**В.С. Прохоров**

# ТЕОРИЯ ИНФОРМАЦИИ

## Лекции

### Содержание

## Введение

Теория информации является одним из курсов при подготовке инженеров, специализирующихся в области автоматизированных систем управления и обработки информации. Функционирование таких систем существенным образом связано с получением, подготовкой, передачей, хранением и обработкой информации, поскольку без осуществления этих этапов невозможно принять правильное решение и осуществить требуемое управляющее воздействие, которое является конечной целью функционирования любой системы.

Возникновение теории информации связывают обычно с появлением фундаментальной работы американского ученого К. Шеннона «Математическая теория связи» (1948). Однако в теорию информации органически вошли и результаты, полученные другими учеными. Например, Р. Хартли, впервые предложил количественную меру информации (1928), акад. В. А. Котельников, сформулировал важнейшую теорему о возможности представления непрерывной функции совокупностью ее значений в отдельных точках отсчета (1933) и разработал оптимальные методы приема сигналов на фоне помех (1946). Акад. А. Н. Колмогоров, внес огромный вклад в статистическую теорию колебаний, являющуюся математической основой теории информации (1941). В последующие годы теория информации получила дальнейшее

развитие в трудах советских ученых (А. Н. Колмогорова, А. Я. Хинчина, В. И. Сифорова, Р. Л. Добрушина, М. С. Пинскера, А. Н. Железнова, Л. М. Финка и др.), а также ряда зарубежных ученых (В. Макмиллана, А. Файнстейна, Д. Габора, Р. М. Фано, Ф. М. Вудворта, С. Гольдмана, Л. Бриллюэна и др.).

К теории информации, в ее узкой классической постановке, относят результаты решения ряда фундаментальных теоретических вопросов. Это в первую очередь: анализ вопросов оценки «количества информации»; анализ информационных характеристик источников сообщений и каналов связи и обоснование принципиальной возможности кодирования и декодирования сообщений, обеспечивающих предельно допустимую скорость передачи сообщений по каналу связи, как при отсутствии, так и при наличии помех.

Если рассматривают проблемы разработки конкретных методов и средств кодирования сообщений, то совокупность излагаемых вопросов называют *теорией информации и кодирования* или *прикладной теорией информации.*

Попытки широкого использования идей теории информации в различных областях науки связаны с тем, что в основе своей эта теория математическая. Основные ее понятия (энтропия, количество информации, пропускная способность) определяются только через вероятности событий, которым может быть приписано самое различное физическое содержание. Подход к исследованиям в других областях науки с позиций использования основных идей теории информации получил название *теоретико-информационного подхода.* Его применение в ряде случаев позволило получить новые теоретические результаты и ценные практические рекомендации. Однако не редко такой подход приводит к созданию моделей процессов, далеко не адекватных реальной действительности. Поэтому в любых исследованиях, выходящих за рамки чисто технических проблем передачи и хранения сообщений, теорией информации следует пользоваться с большой осторожностью. Особенно это касается моделирования умственной деятельности человека, процессов восприятия и обработки им информации.

Содержание конспекта лекций ограничивается рассмотрением вопросов теории и практики кодирования.

## 1. Понятие информации. Задачи и постулаты прикладной теории информации

### 1.1 Что такое информация

С начала 1950-х годов предпринимаются попытки использовать понятие информации (не имеющее до настоящего времени единого определения) для объяснения и описания самых разнообразных явлений и процессов.

В некоторых учебниках дается следующее определение информации:

***Информация*** *- это совокупность сведений, подлежащих хранению, передаче, обработке и использованию в человеческой деятельности*.

Такое определение не является полностью бесполезным, т.к. оно помогает хотя бы смутно представить, о чем идет речь. Но с точки зрения логики оно бессмысленно. Определяемое понятие (*информация*) здесь подменяется другим понятием (*совокупность сведений)*, которое само нуждается в определении.

При всех различиях в трактовке понятия информации, бесспорно, то, что проявляется информация всегда в материально-энергетической форме в виде сигналов.

Информацию, представленную в формализованном виде, позволяющем осуществлять ее обработку с помощью технических средств, называют **данными**.

## 1.2 Этапы обращения информации

Можно выделить следующие этапы обращения информации:

1) восприятие информации;

2) подготовка информации;

3) передача и хранение информации;

4) обработка информации;

5) отображение информации;

6)воздействие информации.

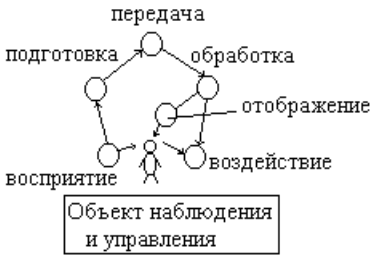



Рис.1.1 Этапы обращения информации

На этапе восприятия информации осуществляется целенаправленное извлечение и анализ информации о каком-либо объекте (процессе), в результате чего формируется образ объекта, проводится его опознание и оценка. При этом отделяют интересующую информацию от шумов.

На этапе подготовки информации получают сигнал в форме, удобной для передачи или обработки (нормализация, аналого-цифровое преобразование и т.д.). На этапе передачи и хранения информация пересылается либо из одного места в другое, либо от одного момента времени до другого. На этапе обработки информации выделяются ее общие и существенные взаимозависимости для выбора управляющих воздействий (принятия решений). На этапе отображения информации она представляется человеку в форме, способной воздействовать на

его органы чувств. На этапе воздействия информация используется для осуществления необходимых изменений в системе.

### 1.3 Информационные системы

В основе решения многих задач лежит обработка информации. Для облегчения обработки информации создаются ***информационные системы (ИС).*** Автоматизированными называют ИС, в которых применяют технические средства, в частности ЭВМ. Большинство существующих ИС являются автоматизированными, поэтому для краткости просто будем называть их ИС. В ***широком понимании*** под определение ИС подпадает любая система обработки информации. По ***области применения*** ИС можно разделить на системы, используемые в производстве, образовании, здравоохранении, науке, военном деле, социальной сфере, торговле и других отраслях. По ***целевой функции*** ИС можно условно разделить на следующие основные категории: управляющие, информационно-справочные, поддержки принятия решений. Заметим, что иногда используется более ***узкая трактовка понятия ИС*** как совокупности аппаратно-программных средств, задействованных для решения некоторой прикладной задачи. В организации, например, могут существовать информационные системы, на которые возложены следующие задачи: учет кадров и материально-технических средств, расчет с поставщиками и заказчиками, бухгалтерский учет и т. п. Эффективность функционирования информационной системы (ИС) во многом зависит от ее архитектуры. В настоящее время перспективной является архитектура клиент-сервер. В распространенном варианте она предполагает наличие компьютерной сети и распределенной базы данных, включающей корпоративную базу данных (КБД) и персональные базы данных (ПБД). КБД размещается на компьютере-сервере, ПБД размещаются на компьютерах сотрудников подразделений, являющихся клиентами корпоративной БД. ***Сервером*** определенного ресурса в компьютерной сети называется компьютер (программа), управляющий этим ресурсом. ***Клиентом*** — компьютер (программа), использующий этот ресурс. В качестве ресурса компьютерной сети могут выступать, к примеру, базы данных, файловые системы, службы печати, почтовые службы. Тип сервера определяется видом ресурса, которым он управляет. Например, если управляемым ресурсом является база данных, то соответствующий сервер называется сервером базы данных. Достоинством организации информационной системы по архитектуре клиент-сервер является удачное сочетание централизованного хранения, обслуживания и коллективного доступа к общей корпоративной информации с индивидуальной работой пользователей над персональной информацией. Архитектура клиент-сервер допускает различные варианты реализации.

### 1.4 Система передачи информации

Информация поступает в систему в форме сообщений. *Под* ***сообщением*** *понимают совокупность знаков или первичных сигналов, содержащих информацию.*

***Источник сообщений*** в общем случае образует совокупность ***источника информации*** (ИИ) (исследуемого или наблюдаемого объекта) и ***первичного преобразователя*** (ПП) (датчика, человека-оператора и т.д.), воспринимающего информацию о протекающем в нем процессе.

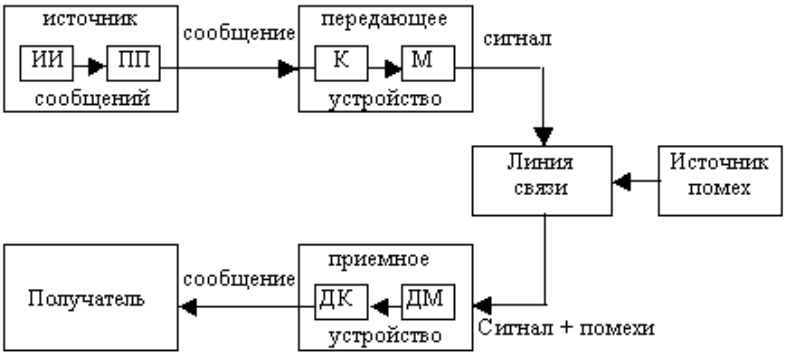



Рис. 1.2. Структурная схема одноканальной системы передачи информации.

Различают дискретные и непрерывные сообщения.

**Дискретные сообщения** формируются в результате последовательной выдачи источником сообщений отдельных элементов - **знаков**.

Множество различных знаков называют **алфавитом источника сообщения**, а число знаков - **объемом алфавита**.

**Непрерывные сообщения** не разделены на элементы. Они описываются непрерывными функциями времени, принимающими непрерывное множество значений (речь, телевизионное изображение).

Для передачи сообщения по каналу связи ему ставят в соответствие определенный сигнал. Под сигналом понимают физический процесс, отображающий (несущий) сообщение.

Преобразование сообщения в сигнал, удобный для передачи по данному каналу связи, называют **кодированием в широком смысле слова**.

Операцию восстановления сообщения по принятому сигналу называют **декодированием**.

Как правило, прибегают к операции представления исходных знаков в другом алфавите с меньшим числом знаков, называемых ***символами***. При обозначении этой операции используется тот же термин "***кодирование***", рассматриваемый *в узком смысле*. Устройство, выполняющее такую операцию, называют кодирующим или ***кодером***. Так как алфавит символов меньше алфавита знаков, то каждому знаку соответствует некоторая последовательность символов, которую называют ***кодовой комбинацией***.

Число символов в кодовой комбинации называют ее **значностью**, число ненулевых символов - **весом**.

Для операции сопоставления символов со знаками исходного алфавита используют термин “**декодирование**”. Техническая реализация этой операции осуществляется декодирующим устройством или ***декодером***.

***Передающее устройство*** осуществляет преобразование непрерывных сообщений или знаков в сигналы, удобные для прохождения по линии связи. При этом один или несколько параметров выбранного сигнала изменяют в соответствии с передаваемой информацией. Такой процесс называют ***модуляцией***. Он осуществляется ***модулятором***. Обратное преобразование сигналов в символы производится ***демодулятором***

Под ***линией связи*** понимают среду (воздух, металл, магнитную ленту и т.д.), обеспечивающую поступление сигналов от передающего устройства к приемному устройству.

Сигналы на выходе линии связи могут отличаться от сигналов на ее входе (переданных) вследствие затухания, искажения и воздействия помех.

**Помехами** называют любые мешающие возмущения, как внешние, так и внутренние, вызывающие отклонение приинятых сигналов от переданных сигналов.

Из смеси сигнала с помехой ***приемное устройство*** выделяет сигнал и посредством декодера восстанавливает сообщение, которое в общем случае может отличаться от посланного. Меру соответствия принятого сообщения посланному сообщению называют ***верностью передачи***.

Принятое сообщение с выхода системы связи поступает к абоненту-получателю, которому была адресована исходная информация.

Совокупность средств, предназначенных для передачи сообщений, называют ***каналом связи***.

### 1.5 Задачи и постулаты прикладной теории информации

К теории информации относят результаты решения ряда фундаментальных теоретических вопросов:

- анализ сигналов как средства передачи сообщений, включающий вопросы оценки переносимого ими «количества информации»;

- анализ информационных характеристик источников сообщений и каналов связи и обоснование принципиальной возможности кодирования и декодирования сообщений, обеспечивающих предельно допустимую скорость передачи сообщений по каналу связи, как при отсутствии, так и при наличии помех.

В теории информации исследуются информационные системы при четко сформулированных условиях (постулатах):

1. Источник сообщения осуществляет выбор сообщения из некоторого множества с определенной вероятностью.

2. Сообщения могут передаваться по каналу связи в закодированном виде. Кодированные сообщения образуют множество, являющееся взаимно однозначным отображением множества сообщений. Правило декодирования известно декодеру (записано в его программе).

3. Сообщения следуют друг за другом, причем число сообщений может быть сколь угодно большим.

4. Сообщение считается принятым верно, если в результате декодирования оно может быть в точности восстановлено. При этом не учитывается, сколько времени прошло с момента передачи сообщения до момента окончания декодирования, и какова сложность операций кодирования и декодирования.

5. Количество информации не зависит от смыслового содержания сообщения, от его эмоционального воздействия, полезности и даже от его отношения к реальной действительности.

## 2. Количественная оценка информации

В качестве основной характеристики сообщения теория информации принимает величину, называемую ***количеством информации***. *Это понятие не затрагивает смысла и важности передаваемого сообщения, а связано со степенью его неопределенности.*

Пусть алфавит источника сообщений состоит из **m** знаков, каждый из которых может служить элементом сообщения. Количество **N** возможных сообщений длины **n** равно числу перестановок с неограниченными повторениями:

$N = m^n$

Если для получателя все **N** сообщений от источника являются равновероятными, то получение конкретного сообщения равносильно для него случайному выбору одного из **N** сообщений с вероятностью **1/N**.

*Ясно, что чем больше* **N***, тем большая степень неопределенности характеризует этот выбор и тем более информативным можно считать сообщение.*

Поэтому число **N** могло бы служить мерой информации. Однако, с позиции теории информации, естественно наделить эту меру свойствами *аддитивности*, т.е. определить ее так, чтобы она бала пропорциональна длине сообщения (например, при передаче и оплате сообщения - телеграммы, важно не ее содержание, а общее число знаков).

***В качестве меры неопределенности выбора состояния источника с равновероятными состояниями принимают логарифм числа состояний:***

$I = \log N = \log m^n = n \log m.$

*Эта логарифмическая функция характеризует* ***количество информации:***

Указанная мера была предложена американским ученым Р.Хартли в 1928 г.

Количество информации, приходящееся на один элемент сообщения (знак, букву), называется **энтропией**:

$$H = \frac{I}{n} = \frac{n \log m}{n} = \log m .$$

В принципе безразлично, какое основание логарифма использовать для определения количества информации и энтропии, т. к. в силу соотношения $\log_a m = \log_a b \log_b m$ переход от одного основания логарифма к другому сводится лишь к изменению единицы измерения.

Так как современная информационная техника базируется на элементах, имеющих два устойчивых состояния, то обычно выбирают основание логарифма равным двум, т.е. энтропию выражают как:

$H_0 = \log_2 m$.

Тогда ***единицу количества информации*** на один элемент сообщения называют ***двоичной единицей*** или ***битом***. При этом единица неопределенности (двоичная единица или бит) представляет собой неопределенность выбора из двух равновероятных событий *(bit* — сокращение от англ. *binary digit* — двоичная единица)

Так как из $\log_2 m = 1$ следует $m = 2$, то ясно, что ***1 бит - это количество информации, которым характеризуется один двоичный элемент при равновероятных состояниях 0 и 1.***

Двоичное сообщение длины **n** содержит **n** бит информации.

Единица количества информации, равная 8 битам, называется **байтом**.

Если основание логарифма выбрать равным десяти, то энтропия выражается в десятичных единицах на элемент сообщения - **дитах**, причем 1 дит = $\log_{10}2$ бит = 3,32 бит.

**Пример1**. Определить количество информации, которое содержится в телевизионном сигнале, соответствующем одному кадру развертки. Пусть в кадре 625 строк, а сигнал, соответствующий одной строке, представляет собой последовательность из 600 случайных по амплитуде импульсов, причем амплитуда импульса может принять любое из 8 значений с шагом в 1 В.

**Решение.** В рассматриваемом случае длина сообщения, соответствующая одной строке, равна числу случайных по амплитуде импульсов в ней: **n = 600**.

Количество элементов сообщения (знаков) в одной строке равно числу значений, которое может принять амплитуда импульсов в строке,: **m = 8.**

Количество информации в одной строке: **I = n log m = 600 log 8,** а количество информации в кадре: $\mathbf{I' = 625\ I = 625\ 600 \log 8 = 1{,}125 \cdot 10^6}$ **бит**

**Пример2**. Определить минимальное число взвешиваний, которое необходимо произвести на равноплечих весах, чтобы среди 27 внешне неотличимых монет найти одну фальшивую, более легкую.

**Решение.** Так как монеты внешне не отличимые, то они представляют источник с равновероятными состояниями, а общая неопределенность ансамбля, характеризующая его энтропию, поэтому составляет: $H1 = Iog_2 27$ бит.

Одно взвешивание способно прояснить неопределенность ансамбля насчитывающего три возможных исхода (левая чаша весов легче, правая чаша весов легче, весы находятся в равновесии).Так как все исходы равновероятны (нельзя заранее отдать предпочтение одному из них), то результат одного взвешивания представляет источник с равновероятными состояниями, а его энтропия составляет: $H2 = Iog_2 3$ бит.

Так как энтропия отвечает требованию аддитивности и при этом $\mathbf{H1=3H2= 3\ 1og_2 3}$, то для определения фальшивой монеты достаточно произвести три взвешивания.

Алгоритм определения фальшивой монеты следующий. При первом взвешивании на каждую чашку весов кладется по девять монет. Фальшивая монета будет либо среди тех девяти монет, которые оказались легче, либо среди тех, которые не взвешивались, если имело место равновесие. Аналогично, после второго взвешивания число монет, среди которых находится фальшивая монета, сократится до трех. Последнее, третье, взвешивание дает возможность точно указать фальшивую монету.

**Рассмотренная выше оценка информации основана на предположении о равновероятности всех знаков алфавита**.

В общем случае каждый из знаков появляется в сообщении с различной вероятностью.

Пусть на основании статистического анализа известно, что в сообщении длины **n** знак $\mathbf{x_i}$ появляется $\mathbf{n_i}$ раз, т.е. вероятность появления знака:

$P_i = \frac{n_i}{n}$ , (i = 1, 2, 3, ... , m).

Все знаки алфавита составляют полную систему случайных событий, поэтому:

$\sum_{i=1}^{m} Pi = 1$ .

Число всех возможных сообщений длины **n**, в которых знак $\mathbf{x_i}$ входит $\mathbf{n_i}$ раз, где **i = 1, 2, 3 ... ,m,** определяется как число перестановок с повторениями из **n** элементов, спецификация которых $\mathbf{\{n_1, n_2, ..., n_m\}}$**.** Поэтому количество возможных сообщений определяют по формуле:

$N = \frac{n!}{n_1! n_2! \ldots n_m!}$ .

Например, план застройки улицы 10 домами, среди которых 3 дома одного типа, 5 другого и 2 третьего, можно представить

$$\frac{10!}{3!5!2!} = 2520 способами$$

.

Количество информации можно найти по формуле:

$I = \log N = \log n! - (\log n_1! + \log n_2! + ... + \log n_m!)$.

Для достаточно больших **n** это выражение можно преобразовать с помощью приближенной формулы Стирлинга:

$\log n! \approx n(\ln n - 1)$.

