические составляющие, а на *вейвлет-компоненты*. Они, в отличие от гармоник, конечны в пространстве или во времени (при ДВП во временной области) и представляют собой растянутые или сжатые, а также сдвинутые по одной из осей ординат или, соответственно, во времени копии некоторого *порождающего* вейвлета. В результате ДВП позволяет не только выявить наличие вейвлет-компонент той или иной длительности в анализируемой последовательности отсчетов, но и определить положение этих компонент в пространстве или во времени.

Известно достаточно большое количество порождающих вейвлетов, каждый из которых предпочтителен для каких-либо конкретных приложений. *JPEG 2000* оговаривает применение семейства вейвлетов, названного по фамилиям предложивших их специалистов вейвлетами *CDF* (*Cohen – Daubechies – Feauveau*) (вейвлеты Коэна – Добеши – Фово). Для вейвлетов *CDF*, как и для всех применяемых в ДВП, характерно наличие двух порождаемых ими функций – аппроксимирующей и детализирующей. Аппроксимирующая функция служит в качестве базисной при представлении низкочастотных составляющих подвергаемой ДВП последовательности отсчетов, а детализирующая – ее высокочастотных компонент. Временные диаграммы данных функций вейвлета *CDF 9/7*, применяемого алгоритмом *JPEG2000* при сжатии с потерями, представлены на рис. 1.16.



а *б*

Рис. 1.16. Временные диаграммы аппроксимирующей (*а*) и детализирующей (*б*) функций вейвлета *CDF 9/7*

Поэтапная декомпозиция на базовые и детализирующие вейвлет-компоненты является наиболее распространенным на практике способом ДВП (рис. 1.17).

Процедуры фильтрации на некотором j-м этапе декомпозиции (обычно называемом j-м уровнем ДВП), совмещенные с двукратным прореживанием (рис. 1.17), реализуются по следующим выражениям:

$$\left.\begin{aligned} b_j[i] &= \sum_k b_{j-1}[k] \times h_d[k+2i]; \\ d_j[i] &= \sum_k b_{j-1}[k] \times g_d[k+2i], \end{aligned}\right\} \qquad (1.30)$$

где $b_j[i]$ и $d_j[i]$ – отсчеты соответственно базовой и детализирующей вейвлет-компонент j-го уровня ДВП;

$h_d[\bullet]$ и $g_d[\bullet]$ – отсчеты импульсных характеристик соответственно низко- и высокочастотного цифровых вейвлет-фильтров *декомпозиции*.

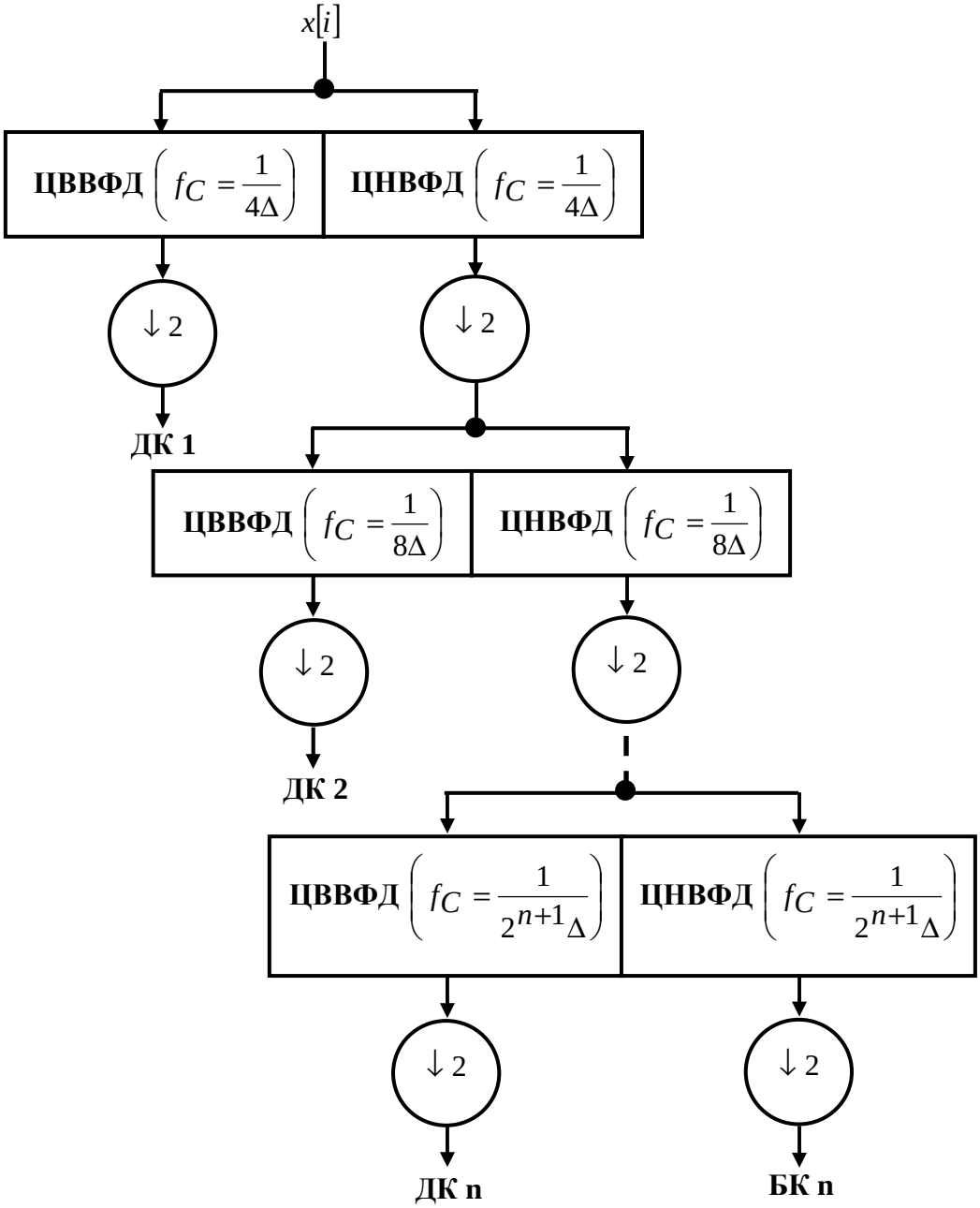
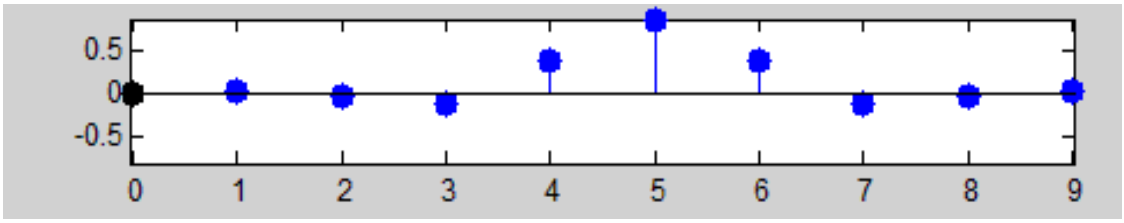


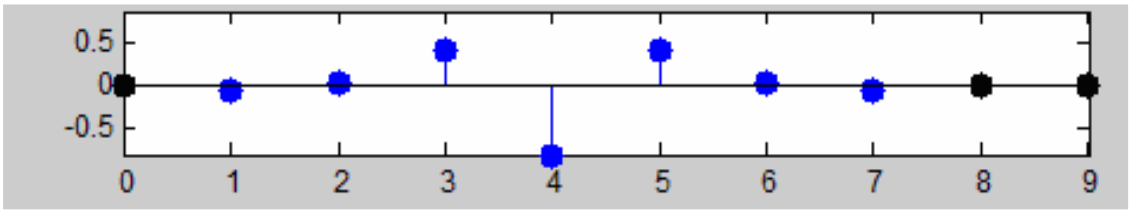
Рис. 1.17. Операционная модель *n*-уровневого одномерного ДВП:
$x[i]$ – входная последовательность отсчетов;
ЦНВФД, ЦВВФД – соответственно низкочастотный и высокочастотный цифровые вейвлет-фильтры декомпозиции;
f_C – частоты среза фильтров;
Δ – период дискретизации (применительно к изображениям – размер пикселя);
$\downarrow 2$ – оператор прореживания выходных отсчетов фильтров в 2 раза;
БК n – базовые компоненты *n*-го уровня ДВП;
ДК 1…ДК n – детализирующие компоненты соответствующих уровней ДВП

Импульсные характеристики низкочастотного и высокочастотного вейвлет-фильтров декомпозиции (рис. 1.18) связаны однозначными зависимостями соответственно с аппроксимирующей и детализирующей функциями вейвлета.

В качестве базовой компоненты нулевого уровня, $b_0[i]$, служит исходная последовательность отсчетов, подвергаемая ДВП.



а



б

Рис. 1.18. Импульсные характеристики ЦНВФД (*а*) и ЦВВФД (*б*) вейвлета *CDF 9/7*

Двумерное ДВП, лежащее в основе *JPEG 2000*, отличается от одномерного тем, что результатами выполнения каждого j-го его этапа (рис. 1.19) являются не две одномерные вейвлет-компоненты, а следующие четыре двумерных массива:

- массив, обозначаемый как jLL, представляющий собой результат низкочастотной вейвлет-фильтрации каждой из строк с последующей низкочастотной вейвлет-фильтрацией каждого из столбцов и прореживанием в 2 раза после каждой из процедур фильтрации;
- массив, обозначаемый как jLH, представляющий собой результат низкочастотной вейвлет-фильтрации каждой из строк с последующей высокочастотной вейвлет-фильтрацией каждого из столбцов и прореживанием в 2 раза после каждой из процедур фильтрации;
- массив, обозначаемый как jHL, представляющий собой результат высокочастотной вейвлет-фильтрации каждой из строк с последующей низкочастотной вейвлет-фильтрацией каждого из столбцов и прореживанием в 2 раза после каждой из процедур фильтрации;

- массив, обозначаемый как jHH, представляющий собой результат высокочастотной вейвлет-фильтрации каждой из строк с последующей высокочастотной вейвлет-фильтрацией каждого из столбцов и прореживанием в 2 раза после каждой из процедур фильтрации.

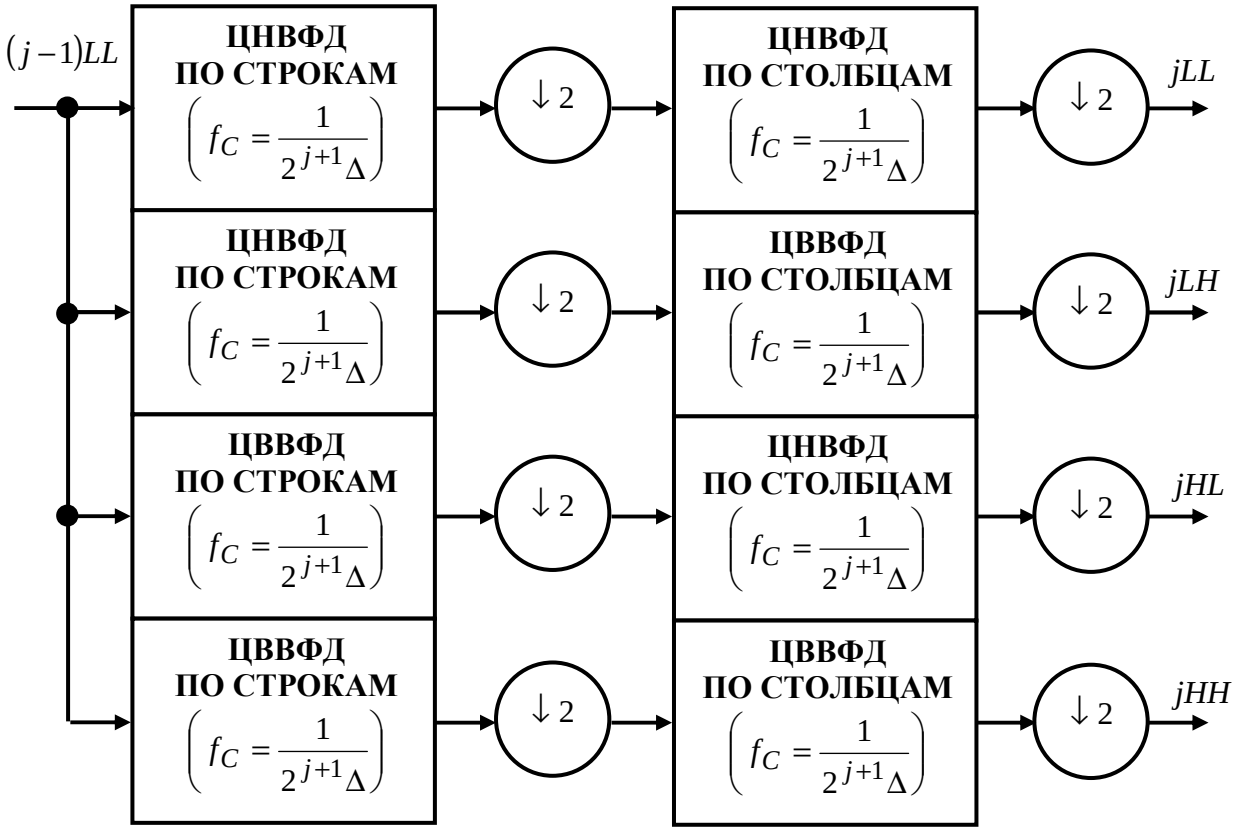


Рис. 1.19. Операционная модель одного этапа двумерного ДВП, применяемого в *JPEG 2000*

При этом входным массивом j-го уровня двумерного ДВП служит массив $(j-1)LL$. В качестве массива $0LL$, в свою очередь, выступают собственно подвергаемые ДВП исходные данные. Массив jLL выступает в качестве базовых компонент j-го уровня декомпозиции, а jLH, jHL и jHH – в качестве детализирующих компонент. Фильтрация по каждой из осей координат осуществляется также в соответствии с выражениями (1.30), с использованием ЦНВФД и ЦВВФД с теми же импульсными характеристиками, что и при одномерном ДВП на основе того же вейвлета. Частоты среза фильтров уменьшаются в 2 раза при переходе на каждый последующий уровень.

На рис. 1.20 приведен пример полутонового изображения, а на рис. 1.21 – результаты реализованного по вышеописанному алгоритму двухуровневого ДВП этого изображения [5]. ДВП осуществлялось на основе вейвлета *CDF 9/7* (см. рис. 1.15 и 1.17).

Рис. 1.20. Исходное изображение

Как указано ранее, при компрессии по алгоритму *JPEG 2000* ДВП подвергается каждая из составляющих системы YC_RC_B или YUV каждого из фрагментов. Число уровней ДВП зависит от выбранного профиля сжатия.

4. Результаты ДВП (элементы полученных массивов) подвергаются квантованию по принципу, аналогичному используемому алгоритмом *JPEG*: менее значимые для восприятия компоненты представляются с меньшей точностью (см. выражение (1.26) и пояснения к нему). Как видно из рис. 1.21, компоненты тем менее значимы, чем меньше номер уровня декомпозиции, к которому они относятся. По-

этому шаг квантования тем больше, чем меньше уровень ДВП. Например, элементы массива $2HH$ представляются с меньшей точностью (с бóльшим шагом квантования), чем массива $1HH$. С наивысшей точностью квантуются элементы массива nLL, наиболее важного для восприятия (см. рис. 1.17, 1.19 и 1.21). При этом, в зависимости от выбранного профиля сжатия, после квантования большинство элементов детализирующих массивов первых уровней ДВП (например, $1HH$, $1HL$ и $1LH$) обнуляется, аналогично тому, как это имеет место при *JPEG*-сжатии. Значения шага квантования результатов каждого из уровней ДВП определяются выбранным профилем сжатия.

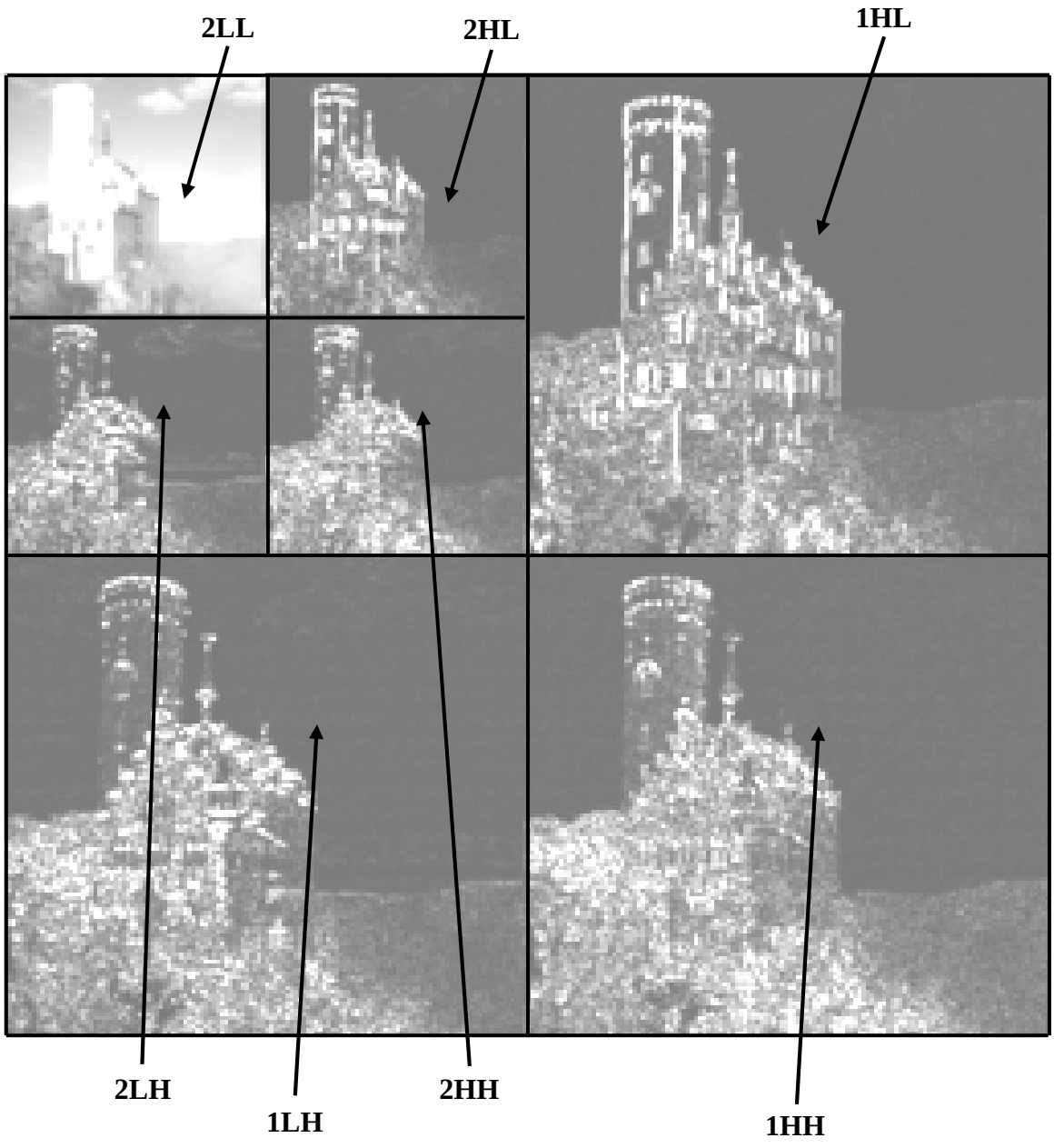


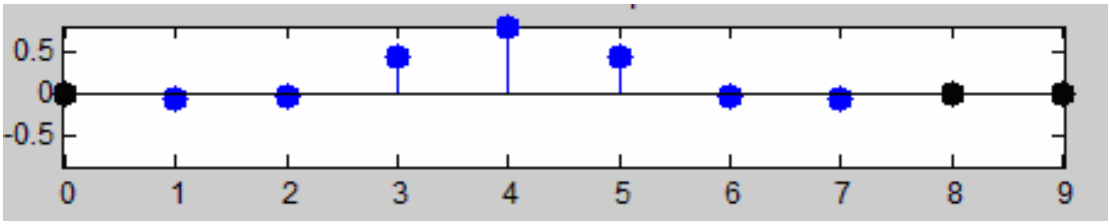
Рис. 1.21. Результаты двухуровневого ДВП изображения, представленного на рис. 1.20 (тип вейвлета – *CDF 9/7*)

5. Результаты квантования подвергаются арифметическому кодированию.

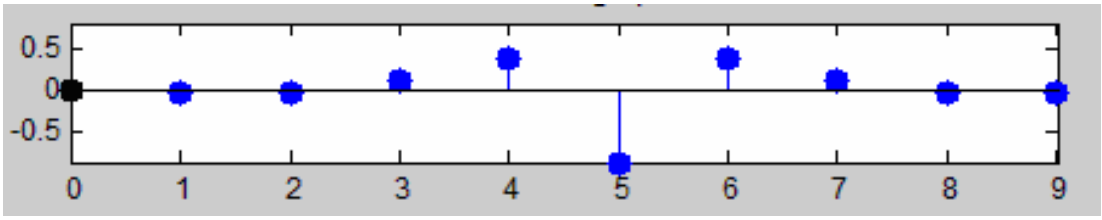
Декодирование изображений, сжатых по алгоритму *JPEG 2000*, осуществляется в порядке, обратном кодированию [2, 3]. Центральной процедурой декодирования является *обратное ДВП* (ОДВП). Его базовое выражение имеет вид

$$b_{j-1}[i] = \sum_k b_j[k] \times h_r[i+2k] + \sum_k d_j[k] \times g_r[i+2k], \qquad (1.31)$$

где $h_r[\bullet]$ и $g_r[\bullet]$ – отсчеты импульсных характеристик соответственно низкочастотного и высокочастотного цифровых вейвлет-фильтров *реконструкции* (рис. 1.22).



а



б

Рис. 1.22. Импульсные характеристики реконструирующих низкочастотного (*а*) и высокочастотного (*б*) цифровых фильтров вейвлета *CDF 9/7*

Операционная модель *одномерного* ОДВП представлена на рис. 1.23.

Двумерное ОДВП, применяемое алгоритмом *JPEG 2000*, отличается тем, что на каждом его этапе (рис. 1.24) выполняются процедуры, обратные реализуемым при двумерном ДВП.

Сжатие по алгоритму *JPEG 2000*, в зависимости от конкретного профиля, может осуществляться как с потерями, так и без потерь [2, 3].

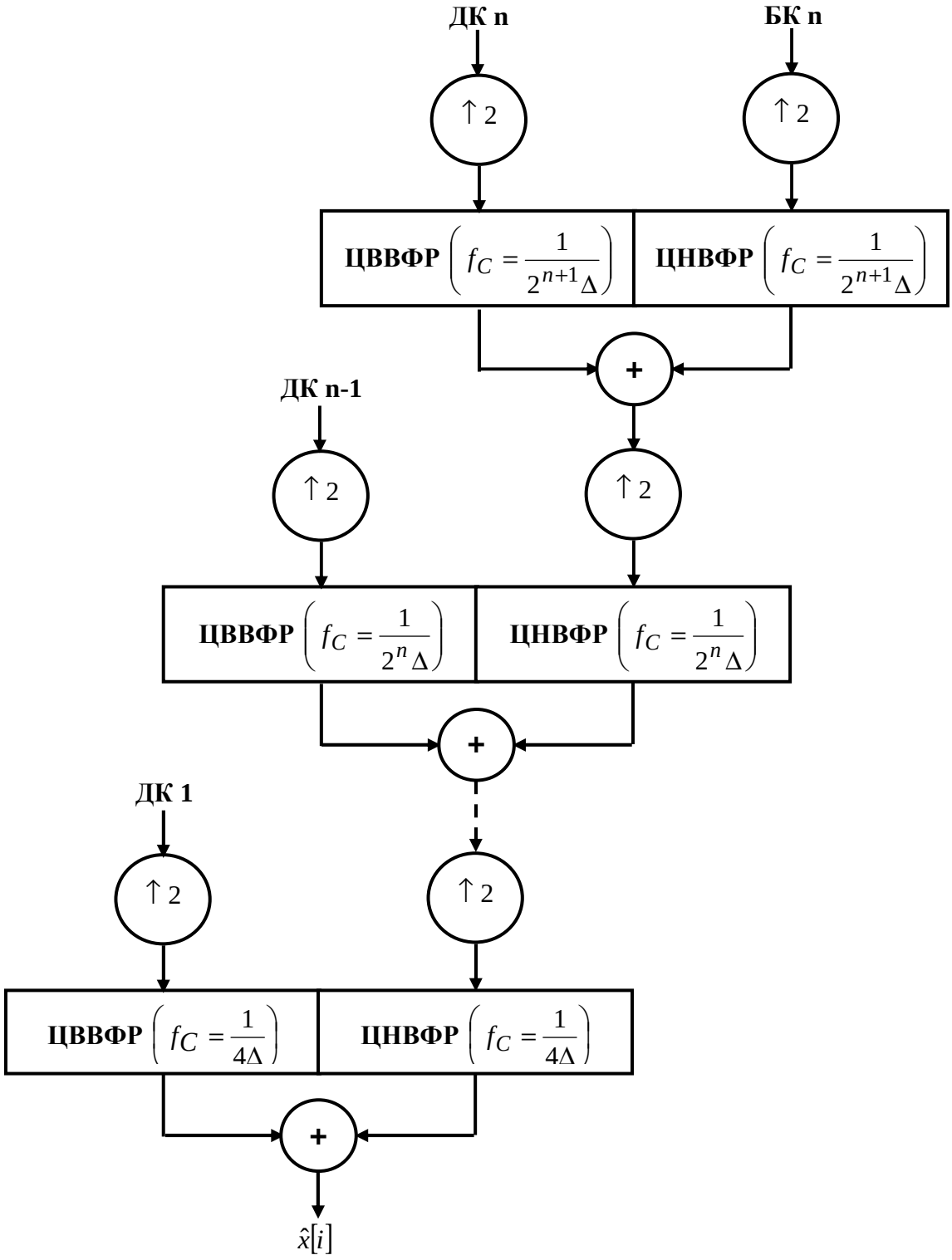


Рис. 1.23. Операционная модель *n*-уровневого одномерного ОДВП:
ЦНВФР, ЦВВФР – соответственно низкочастотный и высокочастотный цифровые вейвлет-фильтры реконструкции;
$\uparrow 2$ – оператор удвоения частоты дискретизации;
$\hat{x}[i]$ – восстановленная последовательность отсчетов

Фрактальное сжатие [3] основывается на представлении кодируемых изображений в целом или их фрагментов в виде *фракталов* – геометрических фигур, обладающих свойством самоподобия, т. е. составленных из нескольких частей, каждая из которых подобна всей фигуре целиком.

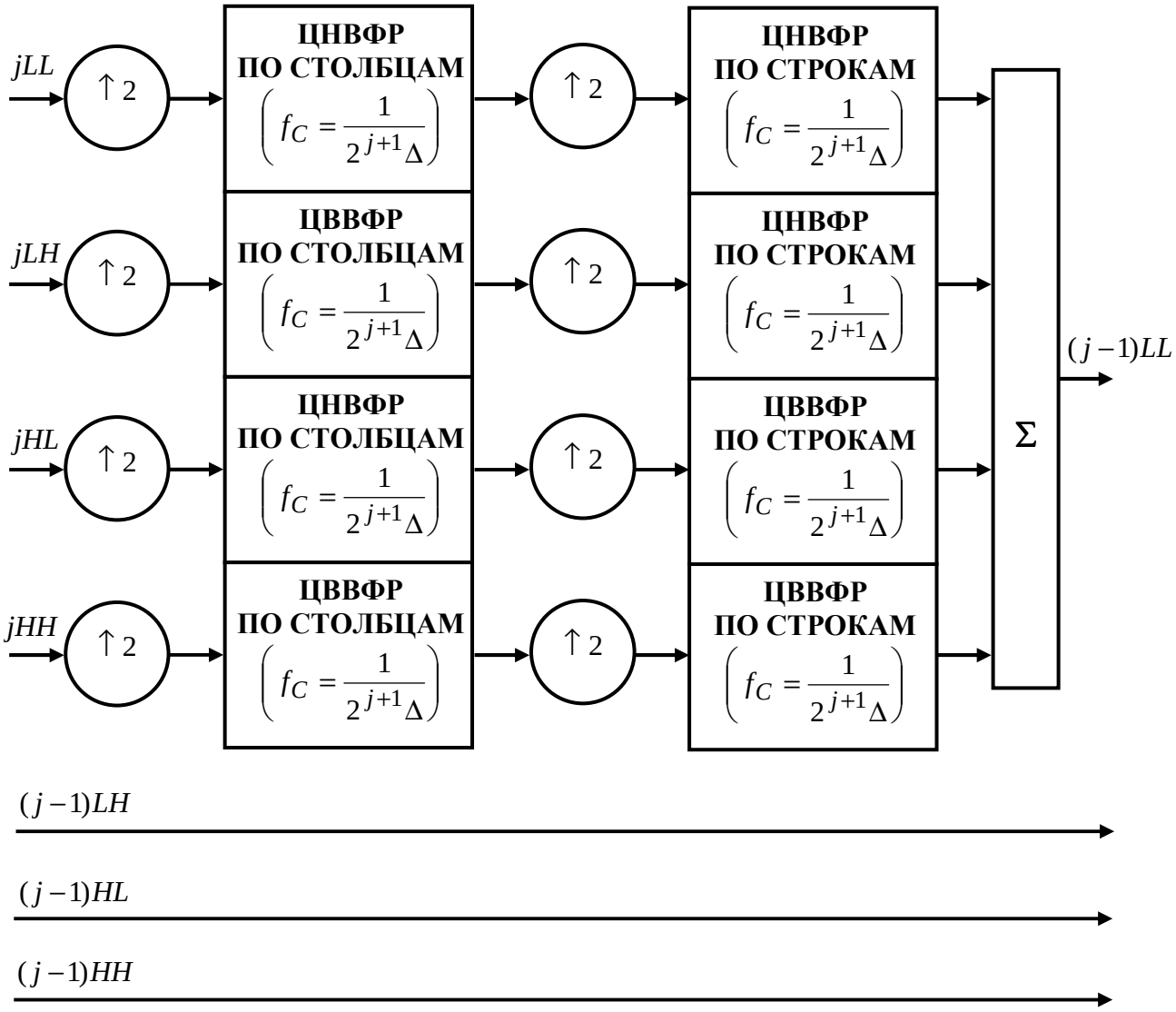


Рис. 1.24. Операционная модель одного этапа двумерного ОДВП, применяемого *JPEG 2000*

На рис. 1.25 в качестве примера представлено одно из простых семейств фракталов – кривые Коха 1-го, 2-го и 4-го порядков. Нетрудно заметить, что кривая более высокого порядка формируется из кривых более низких порядков.

Таким образом, любой фрактал при кодировании может быть представлен как комбинация кода порождающей его геометрической фигуры и параметров алгоритма построения фрактала, число которых, как правило, невелико. Например, построение кривой Коха (см. рис. 1.25)

однозначно описывается 24 коэффициентами. С другой стороны, многие элементы мультимедийных изображений, в первую очередь природные объекты (береговая линия, деревья и т. п.), с приемлемым визуальным качеством могут быть представлены в виде фракталов.

Фрактальное сжатие весьма эффективно при компактном кодировании такого класса изображений, как фотографии природных объектов (достижимые значения коэффициентов сжатия – порядка нескольких сотен). Для кодирования других категорий изображений данный способ распространения не получил.

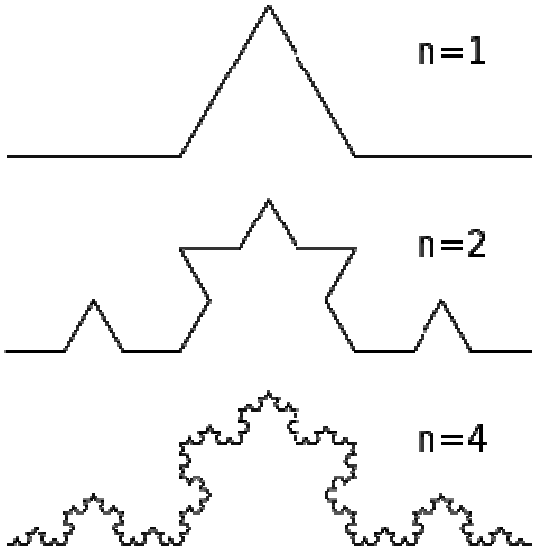


Рис. 1.25. Примеры фракталов (кривые Коха)

Сжатие на основе метода *векторной графики* [2, 3] заключается в представлении изображения в виде совокупности элементарных геометрических фигур (примитивов), описание каждой из которых состоит из набора кодов ее числовых параметров и качественных характеристик (атрибутов). Основными разновидностями примитивов, применяемыми в векторных форматах представления изображений, являются:

- отрезки прямых линий;
- ломаные линии;
- многоугольники;
- окружности, эллипсы и их фрагменты (дуги);
- специальные типы кривых, из которых наиболее широко распространены *кривые Безье*, а также образуемые ими замкнутые фигуры (в частности, *безигоны*, образуемые кривыми Безье);
- фракталы.

Например, такой примитив, как окружность, однозначно описывается следующим набором параметров и характеристик:

- координаты центра;
- значение радиуса;
- коды цвета и толщины контурной линии,

а для однозначного описания многоугольника достаточно указать:

- координаты каждого из углов;
- коды цвета и толщины контурной линии,
- код цвета заливки (при ее наличии);
- код типа штриховки (при ее наличии).

Таким образом, описание как окружности, так и многоугольника в векторной форме значительно (в десятки – сотни раз) более компактно, чем представление их же в растровой форме. То же верно и для других разновидностей примитивов. За счет этого достигается эффект сжатия изображений, представимых в векторной форме [т. е. для которых возможна декомпозиция на относительно небольшое количество геометрических примитивов, удовлетворяющее приведенному далее критерию (1.32)] по сравнению с их представлением в растровой форме. Кроме того, векторное представление изображения обладает дополнительными преимуществами, такими как:

- возможность масштабирования и других преобразований (перемещений, поворотов и т. п.) элементов изображения без потерь качества;
- возможность представления числовых параметров элементов изображения в *аппаратно-независимых единицах,* что обеспечивает возможность его воспроизведения широкой номенклатурой устройств отображения и регистрации;
- простота преобразования изображения из векторной формы в растровую.

Вместе с тем векторное представление обладает следующими недостатками:

- оно обеспечивает существенный выигрыш в объеме по сравнению с растровым только при выполнении условия

$$\sum_{i=1}^{I} N_i << B_{r\min}, \qquad (1.32)$$

где I – общее число примитивов, представляющих изображение;

N_i – число бит, необходимое для кодирования параметров и характеристик i-го примитива;

$B_{r\min}$ – минимальный достижимый общий объем (в битах) растрового представления того же изображения после сжатия;

• преобразование изображения из растровой формы в векторную характеризуется высокой сложностью.

Исходя из вышесказанного представление в векторной форме рационально для следующих категорий изображений:

- технической графической документации (чертежи, схемы, графики, диаграммы и т. п.), которая изначально представляет собой совокупность графических примитивов;

- изображений, искусственно синтезированных методами компьютерной (в том числе фрактальной) графики.

В то же время представление в векторной форме изображений естественных объектов, за исключением фрактального сжатия фотографий природных объектов с потерями, не рационально из-за невозможности обеспечения условия (1.32).

Характеристики наиболее распространенных форматов представления изображений при компактном кодировании приведены в табл. 1.13 [2, 3].

Таблица 1.13

Распространенные форматы представления неподвижных изображений при сжатии

Формат	Протокол / алгоритм сжатия	Способы кодирования, используемые протоколом/алгоритмом
1	2	3
Форматы, основанные на растровом представлении изображения		
*.bmp, *.dib	RLE (опционально)	Кодирование длин серий
*.tga, *tpic		
*.tiff, *.tif	PackBits	
	LZW	Словарное кодирование
	LZ77	
	Deflate	LZ77 → Кодирование Хаффмана
	JBIG	Арифметическое побитовое кодирование
*.tiff, *.tif	JPEG	Цветовая субдискретизация → ДКП → → Адаптивное квантование и упорядочивание результатов ДКП → → RLE → Кодирование Хаффмана
*.jpg	JPEG	

Окончание табл. 1.13

1	2	3
*.gif	LZW	Словарное кодирование
*.png	DPCM + Deflate	Дифференциальная модуляция с предсказанием, аналогичная применяемой *Lossless JPEG* → LZ77 → → Кодирование Хаффмана
*.jp2	JPEG 2000	Цветовая субдискретизация → ДВП → → Адаптивное квантование результатов ДВП → Арифметическое кодирование
Форматы, основанные на векторном представлении изображения		
*.svg	SVG	Представление изображения в виде совокупности масштабируемых графических примитивов следующих типов: - отрезки прямых линий; - дуги; - кривые Безье

1.5. Способы и алгоритмы эффективного кодирования видеоданных

Видеоданные представляют собой, по существу, последовательность взаимно коррелированных неподвижных растровых изображений (*кадров*), на каждом из которых представлено состояние некоторых объектов в определенный момент времени [2, 3]. При последовательном воспроизведении кадров с определенной скоростью (обычно 25 кадров в секунду) создается эффект движения представленных на них объектов. Как правило, видеоданные снабжаются массивом сопутствующих им аудиоданных (звукового сопровождения соответствующей видеозаписи).

Компактное кодирование видеоданных, таким образом, сводится к сжатию последовательностей неподвижных изображений (кадров) и сопутствующих им аудиоданных. При этом видеоданные как объект сжатия характеризуются следующими основными особенностями [2, 3]:

- незначительными взаимными различиями соседних кадров;
- возможностью достаточно точного математического моделирования изменений от кадра к кадру;

- допустимостью сжатия с потерями, обусловленной теми же факторами, что и при сжатии неподвижных изображений (в первую очередь, малозаметностью для глаза изменений яркости и цвета мелких деталей, а также большей информативностью яркостной составляющей, чем цветовых);

- необходимостью декомпрессии в реальном масштабе времени, т. е. непосредственно в процессе воспроизведения, и, как следствие, жесткими требованиями к времени декомпрессии;

- необходимостью аудиовизуальной синхронизации, т. е. синхронизации изображения и звукового сопровождения при декомпрессии;

- рядом специальных требований, основными из которых являются возможность воспроизведения видео с произвольного момента, а также масштабируемость изображения.

Ввиду перечисленных особенностей практически все известные алгоритмы сжатия видеоданных основываются на сочетании таких базовых процедур, как [2, 3]:

- выделение в потоке кадров некоторых *базовых кадров* (например, каждого 15-го или каждого 12-го) и их сжатие как отдельных (независимых) неподвижных изображений по алгоритмам, аналогичным описанным ранее *JPEG* и *JPEG 2000*;

- представление последовательности кадров, расположенных во времени между базовыми, методом ДИКМ, в виде попиксельных разностей каждого из данных кадров и кадра (кадров), служащих для него в качестве опорных.

Указанные базовые процедуры сопровождаются рядом дополнительных операций уменьшения избыточности видеоданных, наиболее распространенными из которых являются [2, 3]:

- цветовая субдискретизация;

- дополнительное компактное кодирование результатов ДИКМ способом *RLE*, статистическими способами, словарными способами или их сочетаниями;

- раздельное кодирование различных составляющих изображения (фон, движущиеся фигуры, заставки и т. п.) способами и алгоритмами, обеспечивающими максимальное сжатие каждой из составляющих, с формированием единого изображения при декодировании путем взаимного наложения указанных составляющих;

- замена ряда компонентов изображения их копиями («спрайтами»), искусственно синтезированными способами векторной графики и, как следствие, требующими значительно меньших затрат памяти для их описания, чем оригиналы;

- представление в векторной форме «искусственных» (не природных) элементов изображений, таких как строительные конструкции, мебель и т. п.;

- использование базовых кадров для быстрого поиска некоторого кадра (без необходимости декомпрессии всего видеофайла) и воспроизведения видео начиная с этого кадра.

Представление *способом ДИКМ* кадров, промежуточных по отношению к базовым, осуществляется на основании общего принципа [2, 3]:

• каждому из промежуточных кадров назначаются кадры, служащие для него в качестве опорных; в распространенных на практике алгоритмах таких кадров один или два (см. рис. 1.26 и пояснения к нему);

• каждый из промежуточных кадров представляется как множество массивов, описываемых следующим обобщенным выражением [2]:

$$\Delta_{ij}(x,y) = I_{ij}(x,y) - \hat{I}_{ij}(x,y) = I_{ij}(x,y) - \\ - F_p\left\{I_{R1ij}\left(x_{p1}(j,x,y), y_{p1}(j,x,y)\right), \quad V_j \times I_{R2ij}\left(x_{p2}(j,x,y), y_{p2}(j,x,y)\right)\right\}, \quad (1.33)$$

где $\Delta_{ij}(x,y)$ – код, на основе которого при декодировании восстанавливается интенсивность i-й цветовой составляющей пикселя с координатами (x,y) j-го кадра;

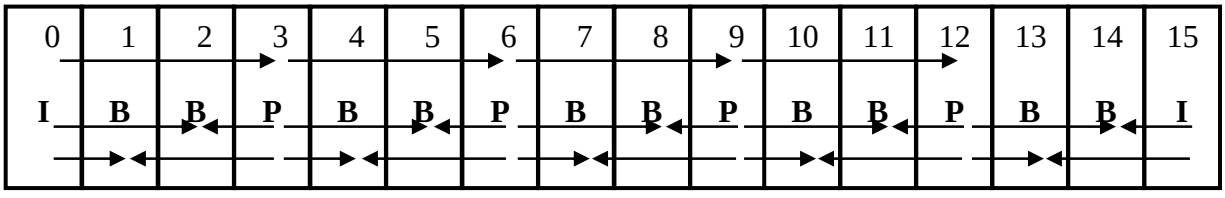


Рис. 1.26. Пример временной структуры подвергаемых сжатию кадров:
I – кадры, кодируемые независимо, без применения опорных (*Infra-Frames*);
P – кадры, кодируемые по обобщенному выражению (1.33), с использованием одного опорного кадра (*Predicted Frames*);
B – кадры, кодируемые по обобщенному выражению (1.33), с использованием двух опорных кадров (*Bidirectional-Predicted Frames*).
Стрелки указывают направление предсказания (от опорного кадра к кодируемому на его основе)

$I_{ij}(x,y)$ – интенсивность i-й цветовой составляющей пикселя с координатами (x,y) j-го кадра, подлежащего сжатию;

$\hat{I}_{ij}(x,y)$ – предсказанное значение указанной интенсивности;

$F_p\{\bullet\}$, $I_{R1ij}\left(x_{p1}(j,x,y), y_{p1}(j,x,y)\right)$ и $I_{R2ij}\left(x_{p2}(j,x,y), y_{p2}(j,x,y)\right)$ – соответственно функция предсказания и служащие ее аргументами интенсивности i-й цветовой составляющей пикселей с некоторыми координатами $\left(x_{p1}(j,x,y), y_{p1}(j,x,y)\right)$ и $\left(x_{p2}(j,x,y), y_{p2}(j,x,y)\right)$ [в общем случае не совпадающими с координатами (x,y)] 1-го и 2-го опорных кадров подвергаемого сжатию j-го кадра;

V_j – логическая переменная, равная нулю или единице соответственно при отсутствии и наличии 2-го опорного кадра.

Вид функции $F_p\{\bullet\}$ определяется конкретным алгоритмом сжатия. Простейшим вариантом данной функции является следующий:

$$\begin{cases} \hat{I}_{ij}(x,y) = I_{R1ij}(x,y) \text{ при } V_j = 0; \\ \hat{I}_{ij}(x,y) = 0{,}5 \times \left\{ I_{R1ij}(x,y) + I_{R2ij}(x,y) \right\} \text{ при } V_j = 1. \end{cases} \quad (1.34)$$

Предсказание по выражениям (1.34) с последующим статистическим, словарным или *RLE*-кодированием массива разностей $\Delta_{ij}(x,y)$ обеспечивают высокие (до нескольких сотен раз) коэффициенты сжатия видеоданных, содержащих неподвижные объекты. Однако изменение положения объекта на каком-либо кадре по отношению к предыдущему кадру даже на несколько пикселей приводит к резкому снижению «регулярности» массива $\Delta_{ij}(x,y)$, полученного по выражениям (1.34), т. е. повышению его энтропии и, как следствие, к существенному уменьшению эффективности сжатия. Лучшие результаты обеспечиваются при использовании функций предсказания, базирующихся на принципе *компенсации движения* (*Motion Compensation – MC*) и описываемых обобщенным выражением [2]:

$$\hat{I}_{ij}(b,x,y) = F_p\left\{ I_{R1ij}\left(B_{1bj}, x + dx_{1bj}, y + dy_{1bj}\right);\ \ V_j \times I_{R2ij}\left(B_{2bj}, x + dx_{2bj}, y + dy_{2bj}\right) \right\}, \quad (1.35)$$

где $\hat{I}_{ij}(b,x,y)$ – предсказанное значение интенсивности i-й цветовой составляющей пикселя с координатами (x,y) b-го *блока* j-го кадра (понятие блока применительно к сжатию видеоданных пояснено далее);

B_{1bj} и B_{2bj} – блоки соответственно 1-го и 2-го опорных кадров, наиболее сходных с j-м по некоторому критерию (например, по минимуму среднеквадратического отклонения интенсивностей пикселей [2]);

dx_{1bj}, dy_{1bj}, dx_{2bj} и dy_{2bj} – смещения блока b по осям x и y относительно блоков B_{1bj} и B_{2bj} соответственно;

$I_{R1ij}\left(B_{1bj}, x+dx_{1bj}, y+dy_{1bj}\right)$ и $I_{R2ij}\left(B_{2bj}, x+dx_{2bj}, y+dy_{2bj}\right)$ – интенсивности i-й цветовой составляющей пикселей с соответствующими координатами блоков B_{1bj} и B_{2bj}.

В качестве *блоков*, в зависимости от конкретного алгоритма сжатия, могут выступать:

- неперекрывающиеся фрагменты фиксированного размера (например, 16×16 пикселей), на которые разбиваются как промежуточные, так и опорные кадры;
- движущиеся объекты целиком.

Простейшим (однако достаточно распространенным на практике) вариантом функции предсказания при использовании принципа компенсации движения является следующий [ср. с выражениями (1.34)] [2]:

$$\begin{cases} \hat{I}_{ij}(b,x,y) = I_{R1ij}\left(B_{1bj}, x+dx_{1bj}, y+dy_{1bj}\right) \text{ при } V_j = 0; \\ \hat{I}_{ij}(b,x,y) = 0{,}5\times\left\{I_{R1ij}\left(B_{1bj}, x+dx_{1bj}, y+dy_{1bj}\right) + I_{R2ij}\left(B_{2bj}, x+dx_{2bj}, y+dy_{2bj}\right)\right\} \text{ при } V_j = 1. \end{cases} \quad (1.36)$$

Таким образом, при использовании принципа компенсации движения блоки, соответствующие изображению какого-либо объекта в некотором j-м кадре, представляются разностями интенсивностей их пикселей не с интенсивностями пикселей блоков опорных кадров с теми же координатами, как при предсказании по выражениям (1.34), а с интенсивностями пикселей блоков, соответствующих изображению *того же объекта* в опорных кадрах. За счет этого существенно снижается энтропия массива разностей $\Delta_{ij}(x,y)$ и, как следствие, повышается эффективность сжатия.

Результаты ДИКМ при использовании принципа компенсации движения содержат:

- массивы разностей $\Delta_{ij}(b,x,y)$, полученных в соответствии с базовым выражением (1.35);
- значения dx_{1bj}, dy_{1bj}, dx_{2bj} и dy_{2bj} для каждого из блоков, называемые *векторами движения* блоков.

В табл. 1.14 приведены наиболее распространенные форматы представления видеоданных при компактном кодировании, а также характеристики используемых данными форматами стандартов/протоколов сжатия [2, 3].

Таблица 1.14

Форматы представления видеоданных при сжатии

Формат	Протокол/ алгоритм сжатия	Основные способы кодирования, используемые протоколом/алгоритмом
*.m2ts, *.MTS *.mpg, *.mpeg	MPEG-2	Цветовая субдискретизация; ДВП кадров быстрого поиска; ДИКМ с компенсацией движения; адаптивное квантование; кодирование результатов преобразований/ДИКМ по Хаффману
*.avi,	XVID (MPEG-4 Part 2, Advanced Simple Profile)	Цветовая субдискретизация; ДВП кадров быстрого поиска; ДКП; ДИКМ с компенсацией движения; адаптивное квантование; арифметическое кодирование результатов преобразований/ДИКМ
*.mp4, *.m4v	H.264 (MPEG-4 Part 10, Advanced Video Coding)	Цветовая субдискретизация; ДВП кадров быстрого поиска; ДКП; ДИКМ с количеством опорных кадров до 16; компенсация движения с переменным размером блока; адаптивное квантование; контекстное бинарное арифметическое кодирование результатов преобразований/ДИКМ
	MPEG-4 Part 12, ISO base media file format	Аналогично H.264 + объектно-ориентированное кодирование
*.wmv	VC-1	Цветовая субдискретизация; ДКП с адаптацией размера блока; ДИКМ с компенсацией движения; адаптивное квантование; кодирование результатов преобразований/ДИКМ по Хаффману

Выводы по разделу 1

1. *Эффективное кодирование (сжатие)* сообщения состоит в его преобразовании в форму, требующую меньшего объема для представления сообщения, чем до преобразования, при возможности однозначного восстановления копии исходного сообщения из результата преобразования, с отклонениями от оригинала, не превышающими некоторые, наперед заданные.

2. Допустимые *отклонения* от оригинала копии сообщения, получаемой при декомпрессии, зависят от характера и назначения сообщения. По степени данных отклонений различают способы и алгоритмы сжатия данных *с потерями* и *без потерь*. Первые обеспечивают восстановление сообщения с точностью до бита, вторые – с отклонениями, определяемыми задаваемым пользователем качеством восстановления. Сжатию без потерь подлежат практически все текстовые данные, а также изображения научного и технического назначения. Сжатие с потерями допустимо для мультимедийных данных (звуковые файлы, изображения не научно-технического назначения, видео и т. п.).

3. В общем случае сжатие данных включает в себя следующие *процедуры*:

- предварительное преобразование подлежащего сжатию сообщения в форму, обеспечивающую более эффективное сжатие или более компактное представление, чем исходная форма представления сообщения;
- кодирование результатов преобразования.

В ряде частных случаев собственно эффективное кодирование обеспечивается только одной из перечисленных процедур.

4. Наиболее распространенными процедурами *предварительного преобразования* исходного сообщения при компактном кодировании являются:

- преобразование дискретного сообщения с целью снижения его энтропии; распространенным алгоритмом такого преобразования является *BWT* (см. п. 1.2.7);
- представление аудиоданных, изображения или видеоданных некоторой математической моделью, например, во временной области, в пространственной области или в виде спектра (см. пп. 1.3 – 1.5); при сжатии с потерями элементы модели или спектральные компоненты, несущественные для восприятия, не включаются в результаты преобразования.

5. *Кодирование* совокупности символов, представляющей сжимаемое сообщение (как при наличии, так и при отсутствии его предварительного преобразования), как правило, осуществляется одним из способов, обеспечивающих существенное снижение объема результата кодирования по сравнению со стандартными способами (например, с представлением текстового файла в виде последовательности *ASCII*-кодов его символов). Наиболее распространенными способами такого кодирования являются следующие:

- способ *кодирования длин серий* (*RLE*), состоящий в замене серии из N повторяющихся символов двумя кодами – повторяющегося символа и числа N; как правило, применяется для кодирования изображений в совокупности с другими способами кодирования (см. п. 1.2.1 и табл. 1.13);

- *статистическое (энтропийное) префиксное неравномерное кодирование*, заключающееся в представлении символов кодами неодинаковой разрядности, которая тем меньше, чем выше частота появления символа. Широко распространенным его способом является кодирование по Хаффману (см. п. 1.2.3), используемое при сжатии данных различных типов, как правило, в сочетании с другими способами кодирования и предварительного преобразования (см. табл. 1.10, 1.11, 1.13 и 1.14);

- *энтропийное арифметическое кодирование* (см. п. 1.2.4), состоящее в представлении кодируемой совокупности символов числом, являющимся функцией от порядка их следования и частоты встречаемости. Рассматривается большинством специалистов как наиболее перспективный способ кодирования. Как и кодирование по Хаффману, широко применяется при сжатии данных различных типов, как правило, в сочетании с другими способами кодирования и предварительного преобразования (см. табл. 1.10, 1.11, 1.13 и 1.14);

- *словарное кодирование*, состоящее в формировании собственного *словаря* сжимаемого сообщения, который включает в себя не только коды стандартных символов (например, букв, цифр и т. п.), но и коды, присваиваемые последовательностям таких символов, встречающимся в сжимаемом сообщении. Используется преимущественно при сжатии дискретных (например, текстовых) сообщений.

При этом фактически все современные алгоритмы энтропийного эффективного кодирования являются:

- *адаптивными*, учитывающими реальные статистические характеристики конкретного сообщения, оцениваемые непосредственно в процессе его сжатия (см. табл. 1.1 – 1.3);

- использующими в той или иной форме *контекстное моделирование* сжимаемого сообщения (см. п. 1.2.5).

6. Практически все современные стандарты (протоколы) сжатия данных основываются на *сочетании* нескольких алгоритмов преобразования и кодирования (см. табл. 1.10, 1.11, 1.13 и 1.14), что позволяет повысить эффективность сжатия.

Вопросы для самопроверки

1. Дайте определение эффективного кодирования (сжатия) данных. Поясните смысл понятий «сжатие без потерь» и «сжатие с потерями».

2. Приведите (с обоснованием) примеры данных, сжатие которых необходимо осуществлять без потерь, и данных, сжатие которых допустимо осуществлять с потерями.

3. Коэффициент сжатия (без потерь) какого из нижеприведенных текстовых сообщений потенциально окажется максимальным, а какого – минимальным при прочих равных условиях?

- статья в научно-техническом журнале
- статья в общественно-политическом издании
- фрагмент литературного произведения
- текст компьютерной программы

Ответ обоснуйте.

4. Установлено, что априорная ИЭ русского литературного текста в прозе равна примерно 1,5 бит/символ. Оцените объем (в байтах) текстового файла в ASCII-кодировке, содержащего фрагмент литературного произведения в прозе на русском языке, после его сжатия без потерь идеальным архиватором, если объем файла до сжатия составляет 480 Кбайт.

5. Обоснуйте отнесение кодирования по Хаффману к способам статистического (энтропийного) префиксного неравномерного кодирования.

6. Определите минимальную разрядность кода символа после кодирования по Хаффману алфавита, состоящего из 32 символов, вероятность одного из которых равна суммарной вероятности остальных.

7. Опишите принцип арифметического кодирования. Обоснуйте отнесение его к энтропийным способам эффективного кодирования.

8. Опишите принцип словарного эффективного кодирования. Объясните, по какой причине коэффициент сжатия сообщения словарными способами тем выше, чем больше его объем.

9. Раскройте принципы неадаптивного, полуадаптивного и неадаптивного энтропийного сжатия.

10. Опишите принцип контекстного моделирования с предсказанием. Объясните эффект повышения коэффициента сжатия алгоритмами группы PPM с ростом объема сжимаемого сообщения.

11. Обоснуйте универсальность способа CTW.

12. Объясните эффект снижения энтропии сообщения после преобразования Барроуза – Уилера.

13. За счет чего достигается эффект сжатия аудиоданных при ДИКМ? В чем состоит принцип АДИКМ? Обоснуйте ее преимущества по сравнению с неадаптивной ДИКМ.

14. Опишите принцип моделирования и сжатия речи способом LPC.

15. По какой причине ДКП, будучи широко применяемым при сжатии аудиоданных и изображений, практически не используется для спектрального анализа?

16. Коэффициент сжатия какого из нижеприведенных звуковых файлов потенциально окажется максимальным, а какого – минимальным при прочих равных условиях?

- запись концерта медленной классической музыки
- запись оперы
- запись рок-концерта

Ответ обоснуйте.

17. Поясните смысл процедуры психоакустического анализа при сжатии аудиоданных.

18. Во сколько раз при прочих равных условиях уменьшится объем файла изображения при цветовой субдискретизации с $a = b = 2$ по каждой из цветоразностных составляющих?

19. Опишите принцип сжатия по алгоритму *JPEG*.

20. Объясните смысл возрастания значений элементов Q_{ij} матрицы квантования [см. выражение (1.26)] с ростом значений i и j. Должны ли значения Q_{ij} быть увеличены или уменьшены для повышения коэффициента сжатия? А для снижения потерь качества при сжатии? Ответ обоснуйте.

21. Опишите принцип сжатия по алгоритму *JPEG 2000*.

22. Будет ли достигнуто существенное снижение объема изображения при его векторном представлении по сравнению с растровым, если изображение является: машиностроительным чертежом; орнаментом; лесным пейзажем?

Ответ обоснуйте.

23. Поясните смысл компенсации движения при сжатии видеоданных.

24. При прочих равных условиях коэффициент сжатия какого из фрагментов футбольного матча, снятого дальним или ближним планом, окажется больше? Ответ обоснуйте.

2. ОСНОВЫ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

2.1. Общие положения

Помехоустойчивым кодированием называют представление цифровых данных в форме, при которой отдельные элементы сообщения (биты или их группы) связаны определенной зависимостью, позволяющей при ее нарушении обнаружить и/или исправить ошибки в сообщении. Все разновидности помехоустойчивых кодов характеризуются общими свойствами [6 – 8]:

- большей разрядностью кодовой комбинации, получаемой в результате кодирования, по сравнению с комбинацией, подвергаемой кодированию, т. е. наличием *избыточности*, без которой обнаружение и исправление ошибок *невозможно в принципе*;

- принадлежностью всех разрядов кода к некоторому *конечному полю* и их формированием посредством математических операций в данном поле (см. п. 2.2); непринадлежность разрядов принятого кода соответствующему полю или/и их несоответствие правилам выполнения операций в нем являются одними из признаков наличия ошибок;

- наличием *разрешенных* (возможных только в отсутствие ошибок) и *запрещенных* (возможных только при наличии ошибок) кодовых комбинаций;

- ограниченным числом обнаруживаемых или/и исправляемых ошибок в сообщении, зависящим от конкретной разновидности и параметров кода.

Обобщенная классификация наиболее распространенных разновидностей помехоустойчивых кодов представлена на рис. 2.1. Более полная и подробная классификация приведена, например, в [8].

Помехоустойчивые коды разделяются на два основных класса: *блочные* и *сверточные* [6 – 8]. Их базовое различие состоит в следующем. Результаты *блочного* помехоустойчивого кодирования представляют собой последовательность n-разрядных *блоков* (слов), каждый из которых формируется из k-разрядного слова ($n > k$) исходной (кодируемой) последовательности независимо от других блоков. *Сверточное* кодирование осуществляется путем преобразования каждого слова разрядностью K кодируемых данных в N-разрядное слово ($N > K$), каждый из разрядов которого формируется как некоторая

функция не только от соответствующего ему K-разрядного слова входной последовательности, но и от предыдущих слов как входной, так и (в общем случае) выходной последовательности.

Сверточные коды, характеризующиеся зависимостью выходных слов только от входных, называют *нерекурсивными*. В свою очередь, сверточные коды, выходные слова которых зависят как от предшествующих им выходных, так и от входных слов, относят к *рекурсивным* (рис. 2.1).

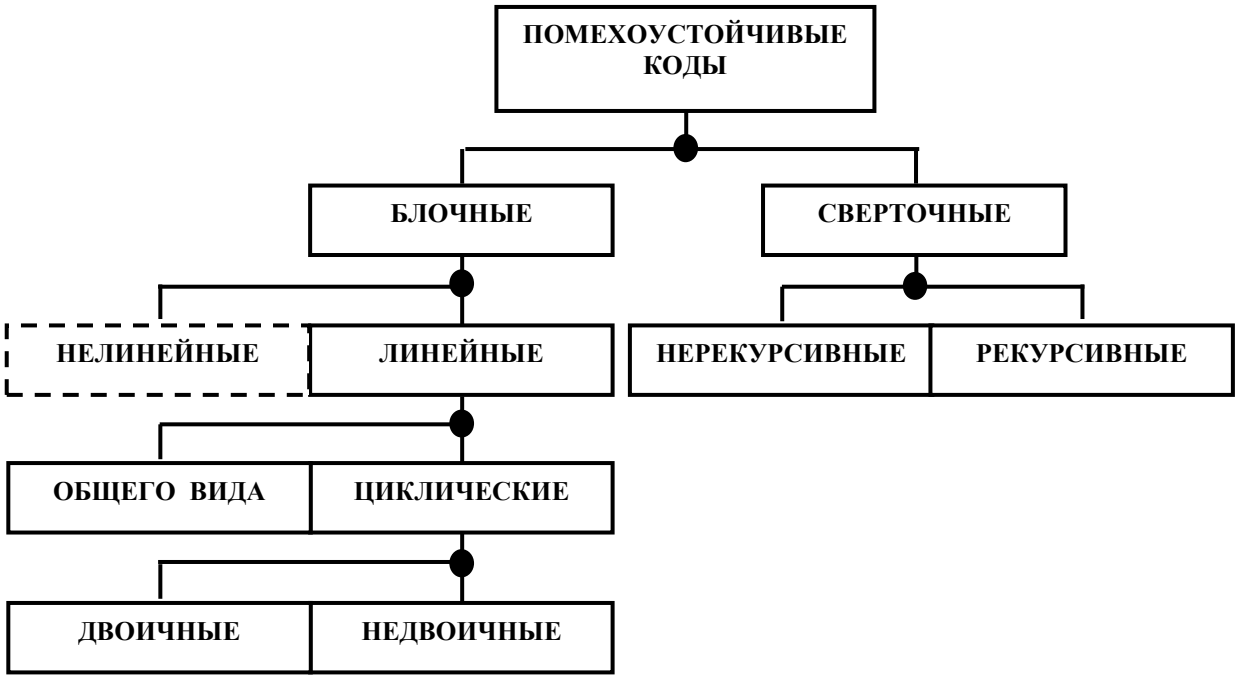


Рис. 2.1. Классификация основных разновидностей помехоустойчивых кодов

При этом как соотношение между параметрами n и k (N и K), так и алгоритмы формирования кодовых слов определяются конкретной разновидностью помехоустойчивого кода.