Воспользовавшись формулой Стирлинга и соотношением $\sum_{i=1}^{m} n_i = n$, получают:

$$I = \ln N = n(\ln n - 1) - \sum_{i=1}^{m} n_i(\ln n_i - 1) = n \ln n - \sum_{i=1}^{m} n_i \ln n_i =$$

$$= -n[-\ln n + \sum_{i=1}^{m} \frac{n_i}{n}(\ln \frac{n_i}{n} + \ln n)] = -n(-\ln n + \sum_{i=1}^{m} \frac{n_i}{n} \ln \frac{n_i}{n} +$$

$$+ \ln n \sum_{i=1}^{m} \frac{n_i}{n}) = -n \sum_{i=1}^{m} \frac{n_i}{n} \ln \frac{n_i}{n}$$

Переходя к вероятностям и произвольным основаниям логарифмов, получают ***формулы Шеннона для количества информации и энтропии:***

$$I = -n \sum_{i=1}^{m} P_i \log P_i;$$

$$H = -\sum_{i=1}^{m} P_i \log P_i$$

***В дальнейшем в выражениях для количества информации I и энтропии H всегда используют логарифмы с основанием 2.***

## 2.1 Свойства энтропии

При равновероятности знаков алфавита **Pi = 1/m** из формулы Шеннона получают:

$$H = -\sum_{i=1}^{m} Pi \log Pi = -\sum_{i=1}^{m} \frac{1}{m} \log \frac{1}{m} = -(m \frac{1}{m})(-\log m) = \log m$$

.

***Из этого следует, что при равновероятности знаков алфовита энтропия определяется исключительно числом знаков m алфавита и по существу является характеристикой только алфавита.***

*Если же знаки алфавита неравновероятны, то алфавит можно рассматривать как дискретную случайную величину, заданную статистическим распределением частот* $\mathbf{n_i}$ *появления знаков* $\mathbf{x_i}$ (*или вероятностей* $\mathbf{Pi = n_i / n}$) табл. 2.1:

Таблица 2.1.

| | | | | |
|---|---|---|---|---|
| Знаки $x_i$ | $x_1$ | $x_2$ | . . . | $x_m$ |
| Частоты $n_i$ | $n_1$ | $n_2$ | . . . | $n_m$ |

Такие распределения получают обычно на основе статистического анализа конкретных типов сообщений (например, русских или английских текстов и т.п.).

Поэтому, если знаки алфавита неравновероятны и хотя формально в выражение для энтропии входят только характеристики алфавита (вероятности появления его знаков), энтропия отражает статистические свойства некоторой совокупности сообщений.

На основании выражения

$$H = -\sum_{i=1}^{m} Pi \log Pi = \sum_{i=1}^{m} Pi \log \frac{1}{Pi} \quad ,$$

величину **log 1/Pi** можно рассматривать как **<u>частную энтропию</u>**, характеризующую информативность знака $\mathbf{x_i}$, а **энтропию H -** как **среднее значение частных энтропий.**

Функция **(Pi · log Pi)** отражает вклад знака $\mathbf{x_i}$ в энтропию **H.** При вероятности появления знака **Pi=1** эта функция равна нулю, затем возрастает до своего максимума, а при дальнейшем уменьшении **Pi** стремится к нулю (функция имеет экстремум): рис.2.1.

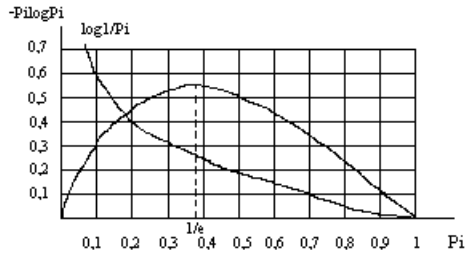



Рис. 2.1. Графики функций log 1/Pi и -Pi · log Pi

Для определения координат максимума этой функции нужно найти производную и приравнять ее к нулю.

Из условия $\frac{\partial}{\partial Pi}(-Pi \log Pi) = -\log Pi - \log e = -\log Pi\, e = 0$ находят: **Pi e = 1**,где **e** - основание натурального логарифма.

Таким образом, функция: **(Pi log Pi)** при **Pi = 1/e = 0,37** имеет максимум: $\frac{1}{e}\log e = 0{,}531$., т.е координаты максимума (0,37; 0,531)

*Энтропия* **H** - *величина вещественная, неотрицательная и ограниченная,* т.е. **H ≥ 0** (это свойство следует из того, что такими же качествами обладают все ее слагаемые **Pi log 1/Pi).**

Энтропия равна нулю, если сообщение известно заранее (в этом случае каждый элемент сообщения замещается некоторым знаком с вероятностью, равной единице, а вероятности остальных знаков равны нулю).

***Энтропия максимальна, если все знаки алфавита равновероятны****, т.е.* $\boldsymbol{H_{max} = \log m}$.

Таким образом, степень неопределенности источника информации зависит не только от числа состояний, но и от вероятностей этих состояний. При неравновероятных состояниях свобода выбора источника ограничивается, что должно приводить к уменьшению неопределенности. Если источник информации имеет, например, два возможных состояния с вероятностями 0,99 и 0,01, то неопределенность выбора у него значительно меньше, чем у источника, имеющего два равновероятных состояния. Действительно, в первом случае результат практически предрешен (реализация состояния, вероятность которого равна 0,99), а во втором случае неопределенность максимальна, поскольку никакого обоснованного предположения о результате выбора сделать нельзя. Ясно также, что весьма малое изменение вероятностей состояний вызывает соответственно незначительное изменение неопределенности выбора.

**Пример3.** Распределение знаков алфавита имеет вид $\mathbf{p(x_1) = 0,1\ p(x_2) = 0,1\ p(x_3) = 0,1\ p(x_4) = 0,7.}$ Определить число знаков другого алфавита, у которого все знаки равновероятны, а энтропия такая же как и у заданного алфавита.

Особый интерес представляют ***бинарные сообщения***, использующие алфавит из двух знаков: **(0,1).** При $\mathbf{m = 2}$ сумма вероятностей знаков алфавита: $\mathbf{P_1+P_2 = 1}$. Можно положить $\mathbf{P_1 = P}$, тогда $\mathbf{P_2 = 1-P.}$

Энтропию можно определить по формуле:

$$H = -P_1 \log P_1 - P_2 \log P_2 = -P \log P - (1 - P) \log(1 - P),$$

*Энтропия бинарных сообщений достигает максимального значения, равного 1 биту, когда знаки алфавита сообщений равновероятны, т.е. при P = 0,5, и ее график симметричен относительно этого значения.(рис.2.2).*

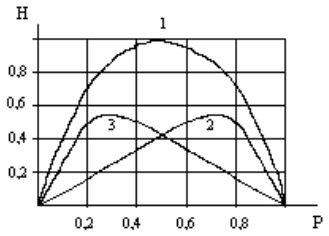



Рис. 2.2. График зависимости энтропии **H** двоичных сообщений (1) и ее составляющих (2,3): **- (1 - P) log (1 - P)** и **- P log P** от **P**.

**Пример 4.** Сравнить неопределенность, приходящуюся на букву источника информации (алфавита русского языка), характеризуемого ансамблем, представленным в таблице 2.2, с неопределенностью, которая была бы у того же источника при равновероятном использовании букв.

Таблица 2.2.

| Буква | Вероятность | Буква | Вероятность | Буква | Вероятность | Буква | Вероятность |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| а | 0,064 | й | 0,010 | т | 0,056 | ы | 0,016 |
| б | 0,015 | к | 0,029 | у | 0,021 | э | 0,003 |
| в | 0,039 | л | 0,036 | ф | 0,02 | ю | 0,007 |
| г | 0,014 | м | 0,026 | х | 0,09 | я | 0,019 |
| д | 0,026 | н | 0,056 | ц | 0,04 | пробел | 0,143 |
| е,ё | 0,074 | о | 0,096 | ч | 0,013 | | |
| ж | 0,008 | п | 0,024 | ш | 0,006 | | |
| з | 0,015 | р | 0,041 | ш | 0,003 | | |
| и | 0,064 | с | 0,047 | ъ,ь | 0,015 | | |

**Решение**. 1. При одинаковых вероятностях появления любой из всех **m = 32** букв алфавита неопределенность, приходящуюся на одну букву, характеризует энтропия

H = log m = log 32 = 5 бит.

2. Энтропию источника, характеризуемого заданным табл. 2.2 ансамблем, находят по формуле:

$H = -\sum_{i=1}^{m} p_i \log p_i =$ -0,064 log 0,064 -0,015log0,015 - 0,143log0,143 ≈ 4,43 бит.

*Таким образом, неравномерность распределения вероятностей использования букв снижает энтропию источника с 5 до 4,42 бит*

**Пример 5.** Заданы ансамбли **X** и **Y** двух дискретных величин**:**

Таблица 2.3.

| Случайные величины $x_i$ | 0,5 | 0,7 | 0,9 | 0,3 |
|---|---|---|---|---|
| Вероятности их появления | 0,25 | 0,25 | 0,25 | 0,25 |

Таблица 2.4.

| Случайные величины $y_j$ | 5 | 10 | 15 | 8 |
|---|---|---|---|---|
| Вероятности их появления | 0,25 | 0,25 | 0,25 | 0,25 |

Сравнить их энтропии.

**Решение**. Энтропия не зависит от конкретных значений случайной величины. Так как вероятности их появления в обоих случаях одинаковы, то

$H(X) = H(Y) = -\sum_{i=1}^{m} p_i \log p_i =$ - 4(0,25log0,25) = -4(1/4log1/4) =

= log 4 = 2 бит

## 2.2 Энтропия при непрерывном сообщении

В предыдущих параграфах была рассмотрена мера неопределенности выбора для дискретного источника информации. На практике в основном встречаются с источниками

информации, множество возможных состояний которых составляет континуум. Такие источники называют *непрерывными* источниками информации.

Во многих случаях они преобразуются в дискретные посредством использования устройств дискретизации и квантования. Вместе с тем существует немало и таких систем, в которых информация передается и преобразуется непосредственно в форме непрерывных сигналов. Примерами могут служить системы телефонной связи и телевидения.

Оценка неопределенности выбора для непрерывного источника информации имеет определенную специфику. Во-первых, значения, реализуемые источником, математически отображаются случайной непрерывной величиной. Во-вторых, вероятности значений этой случайной величины не могут использоваться для оценки неопределенности, поскольку в данном случае вероятность любого конкретного значения равна нулю. Естественно, однако, связывать неопределенность выбора значения случайной непрерывной величины с плотностью распределения вероятностей этих значений. Учитывая, что для совокупности значений, относящихся к любому сколь угодно малому интервалу случайной непрерывной величины, вероятность конечна, попытаемся найти формулу для энтропии непрерывного источника информации, используя операции квантования и последующего предельного перехода при уменьшении кванта до нуля.

Для обобщения формулы **Шеннона** разобьем интервал возможных состояний случайной непрерывной величины **X** на равные непересекающиеся отрезки **Δx** и рассмотрим множество дискретных состояний $\mathbf{x_1, x_2, \dots, x_m}$ с вероятностями $\mathbf{Pi = p(x_i)\Delta x\ (i = 1, 2, \dots, m).}$ Тогда энтропию можно вычислить по формуле:

$$H = -\sum_{i=1}^{m} p(x_i)\Delta x \log p(x_i)\Delta x = -\sum_{i=1}^{m} p(x_i)\Delta x \log p(x_i) - -\sum_{i=1}^{m} p(x_i)\Delta x \log \Delta x$$

В пределе при $\mathbf{\Delta x \to 0}$ с учетом соотношения:

$$\int_{-\infty}^{\infty} p(x)dx = 1,$$

Получим $$H(x) = -\int_{-\infty}^{\infty} p(x) \log p(x)dx - \log\Delta x.$$

Первое слагаемое в правой части соотношения имеет конечное значение, которое зависит только от закона распределения непрерывной случайной величины X и не зависит от шага квантования. Оно имеет точно такую же структуру, как энтропия дискретного источника.

Поскольку для определения этой величины используется только функция плотности вероятности, т. е. дифференциальный закон распределения, она получила название относительной дифференциальной энтропии или просто ***дифференциальной энтропии*** непрерывного источника информации (непрерывного распределения случайной величины X).

Первое слагаемое в этой сумме, называемое также ***приведенной энтропией***, целиком определяет информативность сообщений, обусловленных статистикой состояний их элементов.

Величина **log$\Delta$x** зависит только от выбранного интервала **$\Delta$x,** определяющего точность квантования состояний, и при **$\Delta$x =const** она постоянна.

***Энтропия и количество информации зависят от распределения плотности вероятностей p(x).***

В теории информации большое значение имеет решение вопроса о том, при каком распределении обеспечивается максимальная энтропия **H(x).**

*Можно показать, что при заданной дисперсии:*

$$\sigma^2 = \int_{-\infty}^{\infty} x^2 p(x)\,dx = const,$$

наибольшей информативностью сообщение обладает только тогда, когда состояния его элементов распределены по нормальному закону:

$$p(x) = \frac{1}{\sigma\sqrt{\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

Так как дисперсия определяет среднюю мощность сигнала, то отсюда следуют практически важные выводы.

Передача наибольшего количества информации при заданной мощности сигнала (или наиболее экономичная передача информации) достигается при такой обработке сигнала, которая приближает распределение плотности вероятности его элементов к нормальному распределению.

В то же время, обеспечивая нормальное распределение плотности вероятности элементам помехи, обеспечивают ее наибольшую “ информативность”, т.е наибольшее пагубное воздействие на прохождение сигнала. Найдем значение энтропии, когда состояния элементов источника сообщений распределены по *нормальному закону*:

$$H = -\frac{1}{e\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \log[\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}]dx - \log\Delta x =$$

$$= \log(\sigma\sqrt{2\pi}\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} dx + \frac{\log e}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} dx - \log\Delta x) =$$

$$= \log(\frac{\sigma}{\Delta x}\sqrt{2\pi e})$$

.

Найдем значение энтропии, когда состояния элементов распределены внутри интервала их существования **a ≤ x ≤ b** по *равномерному закону, т.е*

$$\text{p(x)} = \begin{cases} \frac{1}{b-a} & при\ a \le x \le b \\ 0 & при\ x < a, x > b \end{cases}$$

$$H_p(x) = -\frac{1}{b-a}\int_a^b \log(\frac{1}{b-a} - \Delta x)dx = \log\frac{b-a}{\Delta x}$$ .

Дисперсия равномерного распределения $\sigma_p^2 = \frac{(b-a)^2}{12}$, поэтому (b-a) = $2\sqrt{3}\ \sigma_p$. С учетом этого можно записать

$$H_p(x) = \log(\frac{\sigma_p}{\Delta x}2\sqrt{3})$$ .

Сравнивая между собой сообщения с равномерным и нормальным распределением вероятностей при условии $H_н(x) = H_р(x)$, получаем:

$$\sigma_p^2 = \frac{e}{6}\sigma^2 \approx 1{,}42\sigma^2$$

Это значит, что при одинаковой информативности сообщений средняя мощность сигналов для **равномерного распределения** их амплитуд должна быть на 42% больше, чем при **нормальном распределении амплитуд.**

**Пример 7.** Найдите энтропию случайной величины, распределенной по закону с плотностью вероятности

$$p(x) = \begin{cases} 0, & x \le 0 \\ x^2, & 0 < x \le 1 \\ 1, & x > 1 \end{cases}$$

**Пример 8.** При организации мешающего воздействия при передаче информации можно использовать источник шума с нормальным распределением плотности и источник, имеющий в некотором интервале равномерную плотность распределения. Определить, какой источник шума применять экономичнее, каков при этом выигрыш в мощности.

**Решение**. Сравнение источников следует проводить из условия обеспечения равенства энтропий, когда каждый источник вносит одинаковое мешающее воздействие при передаче информации, но, очевидно, затрачивая при этом не одинаковые мощности.

Как было показано выше, значение энтропии, когда состояния элементов распределены по нормальному закону, можно найти по формуле:

$$H_н(x) = \log(\frac{\sigma}{\Delta x}\sqrt{2\pi e}) = \log(\sigma_Г\sqrt{2\pi e})$$ ,

где Δx = 1 Ом, а $\frac{\sigma}{\Delta x} = \sigma_Г$, т.е. $\sigma_г^2$ - дисперсия, характеризующая мощность, выделяемую на резистора с сопротивлением 1Ом.

Для равномерного распределения энтропию можно найти по формуле:

$H_p(x) = \log \frac{b-a}{\Delta x}$

Так как дисперсия равномерного распределения

$\sigma_h^2 = \frac{(b-a)^2}{12}$, *то* $b - a = 2\sqrt{3}\,\sigma_p$, *и, следовательно*

$H_p(x) = \log(\frac{\sigma_p}{\Delta x} 2\sqrt{3}) = \log(\sigma_{pГ} 2\sqrt{3})$, *где*

$\sigma_{pГ} = \frac{\sigma_p}{\Delta x}$, $\Delta x = 1$ *Ом*

$H_н(x) = H_p(x)$, *то* $\log(\sigma_Г \sqrt{2\pi e}) = \log(\sigma_{pГ} 2\sqrt{3})$,

$2\pi e \sigma_Г^2 = 12\sigma_{pГ}$,

Так как $\sigma_{pГ}^2 = \frac{\pi e}{6}\sigma_Г^2 \approx 1{,}42\sigma_Г^2$

Поэтому следует выбирать источник шума с нормальным распределением плотности распределения амплитуд, т.к. при той же неопределенности, вносимой им в канал связи, можно выиграть в мощности 42%.

Поэтому следует выбирать источник шума с нормальным распределением плотности, т.к. при той же неопределенности, вносимой им в канал связи, можно выиграть в мощности 42%.

### 2.3 Условная энтропия

До сих пор предполагалось, что все элементы сообщения независимы, т.е. появление каждого данного элемента никак не связано с предшествующими элементами.

Рассмотрим теперь два ансамбля

$X = (x_1, x_2, \ldots, x_r)$

$Y = (y_1, y_2, \ldots, y_s)$ ,

которые определяются не только собственными вероятностями $p(x_i)$ и $p(y_j)$, но и условными вероятностями $p_{xi}(y_j)$, $p_{yj}(x_i)$, где i = 1, 2, ... , r ; j = 1, 2, ... , s.

Систему двух случайных величин (сообщений) X, Y можно изобразить случайной точкой на плоскости. Событие, состоящее в попадании случайной точки (X, Y) в область D, принято обозначать в виде $(X, Y) \subset D$.

Закон распределения системы двух случайных дискретных величин может быть задан с помощью табл. 2.5.

Таблица 2.5.

| $y_j$ | | | | |
|---|---|---|---|---|
| Y / X | $y_1$ | $y_2$ | . . . | $y_s$ |
| $x_i$ | | | | |

| | | | | |
|---|---|---|---|---|
| $x_1$ | $P_{11}$ | $P_{12}$ | . . . | $P_{1s}$ |
| $x_2$ | $P_{21}$ | $P_{22}$ | . . . | $P_{2s}$ |
| : | : | : | : | : |
| $x_r$ | $P_{r1}$ | $P_{r2}$ | . . . | $P_{rs}$ |

где $P_{ij}$ - вероятность события, заключающегося в одновременном выполнении равенства $X = x_i$, $Y = y_j$. При этом

$$\sum_{i=1}^{r}\sum_{j=1}^{s} P_{ij} = 1 .$$

Закон распределения системы случайных непрерывных величин (X, Y) задают при помощи функции плотности вероятности p(x, y).