На практике применяются как блочные, так и сверточные коды, причем как по отдельности, так и совместно (см. п. 2.6).

Блочные коды, в свою очередь, разделяются на *линейные* и *нелинейные*. Блочные *линейные* помехоустойчивые коды (БЛПК) в настоящее время являются основной разновидностью блочных помехоустойчивых кодов. Они характеризуются следующими свойствами:

- поразрядная сумма в *конечном поле* (см. п. 2.2), к которому принадлежит БЛПК (например, в двоичном поле – сумма по модулю 2), любых двух разрешенных (не содержащих ошибок) слов БЛПК представляет собой также разрешенное слово того же БЛПК;

- произведение любого разрешенного слова БЛПК в конечном поле, к которому он принадлежит, на любой из элементов данного поля (например, в двоичном поле – на нуль или на единицу) представляет собой также разрешенное слово того же БЛПК.

К *нелинейным* относят блочные коды, не обладающие вышеуказанными свойствами. До настоящего времени они не получили сколько-нибудь значительного распространения на практике.

Наиболее распространенными разновидностями БЛПК являются БЛПК *общего вида* и *циклические*. Первые получили свое название от того, что наиболее удобной или единственной формой математического описания алгоритмов их формирования является *матричная* (см. п. 2.3), универсальная для всех БЛПК. Алгоритмы формирования *циклических* БЛПК, в принципе, тоже могут описываться в матричной форме. Однако существует более удобная (в том числе для аппаратной или программной реализации) форма описания указанных алгоритмов – на основе *порождающего полинома* (см. п. 2.4). По множеству чисел, к которому принадлежат его коэффициенты, различают *двоичные* и *недвоичные* циклические БЛПК (см. пп. 2.4.3 и 2.4.4 соответственно). Коэффициенты порождающих полиномов двоичных циклических БЛПК могут принимать только значения, равные нулю или единице, в то время как недвоичных – целочисленные значения от нуля до некоторого максимального, отличного от единицы и определяемого конкретным протоколом кодирования. Естественно, несмотря на название «недвоичный циклический БЛПК», данные коэффициенты в процессах кодирования и декодирования представляются в двоичной системе счисления.

2.2. Математические основы помехоустойчивого кодирования

С математической точки зрения помехоустойчивое кодирование базируется на понятии *конечного поля*. Рассмотрим вкратце базовые положения теории конечных полей [8].

Конечным полем, известным также под названием *поля Галуа* (*Galois Field*) и обозначаемым GF, называется конечное множество элементов, *замкнутое* по отношению к двум заданным в нем математическим операциям – *сложения* и *умножения* элементов. Под *замкнутостью* понимается тот факт, что результаты названных операций принадлежат тому же множеству элементов.

В практике помехоустойчивого кодирования получили распространение:

- *простые* конечные поля, содержащие p целочисленных элементов от нуля до $p-1$ (где p – простое число) и обозначаемые $GF(p)$;

- *расширенные* конечные поля, обозначаемые $GF\left(p^m\right)$ (где m – целое положительное число) и содержащие p^m-1 полиномов степени, меньшей или равной $m-1$, коэффициенты которых принадлежат полю $GF(p)$.

Необходимым условием *замкнутости* поля по отношению к операциям сложения и умножения (см. п. 2.1) является выполнение данных операций по следующим *общим правилам*:

• *Правило 1.* Сложение и умножение в *простом* поле $GF(p)$ его элементов (целых чисел от нуля до $p-1$) выполняются *по модулю* p, т. е. результатом каждой из них служит остаток от деления на p соответственно суммы или произведения, полученных по общепринятым математическим правилам.

В частности, в распространенном на практике поле $GF(2)$ указанные операции выполняются *по модулю 2*. Операция суммирования по модулю 2 обозначается оператором $\oplus$, а, например, $1\oplus 1=0$. В общем случае для указания того, что некоторая математическая операция выполняется по модулю p, применяется *условное обозначение* $(\mathrm{mod}\, p)$. Так, запись $ab=x(\mathrm{mod}\, p)$ означает, что произведение ab равно x по модулю p, т. е. остаток от деления данного произведения на p равен x. К примеру, $3\times 5=1(\mathrm{mod}\, 7)$, так как произведение 3×5 равно 15, а остаток от его деления на 7 равен единице.

• *Правило 2.* Сложение и умножение в *расширенном* поле $GF\left(p^m\right)$ его элементов (полиномов) выполняются *по модулю* $G(x)$, где $G(x)$ – *порождающий полином* соответствующего поля. Он представляет собой многочлен степени m с коэффициентами, принадлежащими полю $GF(p)$, *неприводимый* в данном поле, т. е. не разлагаемый на многочлены степеней, меньших m, с коэффициентами, принадлежащими полю $GF(p)$. Результатом сложения или умножения полиномов в поле $GF\left(p^m\right)$ служит *остаток* от деления суммы или произведения соответствующих полиномов на порождающий полином поля. При этом указанные сумма или произведение в целом вычисляются

по общепринятым правилам сложения или умножения полиномов, за исключением того, что суммирование и умножение их *коэффициентов* осуществляются *по модулю* p.

Например, произведение полиномов x^4+x+1 и $x^4+x^3+x^2+x+1$, коэффициенты которых принадлежат полю $GF(2)$, равно:

$$\left(x^4+x+1\right)\times\left(x^4+x^3+x^2+x+1\right)=$$
$$=x^8+x^7+x^6+x^5+x^4+x^5+x^4+x^3+x^2+x+x^4+x^3+x^2+x+1=$$
$$=x^8+x^7+x^6+(1\oplus 1)x^5+(1\oplus 1\oplus 1)x^4+(1\oplus 1)x^3+(1\oplus 1)x^2+(1\oplus 1)x+1=$$
$$=x^8+x^7+x^6+x^4+1,$$

а в поле $GF(929)$, используемом при формировании помехоустойчивых штрих-кодов (см. п. 2.4.4), справедливо следующее равенство:

$$(x-3)\times\left(x-3^2\right)\times\left(x-3^3\right)\times\left(x-3^4\right)=x^4+809x^3+723x^2+568x+522.$$

В самом деле, нетрудно убедиться, что, например, остаток от деления произведения $3\times 3^2\times 3^3\times 3^4$ на 929 равен 522, т.е. $3\times 3^2\times 3^3\times 3^4=522(\mathrm{mod}\,929)$.

В свою очередь, *деление* полиномов в конечных полях [8], в целом, осуществляется по «классическим» правилам [9], за исключением того, что:

- при модификации полинома-делителя путем его умножения на ax^k (где a – некоторый элемент поля $GF(p)$, а k – целое положительное число) умножение его коэффициентов на a производится *по модулю* p;

- *вычитание* коэффициентов при одноименных членах текущего остатка и модифицированного делителя осуществляется в поле $GF(p)$ на основании следующей аксиомы.

Аксиома 1. Для любого элемента a конечного поля существует *обратный* ему элемент *по сложению* $(-a)$, также принадлежащий данному полю, такой, что *по правилам сложения в данном поле* $a+(-a)=0$, т. е.

$$\begin{cases} a+(-a)=0(\mathrm{mod}\ p) \text{ в простом поле } GF(p); \\ P(x)+(-P(x))=0(\mathrm{mod}\,G(x)) \text{ в расширенном поле } GF\left(p^m\right), \end{cases} \quad (2.1)$$

где $P(x)$ – полином, являющийся элементом поля $GF\left(p^m\right)$.

Наличие обратного элемента по сложению позволяет *вычитать* элементы конечного поля по следующим правилам:

- в *простом* поле $GF(p)$

$$b - a = [b + (-a)] (\mathrm{mod}\ p); \qquad (2.2)$$

- в *расширенном* поле $GF\left(p^m\right)$

$$P_1(x) - P_2(x) = [P_1(x) + (-P_2(x))] (\mathrm{mod}\, G(x)), \qquad (2.3)$$

где $P_1(x)$ и $P_2(x)$ – полиномы, являющиеся элементами поля $GF\left(p^m\right)$.

Суммирование коэффициентов полиномов в выражениях (2.1) и (2.3) осуществляется по модулю p.

Результатами вычитания по выражениям (2.2) и (2.3) являются элементы того же поля.

В частности, в поле $GF(2)$ $-a = a$, поскольку $0 \oplus 0 = 0$, а $1 \oplus 1 = 0$. При этом операция вычитания в данном поле равносильна сложению по модулю 2. В общем случае в поле $GF(p)$ справедливо равенство

$$-a = p - a, \qquad (2.4)$$

так как сумма значений a и $p - a$ равна p и, следовательно, остаток от ее деления на p равен нулю, т. е. $a + p - a = 0 (\mathrm{mod}\ p)$. Так, в поле $GF(929)$

$$-569 = 929 - 569 = 360.$$

В свою очередь, например, в любом поле $GF\left(2^m\right)$ $P(x) = -P(x)$ благодаря суммированию коэффициентов полиномов по модулю 2 при их сложении в полях данного класса:

$$\left(x^4 + x + 1\right) + \left(x^4 + x + 1\right) = (1 \oplus 1)x^4 + (1 \oplus 1)x + (1 \oplus 1) = 0.$$

Вычитание коэффициентов полиномов по выражениям (2.2) и (2.4), а также сложение и умножение данных коэффициентов по правилу 1 позволяют выполнять деление полиномов в конечном поле, обеспечивающее принадлежность остатка от деления к тому же полю. Рис. 2.2 поясняет процедуру деления полиномов в поле $GF\left(2^m\right)$ и нахождения остатка от деления, а рис. 2.3 – аналогичную процедуру в поле $GF(p)$ при $p > 2$.

Рис. 2.2 не требует особых комментариев; вычитание здесь заменено эквивалентным ему в поле $GF(2)$ сложением по модулю 2.

$$
\begin{array}{ll}
 & x^{12} \\
\oplus & x^{12}+x^9+x^8=x^8\left(x^4+x+1\right) \\
\hline
 & x^9+x^8 \\
\oplus & x^9+x^6+x^5=x^5\left(x^4+x+1\right) \\
\hline
 & x^8+x^6+x^5 \\
\oplus & x^8+x^5+x^4=x^4\left(x^4+x+1\right) \\
\hline
 & x^6+x^4 \\
\oplus & x^6+x^3+x^2=x^2\left(x^4+x+1\right) \\
\hline
 & x^4+x^3+x^2 \\
\oplus & x^4+x+1=1\times\left(x^4+x+1\right) \\
\hline
 & x^3+x^2+x+1
\end{array}
\qquad
\begin{array}{|l}
x^4+x+1 \\
\hline
x^8+x^5+x^4+x^2+1
\end{array}
$$

Делимое: x^{12}

Делитель: x^4+x+1

Частное: $x^8+x^5+x^4+x^2+1$

Остаток: x^3+x^2+x+1

Рис. 2.2. Пример деления полиномов в поле $GF\left(2^m\right)$ и нахождения остатка от деления

$$
\begin{array}{ll}
 & 3x^6+2x^5+x^4 \\
- & 3x^6+569x^5+311x^4+775x^3+637x^2 \\
\hline
 & 362x^5+619x^4+154x^3+292x^2 \\
- & 362x^5+223x^4+677x^3+307x^2+377x \\
\hline
 & 396x^4+406x^3+914x^2+552x \\
- & 396x^4+788x^3+176x^2+110x+474 \\
\hline
 & 547x^3+738x^2+442x+455
\end{array}
\qquad
\begin{array}{|l}
x^4+809x^3+723x^2+568x+522 \\
\hline
3x^2+362x+396
\end{array}
$$

Делимое: $3x^6+2x^5+x^4$

Делитель: $x^4+809x^3+723x^2+568x+522$

Частное: $3x^2+362x+396$

Остаток: $547x^3+738x^2+442x+455$

Рис. 2.3. Пример деления полиномов в поле $GF(p)$ при $p>2$ ($p=929$) и нахождения остатка от деления

В более подробных пояснениях нуждается рис. 2.3. Процедура деления полиномов в поле $GF(p)$ при $p > 2$ основывается на выполнении двух базовых действий:

- модификации делителя путем его умножения на ax^k;
- вычитания модифицированного делителя из текущего остатка.

Например, полином $3x^6 + 569x^5 + 311x^4 + 775x^3 + 637x^2$ получен умножением в поле $GF(929)$ делителя $x^4 + 809x^3 + 723x^2 + 568x + 522$ на $3x^2$ (см. рис. 2.3). При этом:

$$569 = (809 \times 3)\,(\mathrm{mod}\,929);$$
$$311 = (723 \times 3)\,(\mathrm{mod}\,929);$$
$$775 = (568 \times 3)\,(\mathrm{mod}\,929);$$
$$637 = (522 \times 3)\,(\mathrm{mod}\,929).$$

В свою очередь, полином $362x^5 + 619x^4 + 154x^3 + 292x^2$ получен вычитанием в поле $GF(929)$ полинома $3x^6 + 569x^5 + 311x^4 + 775x^3 + 637x^2$ из делимого, равного $3x^6 + 2x^5 + x^4$. При этом [см. выражения (2.2) и (2.4)]:

$$362 = (2 + 929 - 569)\,(\mathrm{mod}\,929);$$
$$619 = (1 + 929 - 311)\,(\mathrm{mod}\,929);$$
$$154 = (0 + 929 - 775)\,(\mathrm{mod}\,929);$$
$$292 = (0 + 929 - 637)\,(\mathrm{mod}\,929).$$

Из вышеприведенных пояснений достаточно просто понять последующий ход процедуры деления.

В дальнейшем под *математическими операциями в некотором поле* будут подразумеваться операции, выполняемые по вышеприведенным правилам.

Кроме аксиомы 1, для всех конечных полей также справедливы следующие аксиомы:

Аксиома 2. Для любого ненулевого элемента a конечного поля существует *обратный* ему элемент *по умножению* (a^{-1}), также принадлежащий данному полю, такой, что *в данном поле* $a \times a^{-1} = 1$, т. е.:

$$\begin{cases} a \times a^{-1} = 1(\mathrm{mod}\ p) \text{ в простом поле } GF(p); \\ P(x) \times P^{-1}(x) = 1(\mathrm{mod}\, G(x)) \text{ в расширенном поле } GF\left(p^m\right). \end{cases} \quad (2.5)$$

Например, в поле $GF(929)$ $27^{-1} = 757$, так как $27 \times 757 = 20439$, а остаток от деления числа 20439 на 929 равен единице, т.е. $27 \times 757 = 1(\mathrm{mod}\,929)$.

Наличие обратного элемента по умножению в общем случае позволяет делить элементы конечного поля по следующим выражениям:

- в *простом* поле $GF(p)$

$$b/a = \left\lfloor b \times a^{-1} \right\rfloor (\mathrm{mod}\ p); \tag{2.6}$$

- в *расширенном* поле $GF\left(p^m\right)$

$$P_1(x)/P_2(x) = \left\lfloor P_1(x) \times P_2^{-1}(x) \right\rfloor \left(\mathrm{mod}\, G(x)\right). \tag{2.7}$$

Операции умножения и сложения коэффициентов полиномов в выражениях (2.5) и (2.7) осуществляются по модулю p.

Результатами деления по выражениям (2.5) и (2.7) являются элементы того же поля.

Аксиома 3. Множество элементов любого конечного поля содержит *нулевой элемент*, такой, что для любого элемента a данного поля $a+0=a$ *в соответствующем поле.*

Аксиома 4. Множество элементов любого конечного поля содержит *единичный элемент* (называемый также *мультипликативной единицей*), такой, что для любого элемента a данного поля $a \times 1 = a$ *в соответствующем поле.*

Очевидно, для любого конечного поля нулевым элементом является «обычный» нуль, а единичным – «обычная» единица.

Аксиома 5. Для операций сложения и умножения *в конечном поле* справедливы следующие свойства:

- *коммутативности*: $a+b=b+a$ и $a \times b = b \times a$;
- *ассоциативности*: $a+(b+c)=(a+b)+c$ и $a \times (b \times c) = (a \times b) \times c$;
- *дистрибутивности*: $a \times (b+c) = a \times b + a \times c$.

В теории и простых, и расширенных конечных полей, как и в «классической» алгебре, существует понятие *корня полинома*. Последний определяется как число, принадлежащее полю $GF(p)$, или как принадлежащий полю $GF\left(p^m\right)$ многочлен с принадлежащими полю $GF\left(p\right)$ коэффициентами, при подстановке которого в полином в качестве его переменной последний обращается в нуль *в соответствующем поле.*

Например, одночлен x^3 является корнем полинома $P(x) = x^4 + x^3 + x^2 + x + 1$ в расширенном поле $GF\left(2^4\right)$ с порождающим полиномом $G(x) = x^4 + x + 1$. Посредством процедуры, аналогичной представленной на рис. 2.2, нетрудно убедиться, что

$P(x^3) = x^{12} + x^9 + x^6 + x^3 + 1 = 0\left(\mathrm{mod}\left(x^4 + x + 1\right)\right)$, т. е. остаток от деления полинома $x^{12} + x^9 + x^6 + x^3 + 1$ на полином $x^4 + x + 1$ равен нулю.

Все ненулевые элементы простого конечного поля $GF(p)$ являются целыми степенями по модулю p от 0-й до $(p-2)$-й, а расширенного конечного поля $GF\left(p^m\right)$ – целыми степенями по модулю $G(x)$ от 0-й до $(p^m - 2)$-й некоторого *примитивного элемента* α соответствующего поля. Примитивным элементом простого конечного поля $GF(p)$ по определению является элемент, удовлетворяющий следующим условиям:

$$\begin{cases} \alpha^{p-1} = 1(\mathrm{mod}\ p); \\ \alpha^i \neq 1(\mathrm{mod}\ p) \text{ при } 0 < i < p-1, \end{cases} \tag{2.8}$$

а расширенного конечного поля $GF\left(p^m\right)$ – условиям:

$$\begin{cases} \alpha^{p^m-1} = 1(\mathrm{mod}\, G(x)); \\ \alpha^i \neq 1(\mathrm{mod}\, G(x)) \text{ при } 0 < i < p^m - 1. \end{cases} \tag{2.9}$$

В общем случае в конечном поле может существовать несколько примитивных элементов.

Из условия (2.8) следует, что примитивный элемент простого конечного поля $GF(p)$ обладает следующим свойством:

$$\alpha^{i+n(p-1)}(\mathrm{mod}\ p) = \alpha^i(\mathrm{mod}\ p) \tag{2.10}$$

при любом целом неотрицательном n и любом целом i, удовлетворяющем неравенству $0 \le i \le p-2$.

Из условия (2.9) следует, что примитивный элемент расширенного конечного поля $GF\left(p^m\right)$ удовлетворяет равенству

$$\alpha^{i+n\left(p^m-1\right)}(\mathrm{mod}\, G(x)) = \alpha^i(\mathrm{mod}\, G(x)) \tag{2.11}$$

при любом целом неотрицательном n и любом целом i, удовлетворяющем неравенству $0 \le i \le p^m - 2$.

Например, примитивным элементом поля $GF(2)$ является число 1, а поля $GF(5)$ – число 3, поскольку $3^{5-1} = 81 = 1(\mathrm{mod}\, 5)$. При этом все ненулевые элементы поля $GF(5)$, которыми являются числа 1, 2, 3 и 4, действительно являются целыми степенями числа 3 по модулю 5,

находящимися в диапазоне от нуля до $5-2=3$. Действительно, $3^0=1=1(\mathrm{mod}\,5)$, $3^1=3=3(\mathrm{mod}\,5)$, $3^2=9=4(\mathrm{mod}\,5)$, а $3^3=27=2(\mathrm{mod}\,5)$ (см. правило 1).

С другой стороны, одночлен x является примитивным элементом любого из расширенных полей, принадлежащих к наиболее распространенному на практике классу $GF(2^m)$ с *примитивным* порождающим полиномом. Им по определению является неприводимый двоичный полином, обладающий следующими свойствами:

- примитивный элемент порождаемого им поля является его корнем;
- в порождаемом им поле для каждого ненулевого элемента существует неприводимый полином с порядком, меньшим или равным m, корнем которого является соответствующий элемент.

Примитивные двоичные полиномы затабулированы. Полиномы с порядками от 1 до 8 представлены в табл. 2.1, а более высоких порядков – например, в [8].

Таблица 2.1

Примитивные двоичные многочлены порядков от 1 до 8

Порядок	Примитивный двоичный полином
1	$x+1$
2	x^2+x+1
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^3+x^2+1$

В качестве примера в табл. 2.2 представлены ненулевые элементы β_i расширенного конечного поля $GF(2^4)$ с порождающим полиномом $G(x)=x^4+x+1$ [8]. Данные элементы получены как остатки от деления на $G(x)$ (см. рис. 2.2) примитивного элемента поля $\alpha=x$, возведенного в степень i, где i – номер (индекс) элемента.

Важным свойством ненулевых элементов конечного поля $GF(p^m)$ является следующее:

$$\beta_{i+n\times(p^m-1)}=\beta_i \qquad (2.12)$$

для любого целого положительного n (см. табл. 2.2).

Таблица 2.2

Ненулевые элементы расширенного конечного поля $GF\left(2^4\right)$ *с порождающим полиномом* $G(x) = x^4 + x + 1$

Элемент $\beta_i = \alpha^i \left(\mathrm{mod}\, G(x)\right);\ \alpha = x$	Полиномиальное представление элемента
β_0	1
β_1	x
β_2	x^2
β_3	x^3
β_4	$x+1$
β_5	$x^2 + x$
β_6	$x^3 + x^2$
β_7	$x^3 + x + 1$
β_8	$x^2 + 1$
β_9	$x^3 + x$
β_{10}	$x^2 + x + 1$
β_{11}	$x^3 + x^2 + x$
β_{12}	$x^3 + x^2 + x + 1$
β_{13}	$x^3 + x^2 + 1$
β_{14}	$x^3 + 1$
$\beta_{15} = \beta_0;\ \beta_{16} = \beta_1;\ldots;\ \beta_{29} = \beta_{14};\ \beta_{30} = \beta_{15} = \beta_0;\ \beta_{31} = \beta_{16} = \beta_1;\ldots$	

Еще одним базовым понятием теории конечных полей является понятие *минимального полинома (минимального многочлена) элемента* β_i некоторого конечного поля $GF\left(p^m\right)$ $(m \geq 1)$. Он обозначается как $M_i(x)$ и по определению является приведенным полиномом (многочленом с равным единице коэффициентом при старшем по степени слагаемом) наименьшего порядка с коэффициентами, принадлежащими $GF(p)$, для которого β_i является корнем *в соответствующем поле*, т. е.

$$\begin{cases} M_i(\beta_i) = 0 \left(\mathrm{mod}\ p\right) \text{при } \mathrm{m} = 1; \\ M_i(\beta_i) = 0 \left(\mathrm{mod}\, G(x)\right) \text{при } \mathrm{m} > 1. \end{cases} \qquad (2.13)$$

Например, минимальным полиномом $M_3(x)$ поля $GF\left(2^4\right)$ при $G(x) = x^4 + x + 1$ является полином $x^4 + x^3 + x^2 + x + 1$.

Минимальный полином элемента β_i поля $GF\left(p^m\right)$ является минимальным и для *всех элементов* того же поля с индексами, равными $i \times p^n$, где $n -$ любое целое положительное число [8].

Благодаря свойствам конечных полей, формирование и декодирование помехоустойчивых кодов в данных полях обеспечивают:

- принадлежность разрешенных кодовых комбинаций или/и их разрядов к тому же полю, что и исходных информационных слов (например, принадлежность к полю $GF(2)$, т. е. равенство или нулю, или единице всех разрядов результатов кодирования);

- возможность выявления запрещенных (содержащих ошибки) кодовых слов по признаку непринадлежности к соответствующему конечному полю.

2.3. Блочные линейные помехоустойчивые коды общего вида

Блочные *линейные* помехоустойчивые коды (БЛПК) в настоящее время являются основной разновидностью блочных помехоустойчивых кодов. Они характеризуются свойством *линейности* в конечном поле, к которому принадлежат (см. п. 2.1).

Наиболее универсальной формой задания БЛПК является *матричная*, описываемая одним из следующих выражений [8, 9]:

$$\mathbf{W} = (w_{11} \quad w_{12} \quad \cdots \quad w_{1n}) = \mathbf{A} \times \mathbf{G} = (a_{11} \quad a_{12} \quad \ldots \quad a_{1k}) \times \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{pmatrix} \quad (2.14)$$

или

$$\mathbf{H} \times \mathbf{W}^{\mathbf{T}} = \begin{pmatrix} h_{11} & h_{12} & \ldots & h_{1n} \\ h_{21} & h_{22} & \ldots & h_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ h_{c1} & h_{c2} & \cdots & h_{cn} \end{pmatrix} \times \begin{pmatrix} w_{11} \\ w_{21} \\ \vdots \\ w_{n1} \end{pmatrix} = \mathbf{N}_{\mathbf{c} \times \mathbf{1}} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (2.15)$$

где n и k – разрядность соответственно блока и представляемого им входного слова помехоустойчивого кодера, причем всегда справедливо неравенство $n > k$;

W – матрица-строка, каждый из элементов w_{1j} которой ($j=\overline{1,\ldots,n}$) представляет собой число, равное $(j-1)$-му разряду результирующего кодового слова (блока);

$\mathbf{W^T}$ – транспонированная матрица **W**;

A – матрица-строка, каждый из элементов a_{1i} которой ($i=\overline{1,\ldots,k}$) представляет собой число, равное $(i-1)$-му разряду кодируемого информационного слова;

G – *порождающая матрица* БЛПК, размерность которой равна $k \times n$;

H – *проверочная матрица* БЛПК размерностью $c \times n$;

c – число избыточных разрядов в блоке, равное $n-k$;

$\mathbf{N_{c \times 1}}$ – нулевая матрица-столбец размерностью $c \times 1$.

Все элементы матриц в выражениях (2.14) и (2.15) принадлежат к тому же конечному полю, что и задаваемый ими БЛПК.

Правила формирования матриц **G** и **H** зависят от конкретной разновидности БЛПК. Данные матрицы связаны между собой уравнением

$$\mathbf{G \times H^T = N_{k \times c}},$$

где $\mathbf{N_{k \times c}}$ – нулевая матрица размерностью $k \times c$.

Выражение (2.14) удобно для формирования кодовых слов *неразделимых* БЛПК, а выражение (2.15) – *разделимых* [8]. При этом разделимыми называют БЛПК, в кодовом слове которых существует k разрядов с фиксированными позициями, каждый из которых совпадает с определенным разрядом кодируемого информационного слова. Другими словами, к разделимым относятся БЛПК, в кодовом слове которых в явном виде присутствуют информационные и контрольные разряды, а к неразделимым – БЛПК, не обладающие указанными свойствами.

При формировании слов БЛПК в соответствии с выражением (2.14) их разряды вычисляются следующим образом:

$$w_{1j} = \sum_{i=1}^{k} a_{1i} \times g_{ij},\ j=\overline{1,\ldots,n}, \qquad (2.16)$$

причем суммирование и умножение осуществляются в конечном поле, к которому принадлежит БЛПК (в частности, в $GF(2)$ суммирование производится по модулю 2).

В свою очередь, формирование слов разделимого БЛПК в соответствии с выражением (2.15) сводится к определению их контрольных разрядов как решений системы из c уравнений следующего вида:

$$\begin{cases} \sum\limits_{j=1}^{n} h_{1j} \times w_{j1} = 0, \\ \sum\limits_{j=1}^{n} h_{2j} \times w_{j1} = 0, \\ \vdots \\ \sum\limits_{j=1}^{n} h_{cj} \times w_{j1} = 0 \end{cases} \qquad (2.17)$$

при известных информационных разрядах. Суммирование и умножение, как и в выражении (2.16), осуществляются в конечном поле, к которому принадлежит БЛПК.

Выражениями вида (2.14) или (2.15) могут быть описаны практически все известные разновидности БЛПК. Однако большинство распространенных на практике БЛПК позволяют использовать более простые процедуры кодирования и декодирования. В первую очередь, к таковым относятся *циклические* БЛПК, кодирование и декодирование которых базируется на их полиномиальном представлении. Тем не менее существует ряд БЛПК, формирование которых на основании выражений (2.14) и (2.15) является либо наиболее удобным, либо единственно возможным. Такие БЛПК называют БЛПК *общего вида*.

Практическое применение нашли, в основном, двоичные БЛПК общего вида, разряды которых могут принимать только значения 0 и 1, а все операции их формирования и декодирования выполняются в поле $GF(2)$.

Декодирование БЛПК общего вида, в зависимости от конкретной разновидности кода, осуществляется одним из следующих методов:

- вычислением на приемной стороне m-разрядного кода *синдрома ошибок*, на основании которого делается вывод о наличии или отсутствии ошибок в принятом блоке и определяются позиции ошибочных разрядов;

- методом *максимального правдоподобия*.

Декодирование на основании *синдрома ошибок*, как правило, применяется для разделимых кодов. Синдром вычисляется в соответствии с выражением

$$\mathbf{S}^{\mathbf{T}} = \mathbf{H} \times \mathbf{W}_{\mathbf{R}}^{\mathbf{T}}, \qquad (2.18)$$

где $\mathbf{S}^{\mathbf{T}}$ – матрица-столбец размерностью $c \times 1$, получаемая транспонированием матрицы-строки $\mathbf{S}$, каждый из элементов s_{1i} которой ($i = \overline{1,\ldots,c}$) является i-м разрядом кода синдрома;

$\mathbf{H}$ – проверочная матрица, совпадающая с применявшейся при кодировании [см. выражение (2.15)];

$\mathbf{W}_{\mathbf{R}}^{\mathbf{T}}$ – матрица-столбец размерностью $n \times 1$, получаемая транспонированием матрицы-строки $\mathbf{W}_{\mathbf{R}}$, каждый из элементов w_{r1j} которой ($j = \overline{1,\ldots,n}$) представляет собой число, равное j-му разряду принятого кодового слова (блока), причем в отсутствие ошибок обмена данными $w_{r1j} = w_{1j}$ [см. выражения (2.14) – (2.17)] для всех j.

Сопоставляя выражения (2.15) и (2.17), нетрудно увидеть, что в отсутствие ошибок, т. е. при $w_{r1j} = w_{1j}$ для всех j, все разряды синдрома равны нулю. Таким образом, признаком отсутствия ошибок для декодера служит нулевой синдром.

При количестве ошибочных разрядов в блоке, не превышающем значения [6]:

$$e_{\max \det} = d_{\min} - 1, \qquad (2.19)$$

где $d_{\min}$ – *минимальное кодовое расстояние* БЛПК, равное минимальному количеству различающихся между собой одноименных разрядов в любых двух разрешенных (не содержащих ошибок) словах БЛПК, синдром будет ненулевым.

В свою очередь, при количестве ошибочных разрядов, не превышающем значения [6]:

$$e_{\max corr} = \lfloor (d_{\min} - 1)/2 \rfloor, \qquad (2.20)$$

где $\lfloor \bullet \rfloor$ – оператор округления до ближайшего меньшего целого, по синдрому могут быть однозначно определены позиции ошибочных разрядов. Алгоритм их определения зависит от конкретной разновидности БЛПК.

Однако при количестве ошибок в блоке, большем $d_{\min}-1$ [см. выражение (2.19)], неравенство синдрома нулю не гарантируется. При этом по результатам декодирования может быть сделан *ошибочный вывод* об отсутствии ошибок в блоке. Поэтому выбор значения $d_{\min}$ при разработке протоколов кодирования БЛПК производится таким образом, чтобы оно удовлетворяло условию (2.19) (если протокол помехоустойчивого декодирования предполагает только обнаружение ошибок с их исправлением методом *ARQ*) или условию (2.20) (если данный протокол предполагает исправление ошибок при декодировании) при максимально возможном числе ошибок в блоке. Последнее, как правило, определяется путем экспериментально-статистических исследований.

Декодирование методом *максимального правдоподобия* обычно применяется для неразделимых кодов. Оно состоит в выборе декодером, на основании принятого блока и известной порождающей матрицы, наиболее вероятного варианта входного (информативного) слова кодера. Алгоритм выбора определяется конкретной разновидностью БЛПК. Его типовым примером является алгоритм декодирования кодов Рида – Маллера. Необходимо отметить, что максимальное количество как обнаруживаемых, так и исправляемых ошибок в блоке неразделимого БЛПК также описывается выражениями (2.19) и (2.20) соответственно.

Практическое применение нашли, в основном, следующие разновидности БЛПК общего вида:

- код контроля четности;
- БЛПК Хэмминга общего вида;
- коды Рида – Маллера.

Код контроля четности [6, 8] является простейшим БЛПК общего вида. Его минимальное кодовое расстояние равно 2. Следовательно, в соответствии с выражениями (2.19) и (2.20) теоретически он позволяет достоверно обнаружить одну ошибку в блоке без определения позиции ошибочного разряда. На практике код контроля четности обеспечивает обнаружение любого нечетного количества ошибок в блоке (также без их локализации), однако не позволяет обнаружить ошибки при любом четном числе искаженных разрядов.

Код контроля четности является разделимым. Формат его блока: $i_{k-1}i_{k-2}\ldots i_0c_0$ или $c_0i_{k-1}i_{k-2}\ldots i_0$, где буквой i обозначены информационные разряды, а буквой c – контрольный, являющийся единственным в блоке, независимо от количества информационных

разрядов. Проверочная матрица данного кода имеет размерность $1\times n$, а все ее элементы равны единице:

$$\mathbf{H_{PC}} = (1 \quad 1 \quad \ldots \quad 1 \quad 1).$$

На основании выражений (2.15), (2.17) и (2.18) нетрудно увидеть, что контрольный разряд кодового слова при этом рассчитывается по выражению

$$c_0 = i_0 \oplus i_1 \oplus \ldots \oplus i_{k-2} \oplus i_{k-1},$$

а синдром ошибки кода контроля четности, разрядность которого равна одному биту, определяется следующим образом:

$$s_0 = c_{r0} \oplus i_{r0} \oplus i_{r1} \oplus \ldots \oplus i_{r(k-2)} \oplus i_{r(k-1)},$$

где индекс r указывает на то, что бит является принятым и в общем случае не совпадающим с соответствующим ему переданным битом.

При наличии в принятом блоке любого нечетного количества искаженных битов (не совпадающих с переданными) значение синдрома будет равно единице, что, как указано ранее, является признаком наличия ошибок в блоке. Однако при любом четном количестве искаженных битов, как и при их отсутствии, синдром равен нулю.

Кроме вышеописанной простейшей разновидности контроля четности, на практике достаточно широко применяется *продольный контроль четности* (*Longitudinal Redundancy Check – LRC*) [6]. Он используется при передаче данных в виде потока слов некоторой фиксированной разрядности n (например, 8-битовых). При *LRC*-кодировании после каждого блока из N слов передается *контрольная сумма*, каждый из битов которой вычисляется по выражению

$$c_i = w_{i1} \oplus w_{i2} \oplus \ldots \oplus w_{iN}, \quad i = \overline{0,\ldots,n-1},$$

где w_{ij} ($j = \overline{1,\ldots,N}$) – i-й бит j-го слова.

При этом каждое из N слов, в свою очередь, обычно содержит свой бит контроля четности. Такое сочетание *LRC*-кодирования с контролем четности каждого из слов позволяет существенно повысить достоверность обмена данными по сравнению с использованием только контроля четности каждого из слов. Данное сочетание используется, например, одной из версий протокола обмена данными в промышленных сетях *MODBUS*.

Благодаря простоте кодирования и декодирования и минимальному количеству избыточных разрядов, код контроля четности, несмотря на ограниченные возможности обнаружения ошибок и отсутствие возможности определения позиций ошибочных битов, широко применяется различными протоколами обмена данными (*RS-232*, *RS-485*, *HART*, *MODBUS* и др.).

БЛПК Хэмминга общего вида [7, 8] является разделимым, с минимальным кодовым расстоянием, равным 3. Следовательно, в соответствии с выражениями (2.19) и (2.20), при наличии одного искаженного бита в блоке значение синдрома данного кода позволяет однозначно определить позицию ошибочного бита. При наличии двух искаженных битов синдром будет ненулевым, указывая на наличие ошибок в блоке, однако его значение не позволит определить их позиции. При количестве искаженных битов в блоке, большем трех, ненулевое значение синдрома не гарантируется.

Число c контрольных битов в блоке БЛПК Хэмминга общего вида определяется как минимальное, удовлетворяющее неравенству

$$2^c \geq k + c + 1,$$

а формат блока имеет следующий вид: $\ldots i_{18} i_{17} c_{16} i_{15} i_{14} i_{13} i_{12} i_{11} i_{10} i_9 c_8 i_7 i_6 i_5 c_4 i_3 c_2 c_1$, т. е. биты, номера которых являются целыми степенями числа 2, являются контрольными, а остальные – информационными. При этом, в отличие от общепринятых правил, нумерация разрядов кодового слова БЛПК Хэмминга общего вида осуществляется *начиная с 1-го*, а не с 0-го.

Проверочная матрица БЛПК Хэмминга общего вида имеет размерность $m \times n$ и выглядит следующим образом:

$$\mathbf{H}_{\text{НМ}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \ldots \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & \ldots \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \end{pmatrix}, \qquad (2.21)$$

т. е. каждый j-й столбец данной матрицы ($j = \overline{1, \ldots, n}$) является представленным в двоичной системе счисления номером данного столбца (начиная с 1-го).

На основании вышеприведенного формата слова БЛПК Хэмминга общего вида, а также выражений (2.15) и (2.21) нетрудно получить, что контрольные биты данного БЛПК определяются по выражениям:

$$\begin{cases} c_1 = i_3 \oplus i_5 \oplus i_7 \oplus i_9 \oplus i_{11} \oplus \ldots; \\ c_2 = i_3 \oplus i_6 \oplus i_7 \oplus i_{10} \oplus i_{11} \oplus \ldots; \\ c_4 = i_5 \oplus i_6 \oplus i_7 \oplus i_{12} \oplus i_{13} \oplus \ldots; \\ \vdots \end{cases} \qquad (2.22)$$

т. е. i-й контрольный разряд формируется как сумма по модулю 2 всех информационных разрядов, номера которых, представленные в двоичной системе счисления, содержат единицу в $(\log_2 i)$-м бите.

Исходя из выражений (2.18), (2.21) и (2.22) разряды синдрома БЛПК Хэмминга общего вида рассчитываются следующим образом:

$$\begin{cases} s_0 = c_{r1} \oplus i_{r3} \oplus i_{r5} \oplus i_{r7} \oplus i_{r9} \oplus i_{r11} \oplus \ldots; \\ s_1 = c_{r2} \oplus i_{r3} \oplus i_{r6} \oplus i_{r7} \oplus i_{r10} \oplus i_{r11} \oplus \ldots; \\ s_2 = c_{r4} \oplus i_{r5} \oplus i_{r6} \oplus i_{r7} \oplus i_{r12} \oplus i_{r13} \oplus \ldots; \\ \vdots \end{cases} \qquad (2.23)$$

т.е. i-й разряд синдрома представляет собой сумму по модулю 2 2^i-го контрольного разряда принятого слова и всех его информационных разрядов, входящих в выражение для формирования соответствующего контрольного бита [см. выражение (2.22)].

На основе выражений (2.15), (2.21) и (2.23) нетрудно показать, что при наличии одного искаженного бита в принятом блоке синдром равен представленному в двоичной системе счисления номеру данного бита, что обеспечивает простоту обнаружения и исправления ошибки. При любых двух искаженных битах в блоке синдром является ненулевым, указывая на наличие ошибок в блоке, однако его значение равно побитовой сумме по модулю 2 представленных в двоичной системе счисления номеров искаженных битов. Поэтому по его значению невозможно достоверно определить позиции указанных битов, так как, например, значение синдрома, равное 0111, может иметь место при искажении:

- 3-го и 4-го битов;
- 1-го и 6-го битов;
- 7-го бита.

При искажении трех и более битов блока ненулевое значение синдрома и, следовательно, обнаружение ошибок не гарантировано.

Благодаря простоте процедуры исправления ошибок при их количестве в блоке, не превышающем единицы, БЛПК Хэмминга общего вида применяется в процессах обмена данными, с одной стороны, требующих обнаружения и исправления ошибок в реальном масштабе времени («на лету»), а с другой – характеризуемых пренебрежимо малой вероятностью появления более одной ошибки в блоке. Последнее имеет место, например, при относительно малом размере блока (порядка единиц байт) или/и малой интенсивности битовых ошибок. Типовыми примерами практического применения БЛПК Хэмминга общего вида является помехоустойчивое кодирование данных в *RAID*-массивах жестких дисков, а также при обмене данными между основной оперативной памятью и ЦПУ компьютера.

Примеры кодирования и декодирования БЛПК Хэмминга общего вида приведены, например, в [8].

Коды Рида – Маллера [8] представляют собой типовой пример в общем случае неразделимых БЛПК общего вида с произвольно задаваемым минимальным кодовым расстоянием. Они характеризуются двумя основными численными параметрами, которые обозначаются r и q, а для кода Рида – Маллера с соответствующими параметрами используется условное обозначение $RM(r,q)$. Данные параметры связаны с разрядностью k исходного информационного слова, разрядностью блока n и минимальным кодовым расстоянием $d_{\min}$ соотношениями:

$$\begin{cases} n = 2^q; \\ k = 1 + C_q^1 + C_q^2 + \ldots + C_q^r; \\ d_{\min} = 2^{q-r}. \end{cases} \qquad (2.24)$$

Важной с практической точки зрения разновидностью кодов Рида – Маллера являются коды $RM(1,q)$, характеризуемые минимальным кодовым расстоянием, равным 2^{q-1}, и в соответствии с выражением (2.20) позволяющие исправить до 2^{q-2} ошибок в блоке.

Ввиду того что коды Рида – Маллера являются в общем случае неразделимыми, формирование их слов осуществляется в соответствии с выражением (2.14), на основе порождающей матрицы, формируемой по следующим правилам:

- все элементы ее первой строки, обозначемой G_1, являются единицами;

- каждая из ее строк с G_2 по G_{q+1} представляет собой последовательность из чередующихся 2^{i-2} нулей и 2^{i-2} единиц, где i – номер строки;

- остальные $k-(q+1)$ строк (при их наличии) представляют собой поразрядные логические произведения элементов строк с G_2 по G_{q+1} вида

$$G_2 \wedge G_3, \ldots, G_2 \wedge G_{q+1}, G_3 \wedge G_4, \ldots G_3 \wedge G_{q+1}, \ldots, G_2 \wedge G_3 \wedge G_4, \ldots, G_2 \wedge G_3 \wedge \ldots G_{q+1}.$$

Например, порождающая матрица кода $RM(1,3)$ имеет вид

$$G_{RM(1,3)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix},$$

а порождающая матрица кода $RM(2,3)$ выглядит следующим образом:

$$G_{RM(2,3)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Нетрудно заметить, что строки $G_1, \ldots, G_{q+1}$ порождающей матрицы кодов Рида – Маллера (определяющие остальные строки при их наличии) представляют собой отсчеты бинарных функций, *взаимно ортогональных* в поле $GF(2)$, т. е.

$$(g_{i1} \times g_{j1}) \oplus (g_{i2} \times g_{j2}) \oplus \ldots \oplus (g_{in} \times g_{jn}) = 0 \Big|_{i \neq j;\ i \leq q+1;\ j \leq q+1}. \quad (2.25)$$

Данное свойство порождающей матрицы кодов Рида – Маллера существенно упрощает процедуры поиска и исправления ошибок в кодовых словах. Также, благодаря данному свойству,

коды $RM(1,q)$, порождающая матрица которых содержит только строки $G_1,\ldots,G_{q+1}$, называются *биортогональными* (попарно ортогональными). Данная разновидность кодов Рида – Маллера, с одной стороны, является одной из наиболее распространенных на практике, а с другой – наиболее удобна для пояснения принципов кодирования и декодирования как кодов Рида – Маллера, так и неразделимых БЛПК общего вида в целом. Поэтому указанные принципы можно рассмотреть на примере одного из кодов разновидности $RM(1,q)$ – $RM(1,3)$.

Согласно выражениям (2.24), для кода $RM(1,3)$ $k=4,\quad n=8,\quad d_{\min}=4$. Следовательно, в соответствии с выражениями (2.19) и (2.20) код $RM(1,3)$ позволяет обнаружить до трех ошибок в блоке, а достоверно исправить ошибки – при их количестве в блоке, не превышающем одной.

По обобщенному выражению (2.14) формирование блока кода $RM(1,3)$ осуществляется следующим образом:

$$(w_0 \quad \ldots \quad w_7)=(a_0 \quad \ldots \quad a_3)\times G_{RM(1,3)}=(a_0 \quad \ldots \quad a_3)\times\begin{pmatrix}1&1&1&1&1&1&1&1\\1&0&1&0&1&0&1&0\\1&1&0&0&1&1&0&0\\1&1&1&1&0&0&0&0\end{pmatrix}.$$

В соответствии с выражением (2.16) получаем:

$$\begin{cases}w_0=a_0\oplus a_1\oplus a_2\oplus a_3;\\w_1=a_0\oplus a_2\oplus a_3;\\w_2=a_0\oplus a_1\oplus a_3;\\w_3=a_0\oplus a_3;\\w_4=a_0\oplus a_1\oplus a_2;\\w_5=a_0\oplus a_2;\\w_6=a_0\oplus a_1;\\w_7=a_0.\end{cases} \qquad (2.26)$$

Поскольку код $RM(1,3)$ является неразделимым, как и все коды $RM(1,q)$ и большинство других разновидностей кодов Рида – Маллера, его декодирование выполняется методом *максимального правдоподобия*. Из выражений (2.26) нетрудно получить, что при от-

сутствии ошибок кодирования и передачи блока должны быть верны следующие соотношения:

$$\begin{cases} w_{r0} \oplus w_{r1} = w_{r2} \oplus w_{r3} = w_{r4} \oplus w_{r5} = w_{r6} \oplus w_{r7} = a_1; \\ w_{r0} \oplus w_{r2} = w_{r1} \oplus w_{r3} = w_{r4} \oplus w_{r6} = w_{r5} \oplus w_{r7} = a_2; \\ w_{r0} \oplus w_{r4} = w_{r1} \oplus w_{r5} = w_{r2} \oplus w_{r6} = w_{r3} \oplus w_{r7} = a_3. \end{cases} \quad (2.27)$$

Из выражений (2.27) следует, что при искажении не более одного из разрядов принятого блока определение наиболее правдоподобных значений a_1, a_2 и a_3 может быть реализовано способом *мажоритарности***.** Операционная модель реализации данного способа (как в аппаратной, так и в программной форме) применительно к разряду a_1 приведена на рис. 2.4. Блок с условным обозначением «≥3» является *блоком мажоритарности*. Состояние его выхода равно состоянию, в котором находится абсолютное большинство входов (три и более).

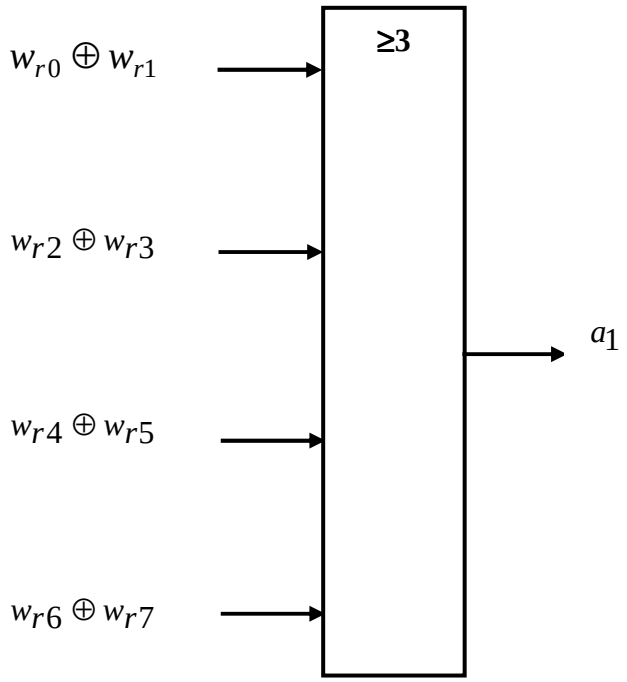


Рис. 2.4. Операционная модель
восстановления разряда a_1 кода $RM(1,3)$

Из выражений (2.27) и рис. 2.4 нетрудно заметить, что корректное восстановление разрядов a_1, a_2 и a_3 исходного кодового слова возможно только при одном искаженном бите принятого блока. При

двух или трех искаженных битах в блоке хотя бы одно из равенств (2.27) не соблюдается, что служит признаком наличия ошибок в блоке. Однако корректное исправление ошибок при этом уже невозможно. При четырех и более ошибках в блоке возможно соблюдение всех равенств (2.27) и, следовательно, неверный вывод об отсутствии ошибок. Вышеизложенное полностью согласуется с выражениями (2.19) и (2.20) с учетом того, что значение $d_{\min}$ кода $RM(1,3)$ равно 4.

После восстановления наиболее правдоподобных значений разрядов a_1, a_2 и a_3 на основании выражений (2.26) восстанавливается наиболее правдоподобное значение разряда a_0 (также способом мажоритарности).

Для возможности достоверного исправления большего количества ошибок в блоке необходимо применять коды $RM(1,q)$ с бóльшим $d_{\min}$. Например, код $RM(1,4)$ характеризуется $d_{\min}$, равным 8, и, следовательно, позволяет достоверно исправить до трех ошибок в блоке или обнаружить в нем до семи ошибок.

Следует отметить, что декодирование кодов $RM(1,q)$ при любом q осуществляется в два этапа, на первом из которых определяются наиболее правдоподобные значения разрядов $a_1,\ldots,a_{k-1}$, а на втором – разряда a_0.

В общем случае декодирование кодов $RM(r,q)$ реализуется (за исключением нескольких специальных их разновидностей) также методом мажоритарности, в $r+1$ этапов, на последнем из которых всегда оценивается наиболее правдоподобное значение разряда a_0.

В целом, кроме кодов Рида – Маллера, вышеописанный принцип мажоритарности применяется при декодировании ряда других классов неразделимых БЛПК методом максимального правдоподобия. Кроме способа мажоритарности, при декодировании указанным методом используются и другие подходы (см. алгоритмы сверточного декодирования в п. 2.5).

Благодаря простоте кодирования и декодирования, коды Рида – Маллера, особенно типа $RM(1,q)$, достаточно широко используются в зашумленных каналах связи для обнаружения и исправления ошибок в реальном масштабе времени («на лету»).

Кроме рассмотренных ранее, существуют и другие классы БЛПК общего вида, менее распространенные на практике [8].

2.4. Циклические помехоустойчивые коды

2.4.1. Общие положения

Циклические помехоустойчивые коды (ЦПК) являются наиболее распространенной на практике разновидностью БЛПК, благодаря возможности обнаружения и/или исправления с их помощью практически любого наперед заданного количества ошибок в блоке, с одной стороны, и относительной простоте как аппаратной, так и программной реализации кодирования и декодирования – с другой [6 – 8]. ЦПК весьма широко применяются практически во всех отраслях информационных технологий: в системах телекоммуникаций, в интерфейсах электронно-вычислительных средств (*PCI Express, USB* и др.), в сетевых технологиях, при записи данных на магнитные и оптические носители и др.

Известно достаточно много разновидностей ЦПК. Однако все они характеризуются рядом общих особенностей, а также аналогичными обобщенными алгоритмами кодирования и декодирования.

Название «*циклический код*» происходит от следующего свойства ЦПК, общего для всех их разновидностей: результатом циклического сдвига любой разрешенной кодовой комбинации ЦПК является также разрешенная кодовая комбинация того же ЦПК.

Все распространенные на практике ЦПК – *разделимые*. Формат их блока следующий: k старших разрядов – информационные, а c младших – контрольные. Разряды блока являются числами, принадлежащими к некоторому простому конечному полю $GF(p)$, причем значение p определяется конкретной разновидностью ЦПК. Соотношение между k и c зависит от максимального количества ошибочных разрядов в блоке, для обнаружения и/или исправления которых предназначается соответствующий ЦПК, а также от его конкретной разновидности.

Максимальное количество обнаруживаемых ошибочных разрядов в блоке ЦПК связано с его минимальным кодовым расстоянием соотношением (2.19), а максимальное количество исправляемых – соотношением (2.20). Следует при этом отметить, что для недвоичных ЦПК, разряды которых принадлежат к полю $GF(p)$ с p, боль-

шим двух, под минимальным кодовым расстоянием понимается минимальное количество различающихся между собой *одноименных разрядов (а не битов)* в любых двух разрешенных словах ЦПК.

Основной характеристикой любого ЦПК является его *порождающий полином* $G(x)$, порядок которого равен m, а коэффициенты принадлежат к тому же конечному полю $GF(p)$, что и разряды ЦПК в целом. Вид порождающего полинома зависит от конкретной разновидности ЦПК.

Обобщенный алгоритм циклического помехоустойчивого кодирования – следующий [6, 8]:

1. Подлежащее кодированию k-разрядное информационное слово представляется полиномом вида

$$I(x) = i_{k-1} \times x^{k-1} + i_{k-2} \times x^{k-2} + \ldots + i_1 \times x + i_0, \qquad (2.28)$$

где $i_{k-1}, i_{k-2}, \ldots, i_0$ – значения разрядов информационного слова с соответствующими номерами.

2. Полином $I(x)$ умножается на x^c.

3. Находится остаток $\mathrm{Re}\, s(x)$ от деления полинома $I(x) \times x^c$ на порождающий полином $G(x)$ *в поле* $GF(p)$, к которому принадлежат их коэффициенты (см. примеры на рис. 2.2 и 2.3).

4. Результат кодирования (кодовое слово ЦПК) формируется следующим образом:

$$T(x) = I(x) \times x^c - \mathrm{Re}\, s(x), \qquad (2.29)$$

причем вычитание также осуществляется *в поле* $GF(p)$.

При формировании кодового слова ЦПК по выражению (2.29) остаток от его деления на порождающий полином $G(x)$ равен нулю. При совпадении слова $T_R(x)$, полученного в результате передачи кодового слова $T(x)$ через канал связи, с переданным словом $T(x)$, т. е. при отсутствии ошибок при передаче, остаток от деления $T_R(x)$ на $G(x)$ также равен нулю. Следовательно, указанный остаток в ЦПК служит *синдромом ошибок***.** Его ненулевое значение является признаком наличия ошибок в блоке [в кодовом слове $T_R(x)$]. Поэтому проверка наличия ошибок в кодовом слове ЦПК сводится к нахождению остатка от деления $T_R(x)$ на $G(x)$ и сравнению его с нулем.

При количестве ошибочных разрядов, превышающем значение, задаваемое выражением (2.19), ненулевое значение синдрома и, как следствие, обнаружение ошибок в блоке не гарантируется. При количестве ошибочных разрядов, не превышающем значения, задаваемого выражением (2.20), по синдрому могут быть однозначно выявлены и исправлены ошибочные разряды. Алгоритмы исправления ошибок зависят от конкретных разновидностей ЦПК. Однако при количестве ошибочных разрядов в блоке, большем определяемого выражением (2.20), указанные алгоритмы не гарантируют исправления ошибок и, более того, их применение может привести к появлению новых ошибок. Поэтому многие из протоколов помехоустойчивого кодирования посредством ЦПК предусматривают только проверку (по синдрому) наличия ошибок в блоке. Исправление ошибок при их обнаружении осуществляется повторной передачей блока по запросу приемника (см. п. 2.6).

Кроме вышеописанной полиномиальной формы представления ЦПК, возможна и матричная форма, универсальная для всех БЛПК [см. выражения (2.14) и (2.15)]. Однако она сложнее как для аппаратной, так и для программной реализации, и, поскольку ЦПК допускают и более простую в реализации полиномиальную форму представления, матричная форма их представления на практике не используется. Ее описание приведено, например, в [8].

Известны два основных класса ЦПК (см. рис. 2.1 и пояснения к нему):

- двоичные ЦПК, коэффициенты полиномов $I(x)$ и $G(x)$ которых являются элементами поля $GF(2)$, т. е. могут принимать только значения 0 и 1;

- недвоичные ЦПК, коэффициенты указанных полиномов которых принадлежат к полю $GF(p)$ с p, большим двух.

При этом существуют классы ЦПК, коэффициенты полиномов которых, в зависимости от конкретной разновидности кода, могут принадлежать как к полю $GF(2)$, так и к полю $GF(p > 2)$. К таковым относятся, например, *ЦПК Боуза – Чоудхури – Хоквингема (коды БЧХ)* [7, 8].

Коды БЧХ являются наиболее распространенным и универсальным классом ЦПК, позволяющим обнаружить или исправить любое наперед заданное количество ошибочных разрядов в блоке, произвольно распределенных по нему.

2.4.2. Коды БЧХ как наиболее универсальный класс ЦПК

Как указано ранее, существуют как двоичные, так и недвоичные коды БЧХ. *Порождающие полиномы* всех ЦПК, относящиеся к данному классу, характеризуются *общим свойством* [7, 8]: наличием $d_{\min}-1$ корней в конечном поле, к которому принадлежит порождающий полином, равных $\beta_i^{l_0}, \beta_i^{l_0+1}, \ldots, \beta_i^{l_0+d_{\min}-2}$. Здесь $d_{\min}$ – минимальное кодовое расстояние ЦПК, задаваемое исходя из выражений (2.19) и (2.20) и для кодов БЧХ выбираемое *нечетным*; β_i – ненулевой элемент i-го порядка конечного поля, к которому принадлежит полином (см. п. 2.2); l_0 – целое неотрицательное число.

На практике обычно выбирается значение l_0, равное единице, при котором порождающий полином имеет при прочих равных условиях наименьшую степень, а кодовое слово БЧХ в целом – наименьшее число избыточных (контрольных) разрядов [8]. Коды БЧХ с l_0, равным единице, известны под названием «*коды БЧХ в узком смысле*» [8].

При корнях порождающего полинома кода, равных целым степеням ненулевого элемента i-го порядка конечного поля $GF(p)$ или $GF(p^m)$, к которому данный полином принадлежит, разрядность n блока кода БЧХ равна $(p^h-1)/i$, где h – целое положительное число. Нетрудно заметить, что при $i=1$, т. е. при использовании в качестве β_i *примитивного элемента* α [см. выражения (2.8) и (2.9)] указанного конечного поля, разрядность n блока при заданном $d_{\min}$ максимальна и равна p^h-1. Коды БЧХ с такими свойствами, ввиду использования примитивного элемента поля в качестве β_i, известны под названием *примитивных* [8]. В целом, коды БЧХ с $l_0=1$ и $\beta_i=\beta_1=\alpha$ наиболее распространены на практике [8].

Порождающий полином рассматриваемой разновидности кодов БЧХ описывается обобщенным выражением [7, 8]:

$$G(x)=HOK\{M_1(x), M_2(x), \ldots, M_{d_{\min}-1}(x)\}, \qquad (2.30)$$

где $HOK\{\bullet\}$ – наименьшее общее кратное;

$M_i(x)$ – минимальный многочлен i-го порядка в конечном поле $GF(p^h)$ [см. выражение (2.13)].

2.4.3. Двоичные коды БЧХ

Коэффициенты как полинома $I(x)$, так и $G(x)$ данных кодов могут принимать только значения 0 или 1, а все математические операции над ними выполняются в конечном поле $GF(2)$. Разрядность блока равна $2^h - 1$, причем $h > 1$. Многочлены $M_i(x)$ являются минимальными в конечном поле $GF(2^h)$ с неприводимым примитивным двоичным порождающим полиномом порядка h (см. табл. 2.1).

В конечном поле $GF(2^h)$ минимальный многочлен $M_i(x)$ равен минимальному многочлену $M_{i\times 2^n}(x)$ при любом целом положительном n [см. пояснения и дополнения к выражению (2.13)]. С учетом этого, а также нечетности $d_{\min}$ кодов БЧХ и на основании обобщенного выражения (2.30) нетрудно получить, что порождающий полином двоичного БЧХ имеет следующий вид:

$$G(x) = M_1(x) \times M_3(x) \times M_5(x) \times \ldots \times M_{d_{\min}-2}(x), \qquad (2.31)$$

причем полином $M_1(x)$ является примитивным двоичным полиномом степени h (см. табл. 2.1), а полиномы $M_3(x), M_5(x), \ldots, M_{d_{\min}-2}(x)$ – двоичными неприводимыми полиномами степени, равной или меньшей h, корнями которых *в поле* $GF(2^h)$, *порождаемом полиномом* $M_1(x)$, являются $x^3, x^5, \ldots$ (остатки от деления данных полиномов на $M_1(x)$ после подстановки в них $x^3, x^5, \ldots$ равны нулю). Для облегчения процесса разработки средств двоичного БЧХ-кодирования порождающие полиномы двоичных кодов БЧХ затабулированы. В табл. 2.3 представлены порождающие полиномы двоичных кодов БЧХ при h, равном от 3 до 8, для различных значений $d_{\min}$, возможных при соответствующем h. Порождающие полиномы двоичных кодов БЧХ при значениях h, больших 8, приведены, например, в [6].

Из табл. 2.3 видно, что при одинаковом $d_{\min}$ чем больше общая разрядность блока n, тем больше соотношение между числом информационных и контрольных разрядов в блоке, т. е. тем меньше информационная избыточность кода. С другой стороны, при слишком большой разрядности блока (более нескольких десятков Кбит) достоверное предсказание максимально возможного числа ошибок в блоке и, следовательно, корректный выбор $d_{\min}$ затруднительны или невозможны.