Вероятность попадания случайной точки (X,Y) в область D определяется равенством

$$P[(X,Y) \subset D] = \iint_D p(x,y)dxdy .$$

Функция плотности вероятности обладает следующими свойствами:

1) $p(x,y) \geq 0$

2) $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(x,y)dxdy = 1$

Если все случайные точки (X,Y) принадлежат области D, то

$$\iint_D p(x,y)dxdy = 1 .$$

***<u>Условным распределением</u>*** составляющей X при $Y = y_j$ (yj сохраняет одно и то же значение при всех возможных значениях X) называют совокупность условных вероятностей $P_{yj}(x_1), P_{yj}(x_2), \ldots, P_{yj}(x_r)$

Аналогично определяется условное распределение составляющей Y.

Условные вероятности составляющих X и Y вычисляют соответственно по формулам:

$$P_{xi}(y_j) = \frac{P(x_i, y_j)}{P(x_i)}$$

$$P_{yj}(x_i) = \frac{P(x_i, y_j)}{P(y_j)}$$

Для контроля вычислений целесообразно убедиться, что сумма вероятностей условного распределения равна единице.

Так как условная вероятность события $y_j$ при условии выполнения события $x_i$ принимается по определению

$$P_{x_i}(y_j) = \frac{P(x_i, y_j)}{P(x_i)},$$

то вероятность совместного появления совокупности состояний

$P(x_i,y_j) = P(x_i)\, P_{xi}(y_j)$.

Аналогично, условимся вероятность события $x_i$ при условии выполнения события $y_j$:

$$P(x_i, y_j) = P(y_j) P_{yi}(x_i)$$

.

Поэтому общую энтропию зависимых ансамблей X и Y определяют по формуле Шеннона:

$$H(X,Y) = -\sum_{i=1}^{r}\sum_{j=1}^{s} P(x_i, y_j)\log P(x_i, y_j) =$$

$$= \sum_{i=1}^{r}\sum_{j=1}^{s} P(x_i) P_{x_i}(y_j)\log[P(x_i) P_{x_i}(y_j)] =$$

$$= \sum_{i=1}^{r} P(x_i)\log P(x_i)\sum_{j=1}^{s} P_{x_i}(y_j) - \sum_{i=1}^{r} P(x_i)\sum_{j=1}^{s} P_{x_i}(y_j)\log P_{x_i}(y_j).$$

С учетом соотношения $\sum_{j=1}^{s} P_{x_i}(y_j) = 1$ получают

$H(X,Y) = H(X) + H_X(Y)$, где $H(X)$ - энтропия ансамбля X;

$H_X(Y)$ - условная энтропия ансамбля Y при условии, что сообщение ансамбля X известны:

$$H_X(Y) = -\sum_{i=1}^{r} P(x_i)\sum_{j=1}^{s} P_{x_i}(y_j)\log P_{x_i}(y_j)$$

Для независимых событий X и Y: $P_{xi}(y_j) = P(y_j)$ и поэтому

$H_X(Y) = H(Y)$ и, следовательно, $H(X,Y) = H(X) + H(Y)$.

Если X и Y полностью зависимы, т.е. при появлении $x_i$ неизбежно следует $y_j$, то $P(x_i,y_j)$ равна единице при $i = j$ и нулю при $i \neq j$. Поэтому $H_X(Y) = 0$, и , следовательно, $H(X,Y) = H(X)$, т.е. при полной зависимости двух ансамблей один из них не вносит никакой информации.

Полученное выражение для условной энтропии

$$H_X(Y) = -\sum_{i=1}^{r} P(x_i)\sum_{j=1}^{s} P_{x_i}(y_j)\log P_{x_i}(y_j)$$

можно использовать и как информативную характеристику одного ансамбля X, элементы которого взаимно зависимы. Положив Y = X, получим

$$H' = -\sum_{i=1}^{r} P(x_i)\sum_{j=1}^{s} P_{x_i}(x_j)\log P_{x_i}(x_j).$$

Например, алфавит состоит из двух элементов 0 и 1. Если эти элементы равновероятны, то количество информации, приходящееся на один элемент сообщения: $H_0 = \log m = \log 2 = 1$ бит. Если же, например, P(0)=s, а P(1) = j, то

$$H = -\sum_{i=1}^{m} P_i \log P_i = -P(0)\log P(0) - P(1)\log P(1) =$$

$$= -\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right) = 0{,}815$$

В случае же взаимной зависимости элементов, определяемой, например, условными вероятностями $P_0(0) = 2/3$; $P_0(1) = 1/3$; $P_1(0) = 1$; $P_1(1) = 0$, то условная энтропия

$$H' = -P(0)[P_0(0)\log P_0(0) + P_0(1)\log P_0(1)] - P(1)[P_1(0)\log P_1(0) +$$

$$+ P_1(1)\log P_1(1)] = -\frac{3}{4}(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}) = 0{,}685$$

Энтропия при взаимно зависимых элементах всегда меньше, чем при независимых, т.е. H'<H.

Пример9: Задано распределение вероятностей случайной дискретной двумерной величины:

Таблица 2.6.

| Y / X | 4 | 5 |
|---|---|---|
| 3 | 0,17 | 0,10 |
| 10 | 0,13 | 0,30 |
| 12 | 0,25 | 0,05 |

Найти законы распределения составляющих X и Y.

Решение: 1) Сложив вероятности "по строкам", получим вероятности возможных значений X:

P(3) = 0,17 + 0,10 = 0,27

P(10) = 0,13 +0,30 = 0,43

P(12) = 0,25 + 0,05 = 0,30.

Запишем закон распределения составляющей X:

Таблица 2.7.

| X | 3 | 10 | 12 |
|---|---|---|---|
| $P(x_i)$ | 0,27 | 0,43 | 0,30 |

Контроль: 0,27 + 0,43 + 0,30 = 1

2) Сложив вероятности "по столбцам", аналогично найдем распределение составляющей Y:

Таблица 2.8.

| Y | 4 | 5 |
|---|---|---|
| $P(y_j)$ | 0,55 | 0,45 |

Контроль: 0,55 + 0,45 = 1

Пример10: Задана случайная дискретная двумерная величина (X,Y):

Таблица 2.9.

| Y / X | $y_1 = 0{,}4$ | $y_2 = 0{,}8$ |
|---|---|---|
| $x_1 = 2$ | 0,15 | 0,05 |

| | | |
|---|---|---|
| $x_2 = 5$ | 0,30 | 0,12 |
| $x_3 = 8$ | 0,35 | 0,03 |

Найти: безусловные законы распределения составляющих; условный закон распределения составляющей X при условии, что составляющая Y приняла значение $y_1 = 0,4$; условный закон распределения составляющей Y при условии, что составляющая X приняла значение $x_2 = 5$

Решение: 1) Сложив вероятности "по строкам", напишем закон распределения X.

Таблица 2.10.

| X | 2 | 5 | 8 |
|---|---|---|---|
| P(x) | 0,20 | 0,42 | 0,38 |

2) Сложив вероятности "по столбцам", найдем закон распределения Y.

Таблица 2.11.

| Y | 0,4 | 0,8 |
|---|---|---|
| P(y) | 0,80 | 0,20 |

3) Найдем условные вероятности возможных значений X при условии, что составляющая Y приняла значение $y_1 = 0,4$

$$P_{y1}(x_1) = \frac{P(x_1, y_1)}{P(y_1)} = \frac{0,15}{0,80} = \frac{3}{16},$$

$$P_{y1}(x_2) = \frac{P(x_2, y_1)}{P(y_1)} = \frac{0,30}{0,80} = \frac{3}{8},$$

$$P_{y1}(x_3) = \frac{P(x_3, y_1)}{P(y_1)} = \frac{0,35}{0,80} = \frac{7}{16}$$

Напишем искомый условный закон распределения X:

Таблица 2.12.

| X | 2 | 5 | 8 |
|---|---|---|---|
| $P_{y1}(x_i)$ | 3/16 | 3/8 | 7/16 |

Контроль: 3/16 + 3/8 + 7/16 = 1

Аналогично найдем условный закон распределения Y:

Таблица 2.13.

| Y | 0,4 | 0,8 |
|---|---|---|
| $P_{x2}(y_j)$ | 5/7 | 2/7 |

Контроль: 5/7 + 2/7 = 1.

Пример11: Закон распределения вероятностей системы, объединяющей зависимые источники информации X и Y, задан с помощью таблицы:

Таблица 2.14.

| Y \ | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|

| X | | | |
|---|---|---|---|
| $x_1$ | 0,4 | 0,1 | 0 |
| $x_2$ | 0 | 0,2 | 0,1 |
| $x_3$ | 0 | 0 | 0,2 |

Определить энтропии H(X), H(Y), $H_X$(Y), H(X,Y).

Решение: 1. Вычислим безусловные вероятности $P(x_i)$ и $P(y_j)$ системы:

а) сложив вероятности “по строкам” , получим вероятности возможных значений X: $P(x_1) = 0,5$

$P(x_2) = 0,3$

$P(x_3) = 0,2$

б) сложив вероятности “по столбцам”, получим вероятности возможных значений Y:

$P(y_1) = 0,4$

$P(y_2) = 0,3$

$P(y_3) = 0,3$

2. Энтропия источника информации X:

$$H(X) = -\sum_{i=1}^{r} P(x_i) \log P(x_i) = -(0,5 \log 0,5 + 0,3 \log 0,3 +$$

$$+ 0,2 \log 0,2) = 1,485 бит$$

3. Энтропия источника информации Y:

$$H(Y) = -\sum_{j=1}^{s} P(y_j) \log P(y_j) = -(0,4 \log 0,4 + 0,3 \log 0,3 +$$

$$+ 0,3 \log 0,3) = 1,57 бит$$

4. Условная энтропия источника информации Y при условии, что сообщения источника X известны:

$$H(Y) = -\sum_{i=1}^{r} P(x_i) \sum_{j=1}^{s} P_{x_i}(y_j) \log P_{x_i}(y_j)$$

Так как условная вероятность события $y_j$ при условии выполнения события $x_i$ принимается по определению

$$P_{x_i}(y_j) = \frac{P(x_i, y_j)}{P(x_i)},$$

поэтому найдем условные вероятности возможных значений Y при условии, что составляющая X приняла значение $x_1$:

$$P_{x_1}(y_1) = \frac{P(x_1, y_1)}{P(x_1)} = \frac{0,4}{0,5} = 0,8$$

$$P_{x_1}(y_2) = \frac{P(x_1, y_2)}{P(x_1)} = \frac{0,1}{0,5} = 0,2$$

$$P_{x_1}(y_3) = \frac{P(x_1, y_3)}{P(x_1)} = \frac{0}{0,5} = 0$$

Для $x_2$:

$$P_{x_2}(y_1) = \frac{P(x_2, y_1)}{P(x_2)} = \frac{0}{0,3} = 0$$

$$P_{x_2}(y_2) = \frac{P(x_2, y_2)}{P(x_2)} = \frac{0,2}{0,3} = 0,67$$

$$P_{x_2}(y_3) = \frac{P(x_2, y_3)}{P(x_2)} = \frac{0,1}{0,3} = 0,33$$

Для $x_3$:

$$P_{x_3}(y_1) = \frac{P(x_3, y_1)}{P(x_3)} = \frac{0}{0,2} = 0$$

$$P_{x_3}(y_2) = \frac{P(x_3, y_2)}{P(x_3)} = \frac{0}{0,2} = 0$$

$$P_{x_3}(y_3) = \frac{P(x_3, y_3)}{P(x_3)} = \frac{0,2}{0,2} = 1$$

Поэтому: $H_X(Y) = - [0,5 (0,8 \log 0,8 + 0,2 \log 0,2) +$

$+0,3 (0,67 \log 0,67 + 0,33 \log 0,33) + 0,2 (1 \log 1)] = 0,635$

5. Аналогично, условная энтропия источника информации X при условии, что сообщения источника Y известны:

$$H_Y(X) = -\sum_{j=1}^{s} P(y_j) \sum_{i=1}^{r} P_{y_j}(x_i) \log P_{y_j}(x_i).$$

$$P_{y_j}(x_i) = \frac{P(x_i, y_j)}{P(y_j)};$$

Для $y_1$:

$$P_{y_1}(x_1) = \frac{P(x_1, y_1)}{P(y_1)} = \frac{0,4}{0,4} = 1$$

$$P_{y_1}(x_2) = \frac{P(x_2, y_1)}{P(y_1)} = \frac{0}{0,4} = 0$$

$$P_{y_1}(x_3) = \frac{P(x_3, y_1)}{P(y_1)} = \frac{0}{0,4} = 0$$

Для $y_2$:

$$P_{y_2}(x_1) = \frac{P(x_1, y_2)}{P(y_2)} = \frac{0,1}{0,3} = 0,33$$

$$P_{y_2}(x_2) = \frac{P(x_2, y_2)}{P(y_2)} = \frac{0,2}{0,3} = 0,67$$

$$P_{y_2}(x_3) = \frac{P(x_3, y_2)}{P(y_2)} = \frac{0}{0,3} = 0$$

$$P_{y_3}(x_1) = \frac{P(x_1, y_3)}{P(y_3)} = \frac{0}{0,3} = 0$$

Для $y_3$:

$$P_{y_3}(x_2) = \frac{P(x_2, y_3)}{P(y_3)} = \frac{0,1}{0,3} = 0,33$$

$$P_{y_3}(x_3) = \frac{P(x_3, y_3)}{P(y_3)} = \frac{0,2}{0,3} = 0,67$$

$$H_Y(X) = -[0,4(1\log 1) + 0,3(0,33\log 0,33 + 0,67\log 0,67) +$$

$$+ 0,3(0,33\log 0,33 + 0,67\log 0,67)] \approx 0,55\ \textit{бит}.$$

6. Общая энтропия зависимых источников информации X и Y:

$$H(X,Y) = -\sum_{i=1}^{r}\sum_{j=1}^{s} P(x_i, y_j)\log P(x_i, y_j) =$$

$$= -(0,4\log 0,4 + 0,1\log 0,1 + 0,2\log 0,2 + 0,1\log 0,1 + 0,2\log 0,2) =$$

$$= 0,529 + 0,332 + 0,464 + 0,332 + 0,464 = 2,12\ \textit{бит}.$$

Проверим результат по формуле:

$H(X,Y) = H(X) + H_X(Y) = 1,485 + 0,635 = 2,12$ бит

$H(X,Y) = H(Y) + H_Y(X) = 1,57 + 0,55 = 2,12$ бит

Пример12: Известны энтропии двух зависимых источников H(X) = 5бит; H(Y) = 10бит. Определить, в каких пределах будет изменяться условная энтропия $H_X(Y)$ в максимально возможных пределах.

Решение: Уяснению соотношений между рассматриваемыми энтропиями источников информации способствует их графическое отображение.

При отсутствии взаимосвязи между источниками информации:

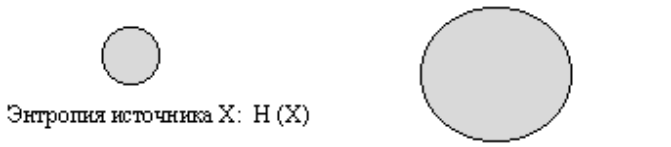



Рис. 2.5.

Если источники информации независимы, то $H_X(Y) = H(Y) = 10$ бит, а $H_Y(X) = H(X) = 5$ бит, и, следовательно, $H(X,Y) = H(X) + H(Y) = 5 + 10 = 15$ бит. Т.е., когда источники независимы $H_X(Y) = H(Y) = 10$ бит и поэтому принимают максимальное значение.

По мере увеличения взаимосвязи источников $H_X(Y)$ и $H_Y(X)$ будут уменьшаться:

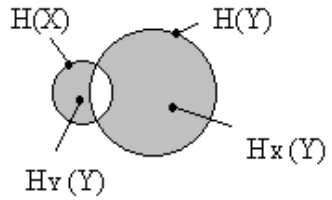



Рис. 2.6.

При полной зависимости двух источников один из них не вносит никакой информации, т.к. при появлении $x_i$ неизбежно следует $y_j$ , т.е. $P(x_i, y_j)$ равно единице при i = j и нулю при $i \neq j$. Поэтому

$H_Y(X) = 0$ и, следовательно, $H(X,Y) = H_X(Y)$ .

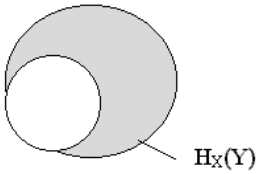



Рис. 2.7.

При этом $H_X(Y) = H(Y) - H(X) = 10 - 5 = 5$ бит. Поэтому $H_X(Y)$ будет изменяться от 10 бит до 5 бит при максимально возможном изменении $H_y(X)$ от 5 бит до 0 бит.

Пример13: Определите H(X) и $H_X(Y)$, если $P(x_1,y_1) = 0,3$; $P(x_1,y_2) = 0,2$;

$P(x_3,y_2) = 0,25$; $P(x_3,y_3) = 0,1$

Пример14: Определите H(X), H(Y), H(X,Y), если $P(x_1,y_1) = 0,2$; $P(x_2,y_1) = 0,4$;

$P(x_2,y_2) = 0,25$; $P(x_2,y_3) = 0,15$

## 2.4 Взаимная энтропия

Пусть ансамбли X и Y относятся соответственно к передаваемому и принимаемому сообщениям. Различия между X и Y обуславливаются искажениями в процессе передачи сообщений под воздействием помех.

При отсутствии помех различий между ансамблями X и Y не будет, а энтропии передаваемого и принимаемого сообщений будут равны: H(X) = H(Y).

Воздействие помех оценивают условной энтропией $H_Y(X)$. Поэтому получаемое потребителем количество информации на один элемент сообщения равно: $E(X,Y) = H(X) – H_Y(X)$

Величину E(X,Y) называют ***взаимной энтропией***.

Если ансамбли X и Y независимы, то это означает, что помехи в канале привели к полному искажению сообщения, т.е. $H_Y(X) = H(X)$, а получаемое потребителем количество информации на один элемент сообщения:E(X,Y)=0.

Если X и Y полностью зависимы, т.е. помехи в канале отсутствуют, то $H_Y(X) = 0$ и $E(X,Y) = H(Y)$.

Так как $H_Y(X) = H(X,Y) – H(Y)$, то $E(X,Y) = H(X) + H(Y) – H(X,Y)$, или

$$E(X,Y) = \sum_{i=1}^{r} \sum_{j=1}^{s} P(x_i, y_j) \log \frac{P_{y_j}(x_i)}{P(x_i)}$$

.

Пример15: Определите H(X) и E(X,Y), если $P(x_1,y_1) = 0,3$; $P(x_1,y_2) = 0,2$; $P(x_2,y_3) = 0,1$; $P(x_3,y_2) = 0,1$; $P(x_3,y_3) = 0,25$.

## 2.5 Избыточность сообщений

Чем больше энтропия, тем большее количество информации содержит в среднем каждый элемент сообщения.

Пусть энтропии двух источников сообщений $H_1<H_2$, а количество информации, получаемое от них одинаковое, т.е. $I = n_1H_1 = n_2H_2$, где $n_1$ и $n_2$ - длина сообщения от первого и второго источников. Обозначим

$$\mu = \frac{n_2}{n_1} = \frac{H_1}{H_2}$$

При передаче одинакового количества информации сообщение тем длиннее, чем меньше его энтропия.

Величина μ, называемая ***коэффициентом сжатия***, характеризует степень укорочения сообщения при переходе к кодированию состояний элементов, характеризующихся большей энтропией.

При этом доля излишних элементов оценивается ***коэффициентом избыточности***:

$$r = \frac{H_2 - H_1}{H_2} = 1 - \frac{H_1}{H_2} = 1 - \mu$$

Русский алфавит, включая пропуски между словами, содержит 32 элемента (см. Пример), следовательно, при одинаковых вероятностях появления всех 32 элементов алфавита, неопределенность , приходящаяся на один элемент, составляет $H_0 = \log 32 = 5$ бит

Анализ показывает, что с учетом неравномерного появления различных букв алфавита H = 4,42 бит, а с учетом зависимости двухбуквенных сочетаний H’ = 3,52 бит, т.е. $H’< H < H_0$

Обычно применяют три коэффициента избыточности:

1) частная избыточность, обусловленная взаимосвязью r’ = 1 - H’/H;
2) частная избыточность, зависящая от распределения r’’ = 1 - H/ $H_0$;
3) полная избыточность $r_0 = 1 - H’/ H_0$

Эти три величины связаны зависимостью $r_0 = r’ + r’’ - r’r’’$

Вследствие зависимости между сочетаниями, содержащими две и больше букв, а также смысловой зависимости между словами, избыточность русского языка (как и других европейских языков) превышает 50% ($r_0$ =1 - H'/ $H_0$ = 1 - 3,52/5 = 0,30).

Избыточность играет положительную роль, т.к. благодаря ней сообщения защищены от помех. Это используют при помехоустойчивом кодировании.

Вполне нормальный на вид лазерный диск может содержать внутренние (процесс записи сопряжен с появлением различного рода ошибок) и внешние (наличие физических разрушений поверхности диска ) дефекты. Однако даже при наличии физических разрушений поверхности лазерный диск может вполне нормально читаться за счет избыточности хранящихся на нем данных. Корректирующие коды С 1, С 2, Q - и Р - уровней восстанавливают все известные приводы, и их корректирующая способность может достигать двух ошибок на каждый из уровней С 1 и С 2 и до 86 и 52 ошибок на уровни Q и Р соответственно. Но затем, по мере разрастания дефектов, корректирующей способности кодов **Рида—Соломона** неожиданно перестает хватать, и диск без всяких видимых причин отказывает читаться, а то и вовсе не опознается приводом. Избыточность устраняют построением оптимальных кодов, которые укорачивают сообщения по сравнению с равномерными кодами. Это используют при архивации данных. Действие средств архивации основано на использовании алгоритмов сжатия, имеющих достаточно длинную историю развития, начавшуюся задолго до появления первого компьютера —/еще в 40-х гг. XX века. Группа ученых-математиков, работавших в области электротехники, заинтересовалась возможностью создания технологии хранения данных, обеспечивающей более экономное расходование пространства. Одним из них был **Клод Элвуд Шеннон**, основоположник современной теории информации. Из разработок того времени позже практическое применение нашли алгоритмы сжатия **Хаффмана** и **Шеннона-Фано**. А в 1977 г. математики **Якоб Зив** и **Абрахам Лемпел** придумали новый алгоритм сжатия, который позже доработал **Терри Велч**. Большинство методов данного преобразования имеют сложную теоретическую математическую основу. Суть работы архиваторов: они находят в файлах избыточную информацию (повторяющиеся участки и пробелы), кодируют их, а затем при распаковке восстанавливают исходные файлы по особым отметкам. Основой для архивации послужили алгоритмы сжатия **Я. Зива** и **А. Лемпела**. Первым широкое признание получил архиватор **Zip**. Со временем завоевали популярность и другие программы: **RAR, ARJ, ACE, TAR, LHA** и т. д.В операционной системе Windows достаточно четко обозначились два лидера: **WinZip** (домашняя страница этой утилиты находится в Internet по адресу **http://www.winzip.com)** и **WinRAR**, созданный российским программистом Евгением Рошалем (домашняя страница **http://www.rarlab.com). WinRAR** активно вытесняет **WinZip** так как имеет: удобный и интуитивно понятный интерфейс; мощную и гибкую систему архивации файлов; высокую

скорость работы; более плотно сжимает файлы. Обе утилиты обеспечивают совместимость с большим числом архивных форматов. Помимо них к довольно распространенным архиваторам можно причислить **WinArj** (домашняя страница **http://www.lasoft-oz.com)**. Стоит назвать **Cabinet Manager** (поддерживает формат **CAB**, разработанный компанией Microsoft для хранения дистрибутивов своих программ) и **WinAce** (работает с файлами с расширением **ace** и некоторыми другими). Необходимо упомянуть программы-оболочки **Norton Commander, Windows Commander или Far Manager.** Они позволяют путем настройки файлов конфигурации подключать внешние DOS-архиваторы командной строки и организовывать прозрачное манипулирование архивами, представляя их на экране в виде обычных каталогов. Благодаря этому с помощью комбинаций функциональных клавиш можно легко просматривать содержимое архивов, извлекать файлы из них и создавать новые архивы. Хотя программы архивации, предназначенные для MS-DOS, умеют работать и под управлением большинства версий Windows (в окне сеанса MS-DOS), применять их в этой операционной системе нецелесообразно. Дело в том, что при обработке файлов DOS-архиваторами их имена урезаются до 8 символов, что далеко не всегда удобно, а в некоторых случаях даже противопоказано.

Выбирая инструмент для работы с архивами, прежде всего, следует учитывать как минимум два фактора: эффективность, т. е. оптимальное соотношение между экономией дискового пространства и производительностью работы, и совместимость, т. е. возможность обмена данными с другими пользователями

Последняя, пожалуй, наиболее значима, так как по достигаемой степени сжатия, конкурирующие форматы и инструменты различаются на проценты, а высокая вычислительная мощность современных компьютеров делает время обработки архивов не столь существенным показателем. Поэтому при выборе программы-архиватора важнейшим критерием становится ее способность "понимать" наиболее распространенные архивные форматы.

При архивации надо иметь в виду, что качество сжатия файлов сильно зависит от степени избыточности хранящихся в них данных, которая определяется их типом. К примеру, степень избыточности у видеоданных обычно в несколько раз больше, чем у графических, а степень избыточности графических данных в несколько раз больше, чем текстовых. На практике это означает, что, скажем, изображения форматов **BMP** и **TIFF,** будучи помещенными в архив, как правило, уменьшаются в размере сильнее, чем документы **MS Word**. А вот рисунки **JPEG** уже заранее компрессированы, поэтому даже самый лучший архиватор для них будет мало эффективен. Также крайне незначительно сжимаются исполняемые файлы программ и архивы.

*Программы-архиваторы* можно разделить на три категории.

1. Программы, используемые для сжатия исполняемых файлов, причем все файлы, которые прошли сжатие, свободно запускаются, но изменение их содержимого, например

русификация, возможны только после их разархивации.

2. Программы, используемые для сжатия мультимедийных файлов, причем можно после сжатия эти файлы свободно использовать, хотя, как правило, при сжатии изменяется их формат (внутренняя структура), а иногда и ассоциируемая с ними программа, что может привести к проблемам с запуском.

3. 3. Программы, используемые для сжатия любых видов файлов и каталогов, причем в основном использование сжатых файлов возможно только после разархивации. Хотя имеются программы, которые "видят" некоторые типы архивов как самые обычные каталоги, но они имеют ряд неприятных нюансов, например, сильно нагружают центральный процессор, что исключает их использование на "слабых машинах".

Принцип работы архиваторов основан на поиске в файле "избыточной" информации и последующем ее кодировании с целью получения минимального объема. Самым известным методом архивации файлов является *сжатие последовательностей одинаковых символов.* Например, внутри вашего файла находятся последовательности байтов, которые часто повторяются. Вместо того, чтобы хранить каждый байт, фиксируется количество повторяемых символов и их позиция. Например, архивируемый файл занимает 15 байт и состоит из следующих символов:

B B B B B L L L L L A A A A A

В шестнадцатеричной системе

42 42 42 42 42 4C 4C 4C 4C 4C 41 41 41 41 41

Архиватор может представить этот файл в следующем виде (шестнадцатеричном):

01 05 42 06 05 4C 0A 05 41

Это значит: с первой позиции пять раз повторяется символ "B", с позиции 6 пять раз повторяется символ "L" и с позиции 11 пять раз повторяется символ "A". Для хранения файла в такой форме потребуется всего 9 байт, что на 6 байт меньше исходного.

Описанный метод является простым и очень эффективным способом сжатия файлов. Однако он не обеспечивает большой экономии объема, если обрабатываемый текст содержит небольшое количество последовательностей повторяющихся символов.

Более изощренный метод сжатия данных, используемый в том или ином виде практически любым архиватором, — это так называемый оптимальный префиксный код и, в частности, кодирование символами переменной длины (алгоритм Хаффмана).

Код переменной длины позволяет записывать наиболее часто встречающиеся символы и группы символов всего лишь несколькими битами, в то время как редкие символы и фразы будут записаны более длинными битовыми строками. Например, в любом английском тексте буква E встречается чаще, чем Z, а X и Q относятся к наименее встречающимся. Таким образом,

используя специальную таблицу соответствия, можно закодировать каждую букву Е меньшим числом битов и использовать более длинный код для более редких букв.

Популярные архиваторы **ARJ, РАК, PKZIP** работают на основе ***алгоритма Лемпела-Зива.*** Эти архиваторы классифицируются как адаптивные словарные кодировщики, в которых текстовые строки заменяются указателями на идентичные им строки, встречающиеся ранее в тексте. Например, все слова какой-нибудь книги могут быть представлены в виде номеров страниц и номеров строк некоторого словаря. Важнейшей отличительной чертой этого алгоритма является использование грамматического разбора предшествующего текста с расположением его на фразы, которые записываются в словарь. Указатели позволяют сделать ссылки на любую фразу в окне установленного размера, предшествующего текущей фразе. Если соответствие найдено, текущая фраза заменяется указателем на своего предыдущего двойника.

При архивации, как и при компрессировании, степень сжатия файлов сильно зависит от формата файла. Графические файлы, типа **TIF** и **GIF**, уже заранее компрессированы (хотя существует разновидность формата **TIFF** и без компрессии), и здесь даже самый лучший архиватор мало чего найдет для упаковки. Совсем другая картина наблюдается при архивации текстовых файлов, файлов **PostScript**, файлов **BMP** и им подобных.

## КОДИРОВАНИЕ ИЗОБРАЖЕНИЯ, ЗВУКА И ВИДИО

### ГРАФИЧЕСКИЕ ФОРМАТЫ ДЛЯ СОХРАНЕНИЯ ИЗОБРАЖЕНИЯ

Все изображения в компьютере хранятся в том или ином графическом формате (более 50 видов). Каждый из них имеет свои особенности. Если работают с графикой или пользуются цифровым фотоаппаратом, то выбор правильного формата во многом определяет качество работ. Все графические форматы делятся на две большие группы: растровые и векторные. Файлы первой из них содержат описание каждой точки изображения. Они представляют собой прямоугольную матрицу (bitmap), состоящую из маленьких квадратиков разного цвета — пикселей. Самыми распространенными форматами этой группы являются GIF и JPEG, использующиеся в Интернете, а также BMP (стандартный формат Windows) и TIFF, применяющийся при хранении отсканированных изображений и в полиграфии. В растровом формате хранятся фотографии, рисунки и обои Рабочего стола. Видео также является последовательностью растровых изображений.

Файлы векторных форматов содержат не растровые точки, как у фотоснимка, а математические формулы, описывающие координаты кривых. Например, прямая линия представлена координатами двух точек, а окружность — координатами центра и радиуса, поэтому достигается очень небольшой размер файла. Векторные графические форматы используют для передачи схем и рисунков, состоящих из набора линий, описываемых математическими формулами. В векторных форматах сохраняются логотипы, схематические

рисунки, текст, предназначенный для вывода на печать, и другие подобные объекты.

Стандартными средствами преобразовать фотоизображение в векторный формат практически невозможно, для этого требуются специальные программы - конверторы. Наиболее распространенные расширения векторных файлов — AI для пакета Adobe Illustrator, CDR для пакета CorelDRAW и WMF (еще один «стандартный» формат Windows).

Если отбросить устаревшие форматы PCX и TIFF, то остается всего четыре – BMP, JPEG, PNG, GIF. Скоро можно будет и GIF считать не рациональным.

**Формат BMP**

BMP (Windows Device Independent Bitmap) — это один из старейших форматов, к тому же являющийся «родным» форматом Windows. Основная область применения BMP — хранение файлов, использующихся внутри операционной системы (например, обоев для Рабочего стола или иконок программ). Файл, сохраненный в этом формате, прочитает любой графический редактор. Файлы, сохраненные в BMP, могут содержать как 256 цветов, так и 16 700 000 оттенков. BMP - самый простой (если не сказать примитивный) графический формат. Записанный в нем файл представляет собой массив данных, содержащий информацию о цвете каждого пиксела, т. е. изображение размером 1024x768 точек с глубиной цвета 24 бита будет занимать 1024x768x3= 2 359 296 байт (без учета служебной информации об объеме и имени файла, составляющей еще несколько сот байт). Основной недостаток формата, ограничивающий его применение, — большой размер BMP-файлов. Конечно, можно попробовать сохранить изображение в формате BMP со сжатием, однако это часто вызывает проблемы при работе с некоторыми графическими пакетами. Даже при нынешних емкостях жестких дисков хранить коллекцию 2,5-Мбайт графических файлов крайне обременительно. Да и передача графики в формате BMP по сети (по крайней мере, без сжатия с помощью различных архиваторов) — обременительно. 10—20 лет назад возможности «железа» были намного скромнее, поэтому вопрос о разработке методов сжатия графической информации до приемлемых размеров стоял очень остро.

**От формата LZW к формату GIF**

Как и BMP, GIF (CompuServe Graphics Interchange Format) является одним из старейших форматов. В 1977 г. израильские математики А. Лемпел и Я. Зив разработали новый класс алгоритмов сжатия без потерь, получивший название метод Лемпела—Зива. Одновременно с самим алгоритмом, именуемым LZ77 (по первым буквам фамилий ученых и последним цифрам года создания), свет увидела статья, где излагались общие идеи данного метода. Впоследствии появилась усовершенствованная версия продукта — LZ78. Многие специалисты стремились доработать его; наибольшее распространение получил вариант LZW, написанный Т. Уэлчем, сотрудником фирмы Unisys, в 1983 г. Он отличался от своего прародителя более высоким

быстродействием. В 1985 г. Unisys запатентовала LZW. Этот алгоритм универсален и может использоваться для сжатия информации любого вида. Однако успешнее всего он справляется с графическими изображениями.

В 1987 г. компания CompuServe разработала на основе алгоритма LZW графический формат GIF (Graphic Interchange Format), позволивший эффективно сжимать изображения с глубиной цвета до 8 бит, а по тем временам 256 оттенков считалось огромным количеством. Он был разработан для передачи растровых изображений по сетям.

В 1989 г. CompuServe выпустила усовершенствованную версию формата, названную GIF89a. В нее были добавлены две функции, которые и по сей день обеспечивают формату популярность в Интернете (GIF позиционируется прежде всего как сетевой формат). Во-первых, был добавлен альфа-канал, где может храниться маска прозрачности, во-вторых, GIF стал анимированным, т. е. в один файл можно поместить несколько изображений, которые будут сменять друг друга через заданный интервал времени. Таким образом, GIF начал поддерживать прозрачность и анимацию. Файлы GIF содержат информацию в сжатом виде, что позволяет заметно уменьшить их размер, особенно если в них есть большие пространства, закрашенные одним цветом. Можно назначить один или несколько цветов прозрачными. Помимо этого GIF может хранить сразу несколько изображений, которые будут отображаться последовательно одно за другим. Объединив эти возможности, можно получить мини-мультфильмы, носящие название «анимированные GIF». Такие изображения можно встретить в оформлении интернет-сайтов, хотя в последнее время все большее количество дизайнеров предпочитают пользоваться технологией Flash.

Еще одна полезная особенность формата — чересстрочная развертка. Во время загрузки изображения сначала показываются первая строка, пятая, десятая и т. д., а затем — вторая, шестая, одиннадцатая. Таким образом создается эффект постепенного проявления картинки на экране. Это позволяет увидеть и оценить изображение еще до завершения загрузки, что особенно выгодно при работе с файлами внушительных размеров.

Область применения GIF известна всем — это изображения с резкими цветовыми переходами и бизнес-графика (логотипы, кнопки, элементы оформления и т. п.). А вот для тех картинок, где важно постепенное изменение оттенков (например, для фотографий), данный формат подходит меньше всего. Первая причина таких ограничений — небольшая по современным меркам максимальная глубина цвета изображения. Для фото 8 бит явно недостаточно. Вторая — не всегда корректное преобразование файлов с плавными цветовыми переливами в палитру 256 цветов и менее. Третья причина ограниченности сферы применения GIF заключается в особенностях метода сжатия информации. Данные об изображении записываются построчно. В итоге все операции происходят с массивом строк высотой в один

пиксел (каждая строка обрабатывается отдельно). Следствие этого — не только крайняя неэффективность сжатия фотографий и любых других изображений, содержащих мало однотонных областей, но и зависимость его результата от расположения объектов. Главный недостаток GIF заключается в том, что изображение может содержать 256 цветов. Это слишком мало для большинства современных задач.

Кстати, алгоритм LZW используется и в формате-TIFF. Однако он по качеству сжатия значительно уступает формату GIF, хотя и поддерживает большее количество цветов в палитре.

**Формат JPEG**

Форматы на основе LZW не справлялись с эффективной обработкой фотографий, и потому появилась идея сжатия с потерей качества. Суть его заключается в том, что часть малозаметных для глаза деталей изображения опускается, а восстановленный после сжатия цифровой массив не полностью соответствует оригиналу. Таким образом, можно добиться довольно большой степени сжатия данных — в 10—20 раз вместо двукратного, производимого без потерь.

В 1991 г. группа Joint Photographic Experts Group, опирающаяся на более чем полувековой опыт исследований в области человеческого зрения, представила первую спецификацию формата JPEG. Через три года она была признана индустриальным стандартом кодирования неподвижных изображений, зарегистрированным как ISO/IEC 10918-1. Впоследствии JPEG лег в основу стандарта сжатия видео MPEG.

JPEG (Joint Photographic Experts Group) на сегодняшний день является одним из наиболее популярных форматов. Всеобщего признания он достиг благодаря одноименному алгоритму сжатия, который показал отличные результаты в соотношении размер/качество.

JPEG сжимает файлы весьма оригинальным образом: он ищет не одинаковые пиксели, а вычисляет разницу между соседними квадратами размером 9x9 пикселей.

Информация, не заслуживающая внимания, отбрасывается, а ряд полученных значений усредняется. В результате получают файл в десятки или сотни раз меньшего размера, чем BMP. Естественно, чем выше уровень компрессии, тем ниже качество.

С помощью формата JPEG лучше всего хранить фотографии и другие похожие на них изображения. Скриншоты, схемы и чертежи лучше хранить в формате TIFF, поскольку при сохранении в JPEG неизбежно появятся помехи и «шумы».