Таблица 2.3

Порождающие полиномы двоичных кодов БЧХ при h, равном от 2 до 8

h	$d_{\min}$	Порождающий полином	Формат блока	
			n	c
3	5	$(x^3+x+1)(x^3+x^2+1)$	7	6
4	5	$(x^4+x+1)(x^4+x^3+x^2+x+1)$	15	8
	7	$(x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)$		10
	9	$(x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)(x^4+x^3+1)$		14
5	5	$(x^5+x^2+1)(x^5+x^4+x^3+x^2+1)$	31	10
	7	$(x^5+x^2+1)(x^5+x^4+x^3+x^2+1)(x^5+x^4+x^2+x+1)$		15
	9	$(x^5+x^2+1)(x^5+x^4+x^3+x^2+1)(x^5+x^4+x^2+x+1)(x^5+x^3+x^2+ +x+1)$		20
	11	$(x^5+x^2+1)(x^5+x^4+x^3+x^2+1)(x^5+x^4+x^2+x+1)(x^5+x^3+x^2+ +x+1)(x^5+x^4+x^2+x+1)$		25
	13	$(x^5+x^2+1)(x^5+x^4+x^3+x^2+1)(x^5+x^4+x^2+x+1)(x^5+x^3+x^2+ +x+1)(x^5+x^4+x^2+x+1)\ (x^5+x^4+x^3+x+1)$		30
6	5	$(x^6+x+1)(x^6+x^4+x^2+x+1)$	63	12
	7	$(x^6+x+1)(x^6+x^4+x^2+x+1)(x^6+x^5+x^2+x+1)$		18
	9	$(x^6+x+1)(x^6+x^4+x^2+x+1)(x^6+x^5+x^2+x+1)(x^6+x^3+1)$		24
	11	$(x^6+x+1)(x^6+x^4+x^2+x+1)(x^6+x^5+x^2+x+1)(x^6+x^3+1)(x^3+ +x^2+1)$		27
	13	$(x^6+x+1)(x^6+x^4+x^2+x+1)(x^6+x^5+x^2+x+1)(x^6+x^3+1)(x^3+ +x^2+1)(x^6+x^5+x^3+x^2+1)$		33
7	5	$(x^7+x^3+1)(x^7+x^3+x^2+x+1)$	127	14
	7	$(x^7+x^3+1)(x^7+x^3+x^2+x+1)(x^7+x^4+x^3+x^2+1)$		21
	9	$(x^7+x^3+1)(x^7+x^3+x^2+x+1)(x^7+x^4+x^3+x^2+1)(x^7+x^6+x^5+ +x^4+x^2+x+1)$		28
	11	$(x^7+x^3+1)(x^7+x^3+x^2+x+1)(x^7+x^4+x^3+x^2+1)(x^7+x^6+x^5+ +x^4+x^2+x+1)(x^7+x^5+x^4+x^3+x^2+x+1)$		35
	13	$(x^7+x^3+1)(x^7+x^3+x^2+x+1)(x^7+x^4+x^3+x^2+1)(x^7+x^6+x^5+ +x^4+x^2+x+1)(x^7+x^5+x^4+x^3+x^2+x+1)(x^7+x^6+x^4+x^2+1)$		42
	15	$(x^7+x^3+1)(x^7+x^3+x^2+x+1)(x^7+x^4+x^3+x^2+1)(x^7+x^6+x^5+ +x^4+x^2+x+1)(x^7+x^5+x^4+x^3+x^2+x+1)(x^7+x^6+x^4+x^2+1)* *(x^7+x+1)$		49
8	5	$(x^8+x^4+x^3+x^2+1)(x^8+x^6+x^5+x^4+x^2 +x+1)$	255	16
	7	$(x^8+x^4+x^3+x^2+1)(x^8+x^6+x^5+x^4+x^2 +x+1)(x^8+x^7+x^6+x^5+ +x^4+x+1)$		24
	9	$(x^8+x^4+x^3+x^2+1)(x^8+x^6+x^5+x^4+x^2 +x+1)(x^8+x^7+x^6+x^5+ +x^4+x+1)(x^8+x^6+x^5+x^3+1)$		32
	11	$(x^8+x^4+x^3+x^2+1)(x^8+x^6+x^5+x^4+x^2 +x+1)(x^8+x^7+x^6+x^5+ +x^4+x+1)(x^8+x^6+x^5+x^3+1)(x^8+x^7+x^5+x^4+x^3+x^2+1)$		40
	13	$(x^8+x^4+x^3+x^2+1)(x^8+x^6+x^5+x^4+x^2 +x+1)(x^8+x^7+x^6+x^5+ +x^4+x+1)(x^8+x^6+x^5+x^3+1)(x^8+x^7+x^5+x^4+x^3+x^2+1)(x^8+ +x^7+x^6+x^5+x^2+x+1)$		48
	15	$(x^8+x^4+x^3+x^2+1)(x^8+x^6+x^5+x^4+x^2 +x+1)(x^8+x^7+x^6+x^5+ +x^4+x+1)(x^8+x^6+x^5+x^3+1)(x^8+x^7+x^5+x^4+x^3+x^2+1)(x^8+ +x^7+x^6+x^5+x^2+x+1)(x^8+x^5+x^3+x+1)$		56
	17	$(x^8+x^4+x^3+x^2+1)(x^8+x^6+x^5+x^4+x^2 +x+1)(x^8+x^7+x^6+x^5+ +x^4+x+1)(x^8+x^6+x^5+x^3+1)(x^8+x^7+x^5+x^4+x^3+x^2+1)(x^8+ +x^7+x^6+x^5+x^2+x+1)(x^8+x^5+x^3+x+1)(x^8+x^7+x^6+x^4+x^2+ +x+1)$		64

Применяемые на практике протоколы БЧХ-кодирования оговаривают разрядность блока порядка от нескольких единиц до нескольких десятков Кбит, что обеспечивает приемлемый компромисс между информационной избыточностью кода и достоверностью предсказания максимального числа ошибок в блоке и, следовательно, корректностью выбора $d_{\min}$.

Пример кодирования двоичным кодом БЧХ. Пусть необходимо осуществить кодирование 15-битового блока двоичным кодом БЧХ, обеспечивающим обнаружение и исправление до двух ошибок или обнаружение без исправления до четырех ошибок в блоке. Тогда в соответствии с выражениями (2.19) и (2.20) $d_{\min}=5$, а, поскольку $15=2^4-1$, $h=4$. Следовательно, согласно табл. 2.3, порождающий полином кода БЧХ имеет следующий вид:

$$G(x)=\left(x^4+x+1\right)\times\left(x^4+x^3+x^2+x+1\right)=x^8+x^7+x^6+x^4+1$$

(см. пример, иллюстрирующий *Правило 2* в п. 2.2). При этом блок будет содержать семь информационных и восемь контрольных разрядов, т. е. $k=7$, $c=8$, а $n=15$.

Пусть информационная часть блока представляет собой двоичный ASCII-код заглавной латинской буквы А – 1000001. Тогда в соответствии с выражением (2.28)

$$I(x)=x^6+1.$$

Умножаем $I(x)$ на x^c, где c – порядок порождающего полинома, в рассматриваемом примере равный 8. Получаем:

$$I(x)\times x^8=x^{14}+x^8.$$

Находим остаток от деления полученного полинома на образующий:

$$\begin{array}{ll} & x^{14}+x^8 \\ \oplus & \underline{x^{14}+x^{13}+x^{12}+x^{10}+x^6=x^6\left(x^8+x^7+x^6+x^4+1\right)} \\ & x^{13}+x^{12}+x^{10}+x^8+x^6 \\ \oplus & \underline{x^{13}+x^{12}+x^{11}+x^9+x^5=x^5\left(x^8+x^7+x^6+x^4+1\right)} \\ & x^{11}+x^{10}+x^9+x^8+x^6+x^5 \\ \oplus & \underline{x^{11}+x^{10}+x^9+x^7+x^3=x^3\left(x^8+x^7+x^6+x^4+1\right)} \\ & x^8+x^7+x^6+x^5+x^3 \\ \oplus & \underline{x^8+x^7+x^6+x^4+1} \\ & x^5+x^4+x^3+1 \end{array} \qquad \begin{array}{|l} \hline x^8+x^7+x^6+x^4+1 \\ \hline x^6+x^5+x^3+1 \end{array}$$

В соответствии с выражением (2.29) остаток, равный $x^5 + x^4 + x^3 + 1$, вычитаем по модулю 2 из полинома, полученного в результате умножения $I(x)$ на x^8, равного $x^{14} + x^8$. С учетом того, что вычитание и сложение по модулю 2 эквивалентны, получаем результат кодирования в полиномиальной форме [см. выражение (2.29)]:

$$T(x) = x^{14} + x^8 + x^5 + x^4 + x^3 + 1.$$

В свою очередь, в двоичной форме данный результат равен 100000100111001. Старшие семь битов данного двоичного блока являются информационными; нетрудно заметить, что они совпадают с подлежащим передаче ASCII-кодом заглавной латинской буквы А. Младшие восемь битов блока – контрольные. В таком виде блок передается через канал связи или записывается на носитель.

Декодирование двоичных кодов БЧХ. Синдром ошибки кода БЧХ, как и других ЦПК, определяется как остаток от деления представленного в полиномиальной форме принятого блока, $T_R(x)$, на порождающий полином, использованный при кодировании. При нулевом остатке делается вывод об отсутствии ошибок обмена данными, при ненулевом – об их наличии. Если протокол декодирования предусматривает только обнаружение ошибок без их исправления – при ненулевом синдроме формируется запрос на повторную передачу блока, например, передатчику блока отправляется сообщение «*Не подтверждение*» (*NACK*). Если же протокол декодирования двоичного кода БЧХ предусматривает исправление обнаруженных ошибок, данная процедура в принципе может быть реализована по одному из универсальных алгоритмов исправления ошибок в кодовых словах ЦПК, в том числе недвоичных [8].

Однако существуют специальные алгоритмы, применимые для исправления ошибок только при кодировании двоичными ЦПК, более простые в реализации, чем универсальные. Блок-схема одного из распространенных на практике алгоритмов исправления ошибок в блоках двоичных кодов БЧХ представлена на рис. 2.5 [8]. Проиллюстрируем работу данного алгоритма следующим примером (пропуская промежуточные выкладки).

Пусть при передаче полученного выше результата кодирования произошло искажение 12-го и 13-го битов, т. е. на приемной стороне получено двоичное слово 111000100111001, или в полиномиальной форме:

$$T_R(x) = x^{14} + x^{13} + x^{12} + x^8 + x^5 + x^4 + x^3 + 1.$$

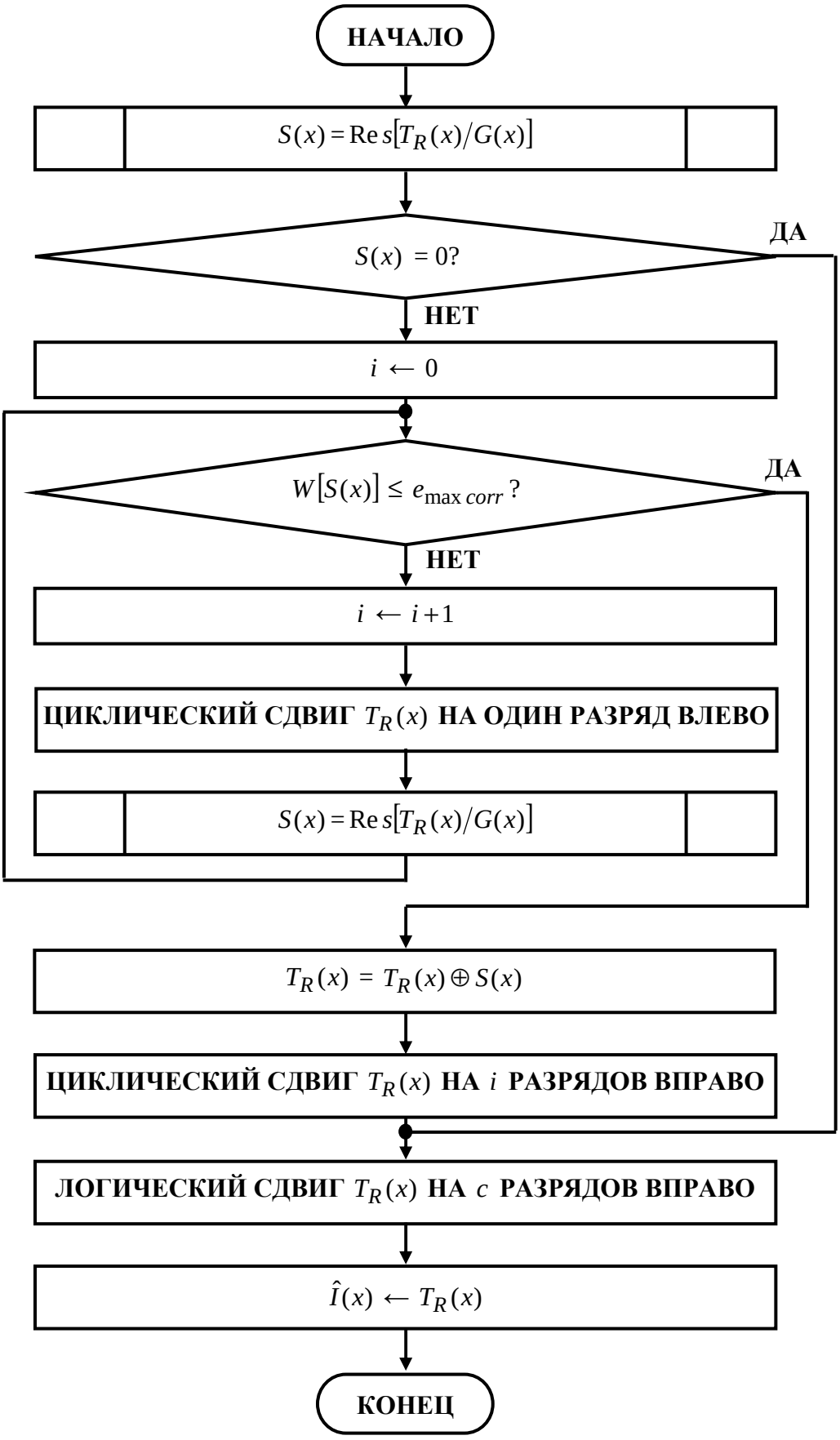


Рис. 2.5. Блок-схема алгоритма декодирования двоичного кода БЧХ:
$S(x)$ – синдром; $W[S(x)]$ – число ненулевых членов синдрома;
i – счетчик сдвигов; $\hat{I}(x)$ – восстановленное информационное поле блока

В результате деления полинома $T_R(x)$ на порождающий полином, использованный при кодировании $(x^8 + x^7 + x^6 + x^4 + 1)$, получаем, что служащий синдромом остаток от деления равен:

$$\mathrm{Re}\,s[T_R(x)/G(x)] = x^6 + x^3 + x^2 + x.$$

Синдром – ненулевой, что указывает на наличие ошибок в блоке. Число ненулевых членов в остатке больше максимального числа исправляемых ошибок, обеспечиваемого рассматриваемым в данном примере кодом БЧХ и равного двум. Поэтому циклически сдвигаем $T_R(x)$ на один разряд влево, получая:

$$T_R(x) = x^{14} + x^{13} + x^9 + x^6 + x^5 + x^4 + x + 1.$$

Делим $T_R(x)$ на *G(x)*. Остаток от деления равен:

$$\mathrm{Re}\,s[T_R(x)/G(x)] = x^7 + x^4 + x^3 + x^2.$$

Число ненулевых членов остатка больше двух, поэтому производим второй циклический сдвиг $T_R(x)$ влево на один разряд, в результате чего получаем:

$$T_R(x) = x^{14} + x^{10} + x^7 + x^6 + x^5 + x^2 + x + 1.$$

Остаток от деления данного полинома на *G(x)* равен:

$$\mathrm{Re}\,s[T_R(x)/G(x)] = x^7 + x^6 + x^5 + x^3 + 1.$$

Число ненулевых членов остатка больше двух, поэтому выполняем третий циклический сдвиг $T_R(x)$ влево на один разряд, получая:

$$T_R(x) = x^{10} + x^8 + x^7 + x^6 + x^3 + x^2 + x + 1.$$

Остаток от деления данного полинома на *G(x)* равен:

$$\mathrm{Re}\,s[T_R(x)/G(x)] = x + 1.$$

Число ненулевых членов в остатке не больше двух. Поэтому суммируем данный остаток с $T_R(x)$. В результате получаем:

$$T_R(x) = x^{10} + x^8 + x^7 + x^6 + x^3 + x^2.$$

После этого циклически сдвигаем $T_R(x)$ на три разряда вправо, получая следующий полином:

$$T_R(x) = x^{14} + x^8 + x^5 + x^4 + x^3 + 1,$$

который совпадает с переданным полиномом, $T(x)$. В результате его логического сдвига на $c = 8$ разрядов вправо получаем исходное информационное слово в полиномиальной форме:

$$\hat{I}(x) = x^6 + 1,$$

которое соответствует двоичному коду 1000001.

Следует отметить, что описанный алгоритм, как и все алгоритмы декодирования ЦПК (и БЛПК в целом), обеспечивает корректное исправление ошибок только при их количестве в блоке, не превышающем верхнего предела, задаваемого выражением (2.20).

2.4.4. Недвоичные коды БЧХ

К недвоичным относятся коды БЧХ, коэффициенты полиномов $I(x)$ и $G(x)$ которых являются элементами поля $GF(p)$, где p – простое число, большее двух, или поля $GF\left(2^m\right)$ с m, бóльшим единицы. При кодировании и декодировании данные коэффициенты представляются в двоичной системе счисления.

Наиболее распространенной на практике разновидностью недвоичных кодов БЧХ являются *коды Рида – Соломона* [6 – 8]. В общем случае их порождающий полином имеет следующий вид:

$$G(x) = \left(x - \alpha^{l_0}\right) \times \left(x - \alpha^{l_0+1}\right) \times \ldots \times \left(x - \alpha^{l_0+d_{\min}-2}\right), \qquad (2.32)$$

где α – примитивный элемент конечного поля $GF(p)$ или $GF\left(2^m\right)$, удовлетворяющий условиям (2.8) или (2.9).

На практике нашли применение, в основном, коды Рида – Соломона с l_0, равным единице, т. е. коды Рида – Соломона в узком смысле (см. п. 2.4.2). Их порождающий полином имеет вид

$$G(x) = (x - \alpha) \times \left(x - \alpha^2\right) \times \ldots \times \left(x - \alpha^{d_{\min}-1}\right) \qquad (2.33)$$

[ср. с выражением (2.30)]. Общее число разрядов в блоке при формировании кода в поле $GF(p)$ может быть до $p-1$, а в поле $GF\left(2^m\right)$ – до $2^m - 1$. Число контрольных разрядов в обоих случаях равно $d_{\min} - 1$.

В качестве как контрольных, так и информационных разрядов блока могут выступать нуль и элементы от β_1 до β_{2^m-2} поля [$GF(p)$ или $GF\left(2^m\right)$], в котором реализуется кодирование (см. табл. 2.2).

Кодирование и обнаружение ошибок осуществляются по описанному в п. 2.4.1 алгоритму, общему для всех ЦПК. При этом деление полиномов $T(x)$ и $T_R(x)$ на порождающий полином осуществляется *в поле* $GF(p)$ или [при кодировании в поле $GF\left(2^m\right)$] $GF(2^m)$. *Исправление* ошибок имеет ряд особенностей по сравнению с аналогичной процедурой при декодировании двоичных ЦПК. Из-за того, что в качестве разрядов принятого кодового слова могут выступать нуль и элементы от β_1 до β_{2^m-2} поля, в котором формируется код, исправление ошибок требует последовательного выполнения трех операций:

- локализации (определения позиций) ошибочных (искаженных) разрядов;

- определения значений поправок, которые необходимо внести в искаженные разряды для исправления ошибок (данная операция не требуется при декодировании двоичных ЦПК, так как исправление их ошибочных разрядов осуществляется операцией инвертирования последних или эквивалентного ему сложения с единицей по модулю 2);

- исправления ошибочных разрядов путем вычитания из них полученных поправок *в поле*, в котором осуществляется кодирование.

Известно несколько алгоритмов как локализации ошибочных разрядов, так и определения поправок, которые необходимо внести в данные разряды. Из алгоритмов *локализации* ошибок одним из наиболее распространенных является *алгоритм Берлекампа – Месси* [6, 8], в принципе, применимый ко всем ЦПК, но наиболее широко используемый при локализации ошибок в кодах Рида – Соломона. Данный алгоритм реализуется следующим образом (изначально предполагается, что синдром, равный остатку от деления $T_R(x)$ на $G(x)$, является ненулевым, т. е. выявлено наличие ошибок в блоке):

1. Полагается, что число ошибок в блоке e равно $e_{\max corr}$, т. е. $\left(d_{\min}-1\right)/2$.

2. В поле $GF(p)$ или, соответственно, $GF\left(2^m\right)$ вычисляются значения $T_R\left(\alpha^i\right)$ при i, равном от единицы до $2e-1$, и определитель следующей матрицы:

$$\mathbf{TR}=\begin{pmatrix} T_R(\alpha) & T_R\left(\alpha^2\right) & \cdots & T_R\left(\alpha^e\right) \\ T_R\left(\alpha^2\right) & T_R\left(\alpha^3\right) & \cdots & T_R\left(\alpha^{e+1}\right) \\ \vdots & \vdots & \vdots & \vdots \\ T_R\left(\alpha^e\right) & T_R\left(\alpha^{e+1}\right) & \cdots & T_R\left(\alpha^{2e-1}\right) \end{pmatrix}. \qquad (2.34)$$

3. Если определитель равен нулю, делается вывод, что число ошибок в блоке меньше e. Значение e уменьшается на единицу, и осуществляется переход к п. 3. Если же данный определитель отличен от нуля, делается вывод, что число ошибок в блоке равно e, и выполняется переход к п. 4.

4. Вычисляются коэффициенты полинома – *локатора ошибок*, как решения *в поле* $GF(p)$ или $GF\left(2^m\right)$ матричного уравнения:

$$\begin{pmatrix} T_R(\alpha) & T_R\left(\alpha^2\right) & \cdots & T_R\left(\alpha^e\right) \\ T_R\left(\alpha^2\right) & T_R\left(\alpha^3\right) & \cdots & T_R\left(\alpha^{e+1}\right) \\ \vdots & \vdots & \vdots & \vdots \\ T_R\left(\alpha^e\right) & T_R\left(\alpha^{e+1}\right) & \cdots & T_R\left(\alpha^{2e-1}\right) \end{pmatrix} \times \begin{pmatrix} \Lambda_e \\ \Lambda_{e-1} \\ \vdots \\ \Lambda_1 \end{pmatrix} = \begin{pmatrix} -T_R\left(\alpha^{e+1}\right) \\ -T_R\left(\alpha^{e+2}\right) \\ \vdots \\ -T_R\left(\alpha^{2e}\right) \end{pmatrix}. \qquad (2.35)$$

5. По результатам решения уравнения (2.35) составляется полином – локатор ошибок, имеющий следующий вид:

$$\Lambda(x)=1+\Lambda_1 x+\Lambda_2 x^2+\ldots+\Lambda_e x^e, \qquad (2.36)$$

и определяются его корни *в поле* $GF(p)$ или $GF\left(2^m\right)$. Очевидно, число данных корней равно e.

6. По значениям корней полинома (2.36) определяются номера ошибочных разрядов блока (откуда название полинома – локатор ошибок), как решения *в поле* $GF(p)$ или $GF\left(2^m\right)$ уравнений вида

$$x_i\alpha^j=1, \qquad (2.37)$$

где x_i – i-й корень полинома (2.36) в поле $GF(p)$ или $GF\left(2^m\right)$;

j – номер ошибочного разряда, на который указывает i-й корень локатора ошибок.

В свою очередь, из алгоритмов *определения поправок*, которые необходимо внести в ошибочные разряды, одним из наиболее распространенных является *алгоритм Форни* [6, 8]. Его входными данными являются полином $T_R(x)$, а также полином – локатор ошибок вида (2.36), полученный, например, по алгоритму Берлекампа – Месси, и его корни.

Алгоритм Форни реализуется следующим образом:

1. Находится полином *частичного синдрома***,** определяемый следующим образом:

$$SP(x) = T_R(\alpha\ ) + T_R(\alpha^2) \times x + \ldots + T_R(\alpha^{d_{\min}-1}) \times x^{d_{\min}-2}, \qquad (2.38)$$

причем все коэффициенты полинома $SP(x)$ рассчитываются *в поле* $GF(p)$ или $GF(2^m)$.

2. Вычисляется *полином оценивания ошибок*, $\Omega(x)$, в соответствии с выражением

$$\Omega(x) = \mathrm{Re}\, s\left[\frac{SP(x) \times \Lambda(x)}{x^{d_{\min}-1}}\right], \qquad (2.39)$$

причем математические операции над коэффициентами полиномов $SP(x)$ и $\Lambda(x)$ при их перемножении осуществляются *в поле* $GF(p)$ или [при кодировании в поле $GF(2^m)$] $GF(2^m)$.

3. Значения поправок, которые необходимо внести в ошибочные разряды для их коррекции, определяются *в поле* $GF(p)$ или $GF(2^m)$ по выражениям следующего вида:

$$e_i = \left[-\frac{\Omega(x_i)}{\Lambda'(x_i)}\right], \qquad (2.40)$$

где $\Lambda'(x_i)$ – значение 1-й производной полинома (2.36) *в соответствующем поле* при $x = x_i$.

Операция *внесения поправок* в ошибочные разряды реализуется путем вычитания из них значений e_i *в поле* $GF(p)$ или $GF(2^m)$.

Вышеприведенный алгоритм предполагает, что число ошибочных разрядов в блоке *не превышает* $(d_{\min}-1)/2$. При большем количестве ошибок в блоке их корректное исправление в общем случае *не гарантируется*.

Пример кодирования. Пусть необходимо закодировать следующее двоичное сообщение:

0000000011 0000000010 0000000001

кодом Рида – Соломона, коэффициенты полиномов $I(x)$ и $G(x)$ которого являются элементами конечного поля $GF(929)$, с $d_{\min}$, равным 5 и с порождающим полиномом

$$G(x)=(x-3)\times\left(x-3^2\right)\times\left(x-3^3\right)\times\left(x-3^4\right) \qquad (2.41)$$

(нетрудно убедиться, что число 3 является примитивным элементом поля $GF(929)$, т. е. остаток от деления 3^{928} на 929 равен единице).

В поле $GF(929)$ данный полином приобретает следующий вид (см. пояснения к правилу 2 в п. 2.2):

$$G(x)=x^4+809x^3+723x^2+568x+522.$$

1) Учитывая, что коэффициенты полиномов $I(x)$ и $G(x)$ в рассматриваемом примере являются элементами конечного поля $GF(929)$, они могут принимать значения от 0 до 928. Поэтому каждый из данных коэффициентов должен представляться 10-битовым двоичным числом, а каждые 10 битов кодируемого сообщения соответствуют одному коэффициенту полинома $I(x)$. Следовательно, в рассматриваемом примере данный полином имеет вид

$$I(x)=3x^2+2x+1.$$

2) Умножаем полином $I(x)$ на x^c, где c – степень порождающего полинома, в рассматриваемом примере равная 4. В результате получаем:

$$I(x)\times x^4=3x^6+2x^5+x^4.$$

3) Делим $I(x)\times x^4$ на порождающий полином. Процесс деления представлен на рис. 2.3. Остаток от деления равен: $547x^3+738x^2+442x+455$.

4) Вычитаем данный остаток *по модулю 929* из полинома $I(x)\times x^4$ и получаем результат кодирования:

$$T(x)=3x^6+2x^5+x^4+382x^3+191x^2+487x+474$$

(см. выражение (2.4) и пояснения к нему).

Пример декодирования. Пусть полученный выше результат кодирования из-за искажения 3-го и 4-го разрядов на приемной стороне приобрел следующий вид:

$$T_R(x) = 3x^6 + 2x^5 + 123x^4 + 456x^3 + 191x^2 + 487x + 474.$$

Естественно, приемнику изначально «не известны» ни номера искаженных разрядов, ни их действительные значения, сформированные на передающей стороне.

Рассмотрим процедуру обнаружения и исправления ошибок на приемной стороне.

1) Делим полином $T_R(x)$ на порождающий полином $G(x)$ (он задается протоколом кодирования и поэтому «известен» декодеру). Остаток от деления равен:

$$\mathrm{Re}\, s[T_R(x)/G(x)] = 779x^3 + 49x^2 + 379x + 417.$$

2) Поскольку остаток не равен нулю, делаем вывод, что принятый блок содержит ошибки. Если протокол помехоустойчивого кодирования оговаривает их исправление методом повторного запроса передачи – такой запрос посылается на передатчик. Если же протокол предполагает их исправление на приемной стороне (предположим последнее) – выполняются локализация ошибочных разрядов, вычисление поправок и их внесение в ошибочные разряды.

3) Локализацию ошибок выполняем по алгоритму Берлекампа – Месси. Вычисляем значения элементов матрицы (2.34), полагая число ошибок в блоке e равным двум, т. е. максимально возможному количеству исправляемых ошибок при равном 5 $d_{\min}$ кода, рассматриваемого в данном примере:

$$T_R(\alpha) = \left\{3\times 3^6 + 2\times 3^5 + 123\times 3^4 + 456\times 3^3 + 191\times 3^2 + 487\times 3 + 474\right\} (\mathrm{mod}\, 929) = 732;$$

$$T_R(\alpha^2) = \left\{3\times 3^{12} + 2\times 3^{10} + 123\times 3^8 + 456\times 3^6 + 191\times 3^4 + 487\times 3^2 + 474\right\} (\mathrm{mod}\, 929) = 637;$$

$$T_R(\alpha^3) = \left\{3\times 3^{18} + 2\times 3^{15} + 123\times 3^{12} + 456\times 3^9 + 191\times 3^6 + 487\times 3^3 + 474\right\} (\mathrm{mod}\, 929) = 762.$$

После этого вычисляем (*по модулю 929*) определитель матрицы (2.34), который в рассматриваемом примере равен:

$$\begin{vmatrix} 732 & 637 \\ 637 & 762 \end{vmatrix} (\mathrm{mod}\, 929) = 588.$$

Поскольку данный определитель не равен нулю, делаем вывод, что число ошибок в принятом блоке действительно равно двум. Для определения позиций ошибочных разрядов составляем матричное уравнение вида (2.35) для определения коэффициентов полинома – локатора ошибок. С учетом того что в рассматриваемом примере

$$-T_R\left(\alpha^{e+1}\right)(\mathrm{mod}\,929)=-T_R\left(\alpha^{3}\right)(\mathrm{mod}\,929)=-762\,(\mathrm{mod}\,929)=929-762=167;$$

$$-T_R\left(\alpha^{e+2}\right)=-T_R\left(\alpha^{2e}\right)=-T_R\left(\alpha^{4}\right)=$$
$$=-\left\{3\times3^{24}+2\times3^{20}+123\times3^{16}+456\times3^{12}+191\times3^{8}+487\times3^{4}+474\right\}(\mathrm{mod}\,929)=$$
$$=-925\,(\mathrm{mod}\,929)=929-925=4$$

[см. выражение (2.4)], получаем следующее матричное уравнение:

$$\begin{pmatrix}732 & 637\\ 637 & 762\end{pmatrix}\times\begin{pmatrix}\Lambda_2\\ \Lambda_1\end{pmatrix}=\begin{pmatrix}167\\ 4\end{pmatrix},$$

которое преобразуем в систему линейных уравнений относительно коэффициентов полинома – локатора ошибок:

$$\begin{cases}732\times\Lambda_2+637\times\Lambda_1=167\\ 637\times\Lambda_2+762\times\Lambda_1=4.\end{cases}$$

Решаем данную систему по применяемому в «классической» алгебре для решения систем линейных уравнений *алгоритму Гаусса*, *но в поле* $GF(929)$. Находим число x, принадлежащее полю $GF(929)$, такое, что $\{732+637x\}(\mathrm{mod}\,929)=0$. Таковым является число 537. Умножив 2-е уравнение данной системы на 537 в поле $GF(929)$, приводим ее к следующему виду:

$$\begin{cases}732\times\Lambda_2+637\times\Lambda_1=167\\ 197\times\Lambda_2+434\times\Lambda_1=290.\end{cases}$$

Почленно складывая 1-е и 2-е уравнения системы в поле $GF(929)$, получаем:

$$142\times\Lambda_1=457.$$

Следовательно, коэффициент Λ_1 равен числу, принадлежащему полю $GF(929)$, произведение которого на 142 по модулю 929 равен 457. Находим, что $\Lambda_1=821$. Подставив данное значение

в 1-е уравнение системы и выполнив необходимые преобразования в поле, получаем уравнение для определения коэффициента Λ_2:

$$732 \times \Lambda_2 = 217.$$

Итак, коэффициент Λ_2 равен числу, принадлежащему полю $GF(929)$, произведение которого на 732 по модулю 929 равен 217. Находим, что $\Lambda_2 = 329$.

Таким образом, искомый полином – локатор ошибок в рассматриваемом примере имеет следующий вид:

$$\Lambda(x) = 329 \times x^2 + 821 \times x + 1. \qquad (2.42)$$

Порядок данного полинома равен двум, поэтому он имеет два корня в поле $GF(929)$, т. е. существует два значения x, принадлежащих полю $GF(929)$, при подстановке которых в полином $\Lambda(x)$ его значение становится равным нулю по модулю 929. Таковыми являются числа $x_1 = 757$ и $x_2 = 562$.

Зная корни полинома – локатора ошибок, определяем номера ошибочных разрядов как значения j, удовлетворяющие условию (2.37) при $\alpha = 3$. Нетрудно показать, что:

$$757 \times 3^3 = 1 \pmod{929};$$
$$562 \times 3^4 = 1 \pmod{929}.$$

Делаем вывод, что ошибочными и подлежащими исправлению являются 3-й и 4-й разряды принятого блока.

4) Исправление ошибочных разрядов будем осуществлять вычитанием из них (*по модулю 929*) поправок, которые, в свою очередь, определим по алгоритму Форни [см. выражения (2.38) – (2.40)].

В соответствии с выражением (2.38), используя вычисленные ранее значения $T_R(\alpha^i)$ $(i = \overline{1,\ldots,4})$, получаем, что в рассматриваемом примере полином частичного синдрома имеет следующий вид:

$$SP(x) = 732 + 637 \times x + 762 \times x^2 + 925 \times x^3.$$

По выражению (2.39) находим *в поле* $GF(929)$ полином оценивания ошибок:

$$\Omega(x) = \mathrm{Re}\,s\left[\frac{\left(732 + 637 \times x + 762 \times x^2 + 925 \times x^3\right) \times \left(329 \times x^2 + 821 \times x + 1\right)}{x^4}\right] = 546 \times x + 732.$$

.

Также *в поле* $GF(929)$ находим 1-ю производную полученного полинома – локатора ошибок (2.42):

$$\Lambda'(x) = 658 \times x + 821.$$

В соответствии с выражением (2.40) рассчитываем поправки, которые необходимо внести в искаженные разряды принятого блока:

$$e_1 = \left[-\frac{\Omega(x_1)\,(\mathrm{mod}\,929)}{\Lambda'(x_1)\,(\mathrm{mod}\,929)} \right] (\mathrm{mod}\,929) = \left[-\frac{(546 \times 757 + 732)\,(\mathrm{mod}\,929)}{(658 \times 757 + 821)\,(\mathrm{mod}\,929)} \right] (\mathrm{mod}\,929) =$$

$$= \left[-\frac{649}{54} \right] (\mathrm{mod}\,929) = [-649(\mathrm{mod}\,929)] \times \left[54^{-1}(\mathrm{mod}\,929)\right] = [280 \times 843](\mathrm{mod}\,929) = 74;$$

$$e_2 = \left[-\frac{\Omega(x_2)\,(\mathrm{mod}\,929)}{\Lambda'(x_2)\,(\mathrm{mod}\,929)} \right] (\mathrm{mod}\,929) = \left[-\frac{(546 \times 562 + 732)\,(\mathrm{mod}\,929)}{(658 \times 562 + 821)\,(\mathrm{mod}\,929)} \right] (\mathrm{mod}\,929) =$$

$$= \left[-\frac{85}{875} \right] (\mathrm{mod}\,929) = [-85(\mathrm{mod}\,929)] \times \left[875^{-1}(\mathrm{mod}\,929)\right] = [844 \times 86](\mathrm{mod}\,929) = 122$$

[при вычислении поправок использованы ранее приведенные выражения (2.4) – (2.6)].

Вычитаем *по модулю 929* каждую из полученных поправок e_i из того ошибочного разряда принятого блока, на который указывает i-й корень полинома – локатора ошибок [см. выражение (2.37)]. В рассматриваемом примере 1-й корень указывает на 3-й разряд, а 2-й – на 4-й. Получаем, что скорректированное значение 3-го разряда равно $(456 - e_1)(\mathrm{mod}\,929) = (456 - 74)(\mathrm{mod}\,929) = 382$, а 4-го разряда – $(123 - e_2)(\mathrm{mod}\,929) = (123 - 122)(\mathrm{mod}\,929) = 1$. Скорректированный блок в полиномиальной форме принимает следующий вид:

$$T_{RC}(x) = 3x^6 + 2x^5 + x^4 + 382x^3 + 191x^2 + 487x + 474.$$

Нетрудно увидеть, что он совпадает с переданным блоком $T(x)$.

Основным достоинством кодов Рида – Соломона является малая избыточность. В самом деле, рассмотренный пример кода Рида – Соломона, формируемый в поле $GF(929)$, с порождающим полиномом (2.41) обеспечивает обнаружение и коррекцию до двух ошибок или обнаружение без коррекции до четырех ошибок в блоке общей разрядностью до 928, из которых четыре разряда – контрольные, а 924 – информационные. При этом каждый из разрядов является 10-битовым целым числом, которое может принимать значения от

нуля до 928. Таким образом, рассмотренный пример кода Рида – Соломона позволяет исправить до 20 или обнаружить до 40 ошибочных битов в 9280-битовом блоке при 40 контрольных битах. С другой стороны, двоичный код БЧХ с тем же количеством обнаруживаемых/исправляемых ошибок и наиболее близкой к 9280 разрядностью блока, $2^{13}-1=8191$, должен содержать 260 контрольных битов [8].

Основной недостаток кодов Рида – Соломона заключается в сложности как аппаратной, так и программной реализации процедур кодирования и декодирования. Однако в настоящее время, благодаря возможности как реализации специализированных кодеров/декодеров на ПЛИС, так и применения высокопроизводительных средств вычислительной техники общего назначения, данный недостаток не существен.

2.5. Сверточные помехоустойчивые коды

2.5.1. Математическое описание и классификация сверточных помехоустойчивых кодов

Сверточное кодирование осуществляется путем преобразования каждого слова разрядностью K кодируемых данных в N-разрядное слово ($N > K$), каждый из разрядов которого формируется как некоторая функция не только от соответствующего ему K-разрядного слова входной последовательности, но и от M предыдущих слов как входной, так и (в общем случае) выходной последовательности (см. п. 2.1). Следовательно, сверточные коды обладают *памятью*. Значение $M+1$ называется при этом *длиной кодового ограничения* (*constraint length*). Сверточный код с разрядностью кодового слова N, разрядностью соответствующего ему фрагмента исходной двоичной последовательности K и длиной кодового ограничения $M+1$ в литературе часто обозначается как сверточный код $(N, K, M+1)$, например, код (2, 1, 3).

Часто сверточное и блочное помехоустойчивое кодирование взаимно дополняют друг друга. Образно говоря, они являются соответственно первым и вторым «рубежами обороны» в борьбе с ошибками обмена данными (см. п. 2.6).

В принципе, существуют как двоичные, так и недвоичные сверточные помехоустойчивые коды (СПК). Однако последние не нашли сколько-нибудь широкого практического применения.

Обобщенное уравнение двоичного сверточного кодирования имеет следующий вид [6]:

$$y_i[j] = \left\{ \sum_{k=0}^{K-1} \sum_{m=0}^{M} a_{ikm} \times x_k[j-m] \right\} \oplus \left\{ \sum_{n=0}^{N-1} \sum_{m=1}^{M} b_{inm} \times y_n[j-m] \right\}, \qquad (2.43)$$

где $y_i[j]$ и $y_n[j-m]$ – соответственно i-й бит j-го и n-й бит $(j-m)$-го N-разрядного слова сверточного кода ($i = 0,..., N-1$);

$x_k[j-m]$ – k-й бит $(j-m)$-го K-разрядного слова исходной двоичной последовательности ($k = 0,..., K-1$);

a_{ikm} и b_{inm} – коэффициенты, зависящие от конкретной разновидности сверточного кода, значения которых могут быть равны нулю или единице;

$M+1$ – длина кодового ограничения,

причем суммирование осуществляется *по модулю 2*.

Таким образом, биты некоторого j-го N-разрядного двоичного слова, получаемого в результате сверточного кодирования, в общем случае являются функциями от битов j-го и M предшествующих ему K-разрядных слов исходной двоичной последовательности, а также M предшествующих ему N-разрядных слов результата кодирования. СПК, обладающие такими свойствами, называют *рекурсивными*. СПК, для которых характерна зависимость результата кодирования только от исходной двоичной последовательности, т. е. у которых все коэффициенты b_{inm} равны нулю, называются *нерекурсивными*. Другими словами, в нерекурсивном сверточном кодере отсутствуют обратные связи.

Одним из простейших примеров *нерекурсивного* СПК является нерекурсивный код (2, 1, 3) [6]. Он характеризуется заменой каждого из битов исходной двоичной последовательности 2-битовым двоичным словом, формируемым в соответствии со следующими выражениями:

$$\left.\begin{aligned} y_0[j] &= x_0[j] \oplus x_0[j-1] \oplus x_0[j-2]; \\ y_1[j] &= x_0[j] \oplus x_0[j-2], \end{aligned}\right\} \qquad (2.44)$$

т. е. коэффициенты a_{000}, a_{001}, a_{002}, a_{100} и a_{102} нерекурсивного сверточного кода (2, 1, 3) равны единице, остальные коэффициенты a_{ikm} и все коэффициенты b_{inm} – нулю [см. выражение (2.43)], а длина кодового

ограничения – 3, т. е. некоторое j-е 2-битовое слово кода (2, 1, 3) является функцией от трех 1-битовых слов (j-го и двух, предшествующих ему) исходной двоичной последовательности.

Простейшим примером *рекурсивного* СПК является рекурсивный код (2, 1, 4) [6]. Для него характерно разбиение подвергаемой кодированию последовательности на 1-битовые двоичные слова, каждое из которых заменяется на 2-битовое слово, формируемое в соответствии со следующими выражениями:

$$\left.\begin{aligned} y_0[j] = x_0[j] \oplus x_0[j-1] \oplus x_0[j-3] \oplus y_0[j-2] \oplus y_0[j-3]; \\ y_1[j] = x_0[j], \end{aligned}\right\} \qquad (2.45)$$

т. е. коэффициенты a_{000}, a_{001}, a_{003}, a_{100}, b_{002} и b_{003} рекурсивного СПК (2, 1, 4) равны единице, остальные коэффициенты a_{ikm} и b_{inm} – нулю, а длина кодового ограничения – 4.

Сопоставляя выражения (2.44) и (2.45), можно заметить следующее. В N-битовом (где $N = 2$) слове рекурсивного СПК (2, 1, 4) в явном виде присутствует соответствующее ему K-битовое ($K = 1$) слово исходной двоичной последовательности. Обладающие таким свойством СПК называются *систематическими* [6]. С другой стороны, в N-битовом слове нерекурсивного СПК (2, 1, 3) исходное K-битовое слово в явном виде отсутствует. Такие сверточные коды известны под названием *несистематических* [6]. Следует отметить, что большинство используемых на практике нерекурсивных СПК относится к категории несистематических, а рекурсивных – к систематическим СПК [6].

2.5.2. Основы сверточного кодирования

Общие принципы рекурсивного и нерекурсивного сверточного кодирования сходны между собой [6], однако первое из них несколько сложнее в реализации и менее наглядно для объяснения. В дальнейшем принципы сверточного кодирования и декодирования будут рассматриваться, в основном, на примере несистематических нерекурсивных кодов. Вопросы рекурсивного сверточного кодирования будут затрагиваться лишь при необходимости.

Ввиду наличия *памяти* у сверточного кодера процессы сверточного кодирования и декодирования удобно описывать *диаграммами*

переходов, подобными диаграммам состояний *конечных автоматов*, т. е. цифровых устройств с памятью [6]. В принципе, сверточные кодеры могут описываться диаграммами, полностью аналогичными «классическим» диаграммам состояний цифровых автоматов. Однако более информативной формой представления процесса сверточного кодирования являются так называемые *решетчатые диаграммы* [6], отображающие последовательность изменений состояний кодера и его выходных сигналов.

Решетчатая диаграмма сверточного кода является графическим представлением правил кодирования. Она отображает возможные переходы между соседними по времени состояниями кодера, т. е. между его состояниями в некотором j-м и следующем за ним $(j+1)$-м тактах кодирования. При этом под *состоянием* сверточного кодера в j-м такте понимается комбинация M K-битовых фрагментов исходной двоичной последовательности, предшествующих некоторому ее j-му K-битовому фрагменту.

Правила построения решетчатой диаграммы сверточного кода иллюстрируются решетчатой диаграммой кода (2, 1, 3), описываемого выражениями (2.44) (рис. 2.6).

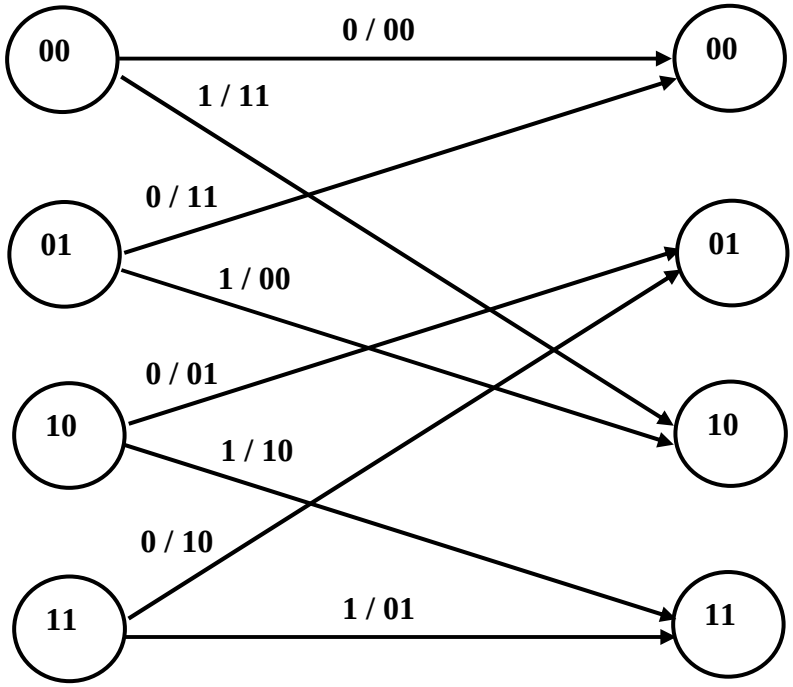


Рис. 2.6. Решетчатая диаграмма СПК (2, 1, 3), описываемого выражениями (2.44)

Состояния кодера обозначаются на диаграмме кружками, называемыми *узлами* диаграммы, с указанием соответствующих им бито-

вых комбинаций внутри кружка или рядом с ним. На рис. 2.6 в качестве старшего бита состояния служит значение $x_0[j-1]$, а младшего – $x_0[j-2]$. Стрелки, снабжаемые обозначениями вида $X[j]/Y[j]$ на них (см. рис. 2.6), указывают направления переходов между состояниями кодера в j-м и в $(j+1)$-м тактах при поступлении на его вход в j-м такте некоторой K-битовой комбинации $X[j]$. $Y[j]$ – выходное N-битовое слово кодера в j-м такте при подаче на его вход двоичного слова $X[j]$ и состоянии кодера, из которого исходит соответствующая стрелка. Например, при состоянии 00 сверточного кодера (2, 1, 3) и поступлении на его вход единицы на выход кодера будет выдано слово 11, и он перейдет в состояние 10 (см. рис. 2.6).

Понятие решетчатой диаграммы существует и для рекурсивных кодов. Правила их построения, в целом, аналогичны вышеописанным. Примеры решетчатых диаграмм рекурсивных кодов представлены, например, в [6].

На основе решетчатой диаграммы сверточного кода описываются процедуры сверточного кодирования и декодирования двоичных последовательностей. Первая из названных процедур представляется *трассой переходов состояний и выходов* кодера при подаче на его вход кодируемой двоичной последовательности. *Начальное состояние* кодера при этом полагается нулевым [6].

В качестве примера на рис. 2.7 представлена подобная трасса кодера сверточного кода (2, 1, 3), описываемого выражениями (2.44), при подаче на его вход двоичной последовательности *1001*.

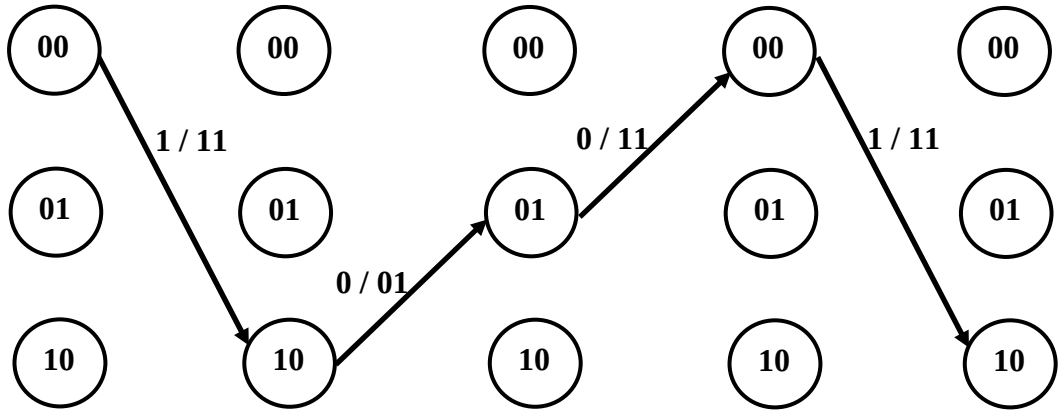


Рис. 2.7. Трасса переходов состояний и выходов кодера СПК (2, 1, 3), описываемого рис. 2.6 (входная последовательность – *1001*, выходная – *11011111*)

2.5.3. Декодирование СПК

Декодирование СПК состоит в восстановлении исходной двоичной последовательности из результата сверточного кодирования, переданного через канал связи или считанного с носителя, т. е. в коррекции ошибок передачи или записи/считывания. При этом использование СПК только для обнаружения ошибок не рационально и на практике не используется. В общем случае результат восстановления может не совпадать с исходной двоичной последовательностью, т. е. содержать остаточные ошибки.

Декодирование СПК может быть реализовано несколькими способами [6], большинство из которых основывается на применении *трасс переходов состояний и выходов* декодера. На рис. 2.8 в качестве простейшего примера приведена подобная трасса декодера нерекурсивного сверточного кода (2, 1, 3), описываемого выражениями (2.44), при поступлении на его вход *разрешенной* двоичной последовательности *11011111* (см. рис. 2.7).

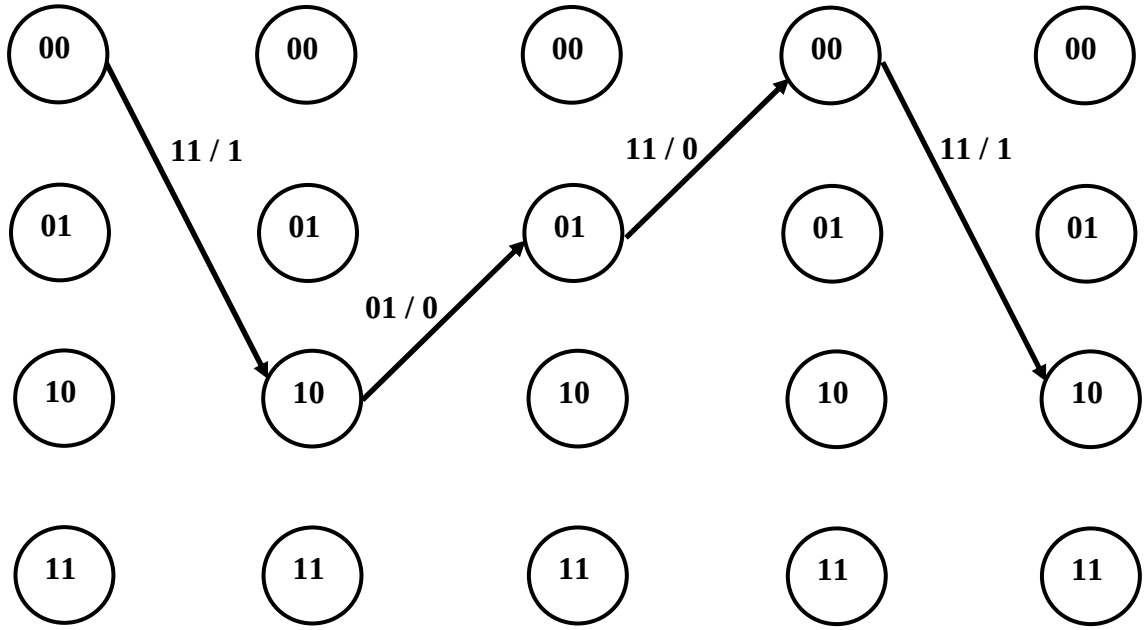


Рис. 2.8. Трасса переходов состояний и выходов декодера нерекурсивного СПК (2, 1, 3), описываемого рис. 2.6, при подаче на вход разрешенной последовательности *11011111* (выходная битовая комбинация декодера – *1001*)

Данная трасса построена на основе решетчатой диаграммы указанного кода (см. рис. 2.6). Переходы между состояниями декодера

определяются очередным N-битовым словом сверточного кода (см. рис. 2.8), а не, как при кодировании, очередной K-битовой комбинацией исходной двоичной последовательности (см. рис. 2.7), которая при декодировании изначально неизвестна и определяется на основании текущего состояния декодера, а также соответствующего ему входного N-битового слова. Начальное состояние декодера, как и кодера, всегда полагается нулевым (см. рис. 2.8).

В общем случае поступающая на вход декодера кодовая комбинация может содержать ошибки, а задача сверточного декодирования заключается в максимально достоверном восстановлении исходной битовой комбинации из входной двоичной последовательности декодера. При этом наличие или отсутствие в ней искаженных битов, а также их позиции изначально «неизвестны» на приемной стороне.

Известен ряд алгоритмов такого восстановления [6, 7]. Большинство из них реализует принцип *максимального правдоподобия*. Он заключается в том, что при декодировании на основе полученной на приемной стороне двоичной последовательности восстанавливается наиболее вероятная (в соответствии с критериями, используемыми конкретным алгоритмом) трасса переходов состояний и выходов кодера. Наиболее простым по сущности (но не с точки зрения аппаратно-программных затрат на реализацию) из алгоритмов этой группы является следующий [6]:

1. На основании решетчатой диаграммы соответствующего сверточного кода строится совокупность *всех* возможных трасс переходов состояний и выходов декодера при разрядности входной двоичной последовательности, равной разрядности декодируемой битовой комбинации.

2. Для каждой из построенных трасс вычисляется *метрика ветвлений* (в дальнейшем – *метрика*). Она определяется как сумма различий между значениями битов реально принятой декодером двоичной последовательности и одноименных битов его входной последовательности, при которой смена состояний декодера описывалась бы соответствующей трассой.

3. Трасса с минимальной метрикой, т. е. соответствующая входной последовательности N-битовых слов декодера с минимальным отклонением от реально принятой, полагается наиболее правдоподобным повторением трассы переходов состояний и выходов кодера. По ней восстанавливается соответствующая ей последовательность K-битовых слов, служащая результатом декодирования. В общем

случае из-за ограниченных корректирующих возможностей сверточных кодов она может содержать ошибки, т. е. не совпадать с входной двоичной последовательностью сверточного кодера передающего абонента.

Следует также отметить, что в ряде частных случаев критерию минимальной метрики может удовлетворять не одна, а несколько трасс. При этом для выбора максимально правдоподобной из них необходимо применение специальных алгоритмов [6].

Вышеописанный алгоритм иллюстрирует рис. 2.9.

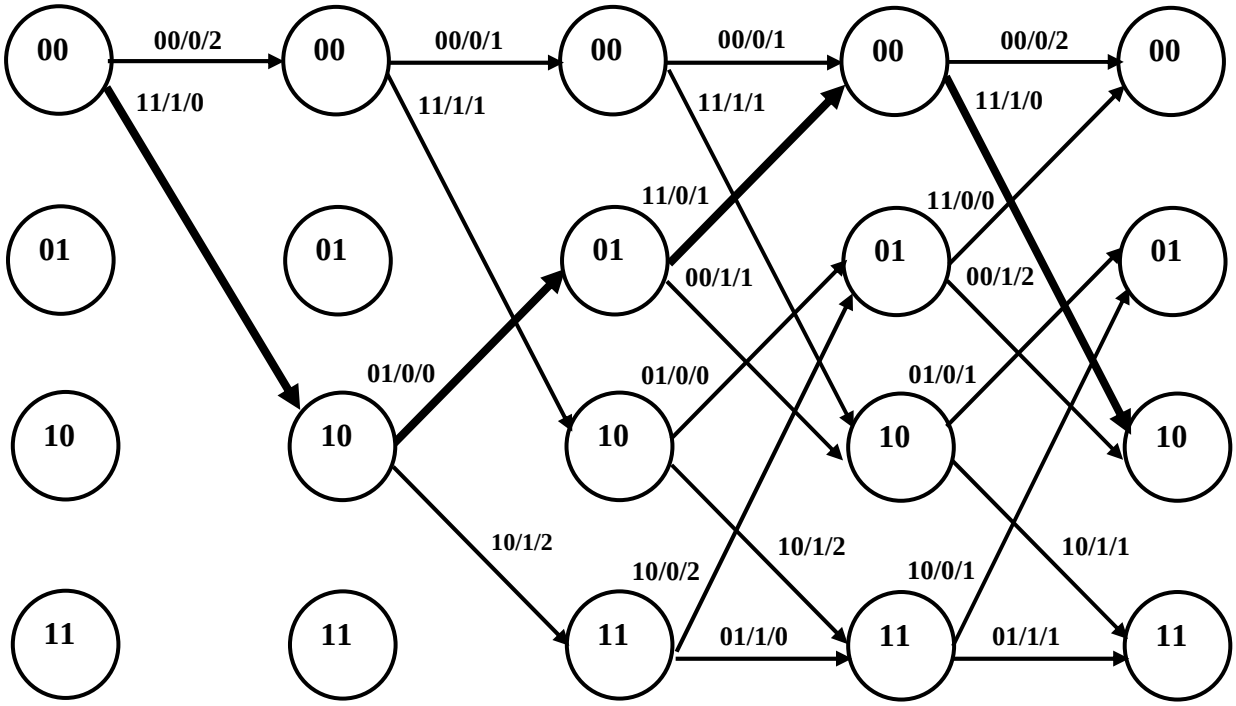


Рис. 2.9. Пример реализации сверточного декодирования
по принципу максимального правдоподобия
(код – нерекурсивный (2, 1, 3), описываемый рис. 2.6;
входная последовательность – *11010111*;
результат декодирования – *1001*)

На рис. 2.9 представлена совокупность всех возможных трасс переходов состояний декодера нерекурсивного сверточного кода (2, 1, 3), описываемого рис. 2.6, при подаче на его вход 8-битовой двоичной последовательности. Предполагается, что реально на его вход поступила *запрещенная* битовая комбинация *11010111*, представляющая собой двоичную последовательность *11011111* с одним искаженным битом. При этом рядом с каждой из стрелок, составляющих трассы, указана, кроме соответствующих ей (т. е. обозначаемому этой стрел-

кой переходу) N-битовой входной и K-битовой выходной комбинации декодера (см. рис. 2.6), также *метрика* данной стрелки. Она, аналогично метрике трассы, определяется как сумма различий между значениями битов N-разрядной входной двоичной последовательности декодера, которая вызвала бы описываемую данной стрелкой смену его состояния, и значениями одноименных битов реально принятого декодером N-разрядного слова, соответствующего указанной смене состояния по его позиции в декодируемой последовательности. Метрика трассы вычисляется как сумма метрик составляющих ее стрелок.

Из рис. 2.9 видно, что наименьшим значением метрики и, следовательно, максимальным правдоподобием обладает трасса, обозначенная стрелками большей интенсивности. Поэтому в соответствии с рассматриваемым алгоритмом принимается решение о том, что при поступлении запрещенной двоичной последовательности *11010111* на вход декодера нерекурсивного сверточного кода (2, 1, 3) наиболее вероятным является отправление передатчиком приемнику битовой комбинации *11011111*, что и имело место на самом деле. Она, в свою очередь, представляет собой результат сверточного кодирования двоичного слова *1001* (см. рис. 2.7 и 2.9), которое при этом и служит результатом декодирования.

В рассмотренном примере результат декодирования не содержит ошибок. В общем случае, как указано выше, он может содержать остаточные ошибки, которые могут быть обнаружены и/или устранены, например, при сочетании сверточного помехоустойчивого кодирования с блочным (см. п. 2.6).

Представленный выше алгоритм является простым по сущности, но «неэкономным» с точки зрения затрат памяти и времени на его реализацию из-за необходимости анализа *всех* возможных трасс переходов состояний и выходов декодера. Поэтому на практике обычно применяются другие алгоритмы сверточного кодирования, также основанные на принципе максимально правдоподобного восстановления трассы переходов состояний декодера, но более «экономные».