Если обрабатывают файл в графическом редакторе, то сохранять его в JPEG нужно лишь после выполнения всех действий по редактированию. В противном случае с каждым сохранением его качество будет ухудшаться, что в конце концов приведет к «размазыванию» изображения.

Процесс обработки графической информации алгоритмом JPEG очень напоминает сжатие

звуковых данных в формате MP3:

1. Исходное изображение делится на блоки размером 16x16 пикселов. Дальнейшие операции выполняются над каждым из них по отдельности, что дает существенный выигрыш в скорости по сравнению с тем случаем, когда картинка обрабатывается как единый массив.

2. Переход к более подходящему для сжатия способу представления цветов. Привычная модель RGB переводится в YCbCr, где Y — сигнал яркости, а Cb и Cr — насыщенность синего и красного соответственно. Такой способ представления цветов будет предпочтительнее и с точки зрения восприятия изображения человеческим глазом. Как известно, зрительная информация воспринимается с помощью сенсоров двух типов: палочек и колбочек. Первые анализируют яркостную составляющую изображения, вторые — цвет. Палочек в 20 раз больше, чем колбочек, и, следовательно, глаз более восприимчив к изменению яркости, чем цвета.

Если учесть эту особенность человеческого зрения, то из матриц значений насыщенности синего и красного следует отбрасывать все четные строки и столбцы. Таким образом теряется 75% информации о распределении цветов. Матрица отчетов о яркости не изменяется, а делится на четыре части, образуя блоки 8x8.

3. Выполняется дискретное косинусное преобразование (Discrete Cosine Transform, DCT), предложенное В. Ченом в 1981 г. Оно сходно с преобразованием Фурье, только у DCT несколько ниже вероятность возникновения ошибки. Применение этого чисто математического приема объясняется тем, что в реальных изображениях соседние блоки достаточно похожи (коэффициент корреляции — 0,9—0,98). DCT преобразует информацию о цвете и яркости каждого пиксела в информацию о скорости изменения этих величин. Это преобразование является обратимым, и, значит, по новой матрице может быть восстановлена исходная с точностью до погрешности данного метода. DCT позволяет значительно сократить объем данных и размер получаемого файла.

В результате DCT графическое изображение описывается двухмерной функцией, показывающей скорости изменения яркости и цвета. Причем для большинства фотографий характерны плавные, мягкие переходы этих параметров на соседних участках. Как выявили исследования, для корректного восприятия таких изображений глазу важнее низкочастотные компоненты матрицы DCT (плавные переходы), а высокочастотные (резкая смена оттенков и яркости) уже не столь существенны. Поэтому последние кодируются с меньшей детальностью, а при превышении определенной пороговой величины, зависящей от выбранного качества сжатия, вообще принимаются равными нулю. Следовательно, при кодировании файлов с низким качеством резкие цветовые переходы смазываются, а изображение становится несколько мозаичным.

4. Кодирование полученных величин методом Хаффмана — последний этап. Оно

заключается в присваивании часто повторяющимся элементам обрабатываемой информации самых коротких кодовых последовательностей, а редко встречающимся — более длинных.

Вследствие этого размер получаемого файла несколько уменьшается.

Такой достаточно сложный алгоритм сжатия графической информации позволил формату JPEG обрабатывать фотографические изображения с большой эффективностью и неплохим качеством. Правда, область применения данного формата лишь фотографиями и ограничивается.

**Формат PNG**

PNG (Portable Network Graphics) обязан своим появлением на свет формату GIF, а точнее, его коммерческому статусу. Дело в том, что GIF основан на запатентованном алгоритме LZW, принадлежащем фирме Unisys, которая в первые годы существования GIF не предъявляла никаких претензий фирме CompuServe, разработавшей этот формат. Но стремительное развитие Интернета в 1993—1994 гг. внесло свои коррективы во взаимоотношения компаний. И из корыстных целей Unisys инициировала судебный процесс против CompuServe. Решение суда обязывало разработчиков ПО, в котором используется формат GIF, платить фирме Unisys лицензионные отчисления. Таким образом, GIF стал платным.

По традиции любому коммерческому программному продукту рано или поздно будет предложена бесплатная альтернатива. Примерами в данном случае могут служить операционные системы Windows и Linux или форматы сжатия звука MP3 и OGG, причем этот список можно продолжать довольно долго. GIF также не суждено было удержать статус безальтернативного формата. Днем рождения PNG можно считать 4 января 1995 г., когда Т. Боутелл предложил в ряде конференций Usenet создать свободный формат, который был бы не хуже GIF. И уже через три недели после публикации идеи были разработаны четыре версии нового формата. Вначале он имел название PBF (Portable Bitmap Format), а нынешнее имя получил 23 января 1995 г. Уже в декабре того же года спецификация PNG версии 0.92 была рассмотрена консорциумом W3C, а с выходом 1 октября 1996 г. версии 1.0 PNG был рекомендован в качестве полноправного сетевого формата. К моменту создания PNG глубина цвета 8 бит, предоставляемая GIF, перестала удовлетворять современным требованиям. Среди предложений довести глубину цвета до 24, 48 и даже до 64 бит был выбран первый вариант. Как показало время, данное решение оказалось рациональным и отвечающим современным запросам. В заголовке файла формата PNG хранится описание всей палитры. Подстроившись таким образом под реальное количество цветов, можно получить достаточно компактный файл на выходе. Формат PNG имеет массу достоинств, выгодно отличающих его от конкурентов. Самое главное из них — предварительная фильтрация обрабатываемых данных, поскольку большинство алгоритмов сжатия рассчитаны на одну вполне определенную модель информации. Например, формат JPEG «заточен» для изображений с плавными цветовыми переходами, а GIF — с большим количеством однотонных областей. И чем

больше структура картинки отличается от эталона, тем ниже эффективность сжатия. Порой изменение всего нескольких бит приводит к тому, что алгоритм начинает работать очень хорошо или, наоборот, очень плохо. Тем временем формат PNG стабильнее воспринимает такие трансформации, и высокий коэффициент сжатия достигается практически для любого рода изображений путем преобразования информации в вид, наиболее приемлемый для обработки.

Формат PNG очень удобен для создания элементов оформления вебстраниц благодаря наличию более совершенного альфа-канала, чем у GIF. Число его уровней доведено до 254 (в большинстве форматов альфа-канал ограничивается всего двумя слоями), а прозрачность каждого из них может варьировать от 0 до 100%. Таким образом, PNG предоставляет веб-дизайнерам очень мощное и удобное средство для построения сложных многослойных изображений. Единственный тип рисунков, где PNG не может потеснить GIF на вебстраницах, — небольшие кнопки и «смайлики» размером до 700—800 байт в формате GIF. За счет служебной информации и описания палитры такие файлы в формате PNG занимают на 10—30% больше места, да и анимированные картинки этот формат не поддерживает. Кстати, статичные «смайлики» чаще используются на страницах журналов, нежели во Всемирной паутине.

Будучи форматом следующего поколения PNG обладает еще несколькими приятными новшествами по сравнению со «старичками» JPEG и GIF. Во-первых, это несколько модифицированный вариант чересстрочной развертки. В отличие от GIF, где не до конца загруженное изображение отображается крупными строками, в формате PNG такой файл показывается большими точками, т. е. развертку следовало бы называть не чересстрочной, а «чересточечной», хотя первый вариант — уже устоявшийся (произошел от английского interlace). Во-вторых, нельзя не отметить встроенную в формат гамма-коррекцию. При переносе графического файла с одного компьютера на другой изображение может выглядеть светлее или темнее, чем в оригинале. Ситуация обостряется тогда, когда на компьютерах установлены различные ОС (MS Windows и Linux) или когда машины построены на базе разных платформ (например, PC и Macintosh). Соотношение между цифровой информацией и реально наблюдаемыми на мониторе цветами называется гаммой. Встроенная в формат PNG гамма-коррекция работает следующим образом: данные о настройках дисплея, видеоплаты и ПО (информация о гамме) записываются в файл, при переносе которого на другой компьютер вид PNG-изображения не изменится.

**Формат TIFF**

Формат TIFF (Tagged Image File Format) наиболее распространен среди тех пользователей, которые превыше всего ставят качество изображения. Именно TIFF используется в издательствах, поскольку при сохранении изображения в этом формате не происходит потерь качества. К тому же этот формат поддерживают практически все программы как PC, так и

Macintosh.

В принципе, TIFF — это наиболее универсальный формат, обеспечивающий среднюю степень сжатия файла. Храня работы в TIFF, можно быть уверенным, что при просмотре файла вы увидите именно то изображение, которое сохраняли. За качество придется расплачиваться достаточно большим объемом файлов. Наиболее актуально это для владельцев цифровых фотоаппаратов, которые сохраняют фотографии на компактные карты памяти. Однако если хотят получить фотографии, которые потом можно будет обработать в Photoshop, сохраняют работы в TIFF.

**Формат RAW**

Существует еще один формат, о котором хотелось бы упомянуть, — RAW. Его название созвучно английскому слову «raw» («сырой»). Действительно, при сохранении в RAW изображение записывается в самом что ни на есть сыром виде — как последовательность байтов, содержащих цветовую информацию о каждом пикселе. Цветовые значения описываются в шестнадцатеричном формате, где О = черный и 255 = белый. При таком способе записи объем файлов оказывается не просто большим, а огромным. Однако именно формат RAW получил популярность среди профессиональных фотографов.

Использование формата RAW в цифровой фотографии позволяет избежать искажений, вызываемых неверными настройками фотокамеры

Дело в том, что цветовой баланс изображения, сохраняемого на карту памяти, во многом зависит от настроек цифрового фотоаппарата, и в первую очередь от установки баланса белого. Если цветовой баланс задан неверно, «вытянуть» фотографию в графическом редакторе будет очень сложно, поскольку при коррекции одного цвета неизбежно «поплывут» другие. При использовании формата RAW цветовой баланс, установленный на фотоаппарате, не будет иметь никакого значения. Вы получите наиболее точную фотографию, которую можно легко обработать на PC.

**Сравнительная характеристика графических форматов**

На первый взгляд, у всех описанных форматов сходное предназначение: добиться уменьшения размера графического файла до разумных пределов, чтобы его было удобно хранить на жестком диске и передавать через сеть. Но алгоритмы сжатия у каждого из них настолько различаются, что и сферы использования GIF, JPEG и PNG оказались разными. И чтобы выяснить оптимальные области применения этих форматов, был проведен анализ с участием достаточно большого количества тестовых изображений.

**Фотографические изображения**. Как оказалось, найти где-либо картинку фотографического качества — задача не из легких даже для владельцев цифровой фотокамеры, поскольку она сохраняет снимки в формате JPEG, да и конвертировать JPEG в GIF или PNG было

бы не совсем корректно. Но удалось решить ее довольно элегантным способом: в сканер была навалена куча мусора и преобразована в цифровую форму. Получившийся «шедевр» размером 2530x3490 точек с глубиной цвета 24 бита, занимающий 25,2 Мбайт, сжимался в форматах JPEG и PNG (вследствие неспособности сохранять изображения с палитрой, включающей более 256 цветов, GIF задействован не был).

Как и следовало ожидать, в данном случае формат JPEG с подобранным экспериментальным путем качеством в 70% намного опередил конкурентов. При минимальной потере качества сжатие более чем в 25 раз достойно похвалы. Причем отличия между оригиналом и сжатым файлом можно заменить, только если сильно увеличить изображение. PNG справился с задачей намного хуже. Последним пришел к финишу универсальный алгоритм сжатия информации, реализованный в программе WinRAR 3.20, который был включен в тест из любопытства.

Наиболее интересные результаты дало сжатие изображения, переведенного в палитру 256 цветов (8,42 Мбайт в формате BMP), благодаря чему в состязании смог участвовать GIF. И на этот раз в лидеры вышел JPEG, но, вопреки предположениям, размер файла получился несколько большим, чем при сжатии той же картинки, но с глубиной цвета 24 бита. Вероятно, увеличение размера файла вызвано огрублением цветовых переходов при переводе в палитру 8 бит. Среди форматов GIF и PNG последний оказался явным фаворитом, однако сжатие изображения архиватором дало просто поразительный результат: RAR-алгоритм превзошел по степени сжатия форматы, специально предназначенные для обработки изображений. Помимо худшего показателя сжатия у GIF проявилась и еще одна отрицательная черта. Несмотря на то что изображение было заранее переведено в палитру 256 цветов, при конвертировании файла в этот формат произошла определенная потеря качества. И без того грубые цветовые переходы стали еще более выделяющимися, а само изображение несколько посветлело. Эксперименты над небольшим фрагментом того же графического файла привели к практически идентичным результатам. Все закономерности остались неизменными, а коэффициенты сжатия у различных форматов лишь немного отличались от предыдущего примера.

**Электрическая схема.** Это изображение с малым количеством цветов, содержащее преимущественно однотонные области и резкие цветовые переходы. Казалось бы, здесь формат GIF должен камня на камне не оставить от конкурентов. Но и тут ситуация сложилась не в пользу детища CompuServe. Явным аутсайдером стал JPEG. Даже при 100%-ном качестве (204 Кбайт) в глаза бросается некоторое осветление закрашенных областей (серого и желтого). При установке же 6%-го качества, когда размер рисунка сопоставим с объемами файлов в форматах PNG и GIF, изображение сильно «поплыло». А вот тот факт, что PNG-файл занимает на 20% меньше места, чем аналогичная картинка в формате GIF, заставляет задуматься об

эффективности последнего. И опять здесь удивил WinRAR, сжавший схему почти в 2 раза лучше все того же GIF.

**«Полосатые» рисунки.** Весьма показательный пример, выявляющий главный недостаток LZW-алгоритма, примененного в формате GIF. Рисунок, состоящий из горизонтальных полос, занимает в формате GIF почти в 4 раза меньше места, чем то же изображение, повернутое на 90°. Предварительная фильтрация информации, реализованная в PNG, сделала этот формат свободным от данного недостатка. Помимо большей степени сжатия по сравнению с GIF различие в размере «полосатых» файлов составило всего 10%. Но самыми компактными снова оказались BMP-файлы, упакованные архиватором. А разница в 2 (!) байта между изображениями с горизонтальными и вертикальными полосами просто поражает.

**Снимок с экрана**. В нем присутствовали достаточно сложные многоцветные элементы и все 256 цветов палитры. Результат: изображение в формате PNG на 40% меньше GIF-файла.

**«Смайлик».** Пожалуй, единственная на ближайшее время гарантия «неприкосновенности» формата GIF — огромная популярность различных графических «смайликов» на широких сетевых просторах, в частности на форумах и в чатах. Порой бывает гораздо легче передать яркую эмоцию с помощью маленькой картинки, нежели сделать это словом и даже целым предложением. Да и по компактности получаемого файла GIF в данном случае вне конкуренции.

**Выводы:** Результаты сравнительного анализа основных графических форматов оказались достаточно предсказуемыми. В области хранения фотоизображений явным лидером, если не сказать монополистом, стал формат JPEG. Он обеспечивает отличное качество картинки при малых размерах файла. Для других целей этот формат совершенно непригоден.

При всей своей популярности GIF, алгоритм сжатия этого формата абсолютно устарел. Более молодой PNG превосходит GIF и по качеству сжатия практически всех изображений. Убогая на нынешний день глубина цвета 8 бит не позволяет хранить в GIF фотографии (некорректная цветопередача — GIF «шалит» с палитрой, портя картинку). Формат PNG годится, например, для промежуточных версий подлежащих редактированию фотоизображений, когда BMP-файлы занимают слишком много места, а каждое последующее сохранение в JPEG приводит к потере качества.

Для нефотографических изображений GIF используется в 95% случаев (в оставшихся 5% применяется PNG). А ведь GIF — единственный платный графический формат, к тому же сильно уступающий по возможностям своему бесплатному конкуренту. Ничем, кроме силы привычки и лени веб-дизайнеров, данный факт объяснить нельзя. Между тем формат PNG наряду с лучшим алгоритмом сжатия может похвастаться многоуровневым альфа-каналом и гамма-коррекцией. Впрочем, популярность PNG растет с каждым днем. Очевидно, что GIF способен удерживаться

на плаву за счет одной своей особенности: в нем позволительно создавать анимированные изображения. Хотя кто знает, может быть, анимированная версия PNG появится уже завтра, а GIF навсегда уйдет в историю.

Очень показательны результаты обработки изображений архиватором WinRAR. Этот постоянно развивающийся, универсальный метод сжатия достаточно сильно оторвался от форматов сжатия графики без потерь. Значит, и графическим форматам есть куда развиваться. Не исключаю, что появится новый, более совершенный формат, основанный на лучшем методе сжатия и имеющий все необходимые функции. Но пока этого не произошло, можно смело говорить, что GIF по-прежнему нужен миру. А что будет дальше, покажет время.

**ХАРАКТЕРИСТИКИ ВИДЕОФАЙЛОВ И ВИДЕОПЛЕЕРОВ**

Операционная система Windows может работать с большим количеством видеофайлов различных форматов, что объясняется многообразием алгоритмов компрессии цифрового видео. Многие из них принадлежат разным производителям и основаны на разных принципах. Например, самый популярный на сегодня стандарт MPEG используется для записи DVD-фильмов и дисков с фильмами в формате DivX, а RealVideo используется для живой телевизионной трансляции в Интернете. Например, телекомпания CNN одной из первых стала вещать в Сети.

Большинство видеофайлов сжаты с использованием стандарта MPEG-4 (сжатие – устранение избыточности цифровой информации).

История стандарта MPEG-4 началась с работы группы экспертов (Motion Picture Experts Group), no аббревиатуре которой и сложились современные названия стандартов, а также расширения многих других видеофайлов. Файлы с расширением AVI, которые используют этот кодек, не имеют постоянных параметров, так как существуют в двух ипостасях: без компрессии и файлы, сжатые, например, кодеком DivX, которые также имеют расширение AVI. Кодек – это программа для кодирования – декодирования потока данных.

Несовершенный человеческий глаз не всегда может уловить визуальную разницу между файлами, сжатыми по тому или иному алгоритму. Она, несомненно, существует, и качество картинки ухудшается в зависимости от области применения кодека. Форматы, используемые в Интернете (наиболее распространенный — RealVideo), обладают небольшим размером файла и самым низким качеством. Поэтому, не особенно загружая канал связи, можно посмотреть последний выпуск теленовостей на сайте выбранной телекомпании.

Залог успешного просмотра видеофайлов на компьютере — это наличие современного программного видеоплеера. В Windows XP по умолчанию это Windows Media Player, причем в комплект его поставки уже входят несколько встроенных кодеков. Установка двух-трех плееров обеспечивает гарантированное воспроизведение видеофайлов различных типов, потому что

политика некоторых разработчиков делает их форматы в других программах недоступными.

Рассмотрим популярные форматы файлов и видеоплееры, которые они могут проигрывать. Каждый из них обеспечивает воспроизведение не только файлов в собственном формате, но и почти всех видеофайлов ближайших конкурентов. Особняком здесь стоит только Real Player, потому что данные в формате RealVideo и RealAudio (файлы с расширением RM, RA, RAM) может воспроизводить только этот плеер.

Формат компрессии DivX (технология компрессии от DivXNetworks): модификация MPEG-4, использует высокую степень компрессии с приемлемым качеством.