Наиболее распространенными из них являются различные варианты *алгоритма Витерби* [6, 7]. Их основным отличием от вышеописанного алгоритма является исключение заведомо наименее правдоподобных трасс из процесса дальнейшего анализа на каждом шаге декодирования. Один из простейших вариантов алгоритма Витерби основывается на исключении указанных трасс из рассмотрения по

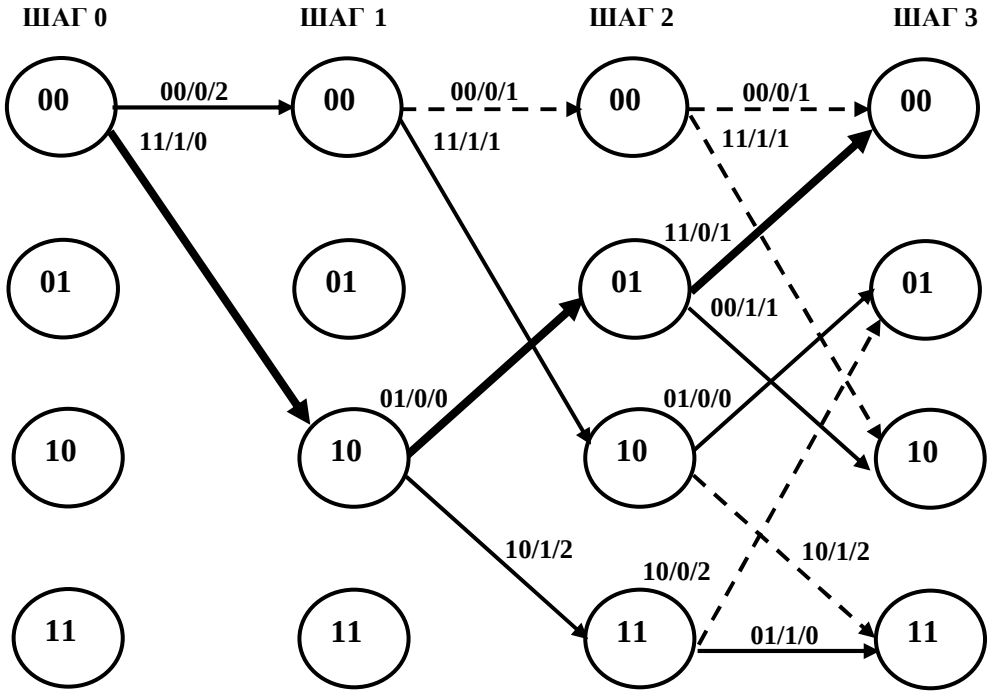
следующему правилу. Если на некотором шаге декодирования в какой-либо узел входит две или несколько различных трасс, то из дальнейшего анализа исключаются все из них, кроме обладающей наименьшей среди них метрикой. Если существует несколько трасс, удовлетворяющих этому критерию, то по простейшему варианту алгоритма оставляются все они. При этом трассы, оставшиеся на последнем шаге декодирования после исключения всех наименее правдоподобных, называются *«выжившими»*.

Трасса, обладающая минимальной метрикой среди «выживших», полагается наиболее правдоподобной. Как и при использовании ранее рассмотренного алгоритма (см. рис. 2.9), критерию минимальной метрики могут удовлетворять несколько из «выживших» трасс. Наиболее правдоподобная из них выбирается по специальным алгоритмам [6, 7].

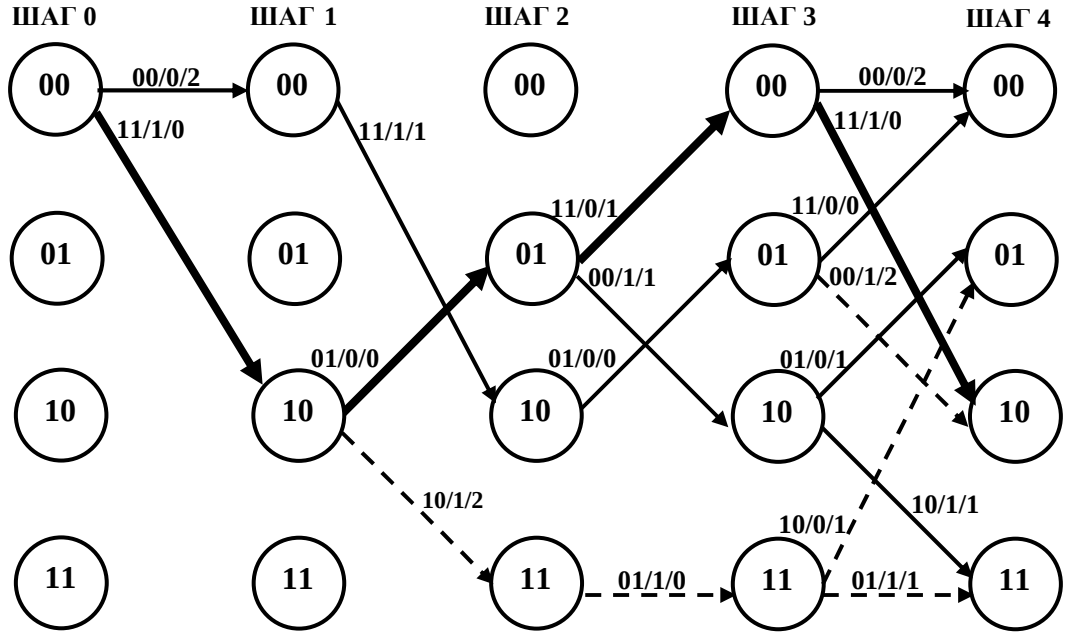
Вышеописанный простейший вариант алгоритма Витерби иллюстрирует рис. 2.10, на котором представлены трассы переходов состояний и выходов декодера нерекурсивного сверточного кода (2, 1, 3) на 3-м и 4-м шаге декодирования, с указанием исключаемых трасс пунктирными стрелками. На 0-м, 1-м и 2-м шаге трассы, подлежащие исключению в соответствии с вышеприведенным правилом, еще не могут быть выявлены (см. рис. 2.9). Минимальной метрикой среди «выживших» характеризуется трасса, отмеченная стрелками большей интенсивности. Ей соответствует результат декодирования *1001*, совпадающий с полученным по ранее рассмотренному алгоритму (см. рис. 2.9).

Алгоритм Витерби характеризуется меньшими затратами памяти и времени анализа, чем ранее описанный алгоритм, за счет исключения заведомо неправдоподобных трасс. С другой стороны, он более сложен в реализации (что, однако, не является серьезным недостатком при существующем уровне развития аппаратно-программных средств кодирования/декодирования). Поэтому алгоритм Витерби достаточно широко применяется на практике [6, 7]. Он не гарантирует отсутствия остаточных ошибок в декодированной двоичной последовательности, которые могут быть обнаружены и/или устранены, например, при сочетании сверточного помехоустойчивого кодирования с блочным (см. п. 2.6).

Кроме вышерассмотренных, существуют и другие алгоритмы сверточного декодирования (в том числе более «развитые» варианты алгоритма Витерби) [6, 7].



а



б

Рис. 2.10. Пояснения к процессу сверточного декодирования по алгоритму Витерби на шаге 3 (*а*) и 4 (*б*) (код – нерекурсивный (2, 1, 3), описываемый рис. 2.6; входная последовательность – *11010111*; результат декодирования – *1001*; пунктиром обозначены исключаемые из рассмотрения трассы)

Декодирование *рекурсивных* сверточных кодов, в целом, реализуется по тем же общим принципам, что и нерекурсивных, в том числе на основе принципа максимального правдоподобия. Однако конкретные алгоритмы декодирования рекурсивных кодов отличаются от таковых нерекурсивных кодов [7, 8]. В частности, на практике достаточно широко распространены алгоритмы группы *MAP* (*Maximum A Posteriori – максимальная апостериорная вероятность*) [6, 7].

2.5.4. Корректирующие способности СПК

Как указано ранее, СПК используются только для коррекции (но не для обнаружения) ошибок. Поэтому практический смысл имеет только выражение для определения максимального количества ошибок, *корректируемых* СПК.

Не существует подобного выражения, справедливого для всех разновидностей СПК и для всех возможных вариантов распределения ошибочных битов в декодируемой последовательности, подобного выражению (2.20) для блочных кодов. Приближенно корректирующая способность СПК может быть оценена по следующему выражению [6] [ср. с выражением (2.20)]:

$$e_{\max corr} = \lfloor (d_{f\min} - 1)/2 \rfloor, \qquad (2.46)$$

где $e_{\max corr}$ – максимальное количество корректируемых ошибок в фрагменте разрядностью $(M+1)N/K$ входной битовой последовательности;

$d_{f\min}$ – *минимальное свободное кодовое расстояние* или *минимальный просвет* (*minimum free distance*) СПК, определяемый как суммарное количество единичных битов в выходном коде решетчатого кодера соответствующего СПК, описываемом трассой переходов состояний минимальной длины, начинающейся и заканчивающейся в нулевом состоянии [6].

Такая трасса для кода, описываемого рис. 2.6, представлена на рис. 2.11, из которого следует, что значение $d_{f\min}$ данного СПК равно 5. Следовательно, данный СПК позволяет исправить максимум две ошибки в каждом из фрагментов разрядностью $(3\times 2)/1 = 6$ бит входной битовой последовательности декодера.

Для ряда вариантов распределения ошибочных битов в декодируемой последовательности выражение (2.46) дает *завышенную* оценку числа исправляемых битов. Методики более точной оценки корректирующих способностей СПК описаны, например, в [6].

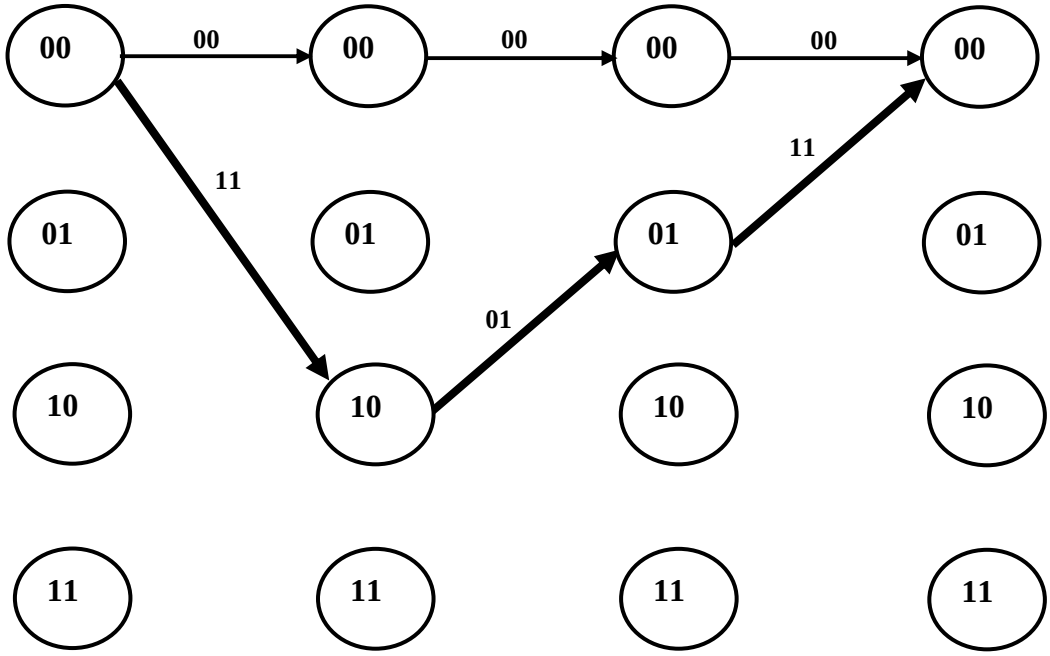


Рис. 2.11. Пояснение понятия минимального просвета СПК

2.5.5. Повышение эффективности СПК

Основными *недостатками* описываемых выражением (2.43) «классических» сверточных кодов, ограничивающими их эффективность, являются:

- увеличение разрядности исходной двоичной последовательности при кодировании в N / K раз, например, при (2, 1, 3)-кодировании – в 2 раза, что приводит к снижению эффективной скорости обмена данными в такое же количество раз, так как N-битовое двоичное слово сверточного кода реально несет информацию только о K битах исходной последовательности (в то время как для используемых на практике блочных кодов характерно снижение скорости обмена на 10 – 20 % [6 – 8]);
- ограниченные возможности или невозможность достоверной (без остаточных ошибок) коррекции *групповых искажений битов*, т.е. следующих подряд двух или нескольких искаженных битов (что весьма часто имеет место на практике).

Однако современные методы и алгоритмы сверточного кодирования позволяют частично устранить перечисленные недостатки. Основным методом устранения *первого* из них является так называемое *выкалывание*, или *перфорация* (*code puncturing*) [7]. Принцип выкалывания заключается в избирательном удалении некоторых избыточных битов из двоичной последовательности, являющейся результатом сверточного кодирования. Такой подход в ряде случаев позволяет достигнуть рационального компромисса между возможностью исправления ошибок и степенью избыточности сверточного кода (т. е. степенью увеличения разрядности результата сверточного кодирования по сравнению с исходной двоичной последовательностью).

Общий принцип реализации выкалывания состоит в следующем. Позиции удаляемых битов выходной последовательности кодера задаются так называемой *матрицей выкалывания*, определяемой типом сверточного кода и конкретным протоколом кодирования. Естественно, данная матрица должна быть «известна» как на передающей, так и на приемной стороне. Число строк матрицы выкалывания равно разрядности N выходного слова кода, а число столбцов – периоду выкалывания, т. е. периоду повторения позиций удаляемых битов. Ниже представлен один из примеров матрицы выкалывания, используемой в сверточном кодировании при разрядностях входного и выходного слова, равных одному и двум битам соответственно:

$$\boldsymbol{P}=\begin{pmatrix}1 & 0 & 1\\ 1 & 1 & 0\end{pmatrix}. \qquad (2.47)$$

Данная матрица соответствует следующему алгоритму выкалывания: в последовательности выходных 2-битовых слов в каждом $3j$-м слове не удаляется ни один бит, в каждом $(3j+1)$-м – младший бит, а в каждом $(3j+2)$-м – старший (где j – целое неотрицательное число). Таким образом, каждым трем битам входной двоичной последовательности соответствуют четыре бита выходной; поэтому говорят, что относительная скорость передачи данных после выкалывания равна 3/4.

Следует отметить, что применяемые на практике матрицы выкалывания, в том числе вышеприведенная, составляются таким образом, чтобы минимизировать обусловленные выкалыванием ошибки декодирования.

На приемной стороне перед декодированием в позиции удаленных при выкалывании битов заносятся некоторые значения (нули или

единицы), определяемые конкретным алгоритмом декодирования. Этим битам присваивается *нулевой уровень доверия*, что учитывается при декодировании [7]. Например, они могут игнорироваться в процессе вычисления метрик ветвлений при декодировании по алгоритмам Витерби. Естественно, это снижает достоверность декодирования. Однако при рациональном выборе матрицы выкалывания и алгоритма декодирования может быть обеспечено приемлемое сочетание скорости обмена данными и вероятности ошибок декодирования.

Возможность коррекции *групповых искажений битов* при применении сверточных кодов улучшается при использовании процедуры *перемежения* (*interleaving*) [6, 7]. В простейшем случае оно состоит в следующем. На передающей стороне биты полученной в результате сверточного кодирования двоичной последовательности *переставляются местами* по определенным правилам, зависящим от конкретного алгоритма перемежения. В частности, на практике достаточно широко применяется *псевдослучайная перестановка*, реализуемая в соответствии с выражением

$$Inr[i] = Cd[Psr(i)], \qquad (2.48)$$

где $Inr[i]$ – i-й бит подвергнутой перемежению двоичной последовательности, причем $i = 0,..., L-1$, где L – ее разрядность (в битах);

$Psr(i)$ – i-й элемент последовательности целых псевдослучайных чисел, находящихся в пределах от 0 до $L-1$;

$Cd[Psr(i)]$ – бит с номером $Psr(i)$ исходной (подвергаемой перемежению) двоичной последовательности разрядностью L.

Псевдослучайная последовательность чисел $Psr(i)$ формируется генератором псевдослучайной последовательности (ГПП), реализуемым в аппаратной или программной форме. Известны и другие алгоритмы перемежения, например: на основе циклического сдвига исходной последовательности; четно-нечетной перестановки и ряд других [6, 7].

После выполнения операции перемежения смежные биты результата сверточного кодирования оказываются разнесенными между собой на несколько позиций, минимальное количество которых также определяется алгоритмом перемежения. После процедуры перемежения двоичная последовательность преобразуется в сигнал-носитель, посредством которого осуществляется ее передача через канал связи или ее запись на накопитель информации. На приемной стороне, после извлечения из сигнала-носителя представляемой им битовой

последовательности, она подвергается процедуре *деперемежения*, т. е. восстановления порядка следования битов, который имел место до выполнения операции перемежения на передающей стороне (рис. 2.12).

Рис. 2.12. Операционная модель сверточного кодирования/декодирования с перемежением

Например, при перемежении методом псевдослучайной перестановки деперемежение осуществляется посредством преобразования, обратного описываемому выражением (2.48), с применением ГПП, идентичного используемому при перемежении. Полученная в результате деперемежения двоичная последовательность затем подвергается сверточному декодированию. При этом, если в процессе передачи будут искажены несколько соседних битов представленной сигналом-носителем двоичной последовательности (вероятность чего велика), после деперемежения они окажутся разнесенными между собой (рис. 2.13). Это повышает достоверность коррекции таких искажений при сверточном декодировании [7].

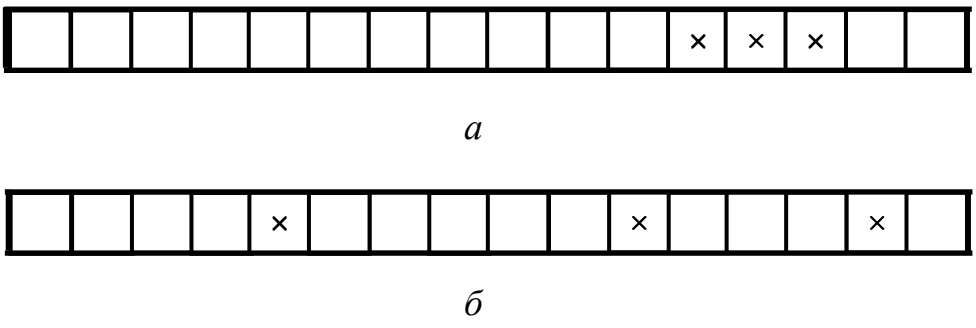


Рис. 2.13. Примеры последовательностей битов, полученных в результате детектирования (*а*) и последующего деперемежения (*б*) (косым крестом отмечены искаженные биты)

Применяется также перемежение и с детерминированным, определяемым протоколом кодирования, назначением позиций битов при их перестановке [6, 7].

2.6. Применение блочных и сверточных кодов. Основные методы повышения эффективности помехоустойчивого кодирования

2.6.1. Сопоставление блочных и сверточных кодов

Большинство распространенных на практике *блочных* помехоустойчивых кодов характеризуется следующими особенностями [6 – 8]:

- при относительно небольшой избыточности (на практике – не более 10 – 20 %) они позволяют эффективно *обнаруживать* до $d_{\min} - 1$ ошибок в блоке на приемной стороне [см. выражение (2.19)];

- блочные коды во многих практических случаях не позволяют выявить факт превышения количеством ошибок в блоке значения $\lfloor (d_{\min} - 1)/2 \rfloor$ [см. выражение (2.20)] после их обнаружения; как следствие, *исправление* ошибок *на приемной стороне* применимо только при пренебрежимо малой вероятности появления в блоке более $\lfloor (d_{\min} - 1)/2 \rfloor$ ошибок (что часто не может быть гарантировано на практике);

- ввиду предыдущего свойства блочных кодов, при помехоустойчивом кодировании, основанном на применении только «классических» блочных кодов, например ЦПК, исправление ошибок в большинстве практических случаев рационально осуществлять методом *ARQ* (см. рис. 1), т. е. повторной передачи или считывания блока, в котором обнаружены ошибки, по запросу приемника;

- блочные коды требуют приема всего блока до начала процедуры декодирования, что в совокупности с относительно большими размерами блоков, особенно при использовании ЦПК (см. п. 2.4), в ряде практических случаев неприемлемо при обмене данными *в реальном масштабе времени*.

В свою очередь, *сверточные* коды отличаются такими особенностями, как [6 – 8]:

- их назначение не обнаружение, а *коррекция* ошибок на приемной стороне *в реальном масштабе времени* (непосредственно в процессе приема сообщения, с минимальной задержкой между входной и выходной битовыми последовательностями декодера);

- для сверточных кодов в общем случае характерно наличие остаточных ошибок в результатах декодирования.

Из вышеизложенного можно сделать следующие выводы:

1. В системах обмена данными, характеризуемых:

- малой вероятностью искажения бита, *Bit Error Rate* (*BER*) (от 10^{-5} – 10^{-6} и менее), в процессе передачи или записи/считывания данных;

- недопустимостью остаточных ошибок в окончательных результатах восстановления исходных данных на приемной стороне;

- отсутствием необходимости в обмене данными в реальном масштабе времени (допустимостью задержек при обработке данных на приемной стороне, в частности, исправления ошибок путем повторной передачи блока),

помехоустойчивое кодирование рационально осуществлять посредством *блочных* помехоустойчивых кодов, в первую очередь ЦПК.

К таким системам относятся, например, кабельные каналы связи вычислительных систем и локальных вычислительных сетей. Применение именно данного класса кодов предусматривают реальные протоколы помехоустойчивого кодирования подобных систем. При этом помехоустойчивое декодирование на приемной стороне, как правило, сводится только к проверке наличия ошибок в блоке, а их исправление осуществляется повторной передачей или считыванием искаженного блока по запросу приемника. Благодаря высокой эффективности блочных кодов при обнаружении ошибок такой подход позволяет свести к минимуму остаточные ошибки. С другой стороны, благодаря малой вероятности появления ошибок в блоках частота их повторной передачи или считывания невысока. Поэтому данные процедуры не оказывают существенного влияния на скорость обмена данными в целом.

2. При процедурах обмена данными, характеризуемых:

- относительно высокими значениями BER (порядка 10^{-3} – 10^{-4}) в процессе передачи или записи/считывания данных;

- обменом в реальном масштабе времени (необходимостью минимизации задержек обработки данных на приемной стороне);

- допустимостью остаточных ошибок в окончательных результатах восстановления исходных данных на приемной стороне,

помехоустойчивое кодирование рационально осуществлять посредством *сверточных* кодов.

К подобным процедурам относится, в частности, обмен речевыми данными по беспроводным каналам связи. Протоколы обмена такими данными предусматривают применение именно СПК в качестве основного способа помехоустойчивого кодирования. С одной стороны, СПК позволяют в реальном масштабе времени скорректировать часть ошибок, а с другой – остаточные ошибки, вероятность которых при помехоустойчивом кодировании на основе СПК достаточно высока, не оказывают существенного влияния на достоверность обмена речевыми сообщениями (в отличие от, например, достоверности обмена текстовой информацией).

3. В системах обмена данными, характеризуемых:

- относительно высокими значениями *BER* (порядка 10^{-3} – 10^{-4}) в процессе передачи или записи/считывания данных;

- недопустимостью остаточных ошибок обмена данными (к ним относятся, например, беспроводные каналы связи локальных вычислительных сетей), ни «классические» блочные, ни «классические» сверточные помехоустойчивые коды, описанные ранее, по отдельности не позволяют достигнуть приемлемого сочетания достоверности и скорости обмена. Поэтому в таких системах необходимо применение специальных методов *повышения эффективности* помехоустойчивого кодирования. Основными из них являются [6, 7]:

- *каскадное помехоустойчивое кодирование;*

- *турбо-кодирование.*

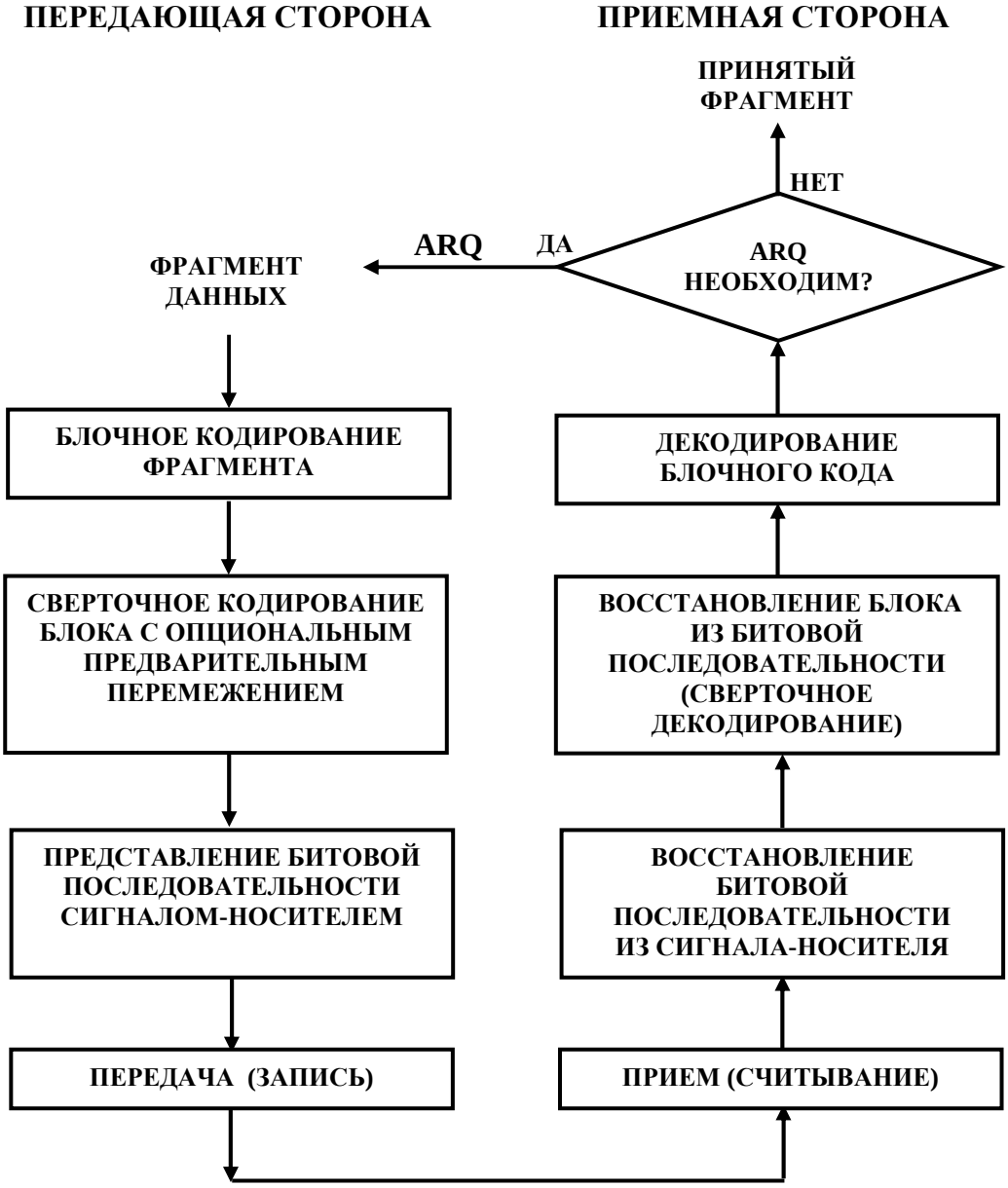
2.6.2. Каскадное помехоустойчивое кодирование

Принцип *каскадного помехоустойчивого кодирования* [6, 7] был предложен и получил распространение раньше, чем турбо-кодирование. Данный принцип состоит в сочетании блочного и сверточного кодирования (рис. 2.14).

Подлежащий передаче или записи фрагмент вначале подвергается блочному помехоустойчивому кодированию. Затем сформированная в его результате двоичная последовательность (блок) подвергается сверточному кодированию, преобразуется в сигнал-носитель и передается принимающему абоненту или записывается на носитель.

На приемной стороне из сигнала-носителя восстанавливается представляемая им битовая последовательность, которая в общем случае не совпадает с переданной последовательностью из-за ошибок

передачи или записи/считывания. Затем осуществляются выявление и коррекция (в общем случае – частичная) ошибок в принятой двоичной последовательности, реализуемые в два этапа.



ARQ (*Automatic Repetition reQuest*)
– автоматический запрос повторной передачи

Рис. 2.14. Операционная модель каскадного помехоустойчивого кодирования/декодирования

Первым этапом является сверточное декодирование, в процессе которого указанные ошибки частично корректируются. Результатом сверточного декодирования является блок данных, содержащий остаточные ошибки, что обусловлено ограниченными корректирующими возможностями сверточного декодирования.

Наличие остаточных ошибок в блоке выявляется на втором этапе помехоустойчивого декодирования – при декодировании блочного кода. Оно, как указано ранее, позволяет достаточно эффективно обнаруживать ошибки в блоке, однако в общем случае не обеспечивает их достоверной коррекции на приемной стороне. Поэтому большинство протоколов каскадного помехоустойчивого кодирования оговаривают передачу запроса на повторную передачу или считывание фрагмента при выявлении в нем ошибок на этапе декодирования блочного кода (см. рис. 2.14). Например, в вычислительных системах и сетях в качестве данного запроса обычно служит специальный пакет *NACK* (неподтверждение), передаваемый отправителем получателю.

Каскадное кодирование позволяет достаточно рационально решить проблему помехоустойчивого обмена данными при относительно высоких изначальных значениях *BER* физического канала связи (ФКС), с одной стороны, и недопустимости остаточных ошибок обмена данными – с другой. В самом деле, при указанных порядках *BER* применение только блочного кодирования привело бы к недопустимому снижению скорости обмена из-за необходимости частого повторения передачи фрагментов, в которых обнаружены ошибки.

С другой стороны, использование только сверточного кодирования привело бы к недопустимо высокому количеству пропущенных ошибок в принятом кадре из-за ограниченных возможностей сверточных кодов по обнаружению и исправлению искаженных битов. Таким образом, сверточное и блочное помехоустойчивое кодирование при этом рационально дополняют друг друга и, образно говоря, служат первым и вторым «рубежами обороны» против ошибок.

Основным недостатком каскадного помехоустойчивого кодирования является необходимость *повторной передачи или считывания* фрагмента данных при обнаружении в нем ошибок. Поэтому оно применимо только в системах, в которых такая процедура допустима (например, в беспроводных каналах связи вычислительных сетей).

2.6.3. Турбо-кодирование

От вышеуказанного недостатка каскадного кодирования свободен сравнительно новый способ помехоустойчивого кодирования, пред-

ложенный в 1993 году, – *турбо-кодирование* [6, 7]**.** Его общий принцип состоит в многократном помехоустойчивом кодировании одного и того же фрагмента данных, реализуемом по алгоритмам, обеспечивающим частичную или полную *взаимную статистическую независимость* результатов кодирования. Благодаря статистической независимости результатов кодирования, вероятность появления в них одинаковых ошибок при передаче или записи/считывании пренебрежимо мала. Поэтому путем взаимного «сопоставления» (по определенным алгоритмам) результатов декодирования каждого из них возможно восстановление исходной двоичной последовательности с достоверностью, повышенной по сравнению с «классическими» помехоустойчивыми кодами. При этом приемлемые остаточные значения BER достигаются без необходимости повторной передачи или считывания данных.