Формат компрессии RealVideo (собственный формат компании RealNetworks): RealVideo, использует очень высокую степень сжатия видео для прямой трансляции в Интернете.

Формат компрессии Windows Media (собственный формат файлов Microsoft): аналогичен MPEG-4, собственный формат фирмы Microsoft.

Программный видеоплеер Windows Media Player: типы воспроизводимых файлов WMV, ASF, AVI. MPEG, MPG, IVF, M1V, MP1, MP2, VOB и др.

Программный видеоплеер Real Player: типы воспроизводимых файлов RM, RA, RAM, MPG, MPEG, AVI, ASF, MID, MOV и др.

Программный видеоплеер Quick Time Player: типы воспроизводимых файлов QT, MOV, PNG, AVI, FIX, MPG, MPEG, MP1, MK и др.

В комплекте Windows имеется программный пакет для домашнего видеомонтажа Movie Maker. В нем реализованы все основные этапы работы с цифровым видео: захват, монтирование и сохранение (отправление в Интернет) видеофайлов. Существует достаточно много и других профессиональных программ. Их обычно начинают применять, когда компьютер превращается в домашнюю видеостудию: с приобретением ТВ-тюнера, цифровой видеокамеры, Firewire - оборудования или специальных плат видеозахвата.

Firewire – это стандарт обмена данными между периферийными устройствами.

Видеомонтаж – программная обработка видеоматериала.

Захват видеосигнала: перенос аналогового или цифрового сигнала на жесткий диск ПК.

**КОДИРОВАНИЕ ЗВУКА**

Для сохранения звука пользуются специальными программами — кодаками. Хотя качество результата у всех кодаков разное, принцип работы у них один.

Поклонники того или иного кодека не могут прийти к единому мнению, какой же из них лучше. Объясняется это просто: помимо таких объективных параметров, как размер файла и качество кодирования звука, существуют и субъективные факторы — восприятие каждого человека индивидуально.

Существует два основных стандарта, которые используют любители музыки во всем мире:

MP3 и WMA. Если стандарт WMA разрабатывается исключительно фирмой Microsoft, то кодек для сжатия цифрового звука в стандарте MP3 может создать любой программист. В результате это привело к появлению большого количества алгоритмов кодирования. В условиях конкурентной борьбы за качество MP3-файлов на первое место вышел проект нескольких программистов — Lame.

**Кодек Lame**

Кодек Lame используют с 1998 году, когда с развитием Интернета между пользователями начался активный обмен файлами с данными и возникла проблема передачи звука через Сеть.

Многие фирмы стали разрабатывать программы для кодирования аудиосигнала с наименьшими потерями в качестве. Группа специалистов (MPEG), входящая в состав Международной организации стандартов (ISO), еще в конце 80-х годов разработала стандарт MPEG, на основе которого был создан современный стандарт сжатия звука MPEG 1.0 Audio Layer III — впоследствии он стал известен как MP3. Так как изначально утилиты для создания MP3-файлов распространялись за деньги, программисты стали искать способ бесплатно использовать алгоритмы работы кодека. Так среди многих других проектов появился кодек Lame, постепенно получивший огромную популярность. Основной особенностью Lame стало то, что разработчики сделали акцент на улучшении алгоритма специальной психоакустической модели. Принцип ее действия заключается в том, что из звукового файла удаляются те частоты, которые человеческое ухо воспринимать не в состоянии.

При кодировании цифрового звука (файлы с расширением WAV) основным параметром, влияющим на качество результата, является величина **битрейта**. Если использовать максимальное значение этого параметра, то получают звук, наиболее соответствующий оригиналу. Примерно до 2000 года широко использовалось значение битрейта 128 Кбит/с, а затем, с возросшей пропускной способностью современных каналов связи, наиболее распространенным стал битрейт 192 Кбит/с.

Еще один параметр, который может существенно улучшить качество звука, — функция VBR, позволяющая кодировать информацию с переменным битрейтом в зависимости от характера сигнала. Например, если в музыке присутствуют различные высокочастотные звуки (качественно сжать которые труднее всего), кодек принимает решение использовать для них самый высокий битрейт 320 Кбит/с.

**Кодек WMA фирмы Microsoft**

Кодек WMA был разработан фирмой Microsoft как стандарт хранения сжатой аудиоинформации в операционной системе Windows. Microsoft не стала пользоваться разработками организации MPEG, поэтому стандарт WMA является закрытым для использования другими разработчиками.

Microsoft стремится максимально использовать коммерческий потенциал WMA и поэтому защищает музыку в этом формате от нелегального копирования: WMA-файлы нельзя переконвертировать в файлы с расширением WAV.

Тем не менее популярность WMA растет. Это связано с тем, что при том же качестве звука, что и у стандарта MP3 (по заявлению самой Microsoft), этот кодек позволяет получить меньший размер файла. Такой результат достигается за счет использования более низкого значения параметров битрейта, чем у MP3-файлов.

В последней, девятой, версии кодека создатели значительно переработали психоакустическую модель кодирования аудиоинформации, однако на максимальных значениях битрейта кодек от Microsoft до сих пор не может сравниться по качеству с кодеком Lame.

**Выбор кодека для звука**

При выборе кодека нужно руководствоавться двумя соображениями: если нужно получить файлы как можно меньшего размера — используют кодек от Microsoft, а если цель — получить максимальное качество звука, невзирая на размер файлов, выбирают Lame.

**Другие кодеки для звука**

Программистам и компаниям хочется сделать кодек, который станет всеобщим стандартом. Однако до сих пор ни один разработанный формат, отличный от WMA и MP3, не смог получить широкого распространения. Это произошло по двум причинам: либо кодек не очень хорошо сжимал аудиофайлы, либо программное обеспечение для его использования было полностью платным. Список кодеков, пытавшихся заменить MP3, достаточно велик:

**AAC** — улучшенный вариант стандарта MPEG по соотношению «размер файла/качество звука». Не получил особого распространения, потому что при кодировании аудиофайлов сильно загружает ресурсы компьютера.

**Liquid Audio** — коммерческий кодек, в котором содержание файла шифруется для пресечения нелегального копирования. Несмотря на то, что этот кодек по качеству результата превосходит MP3, отсутствие бесплатных программ для кодирования файлов помешало распространению этого формата.

**MP3pro** — кодек, разработанный фирмой Tomson Multimedia. Был призван улучшить качество формата MP3 на низких битрейтах. Особой популярности не получил из-за достаточно высокой цены и узкой области применения.

**OGG Vorbis** — этот сравнительно недавно появившийся кодек стоит особняком среди всех перечисленных форматов. Получаемые с его помощью файлы как по качеству, так и по размеру значительно превосходят стандарт MP3, что дает разработчикам право надеяться на то, что в будущем файлы с расширением OGG станут единственным средством хранения и распространения музыки через Интернет. Однако ни один современный портативный MP3-плеер

не умеет воспроизводить файлы в этом формате — так же, как и большинство программных мультимедийных проигрывателей. **TwinVQ** — стандарт, разработанный фирмой Yamaha. Предназначался для кодирования звука с очень низким битрейтом и более высоким качеством, чем MP3. Не пользовался популярностью из-за больших требований к системным ресурсам как при воспроизведении музыки в этом формате, так и при кодировании файлов.

**ФОРМАТ PDF**

Часто документ Word «разваливается» на компьютере с другой версией операционной системы или MS Office, так что приходится приводить его в изначальный вид, восстанавливая потерянное форматирование. Но этого можно избежать.

**Назначение формата PDF**

Portable Document Format или просто PDF, был создан специально для ликвидации проблем с отображением информации в файлах. В чем же его преимущество? Во-первых, документ, сохраненный в формате PDF, будет одинаково выглядеть в любой операционной системе: в Windows XP, в Windows 95, и в Linux. Во-вторых, PDF использует качественные алгоритмы сжатия: если объем файла Word, содержащего пару картинок, вряд ли получится меньше мегабайта, то точно такой же PDF вполне уместится в 300-400 Кбайт. В-третьих, PDF умеет встраивать в себя все используемые в документе шрифты — будь то рукописные, готические или славянская вязь, так что можно забыть о тех случаях, когда какого-либо шрифта не оказывается на компьютере. В-четвертых, в формат PDF можно преобразовать любой электронный документ. А для прочтения и печати PDF понадобится бесплатная программа Acrobat Reader.

**Создание PDF-файла**

Сделать PDF просто. Первое, что необходимо, — программа Adobe Acrobat Professional, желательно последней версии 6.0. На практике все происходит очень просто, а во всех приложениях Microsoft Office и продуктах Adobe это можно сделать одним нажатием кнопки.

После инсталляции пакета Acrobat Professional в этих программах появится новая Панель инструментов, изначально она содержит три кнопки.

Открывают файл, который нужно преобразовать в PDF-формат, нажимают кнопку Convert to Adobe PDF на вышеуказанной Панели инструментов. В открывшемся окне указывают желаемое Имя создаваемого в формате PDF файла и Путь (месторасположение), где он будет находиться. Нажимают ОК, и через некоторое время новый файл будет готов и автоматически откроется для просмотра. Не сложнее этот процесс и во многих других программах. После установки Acrobat Professional в системе появляется новый виртуальный принтер под названием Adobe PDF —можно увидеть его ярлычк, открыв папку Принтеры через Панель управления. Он называется виртуальным, потому что реально не существует и используется именно для создания

PDF-файлов. Поэтому печатают на нем тоже виртуально, но результат при этом будет вполне реальным. Открывают программу, из документа которой нужно получить PDF, и выбирают опцию Файл -> Печать. Далее открывают список доступных принтеров, выбирают Adobe PDF и нажмают ОК. После этого задают имя нового файла и его месторасположение, и через несколько секунд он будет готов.

**Дополнительные настройки при создании PDF-файла**

Кликают по значку принтера Adobe PDF правой кнопкой мыши и выбирают Настройку печати. Все самое полезное размещено на вкладке Adobe PDF Settings.Можно выбрать шаблон, по которому будут создаваться файлы: High Quality (для создания PDF с высоким разрешением и повышенным качеством) или Smallest file size (для уменьшения объема файла). А можно создать собственные шаблоны, нажав на кнопку Edit.

Здесь же можно задать настройки безопасности. Можно установить отсутствие защиты (None) или предписать программе спрашивать об опциях безопасности при каждом создании PDF (Reconfirm security for each job) — в этом случае необходимо нажать расположенную рядом кнопку Edit и задать пароль, которым будет блокироваться доступ к новым файлам. Там же скрываются такие полезные функции, как парольный запрет некоторых возможностей работы с файлом: можно разрешить открытие файла, но запретить его печать или сохранение в др

## 3. Эффективное кодирование

При кодировании каждая буква исходного алфавита представляется различными последовательностями, состоящими из кодовых букв (цифр).

Если исходный алфавит содержит m букв, то для построения равномерного кода с использованием k кодовых букв необходимо удовлетворить соотношение $m \le k_q$ , где q - количество элементов в кодовой последовательности.

Поэтому

$$q \ge \frac{\log m}{\log k} = \log_k m$$

Для построения равномерного кода достаточно пронумеровать буквы исходного алфавита и записать их коды как q - разрядные числа в k-ичной системе счисления. Например, при двоичном кодировании 32 букв русского алфавита используется $q = \log_2 32 = 5$ разрядов, на чем и основывается телетайпный код. Кроме двоичных кодов, наибольшее распространение получили восьмеричные коды. Пусть, например, необходимо закодировать алфавит, состоящий из 64 букв. Для этого потребуется $q = \log_2 64 = 6$ двоичных разрядов или $q = \log_8 64 = 2$ восьмеричных разрядов. При этом буква с номером 13 при двоичном кодировании получает код 001101, а при восьмеричном кодировании 15. Общепризнанным в настоящее время является позиционный

принцип образования системы счисления. Значение каждого символа (цифры) зависит от его положения - позиции в ряду символов, представляющих число.

Единица каждого следующего разряда больше единицы предыдущего разряда в m раз, где m - основание системы счисления. Полное число получают, суммируя значения по разрядам:

$$Q = \sum_{i=1}^{l} a_i m^{i-1} = a_l m^{l-1} + a_{l-1} m^{l-2} + \ldots + a_2 m^1 + a_1 m^0,$$

где i - номер разряда данного числа; l - количество рядов; $a_i$ - множитель, принимающий любые целочисленные значения в пределах от 0 до m-1 и показывающий, сколько единиц i - ого ряда содержится в числе.

Часто используются двоично-десятичные коды, в которых цифры десятичного номера буквы представляются двоичными кодами. Так, например, для рассматриваемого примера буква с номером 13 кодируется как 0001 0011. Ясно, что при различной вероятности появления букв исходного алфавита равномерный код является избыточным, т.к. его энтропия (полученная при условии, что все буквы его алфавита равновероятны): $\log_k m = H_0$

всегда больше энтропии H = log m данного алфавита (полученной с учетом неравномерности появления различных букв алфавита, т.е. информационные возможности данного кода используются не полностью).

Например, для телетайпного кода $H0 = \log_k m = \log_2 32 = 5$ бит, а с учетом неравномерности появления различных букв исходного алфавита $H \approx 4{,}35$ бит. Устранение избыточности достигается применением неравномерных кодов, в которых буквы, имеющие наибольшую вероятность, кодируются наиболее короткими кодовыми последовательностями, а более длинные комбинации присваиваются редким буквам. Если i-я буква, вероятность которой $P_i$, получает кодовую комбинацию длины $q_i$, то средняя длина комбинации

$$q_{cp} = \sum_{i=1}^{m} p_i q_i$$

Считая кодовые буквы равномерными, определяем наибольшую энтропию закодированного алфавита как $q_{cp} \log m$, которая не может быть меньше энтропии исходного алфавита H, т.е. $q_{cp} \log m \geq H$.

Отсюда имеем

$$q_{cp} \geq \frac{H}{\log m}$$

При двоичном кодировании (m=2) приходим к соотношению $q_{cp} \geq H$, или

$$\sum_{i=1}^{m} P_i q_i \geq -\sum_{i=1}^{m} P_i \log P_i$$

Чем ближе значение $q_{cp}$ к энтропии H, тем более эффективно кодирование. В идеальном случае, когда $q_{cp} \approx H$, код называют ***эффективным***.

Эффективное кодирование устраняет избыточность, приводит к сокращению длины сообщений, а значит, позволяет уменьшить время передачи или объем памяти, необходимой для их хранения.

При построении неравномерных кодов необходимо обеспечить возможность их однозначной расшифровки. В равномерных кодах такая проблема не возникает, т.к. при расшифровке достаточно кодовую последовательность разделить на группы, каждая из которых состоит из q элементов. В неравномерных кодах можно использовать разделительный символ между буквами алфавита (так поступают, например, при передаче сообщений с помощью азбуки Морзе).

Если же отказаться от разделительных символов, то следует запретить такие кодовые комбинации, начальные части которых уже использованы в качестве самостоятельной комбинации. Например, если 101 означает код какой-то буквы, то нельзя использовать комбинации 1, 10 или 10101.

**Практические методы оптимального кодирования просты и основаны на очевидных соображениях ( *метод Шеннона – Фано )***.

Прежде всего, буквы (или любые сообщения, подлежащие кодированию) исходного алфавита записывают в порядке убывающей вероятности. Упорядоченное таким образом множество букв разбивают так, чтобы суммарные вероятности этих подмножеств были примерно равны. Всем знакам (буквам) верхней половины в качестве первого символа присваивают кодовый элемент 1, а всем нижним 0. Затем каждое подмножество снова разбивается на два подмножества с соблюдением того же условия равенства вероятностей и с тем же условием присваивания кодовых элементов в качестве второго символа. Такое разбиение продолжается до тех пор, пока в подмножестве не окажется только по одной букве кодируемого алфавита. При каждом разбиении буквам верхнего подмножества присваивается кодовый элемент 1, а буквам нижнего подмножества - 0.

Пример16: Провести эффективное кодирование ансамбля из восьми знаков:

Таблица 3.1.

| Буква (знак) $x_i$ | Вероят-ность Pi | Кодовые последовательности | | | | Длина $q_i$ | $p_i q_i$ | $-p_i \log p_i$ |
|---|---|---|---|---|---|---|---|---|
| | | Номер разбиения | | | | | | |
| | | 1 | 2 | 3 | 4 | | | |
| x1 | 0,25 | 1 | 1 | | | 2 | 0,5 | 0,50 |
| x2 | 0,25 | 1 | 0 | | | 2 | 0,5 | 0,50 |
| x3 | 0,15 | 0 | 1 | 1 | | 3 | 0,45 | 0,41 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| x4 | 0,15 | 0 | 1 | 0 | | 3 | 0,45 | 0,41 |
| x5 | 0,05 | 0 | 0 | 1 | 1 | 4 | 0,2 | 0,22 |
| x6 | 0,05 | 0 | 0 | 1 | 0 | 4 | 0,2 | 0,22 |
| x7 | 0,05 | 0 | 0 | 0 | 1 | 4 | 0,2 | 0,22 |
| x8 | 0,05 | 0 | 0 | 0 | 0 | 4 | 0,2 | 0,22 |

$$q_{cp} = \sum_{i=1}^{m} p_i q_i = 2,7$$

$$H = -\sum_{i=1}^{m} P_i \log P_i = 2,7$$

Как видно, $q_{ср} = H$, следовательно, полученный код является оптимальным.

Пример17: Построить код Шеннона - Фано, если известны вероятности: $P(x_1) = 0,5$; $P(x_2) = 0,25$; $P(x_3) = 0,125$; $P(x_4) = 0,125$

Пример18: Провести эффективное кодирование ансамбля из восьми знаков (m=8), используя метод Шеннона - Фано.

Решение: При обычном (не учитывающем статистических характеристик) двоичном кодировании с использованием k=2 знаков при построении равномерного кода количество элементов в кодовой последовательности будет $q \geq \log_k m = \log_2 8 = 3$, т.е. для представления каждого знака использованного алфавита потребуется три двоичных символа.

Метод Шеннона - Фано позволяет построить кодовые комбинации, в которых знаки исходного ансамбля, имеющие наибольшую вероятность, кодируются наиболее короткими кодовыми последовательностями. Таким образом, устраняется избыточность обычного двоичного кодирования, информационные возможности которого используются не полностью.

Таблица 3.2.

| Знаки (буквы) $x_i$ | Вероятность Pi | Кодовые комбинации | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | номер разбиения | | | | | | |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| x1 | 1/2 | 1 | | | | | | |
| x2 | 1/4 | 0 | 1 | | | | | |
| x3 | 1/8 | 0 | 0 | 1 | | | | |
| x4 | 1/16 | 0 | 0 | 0 | 1 | | | |
| x5 | 1/32 | 0 | 0 | 0 | 0 | 1 | | |
| x6 | 1/64 | 0 | 0 | 0 | 0 | 0 | 1 | |
| x7 | 1/128 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| x8 | 1/128 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Так как вероятности знаков представляют собой отрицательные целочисленные степени двойки, то избыточность при кодировании устранена полностью.

Среднее число символов на знак в этом случае точно равно энтропии. В общем случае для алфавита из восьми знаков среднее число символов на знак будет меньше трех, но больше энтропии алфавита. Вычислим энтропию алфавита:

$$H = -\sum_{i=1}^{m=8} P(x_i)\log P(x_i) = 1\frac{63}{64}$$

Вычислим среднее число символов на знак:

$$q_{cp} = \sum_{i=1}^{m=8} P(x_i)q(x_i) = 1\frac{63}{64},$$

где $q(x_i)$ - число символов в кодовой комбинации, соответствующей знаку $x_i$.

Пример19: Определить среднюю длину кодовой комбинации при эффективном кодировании по методу Шеннона - Фано ансамбля - из восьми знаков и энтропию алфавита.

Таблица 3.3.

| Знаки (буквы) $x_i$ | Вероятность Pi | Кодовые комбинации | | | | |
|---|---|---|---|---|---|---|
| | | номер разбиения | | | | |
| | | 1 | 2 | 3 | 4 | 5 |
| x1 | 0,22 | 1 | 1 | | | |
| x2 | 0,20 | 1 | 0 | 1 | | |
| x3 | 0,16 | 1 | 0 | 0 | | |
| x4 | 0,16 | 0 | 1 | | | |
| x5 | 0,10 | 0 | 0 | 1 | | |
| x6 | 0,10 | 0 | 0 | 0 | 1 | |
| x7 | 0,04 | 0 | 0 | 0 | 0 | 1 |
| x8 | 0,02 | 0 | 0 | 0 | 0 | 0 |

Решение: 1. Средняя длина кодовых комбинаций

$$q_{cp} = \sum_{i=1}^{m=8} P_i q_i = 2{,}84$$

2. Энтропия алфавита

$$H = -\sum_{i=1}^{m=8} P_i \log P_i = 2{,}76$$

При кодировании по методу Шеннона - Фано некоторая избыточность в последовательностях символов, как правило, остается ($q_{cp} > H$).

**Эту избыточность можно устранить, если перейти к кодированию достаточно большими блоками.**

Пример20: Рассмотрим процедуру эффективного кодирования по методике Шеннона - Фано сообщений, образованных с помощью алфавита, состоящего всего из двух знаков $x_1$ и $x_2$ с вероятностями появления соответственно $P(x_1) = 0,9$; $P(x_2) = 0,1$.

Так как вероятности не равны, то последовательность из таких букв будет обладать избыточностью. Однако, при побуквенном кодировании мы никакого эффекта не получим. Действительно, на передачу каждой буквы требуется символ либо 1, либо 0, в то время как энтропия равна

$$H = -\sum_{i=1}^{m=2} P_i \log P_i = 0,47,$$

т.е. оказывается $q_{cp} = \sum_{i=1}^{m} P_i q_i = 1 \rangle H = 0,47.$

При кодировании блоков, содержащих по две буквы, получим коды:

Таблица 3.4.

| Блоки | Вероятности | Кодовые комбинации | | |
|---|---|---|---|---|
| | | номер разбиения | | |
| | | 1 | 2 | 3 |
| x1x1 | 0,81 | 1 | | |
| x1x2 | 0,09 | 0 | 1 | |
| x2x1 | 0,09 | 0 | 0 | 1 |
| x2x2 | 0,01 | 0 | 0 | 0 |

Так как знаки статистически не связаны, вероятности блоков определяют как произведение вероятностей составляющих знаков.

Среднее число символов на блок

$$q_{cp} = \sum_{i=1}^{m=4} P_i q_i = 1,29,$$

а на букву 1,29/2 = 0,645, т.е. приблизилось к H = 0,47 и таким образом удалось повысить эффективность кодирования.

Кодирование блоков, содержащих по три знака, дает еще больший эффект:

Таблица 3.5.

| Блоки | Вероятность Pi | кодовые комбинации | | | | |
|---|---|---|---|---|---|---|
| | | номер разбиения | | | | |
| | | 1 | 2 | 3 | 4 | 5 |
| x1x1x1 | 0,729 | 1 | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| x2x1x1 | 0,081 | 0 | 1 | 1 | | |
| x1x2x1 | 0,081 | 0 | 1 | 0 | | |
| x1x1x2 | 0,081 | 0 | 0 | 1 | | |
| x2x2x1 | 0,009 | 0 | 0 | 1 | 1 | |
| x2x1x2 | 0,009 | 0 | 0 | 0 | 1 | 0 |
| x1x2x2 | 0,009 | 0 | 0 | 0 | 0 | 1 |
| x2x2x2 | 0,001 | 0 | 0 | 0 | 0 | 0 |

Среднее число символов на блок равно 1,59, а на знак - 0,53, что всего на 12% больше энтропии.

Следует подчеркнуть, что увеличение эффективности кодирования при укрупнении блоков не связано с учетом все более далеких статистических связей, т.к. нами рассматривались алфавиты с независимыми знаками.

**Повышение эффективности определяется лишь тем, что набор вероятностей получившихся при укрупнении блоков можно делить на более близкие по суммарным вероятностям подгруппы.**

Рассмотренная методика **Шеннона - Фано** не всегда приводит к однозначному построению кода, т.к. при разбиении на подгруппы можно сделать большей по вероятности как верхнюю, так и нижнюю подгруппы:

Таблица 3.6.

| Знаки (буквы) $x_i$ | Вероятность Pi | 1-е кодовые комбинации | | | | | 2-е кодовые комбинации | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | номер разбиения | | | | | номер разбиения | | | | |
| | | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| x1 | 0,22 | 1 | 1 | | | | 1 | 1 | | | |
| x2 | 0,20 | 1 | 0 | 1 | | | 1 | 0 | | | |
| x3 | 0,16 | 1 | 0 | 0 | | | 0 | 1 | 1 | | |
| x4 | 0,16 | 0 | 1 | | | | 0 | 1 | 0 | | |
| x5 | 0,10 | 0 | 0 | 1 | | | 0 | 0 | 1 | | |
| x6 | 0,10 | 0 | 0 | 0 | 1 | | 0 | 0 | 0 | 1 | |
| x7 | 0,04 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| x8 | 0,02 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

От указанного недостатка свободен ***метод Хаффмона***.

Эта методика гарантирует однозначное построение кода с наименьшим для данного распределения вероятностей средним числом символов на букву.

Для двоичного кода метод **Хаффмона** сводится к следующему. Буквы алфавита сообщений выписывают в основной столбец таблицы в порядке убывания вероятностей. Две последние буквы объединяют в одну вспомогательную букву, которой приписывают суммарную вероятность. Вероятности букв, не участвующих в объединении и полученная суммарная вероятность снова располагаются в порядке убывания вероятностей в дополнительном столбце, а две последние объединяются.

Процесс продолжается до тех пор, пока не получат единственную вспомогательную букву с вероятностью, равной единице.

Используя метод ***Хаффмона,*** осуществим эффективное кодирование ансамбля из восьми знаков:

Таблица 3.7.

| Знаки | Вероятности | Вспомогательные столбцы | | | | | | | Новая комбинация |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| x1 | 0,22 | 0,22 | 0,22 | 0,26 | 0,32 | 0,42 | 0,58 | 1 | 01 |
| x2 | 0,20 | 0,20 | 0,20 | 0,22 | 0,26 | 0,32 | 0,42 | | 00 |
| x3 | 0,16 | 0,16 | 0,16 | 0,20 | 0,22 | 0,26 | | | 111 |
| x4 | 0,16 | 0,16 | 0,16 | 0,16 | 0,20 | | | | 110 |
| x5 | 0,10 | 0,10 | 0,10 | 0,16 | | | | | 100 |
| x6 | 0,10 | 0,10 | | | | | | | 1011 |
| x7 | 0,04 | 0,06 | | | | | | | 10101 |
| x8 | 0,02 | | | | | | | | 10100 |

Для наглядности построим *кодовое дерево*. Из точки, соответствующей вероятности 1, направляем две ветви, причем ветви с большей вероятностью присваиваем символ 1, а с меньшей 0.

Такое последовательное ветвление продолжаем до тех пор, пока не дойдем до вероятности каждой буквы.

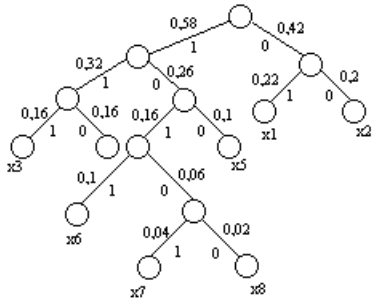



Рис. 3.1.

Двигаясь по кодовому дереву сверху вниз, можно записать для каждой буквы соответствующую ей кодовую комбинацию.

Пример21: Используя метод ***Шеннона - Фано*** и метод ***Хаффмона*** осуществить эффективное кодирование алфавита русского языка, характеризующегося ансамблем, представленным в примере 4.

## 4. Кодирование информации для канала с помехами

*Ошибка в кодовой комбинации появляется при ее передаче по каналу связи вследствие замены одних элементов другими под воздействием помех.* Например, 2-кратная ошибка возникает при замене (искажении) двух элементов. Например, если кодовая комбинация 0110111 принята как 0100110, то имеет место двукратная ошибка.

*Теория помехоустойчивого кодирования базируется на результатах исследований, проведенных* **Шенноном** *и сформулированных в виде теоремы:*

1. При любой производительности источника сообщений, меньшей, чем пропускная способность канала, существует такой способ кодирования, который позволяет обеспечить передачу всей информации, создаваемой источником сообщений, со сколь угодно малой вероятностью ошибки.

2. Не существует способа кодирования, позволяющего вести передачу информации со сколь угодно малой вероятностью ошибки, если производительность источника сообщений больше пропускной способности канала.

Из теоремы следует, что помехи в канале не накладывают ограничений на точность передачи. Ограничение накладывается только на скорость передачи, при которой может быть достигнута сколь угодно высокая точность передачи.

Теорема не затрагивает вопроса о путях построения кодов, обеспечивающих идеальную передачу информации, но, обосновав принципиальную возможность такого кодирования, позволяет вести разработку конкретных кодов.

При любой конечной скорости передачи информации вплоть до пропускной способности канала, сколь угодно малая вероятность ошибки достигается лишь при безграничном увеличении длительности кодируемых последовательностей знаков. Таким образом, *безошибочная передача при наличии помех возможна лишь теоретически.*

Обеспечение передачи информации с весьма малой вероятностью ошибки и достаточно высокой эффективностью возможно при кодировании чрезвычайно длинными последовательностями знаков.

На практике точность передачи информации и эффективность каналов связи ограничивается двумя факторами:

1) размером и стоимостью аппаратуры кодирования/декодирования;
2) временем задержки передаваемого сообщения.

### 4.1 Разновидности помехоустойчивых кодов

*Коды, которые обеспечивают возможность обнаружения и исправления ошибки, называют* ***помехоустойчивыми****.*

Эти коды используют для:

1) исправления ошибок – ***корректирующие коды***;
2) обнаружения ошибок.

*Корректирующие коды основаны на введении избыточности.*

У подавляющего большинства помехоустойчивых кодов помехоустойчивость обеспечивается их алгебраической структурой. Поэтому их называют ***алгебраическими кодами***.

***Алгебраические коды*** подразделяются на два класса:

1) блоковые;
2) непрерывные.

В случае блоковых кодов процедура кодирования заключается в сопоставлении каждой букве сообщения (или последовательности из k символов, соответствующей этой букве) блока из n символов. В операциях по преобразованию принимают участие только указанные k символов, и выходная последовательность не зависит от других символов в передаваемом сообщении.

Блоковый код называют ***равномерным***, если n остается постоянным для всех букв сообщения.

Различают *разделимые и неразделимые блоковые коды*. При кодировании *разделимыми кодами* выходные последовательности состоят из символов, роль которых может быть отчетливо разграничена. Это информационные символы, совпадающие с символами последовательности, поступающей на вход кодера канала, и избыточные (проверочные) символы, вводимые в исходную последовательность кодером канала и служащие для обнаружения и исправления ошибок.

При кодировании *неразделимыми кодами* разделить символы входной последовательности на информационные и проверочные невозможно.

***Непрерывными*** (древовидными) называют такие коды, в которых введение избыточных символов в кодируемую последовательность информационных символов осуществляется непрерывно, без разделения ее на независимые блоки. Непрерывные коды также могут быть разделимыми и неразделимыми.

### 4.2 Общие принципы использования избыточности

*Способность кода обнаруживать и исправлять ошибки обусловлена наличием в нем избыточных символов.*

На вход кодирующего устройства поступает последовательность из k информационных двоичных символов. На выходе ей соответствует последовательность из n двоичных символов, причем n>k.

Всего может быть $2^k$ различных входных и $2^n$ различных выходных последовательностей.

Из общего числа $2^n$ выходных последовательностей только $2^k$ последовательностей соответствуют входным. Их называют ***разрешенными кодовыми комбинациями***.

Остальные $2^n$-$2^k$ возможных выходных последовательностей для передачи не используются. Их называют **запрещенными кодовыми комбинациями**.

Искажения информации в процессе передачи сводятся к тому, что некоторые из передаваемых символов заменяются другими – неверными.

Так как каждая из $2^k$ разрешенных комбинаций в результате действия помех может трансформироваться в любую другую, то всегда имеется $2^k*2^n$ возможных случаев передачи. В это число входят:

1) $2^k$ случаев безошибочной передачи;

2) $2^k(2^k -1)$ случаев перехода в другие разрешенные комбинации, что соответствует необнаруженным ошибкам;

3) $2^k(2^n - 2^k)$ случаев перехода в неразрешенные комбинации, которые могут быть обнаружены.

Следовательно, часть обнаруживаемых ошибочных кодовых комбинаций от общего числа возможных случаев передачи составляет

$$\frac{2^k(2^n-2^k)}{2^k*2^n}=1-\frac{2^k}{2^n}.$$

Пример22: Определить обнаруживающую способность кода, каждая комбинация которого содержит всего один избыточный символ (n=k+1).

Решение : 1. Общее число выходных последовательностей составляет $2^{k+1}$, т.е. вдвое больше общего числа кодируемых входных последовательностей.

2. За подмножество разрешенных кодовых комбинаций можно принять, например, подмножество $2^k$ комбинаций, содержащих четное число единиц (или нулей).

3. При кодировании к каждой последовательности из k информационных символов добавляют один символ (0 или 1), такой, чтобы число единиц в кодовой комбинации было четным. Исполнение любого нечетного числа символов переводит разрешенную кодовую комбинацию в подмножество запрещенных комбинаций, что обнаруживается на приемной стороне по нечетности числа единиц. Часть опознанных ошибок составляет

$$1-2^k/2^{k+1}=1/2.$$

Любой метод декодирования можно рассматривать как правило разбиения всего множества запрещенных кодовых комбинаций на $2^k$ пересекающихся подмножеств $M_i$, каждая из которых ставится в соответствие одной из разрешенных комбинаций. При получении запрещенной комбинации, принадлежащей подмножеству $M_i$, принимают решение, что передавалась запрещенная комбинация $A_i$. Ошибка будет исправлена в тех случаях, когда полученная комбинация действительно образовалась из $A_i$, т.е. $2^n$-$2^k$ случаях.

Всего случаев перехода в неразрешенные комбинации $2^k(2^n - 2^k)$. Таким образом, *при наличии избыточности любой код способен исправлять ошибки.*

Отношение числа исправляемых кодом ошибочных кодовых комбинаций к числу обнаруживаемых ошибочных комбинаций равно

$$\frac{2^n - 2^k}{2^k(2^n - 2^k)} = \frac{1}{2^k}.$$

Способ разбиения на подмножества зависит от того, какие ошибки должны направляться конкретным кодом.

Большинство разработанных кодов предназначено для корректирования взаимно независимых ошибок определенной кратности и пачек (пакетов) ошибок.

***Взаимно независимыми ошибками*** называют такие искажения в передаваемой последовательности символов, при которых вероятность появления любой комбинации искаженных символов зависит только от числа искаженных символов r и вероятности искажения обычного символа p.

При взаимно независимых ошибках вероятность искажения любых r символов в n-разрядной кодовой комбинации:

$$p_r = c_n^r p^r (1-p)^{n-r},$$

где $p$ – вероятность искажения одного символа; $r$ – число искаженных символов; $n$ – число двоичных символов на входе кодирующего устройства; $c_n^r$ – число ошибок порядка r.

Если учесть, что p<<1, то в этом случае наиболее вероятны ошибки низшей кратности. Их следует обнаруживать и исправлять в первую очередь.

### 4.3 Связь корректирующей способности кода с кодовым расстоянием

При взаимно независимых ошибках наиболее вероятен переход в кодовую комбинацию, отличающуюся от данной в наименьшем числе символов.

Степень отличия любых двух кодовых комбинаций характеризуется расстоянием между ними в смысле ***Хэмминга*** или просто ***кодовым расстоянием***.

*Кодовое расстояние выражается числом символов, в которых комбинации отличаются одна от другой, и обозначается через d.*

Чтобы получить кодовое расстояние между двумя комбинациями двоичного кода, достаточно подсчитать число единиц в сумме этих комбинаций по модулю 2. Например

$$\begin{array}{r} 1001111101 \\ \oplus\ 1100001010 \\ \hline 0101110111 \end{array}$$

(Сложение ”по модулю 2”: $y= x_1 \oplus x_2$, сумма равна 1 тогда и только тогда, когда $x_1$ и $x_2$ не совпадают ).

Минимальное расстояние, взятое по всем парам кодовых разрешенных комбинаций кода, называют ***минимальным кодовым расстоянием.***

Более полное представление о свойствах кода дает ***матрица расстояний*** D, элементы которой $d_{ij}$ (i,j = 1,2,…,m) равны расстояниям между каждой парой из всех m разрешенных комбинаций.

Пример23: Представить симметричной матрицей расстояний код $x_1 = 000$; $x_2 = 001$; $x_3 = 010$; $x_4 = 111$.

Решение. 1. Минимальное кодовое расстояние для кода d=1.

2. Симметричная матрица четвертого порядка для кода

Таблица 4.1.

| | | x1 | x2 | x3 | x4 |
|---|---|---|---|---|---|
| | | 000 | 001 | 010 | 111 |
| x1 | 000 | | 1 | 1 | 3 |
| x2 | 001 | 1 | | 2 | 2 |
| x3 | 010 | 1 | 2 | | 2 |
| x4 | 111 | 3 | 2 | 2 | |

Декодирование после приема производится таким образом, что принятая кодовая комбинация отождествляется с той разрешенной, которая находится от нее на наименьшем кодовом расстоянии.

Такое декодирование называется декодированием ***по методу максимального правдоподобия***.

Очевидно, что при кодовом расстоянии d=1 все кодовые комбинации являются разрешенными. Например, при n=3 разрешенные комбинации образуют следующее множество: 000, 001, 010, 011, 100, 101, 110, 111. Любая одиночная ошибка трансформирует данную комбинацию в другую разрешенную комбинацию. Это случай безызбыточного кода, не обладающего корректирующей способностью. Если d = 2, то ни одна из разрешенных кодовых комбинаций при одиночной ошибке не переходит в другую разрешенную комбинацию. Например, подмножество разрешенных кодовых комбинаций может быть образовано по принципу четности в нем числа единиц. Например, для n=3:

000, 011, 101, 110 – разрешенные комбинации;

001, 010, 100, 111 – запрещенные комбинации.

Код обнаруживает одиночные ошибки, а также другие ошибки нечетной кратности (при n=3 тройные).

В общем случае при необходимости обнаруживать ошибки кратности до r включительно минимальное хэммингово расстояние между разрешенными кодовыми комбинациями должно быть по крайней мере на единицу больше r, т.е $d_{0\,min} \geq r+1$.