Восстановление исходной двоичной последовательности из результата турбо-кодирования аналогично восстановлению текста на некотором языке на основании выполненных независимо друг от друга переводов этого текста на несколько других языков, каждый из которых содержит свои неточности. При этом достоверность восстановления будет существенно выше, чем при восстановлении оригинального текста из его перевода только на какой-либо один из языков.

Известны три основных способа турбо-кодирования [6, 7]:

- сверточное турбо-кодирование на основе *параллельной конкатенации* (*parallel concatenated convolutional coding – PCCC*);

- сверточное турбо-кодирование на основе *последовательной конкатенации* (*serial concatenated convolutional coding – SCCC*);

- блочное турбо-кодирование (*turbo product coding – TPC*).

Также на практике применяются сочетания вышеперечисленных способов. В частности, известно *гибридное* сверточное турбо-кодирование (*hybrid concatenated convolutional coding – HCCC*), основанное на сочетании способов PCCC и SCCC.

Операционная модель турбо-кодера, реализующего *способ PCCC*, представлена на рис. 2.15 [6, 7].

Из подлежащей кодированию информационной битовой последовательности способом псевдослучайного перемежения [см. выражение (2.48)], опционально сочетаемого с задержкой, формируется M статистически независимых последовательностей. Каждая из них

подвергается сверточному кодированию; как правило, все сверточные кодеры при этом идентичны. Полученные в результате СПК последовательности опционально подвергаются перфорации с целью снижения информационной избыточности и посредством мультиплексирования объединяются в битовую последовательность, передаваемую через канал связи или записываемую на носитель.

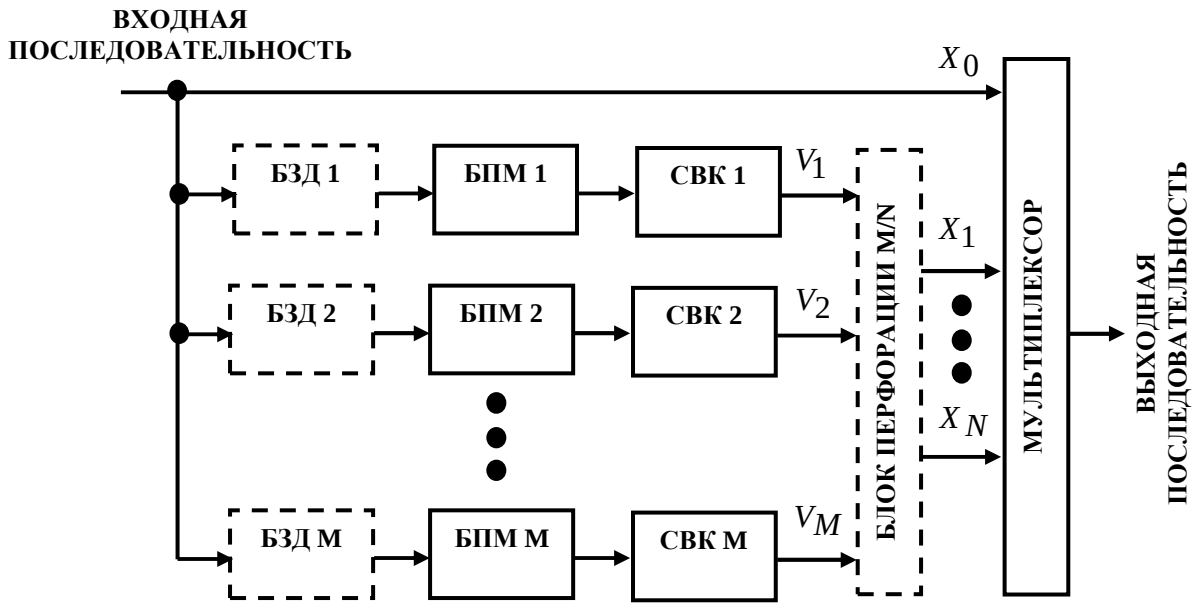


Рис. 2.15. Операционная модель турбо-кодера, реализующего способ *PCCC* (пунктирной линией обозначены блоки, которые могут отсутствовать в ряде частных случаев):
БЗД – блоки задержки;
БПМ – блоки перемежения;
СВК – сверточные кодеры;
$V_0, V_1, \ldots, V_M$ и $X_0, X_1, \ldots, X_N$ – битовые последовательности

Принятая/считанная битовая последовательность, в общем случае содержащая ошибки передачи или записи/считывания, на приемной стороне подвергается демультиплексированию. Из полученных в его результате битовых последовательностей, посредством сверточного декодирования и «сопоставления» его результатов, восстанавливается подвергшаяся турбо-кодированию информационная битовая последовательность. Как правило, восстановление осуществляется за несколько итераций, с последовательным «уточнением» восстанавливаемой битовой последовательности по результатам предыдущих итераций [7].

Способ SCCC реализуется в соответствии с обобщенной операционной моделью, представленной на рис. 2.16 [6, 7].

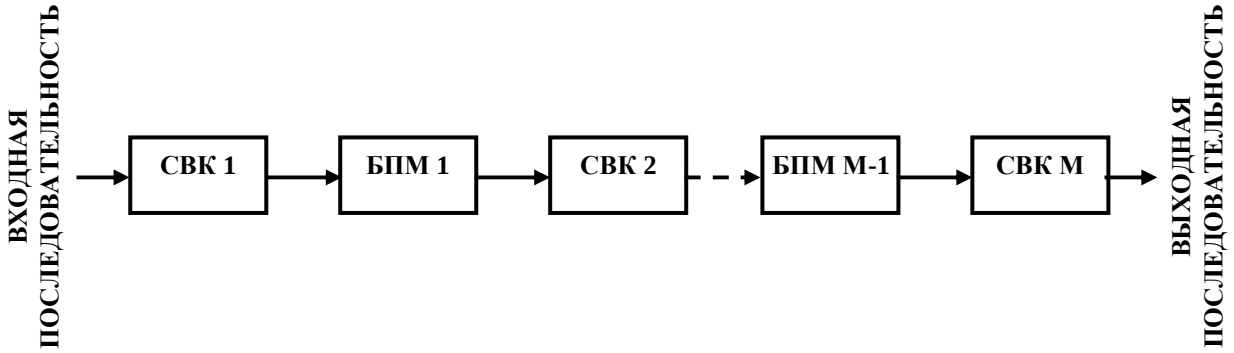


Рис. 2.16. Обобщенная операционная модель турбо-кодера, реализующего способ *SCCC*:
НСВК – несистематический сверточный кодер;
ССВК – систематический сверточный кодер

На практике обычно применяется наиболее простой вариант операционной модели *SCCC*-кодера (рис. 2.17).

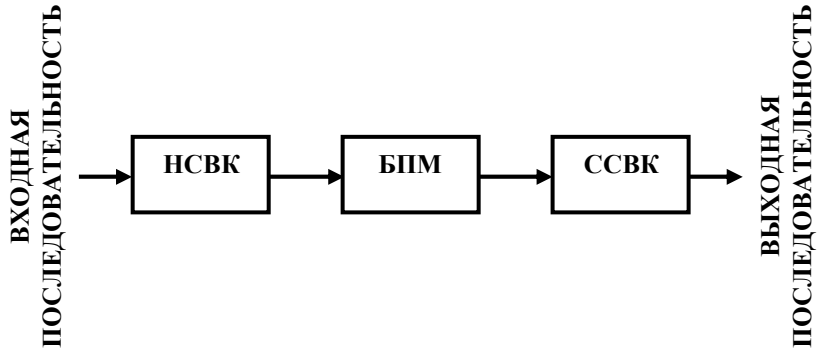


Рис. 2.17. Простейшая операционная модель турбо-кодера, реализующего способ *SCCC*

При *SCCC*-кодировании формирование статистически независимых последовательностей осуществляется не параллельно, как при *PCCC*, а путем последовательного выполнения операций несистематического СПК исходного сообщения и перемежения результатов СПК. В качестве данных последовательностей служат входной и выходной потоки битов каждого из блоков перемежения.

Декодирование SCCC-кодов, как и *PCCC*-кодов, осуществляется по итерационному алгоритму, с последовательным «уточнением» восстанавливаемой битовой последовательности по результатам пре-

дыдущих итераций. На рис. 2.18 приведен один из вариантов операционной модели декодирования результатов *SCCC*-кода, полученного в соответствии с рис. 2.17. При этом декодеры как систематического, так и несистематического сверточного кода реализуют алгоритмы «мягкого» декодирования (алгоритмы, основанные на *fuzzy*-логике), с использованием априорной информации об ожидаемых выходных данных декодера. В их качестве выступает выходная последовательность декодера, полученная на предыдущей итерации (см. рис. 2.18).

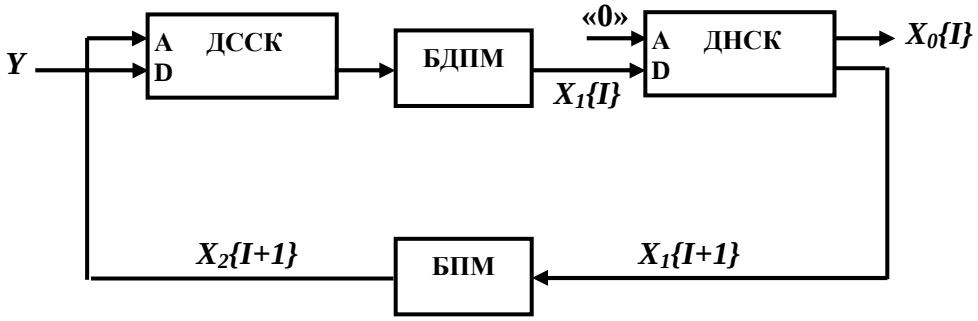


Рис. 2.18. Операционная модель декодирования *SCCC*-кода, полученного в соответствии с рис. 2.17:
ДССК – декодер систематического сверточного кода;
БДПМ – блок деперемежения;
ДНСК – декодер несистематического сверточного кода;
ПМ – блок перемежения;
D и A – входы декодируемых и априорных данных;
Y – принятая (считанная) последовательность;
$X_1\{I\}$ и $X_1\{I+1\}$ – выходная последовательность ДНСК, полученная на I-й и $(I+1)$-й итерации;
$X_2\{I+1\}$ – выходная последовательность ДССК, полученная на $(I+1)$-й итерации;
$X_0\{I\}$ – восстановленная исходная (информативная) последовательность на I-й итерации

На практике также применяется *гибридное* сверточное турбо-кодирование (*hybrid concatenated convolutional coding – HCCC*), основанное на сочетании способов *PCCC* и *SCCC* (рис. 2.19).

Принцип турбо-кодирования *способом ТРС* состоит в следующем. Подлежащая кодированию информационная последовательность преобразуется в 2-мерный или 3-мерный массив. После этого независимому кодированию некоторым блочным кодом (называемым базовым) подвергаются каждая строка и каждый столбец полученно-

го массива. В свою очередь, блочному кодированию тем же базовым кодом подвергаются каждая строка и каждый столбец полученных контрольных разрядов.

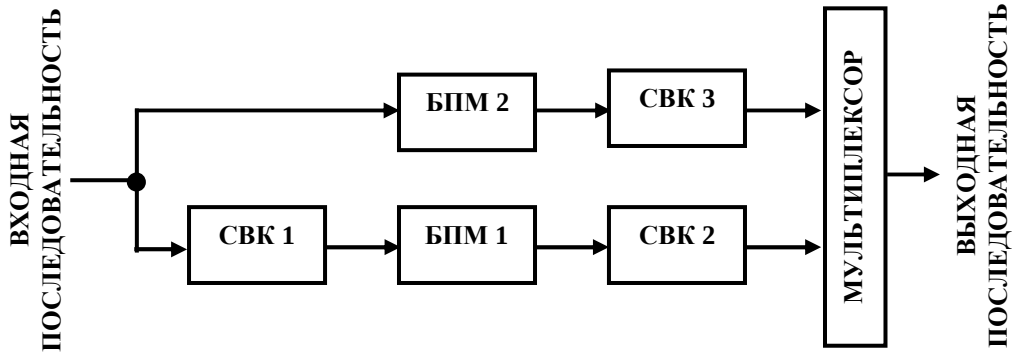


Рис. 2.19. Типовая операционная модель турбо-кодирования способом *HCCC*

При этом, например, в случае организации информационной последовательности в виде 2-мерного массива результат *TPC*-кодирования имеет структуру, представленную на рис. 2.20 [7]. Таким образом, формируется ряд независимых результатов помехоустойчивого кодирования, из которых с значительно большей достоверностью, чем при использовании базового блочного кода, может быть восстановлена исходная последовательность.

Массив информационных битов размерностью k_1 строк $\times$ k_2 столбцов	Массив контрольных битов размерностью k_1 строк $\times$ $\times$ m_2 столбцов
Массив контрольных битов размерностью m_1 строк $\times$ k_2 столбцов	Массив контроля контрольных битов размерностью m_1 строк $\times$ $\times m_2$ столбцов

Рис. 2.20. Структура 2-мерного *TPC*-кода:
m_1, m_2 – число контрольных битов базового блочного кода при количестве информационных битов, равном k_1 и k_2 соответственно

Принцип турбо-кодирования способом *ТРС* поясняет рис. 2.21. На нем представлен один из простейших примеров *ТРС*-кода. Его базовым кодом служит код контроля четности. Подлежащее кодированию информационное слово имеет разрядность 9 бит; из него сформирован двумерный массив размерностью 3×3 бита. Каждая строка и каждый столбец этого массива снабжены дополнительным контрольным битом, вычисленным как сумма по модулю 2 информационных битов строки или, соответственно, столбца. Еще один контрольный бит (C_6) является суммой по модулю 2 контрольных битов всех строк, равной также (естественно, в отсутствие ошибок) сумме по модулю 2 контрольных битов всех столбцов.

I_0	I_1	I_2	$C_0 = I_0 \oplus I_1 \oplus I_2$
I_3	I_4	I_5	$C_1 = I_3 \oplus I_4 \oplus I_5$
I_6	I_7	I_8	$C_2 = I_6 \oplus I_7 \oplus I_8$
$C_3 = I_0 \oplus I_3 \oplus I_6$	$C_4 = I_1 \oplus I_4 \oplus I_7$	$C_5 = I_2 \oplus I_5 \oplus I_8$	$C_6 = C_0 \oplus C_1 \oplus C_2 =$ $= C_3 \oplus C_4 \oplus C_5$

Рис. 2.21. Пример турбо-кодирования способом *ТРС* (базовый блочный код – код контроля четности, число информационных битов в кодируемом фрагменте – 9)

На рис. 2.22 приведен пример, иллюстрирующий повышение достоверности информативности *ТРС*-кода при декодировании по сравнению с его базовым блочным кодом. В данном примере предполагается, что при передаче искажению подвергся выделенный на рис. 2.22 серым цветом 0-й информационный бит приведенного на рис. 2.21 результата *ТРС*-кодирования. Следует напомнить, что искажение бита равносильно его инверсии (см. рис. 2.22). На наличие искаженного бита укажут контрольные биты C_0 и C_3, выделенные двойной граничной линией и являющиеся битами контроля четности соответственно строки и столбца, к которым принадлежит искаженный бит. При этом делается вывод, что искажен бит, находящийся на пересечении данных строки и столбца, а исправление ошибки реализуется инвертированием искаженного бита. Таким образом, в рассматриваемом примере при наличии только одной ошибки в результате *ТРС*-кодирования она однозначно локализуется и исправляется. Следовательно, *ТРС*-код, использующий код контроля четности в ка-

честве базового, дает возможность, как минимум, *исправлять* при декодировании одиночную ошибку в блоке, что в принципе не позволяет его базовый код.

$\bar{I}_0$	I_1	I_2	$C_0 \neq \bar{I}_0 \oplus I_1 \oplus I_2$
I_3	I_4	I_5	$C_1 = I_3 \oplus I_4 \oplus I_5$
I_6	I_7	I_8	$C_2 = I_6 \oplus I_7 \oplus I_8$
$C_3 \neq \bar{I}_0 \oplus I_3 \oplus I_6$	$C_4 = I_1 \oplus I_4 \oplus I_7$	$C_5 = I_2 \oplus I_5 \oplus I_8$	$C_6 = C_0 \oplus C_1 \oplus C_2 = = C_3 \oplus C_4 \oplus C_5$

Рис. 2.22. Пример, поясняющий повышение достоверности декодирования при использовании турбо-кодирования способом *TPC*

На практике используются *TPC*-коды на основе базовых блочных кодов, обладающих значительно бóльшими возможностями обнаружения и исправления ошибок, чем код контроля четности (например, на основе кодов БЧХ в качестве базовых). Благодаря *TPC*-кодированию их возможности по исправлению ошибок многократно возрастают, что позволяет практически при любом реально возможном значении *BER* подобрать тип и параметры базового кода и *TPC*-кода, обеспечивающие достоверное исправление ошибок на приемной стороне. В общем случае оно может быть реализовано, например, на основе декодера табличного типа, входами которого служат информационные и все контрольные разряды принятого блока, а выходами – восстановленные информационные разряды.

В целом, благодаря повышенной (по сравнению с «классическими» помехоустойчивыми кодами) способности коррекции ошибок, обусловленных шумами в канале связи и его ограниченной полосой пропускания, турбо-коды позволяют *вплотную приблизиться* к теоретически максимально возможной пропускной способности зашумленного канала связи, задаваемой *теоремой Шеннона – Хартли* [6]:

$$C = \Delta f \times \log_2\left(1 + \frac{P_s}{P_n}\right), \qquad (2.49)$$

где C – максимальная пропускная способность канала, информационных бит в секунду;

Δf – ширина полосы пропускания канала;

P_s и P_n – мощность соответственно полезного сигнала и шума в канале.

Основным недостатком турбо-кодов является относительно высокая сложность реализации кодирования и декодирования. Однако при современном уровне развития средств обработки цифровых данных этот недостаток не существен. Поэтому турбо-коды находят широкое применение, в первую очередь – в системах связи с зашумленными каналами, в том числе в системах мобильной и спутниковой связи.

Выводы по разделу 2

1. Помехоустойчивым кодированием называют представление цифровых данных в форме, при которой отдельные биты сообщения или их группы связаны определенной зависимостью, позволяющей при ее нарушении обнаружить и/или исправить ошибки в сообщении.

2. Все разновидности помехоустойчивых кодов характеризуются следующими общими свойствами:

- большей разрядностью сообщения, получаемого в результате кодирования, по сравнению с исходным сообщением, т. е. наличием избыточности, без которой обнаружение и исправление ошибок невозможно в принципе;
- принадлежностью всех разрядов кода к некоторому конечному полю и их формированием посредством математических операций в данном поле (см. п. 2.2); непринадлежность разрядов принятого кода соответствующему полю или/и их несоответствие правилам выполнения операций в нем являются одними из признаков наличия ошибок;
- наличием разрешенных (возможных только в отсутствие ошибок) и запрещенных (возможных только при наличии ошибок) кодовых комбинаций;
- ограниченным числом обнаруживаемых или/и исправляемых ошибок в сообщении, зависящим от конкретной разновидности и параметров кода.

3. Помехоустойчивые коды разделяются на два основных класса: блочные и сверточные. Результаты блочного помехоустойчивого кодирования представляют собой последовательность n-разрядных слов (блоков), каждый из которых формируется из k-разрядного слова ($n > k$) исходной (кодируемой) последовательности независимо от других блоков. Сверточное кодирование осуществляется путем пре-

образования каждого слова разрядностью K кодируемых данных в N-разрядное слово ($N > K$), каждый из разрядов которого формируется как некоторая функция не только от соответствующего ему K-разрядного слова входной последовательности, но и от предыдущих слов как входной, так и (в общем случае) выходной последовательности.

4. Наиболее распространенными на практике разновидностями блочных помехоустойчивых кодов являются коды контроля четности, циклические помехоустойчивые коды (ЦПК) и коды Рида – Маллера (см. пп. 2.3 и 2.4). Первые две разновидности блочных кодов являются разделимыми, т. е. в их кодовых словах в явном виде присутствуют информационные и контрольные разряды, а коды Рида – Маллера – неразделимыми. Блочные коды позволяют эффективно обнаруживать ошибки в блоках: разделимые коды – на основании синдрома ошибок, а неразделимые – на основании принципа максимального правдоподобия. Однако во многих практических случаях блочные коды не обеспечивают достоверного исправления ошибок на приемной стороне. Поэтому помехоустойчивое кодирование посредством только блочных кодов рационально применять в системах обмена данными, характеризуемых:

- малыми значениями *BER* (от 10^{-5} – 10^{-6} и менее) в процессе передачи или записи/считывания данных;
- недопустимостью остаточных ошибок в окончательных результатах восстановления исходных данных на приемной стороне;
- отсутствием необходимости в обмене данными в реальном масштабе времени.

К таким системам относятся, например, выделенные кабельные линии вычислительных сетей или интерфейса электронно-вычислительных систем. В них помехоустойчивые коды (как правило, ЦПК) выполняют только функции обнаружения ошибок, а их исправление осуществляется путем повторной передачи/считывания блока по запросу приемника. Благодаря малым значениям *BER* данные процедуры не оказывают существенного влияния на скорость обмена данными.

5. Наиболее универсальными из блочных кодов являются ЦПК БЧХ (см. п. 2.4.3), позволяющие обнаружить или исправить любое наперед заданное количество ошибок в блоке, произвольно распределенных по нему. В свою очередь, наиболее эффективными из ЦПК БЧХ являются недвоичные ЦПК Рида – Соломона (см. п. 2.4.4), характери-

зуемые минимальной избыточностью при заданном количестве обнаруживаемых (исправляемых) ошибок в блоке. Основной недостаток кодов Рида – Соломона по сравнению с двоичными ЦПК БЧХ – повышенная сложность кодирования/декодирования.

6. Сверточные помехоустойчивые коды (СПК) (см. п. 2.5) используются, в основном, для исправления ошибок в реальном масштабе времени. Исправление, как правило, осуществляется на основе принципа максимального правдоподобия. Помехоустойчивое кодирование на базе только СПК рационально применять в системах обмена данными, характеризуемых:

- относительно высокими значениями *BER* (порядка 10^{-3} – 10^{-4}) в процессе передачи или записи/считывания данных;
- обменом в реальном масштабе времени (необходимостью минимизации задержек обработки данных на приемной стороне);
- допустимостью остаточных ошибок в окончательных результатах восстановления исходных данных на приемной стороне.

К таким системам относятся, например, системы цифровой радиотелефонии.

7. По сравнению с блочными кодами СПК характеризуются повышенной информационной избыточностью, а также меньшей эффективностью при наличии пакетов ошибок (искажении нескольких смежных друг с другом битов). Первый из данных недостатков частично устраняется способом перфорации (выкалывания) [см. выражение (2.47)], а второй – способом перемежения (см. рис. 2.12 и 2.13).

8. В системах обмена данными, характеризуемых:

- относительно высокими значениями *BER* (порядка 10^{-3} – 10^{-4}) в процессе передачи или записи/считывания данных;
- недопустимостью остаточных ошибок обмена данными (к ним относятся, например, беспроводные каналы связи локальных вычислительных сетей), ни «классические» блочные, ни «классические» сверточные помехоустойчивые коды по отдельности не позволяют достигнуть приемлемого сочетания достоверности и скорости обмена. Поэтому в таких системах необходимо применение специальных методов повышения эффективности помехоустойчивого кодирования. Основными из них являются:
- каскадное помехоустойчивое кодирование;
- турбо-кодирование.

9. Каскадное помехоустойчивое кодирование (см. рис. 2.14) состоит в кодировании блочным кодом подлежащего передаче/записи

фрагмента данных с последующим сверточным кодированием результата блочного кодирования. Декодирование осуществляется в обратном порядке. При этом сверточное и блочное кодирование взаимно дополняют друг друга, служа первым и вторым «рубежами обороны» против ошибок.

10. Основной недостаток каскадного помехоустойчивого кодирования состоит в необходимости повторной передачи или считывания фрагмента данных при обнаружении в нем ошибок. От данного недостатка свободно турбо-кодирование (см. п. 2.6.3). Его общий принцип состоит в многократном помехоустойчивом кодировании одного и того же фрагмента данных, реализуемом по алгоритмам, обеспечивающим частичную или полную взаимную статистическую независимость результатов кодирования.

Благодаря статистической независимости результатов кодирования, вероятность появления в них одинаковых ошибок при передаче или записи/считывании пренебрежимо мала. Поэтому путем взаимного «сопоставления» (по определенным алгоритмам) результатов декодирования каждого из них возможно восстановление исходной двоичной последовательности с достоверностью, повышенной по сравнению с «классическими» помехоустойчивыми кодами. При этом приемлемые остаточные значения BER достигаются без необходимости повторной передачи или считывания данных.

Известно как блочное, так и сверточное турбо-кодирование (см. п. 2.6.3).

Турбо-коды позволяют *вплотную приблизиться* к теоретически максимально возможной пропускной способности зашумленного канала связи, задаваемой теоремой Шеннона – Хартли [см. выражение (2.49)].

Основным недостатком турбо-кодов является относительно высокая сложность кодирования/декодирования.

Вопросы для самопроверки

1. По какой причине результат кодирования помехоустойчивым кодом в обязательном порядке должен содержать избыточные разряды?

2. Поясните разницу между блочными и сверточными помехоустойчивыми кодами.

3. Обоснуйте необходимость выполнения в некотором *конечном* поле всех математических операций помехоустойчивого кодирования/декодирования.

4. Какое из нижеприведенных чисел не может быть правильным результатом выполнения некоторой математической операции в конечном поле $GF(929)$?

18; 12; 900; 1000.

5. Обоснуйте, по какой причине в поле $GF(2)$ верно равенство

$$-x = x.$$

6. Обоснуйте правильность следующего равенства:

$$0 - 1 = 1 (\mathrm{mod}\, 2).$$

7. Обоснуйте правильность равенства

$$3^2 = 1 (\mathrm{mod}\, 8).$$

8. Поясните смысл минимального кодового расстояния помехоустойчивого кода.

9. Некоторый блочный код характеризуется минимальным кодовым расстоянием, равным 5. Возможно ли обнаружение с помощью данного кода одной ошибки в блоке, двух ошибок, четырех ошибок, шести ошибок?

10. Опишите принцип блочного помехоустойчивого кодирования/декодирования на основе проверочной матрицы.

11. Содержит ли ошибки двоичное слово 110000111, если известно, что его биты с 1-го по 8-й являются информационными, а 0-й бит является битом контроля четности?

12. Возможно ли корректное восстановление информационных разрядов из кода $RM(1,3)$ по принципу мажоритарности при наличии двух ошибок в блоке?

13. Поясните общий принцип помехоустойчивого кодирования/декодирования посредством ЦПК. Обоснуйте смысл использования остатка от деления принятого слова на порождающий полином в качестве синдрома.

14. Представьте двоичное слово 1100001 в виде полинома с коэффициентами, принадлежащими полю $GF(2)$.

15. На сколько битов влево должно быть сдвинуто исходное информационное слово при его кодировании двоичным ЦПК БЧХ с порождающим полиномом

$$G(x) = \left(x^6 + x + 1\right) \times \left(x^6 + x^4 + x^2 + x + 1\right) \times \left(x^6 + x^5 + x^2 + x + 1\right)?$$

16. Приведите порождающий полином $G(x)=\left(x^6+x+1\right)\times\left(x^6+x^4+x^2+x+1\right)$ ЦПК БЧХ (см. табл. 2.3) к окончательному виду, применимому при кодировании (см. пример в п. 2.4.3). Сколько контрольных битов будет при этом содержать блок, получаемый в результате кодирования?

17. Определите разрядность (в битах) контрольного поля блока, кодируемого ЦПК Рида – Соломона, формируемым в поле $GF(929)$ и обладающим возможностью обнаруживать до шести ошибок в блоке.

18. Определите разрядность (в битах) информационного поля блока из предыдущего примера.

19. Поясните общий принцип сверточного помехоустойчивого кодирования. В чем состоят различия между нерекурсивными и рекурсивными СПК?

20. Опишите назначение и правила построения решетчатой диаграммы нерекурсивного СПК.

21. Опишите процесс декодирования СПК по принципу максимального правдоподобия.

22. Опишите принцип декодирования СПК по алгоритму Витерби.

23. Обоснуйте эффективность перемежения для исправления серий ошибок при сверточном помехоустойчивом кодировании.

24. Поясните принцип каскадного помехоустойчивого кодирования. Укажите типы систем обмена данными, в которых его применение рационально.

25. Опишите общий принцип турбо-кодирования. Укажите типы систем обмена данными, в которых его применение рационально.

ЛИТЕРАТУРА

1. Теория информации: учебник для вузов / В.Т. Еременко, В.А. Минаев, А.П. Фисун и др. – Орел: ОрелГТУ, ОГУ, 2010. – 443 с.

2. Сэломон, Д. Сжатие данных, изображений и звука: [пер. с англ.] / Д. Сэломон. – М.: Техносфера, 2006. – 368 с.

3. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М.: Диалог-МИФИ, 2003. – 384 с.

4. Семенюк, В.В. Экономное кодирование дискретной информации / В.В. Семенюк. – СПб.: СПбГИТМО (ТУ), 2001. – 115 с.

5. Воробьев, В.И. Теория и практика вейвлет-преобразования / В.И. Воробьев, В.Г. Грибунин. – СПб.: ВУС, 1999. – 204 с.

6. Скляр, Б. Цифровая связь. Теоретические основы и практическое применение: [пер. с англ.] / Б. Скляр. – Изд. 2-е, испр. – М.: Издательский дом «Вильямс», 2003. – 1104 с.

7. Золотарев, В.В. Помехоустойчивое кодирование. Методы и алгоритмы: справочник / В.В. Золотарев, Г.В. Овечкин; под ред. чл.-корр. РАН Ю.Б. Зубарева. – М.: Горячая линия – Телеком, 2004. – 126 с.

8. Теория электрической связи: учебное пособие / К.К. Васильев, В.А. Глушков, А.В. Дормидонтов, А.Г. Нестеренко; под общ. ред. К.К. Васильева. – Ульяновск: УлГТУ, 2008. – 452 с.

9. Бронштейн, И.Н. Справочник по математике для инженеров и учащихся втузов / И.Н. Бронштейн, К.А. Семендяев. – Изд. 13-е, испр. – М.: Наука, Гл. ред. физ.-мат. лит., 1986. – 544 с.

Учебное издание

Тютякин Александр Васильевич

ОСНОВЫ ЭФФЕКТИВНОГО И ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ СООБЩЕНИЙ

Учебное пособие

Редактор Т.Д. Васильева
Технический редактор Т.П. Прокудина

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Государственный университет - учебно-научно-производственный комплекс»

Подписано к печати 19.12.2014 г. Формат 60×84 1/16.
Усл. печ. л. 11,3. Тираж 100 экз.
Заказ №_____

Отпечатано с готового оригинал-макета
на полиграфической базе ФГБОУ ВПО «Госуниверситет - УНПК»,
302030, г. Орел, ул. Московская, 65.