Действительно, в этом случае ошибка, кратность которой не превышает r, не в состоянии перевести одну разрешенную кодовую комбинацию в другую.

***Для исправления*** одиночной ошибки кодовой комбинации необходимо сопоставить подмножество запрещенных кодовых комбинаций.

Чтобы эти подмножества не пересекались, хэммингово расстояние между разрешенными кодовыми комбинациями должно быть не менее трех. При n=3 за разрешенные кодовые комбинации можно, например, принять 000 и 111. Тогда разрешенной комбинации 000 необходимо приписать подмножество запрещенных кодовых комбинаций 001, 010, 100, образующихся в результате двоичной ошибки в комбинации 000.

Подобным же образом разрешенной комбинации 111 необходимо приписать подмножество запрещенных кодовых комбинаций: 110, 011, 101, образующихся в результате возникновения единичной ошибки в комбинации 111:

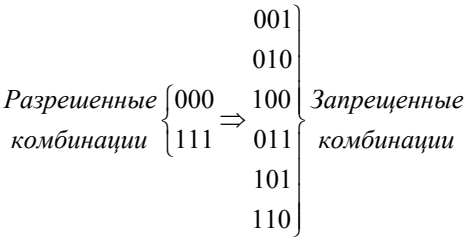


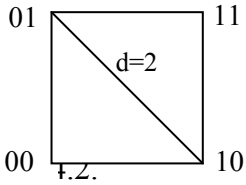

Рис. 4.1.

В общем случае для обеспечения возможности исправления всех ошибок кратности до s включительно при декодировании по методу максимального правдоподобия, каждая из ошибок

должна приводить к запрещенной комбинации, относящейся к подмножеству исходной разрешенной кодовой комбинации.

Любая n-разрядная двоичная кодовая комбинация может быть интерпретирована как вершина m-мерного единичного куба, т.е. куба с длиной ребра, равной 1.

При n=2 кодовые комбинации располагаются в вершинах квадрата:




При n=3 кодовые комбинации располагаются в вершинах единичного куба:

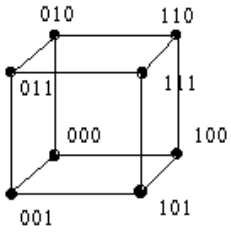



Рис. 4.3.

В общем случае n-мерный единичный куб имеет $2^n$ вершин, что равно наибольшему возможному числу кодовых комбинаций.

Такая модель дает простую геометрическую интерпретацию и кодовому расстоянию между отдельными кодовыми комбинациями. Оно соответствует наименьшему числу ребер единичного куба, которые необходимо пройти, чтобы попасть от одной комбинации к другой.

Ошибка будет не только обнаружена, но и исправлена, если искаженная комбинация остается ближе к первоначальной, чем к любой другой разрешенной комбинации, то есть должно быть: $r < \frac{1}{2} d$ или $d \geq 2r+1$. В общем случае для того, чтобы код позволял обнаруживать все ошибки кратности r и исправлять все ошибки кратности s (r>s), его кодовое расстояние должно удовлетворять неравенству d ≥ r+s+1 (r≥s). Метод декодирования при исправлении одиночных независимых ошибок можно пояснить следующим образом. В подмножество каждой разрешенной комбинации относят все вершины, лежащие в сфере с радиусом (d-1)/2 и центром в вершине, соответствующей данной разрешенной кодовой комбинации. Если в результате действия помехи комбинация переходит в точку, находящуюся внутри сферы (d-1)/2, то такая ошибка может быть исправлена. Если помеха смещает точку разрешенной комбинации на границу двух сфер (расстояние d/2) или больше (но не в точку, соответствующую другой разрешенной комбинации), то такое искажение может быть обнаружено. Для кодов с независимым искажением символов лучшие корректирующие коды – это такие, у которых точки,

соответствующие разрешенным кодовым комбинациям, расположены в пространстве равномерно. Проиллюстрируем построение корректирующего кода на следующем примере. Пусть исходный алфавит, состоящий из четырех букв, закодирован двоичным кодом: $x_1 = 00$; $x_2 = 01$; $x_3 = 10$; $x_4 = 11$. Этот код использует все возможные комбинации длины 2, и поэтому не может обнаруживать ошибки (так как d=1). Припишем к каждой кодовой комбинации один элемент 0 или 1 так, чтобы число единиц в нем было четное, то есть $x_1 = 000$; $x_2 = 011$; $x_3 = 101$; $x_4 = 110$. Для этого кода d=2, и, следовательно, он способен обнаруживать все однократные ошибки. Так как любая запрещенная комбинация содержит нечетное число единиц, то для обнаружения ошибки достаточно проверить комбинацию на четность (например, суммированием по модулю 2 цифр кодовой комбинации). Если число единиц в слове четное, то сумма по модулю 2 его разрядов будет 0, если нечетное – то 1. Признаком четности называют инверсию этой суммы.

Рассмотрим общую схему организации контроля по четности (контроль по нечетности, parity check – контроль по паритету).

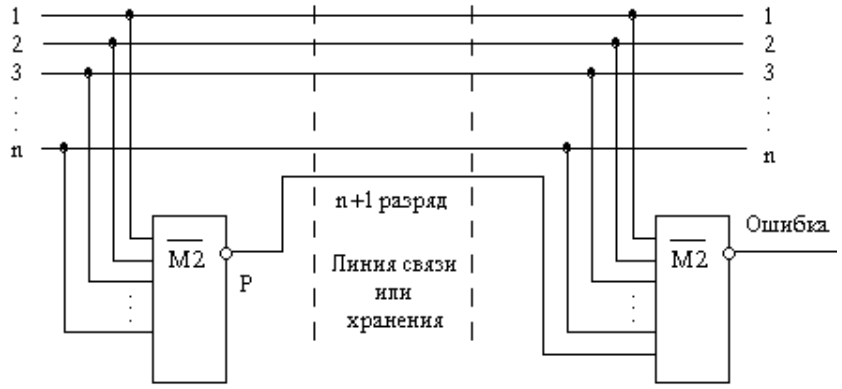



Рис. 4.4.

На n-входовом элементе $\overline{M2}$ формируется признак четности P числа, который в качестве дополнительного, (n+1)-го контрольного разряда (parity bit) отправляется вместе с передаваемым словом в линию связи или запоминающее устройство. Передаваемое (n+1)-разрядное слово имеет всегда нечетное число единиц. Если в исходном слове оно было нечетным, то инверсия функции M2 от такого слова равна 0, и нулевое значение контрольного разряда не меняет число единиц при передаче слова. Если же число единиц в исходном слове было четным, то контрольный разряд P для такого числа будет равен 1, и результирующее число единиц в передаваемом (n+1)-разрядном слове станет нечетным. Вид контроля, когда по линии передается нечетное число единиц, по строгой терминологии называют ***контролем по нечетности***.

На приемном конце линии или из памяти от полученного (n+1)-разрядного слова снова берется свертка по четности, Если значение этой свертки равно 1, то или в передаваемом слове, или в контрольном разряде при передаче или хранении произошла ошибка. Столь простой

контроль не позволяет исправить ошибку, но он, по крайней мере, дает возможность при обнаружении ошибки исключить неверные данные, затребовать повторную передачу и т.д.

Систему контроля можно построить на основе не только инверсии функции М2, но и прямой функции М2 (строго контроль по четности). Однако, в этом случае исходный код “все нули” будет иметь контрольный разряд, равный 0. В линию отправится посылка из сплошных нулей, и на приемном конце она будет неотличима от весьма опасной неисправности – полного пропадания связи. Поэтому контроль по четности в своем чистом виде почти никогда не применяют, контрольный разряд формируют как четности, и в нестрогой терминологии “контролем по четности” называют то, что, строго говоря, на самом деле является контролем по нечетности.

Контроль по четности основан на том, что одиночная ошибка (безразлично – пропадание единицы или появление линией) инвертирует признак четности. Однако две ошибки проинвертируют его дважды, то есть оставят без изменения, поэтому двойную ошибку контроль по четности не обнаруживает. Рассуждая аналогично, легко прийти к выводу, что контроль по четности обнаруживает все нечетные ошибки и не реагирует на любые четные. Пропуск четных ошибок – это не какой-либо дефект системы контроля. Это следствие предельно малой избыточности, равной всего одному разряду. Для более глубокого контроля требуется соответственно и большая избыточность. Если ошибки друг от друга не зависят, то из не обнаруживаемых чаще всего будет встречаться двойная ошибка, а при вероятности одиночной ошибки, равной P, вероятность двойной будет $P^2$. Поскольку в нормальных цифровых устройствах $P<<1$, необнаруженные двойные ошибки встречаются значительно реже, чем обнаруженные одиночные. Поэтому далее при таком простом контроле качество работы устройства существенно возрастает. Это верно лишь для взаимно независимых ошибок.

Признаки четности можно использовать для контроля только неизменяемых данных. При выполнении над данными каких-либо логических операций признаки четности слов в общем изменяются, и попытки компенсировать эти изменения оказываются неэффективными. Счастливое исключение – операция арифметического сложения: сумма по модулю 2 признаков четности двоичных слагаемых и всех, возникших в процессе сложения переносов, равна признаку четности кода арифметической суммы этих слагаемых.

Контроль по четности – самый дешевый по аппаратурным затратам вид контроля, и применяется он очень широко. Практически любой канал передачи цифровых данных или запоминающее устройство, если они не имеют какого-либо более сильного метода контроля, защищены контролем по четности.

Продолжим рассмотрение примера построения корректирующего метода. Чтобы код был способен и исправлять однократные ошибки, необходимо добавить еще не менее двух разрядов. Это можно сделать различными способами, например, повторить первые две цифры:

$x_1 = 00000$; $x_2 = 01101$; $x_3 = 10110$; $x_4 = 11011$;

Матрица расстояний этого кода:

Таблица 4.2.

D=

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| $x_1$ | | 3 | 3 | 4 |
| $x_2$ | 3 | | 4 | 3 |
| $x_3$ | 3 | 4 | | 3 |
| $x_4$ | 4 | 3 | 3 | |

Видно, что $d \geq 3$, что отвечает неравенству ($d \geq 2+3+1$).

Пример23: Постройте корректирующий код для передачи двух сообщений:

1) обнаруживающий одну ошибку;
2) обнаруживающий и исправляющий одну ошибку;
3) обнаруживающий две и исправляющий одну ошибку.

## КОНТРОЛЬ ПРАВИЛЬНОСТИ РАБОТЫ ЦИФРОВЫХ УСТРОЙСТВ

### Общие сведения

В процессе работы цифрового устройства возникают ошибки, искажающие информацию. Причинами таких ошибок могут быть:

- выход из строя какого-либо элемента, из-за чего устройство теряет работоспособность;
- воздействие различного рода помех, возникающих из-за проникновения сигналов из одних цепей в другие через различные паразитные связи.

Выход из строя элемента устройства рассматривается как неисправность. При этом в устройстве наблюдается постоянное искажение информации.

Иной характер искажений информации имеет место под воздействием помех. Вызвав ошибку, помехи могут затем в течение длительного времени не проявлять себя. Такие ошибки называют случайными сбоями.

В связи с возникновением ошибок необходимо снабжать цифровые устройства системой контроля правильности циркулирующей в ней информации. Такие системы контроля могут предназначаться для решения задач двух типов:

- задачи обнаружения ошибок;
- задачи исправления ошибок.

Система обнаружения ошибок, производя контроль информации, способна лишь выносить решения: нет ошибок и есть ошибка, причем в последнем случае она не указывает, какие разряды

слов искажены.

Система исправления ошибок сигнализирует о наличии ошибок и указывает, какие из разрядов искажены. При этом непосредственное исправление цифр искаженных разрядов представляет собой уже несложную операцию. Если известно, что некоторый разряд двоичного слова ошибочен, то появление в нем ошибочного *лог.*0 означает, что правильное значение — *лог.* 1 и наоборот.

Трудно локализовать ошибку, т.е. указать, в каких разрядах слова она возникла. После решения этой задачи само исправление сводится лишь к инверсии цифр искаженных разрядов, поэтому обычно под исправлением ошибок понимают решение задачи локализации ошибок.

При постоянном нарушении правильности информации, обнаружив ошибку, можно принять меры для поиска неисправного элемента и заменить его исправным. Причины же случайных сбоев обычно выявляются чрезвычайно трудно, и такие изредка возникающие ошибки желательно было бы устранять автоматически, восстанавливая правильное значение слов с помощью системы исправления ошибок. Однако следует иметь в виду, что система исправления ошибок требует значительно большего количества оборудования, чем система обнаружения ошибок.

Ниже раздельно рассматриваются методы контроля:

- цифровых устройств хранения и передачи информации;
- цифровых устройств обработки информации.

К устройствам первого типа могут быть отнесены запоминающие устройства, регистры, цепи передачи и другие устройства, в которых информация не должна изменяться. На выходе этих устройств информация та же, что и на входе. К устройствам второго типа относятся устройства, у которых входная информация не совпадает с выходной и в тех случаях, когда ошибки не возникают. Примером могут служить арифметические и логические устройства.

**Обнаружение одиночных ошибок в устройствах хранения и передачи информации**

Для дальнейшего изложения потребуется понятие кодовое расстояние по Хеммингу. Для двух двоичных слов кодовое расстояние по Хеммингу есть число разрядов, в которых разнятся эти слова. Так, для слов 11011 и 10110 кодовое расстояние $\boldsymbol{d} = 3$, так как эти слова различаются в трех разрядах (первом, третьем и четвертом).

Пусть используемые слова имеют $\boldsymbol{m}$ разрядов. Для представления информации можно использовать все $2^m$ возможных комбинаций от 00 ... 0 до 11 ... 1. Тогда для каждого слова найдутся другие такие слова, которые отличаются от данного не более чем в одном разряде. Например, для некоторого слова 1101 можно найти следующие слова: 0101, отличающиеся только в четвертом разряде; 1001, отличающееся только в третьем разряде, и т.д. Таким образом, минимальное кодовое расстояние $d_{\min} = 1$. Обнаружить ошибки в таких словах невозможно.

Например, если передавалось слово $N1$ = 1101, а принято $N2$ = 0101, то в принятом слове невозможно обнаружить никаких признаков наличия ошибки (ведь могло бы быть передано и слово $N2$ = 0101). Для того чтобы можно было обнаружить одиночные ошибки (ошибки, возникающие не более чем в одном из разрядов слова), минимальное кодовое расстояние должно удовлетворять условию $d_{min} \geq 2$. Это условие требует, чтобы любая пара используемых слов отличалась друг от друга не менее чем в двух разрядах. При этом, если возникает ошибка, она образует такую комбинацию цифр, которая не используется для представления слов, т.е. образует так называемую запрещенную комбинацию.

Для получения $d_{min} = 2$ достаточно к словам, использующим любые комбинации из ***m*** информационных двоичных разрядов, добавить один дополнительный разряд, называемый контрольным. При этом значение цифры контрольного разряда выбирают таким, чтобы общее число единиц в слове было четным. Например:

| Информационные разряды | Контрольные разряды |
|---|---|
| 11001110111 | 0 |
| 11010100111 | 1 |

В первом из приведенных примеров число единиц информационной части четно (8), поэтому контрольный разряд должен содержать 0. Во втором примере число единиц в информационной части слова нечетно (7), и для того, чтобы общее число единиц в слове было четным, контрольный разряд должен содержать единицу. Таким способом во все слова вводится определенный признак — четность числа единиц. Принятые слова проверяются на наличие в них этого признака, и, если он оказывается нарушенным (т.е. обнаруживается, что число содержащихся в разрядах слова единиц нечетно), принимается решение, что слово содержит ошибку.

Этот метод позволяет обнаруживать ошибку. Но с его помощью нельзя определить, в каком разряде слова содержится ошибка, т.е. нельзя исправить ее. Кроме того, при этом методе не могут обнаруживаться ошибки четной кратности, т.е. ошибки одновременно в двух, четырех и т.д. разрядах, так как при таком четном числе ошибок не нарушается четность числа единиц в разрядах слова. Однако наряду с одиночными ошибками могут обнаруживаться ошибки, возникающие одновременно в любом нечетном числе разрядов.

На практике часто вместо признака четности используется признак нечетности, т.е. цифра контрольного разряда выбирается такой, чтобы общее число единиц в разрядах слова было нечетным. При этом, если имеет место, например, обрыв линии связи, это обнаруживается, так как принимаемые слова будут иметь 0 во всех разрядах и нарушится принцип нечетности числа единиц.

Рассмотрим схемы, выполняющие операцию проверки на четность (нечетность). *Проверка*

*на четность требует суммирования по модулю 2 цифр разрядов слова.* Если $a^1$, $a^2$,…, $a^n$ — цифры разрядов, результат проверки на четность определится выражением $p = a^1 \oplus a^2 \oplus a^3 \oplus$ ,…, $\oplus a^n$. Если $p = 0$, то число единиц в разрядах слова четно, в противном случае оно нечетно.

Наиболее просто эта операция реализуется, когда контролируемое слово передается в последовательной форме. Суммирование в этом случае может быть выполнено в последовательности $p = ...((a^1 \oplus a^2) \oplus a^3) \oplus ... \oplus a^n$. К результату суммирования $p^i$ первых разрядов прибавляется цифра очередного поступающего разряда $(i + 1)$, находится результат суммирования $(i + 1)$ разрядов $p^{i+1} = p^i \oplus a^{i+1}$, и так до тех пор, пока не будут просуммированы цифры всех разрядов.

Из таблицы истинности

Таблица 1

| $p_i$ | $p_{i+1}$ | $p_{i+1} = p_i \oplus a_{i+1}$ | $p_i$ | $p_{i+1}$ | $p_{i+1} = p_i \oplus a_{i+1}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |

для операции $p^{i+1} = p^i \oplus a^{i+1}$ (табл. 1) видно, что лог.0 не должен менять состояния устройства суммирования ($p^{i+1} = p^i$), лог.1 переводит устройство в новое состояние ($p^{i+1} = \overline{p_i}$). Эта логика соответствует работе триггера со счетным входом (рис.1).

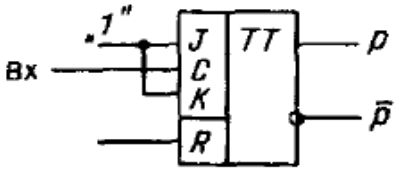



Рис.1. Триггер со счетным входом.

Действительно, пусть триггер был предварительно установлен в состояние 0, после чего на его синхронизирующий вход стали поступать логические уровни, соответствующие цифрам контролируемого слова. При этом первая лог.1 переведет триггер в состояние 1, вторая лог. 1 вернет триггер в состояние 0 и т.д. Следовательно, после подачи четного числа единиц триггер окажется в состоянии $p = 0$; при поступлении нечетного числа единиц — в состоянии $p = 1$.

Если разряды контролируемого слова передаются в параллельной форме, то последовательность действий при проверке на четность может быть следующая:

$$p = ... (((a_1 \oplus a_2) \oplus (a_3 \oplus a_4)) \oplus ((a_5 \oplus a_6) \oplus (a_7 \oplus a_8)))\oplus ...$$

Согласно этому выражению для нахождения $p$ вначале попарно суммируются по модулю 2 цифры разрядов контролируемого слова, далее полученные результаты также суммируются попарно и т.д.

Этот принцип вычисления $p$ использован в схеме проверки на четность на рис.2.

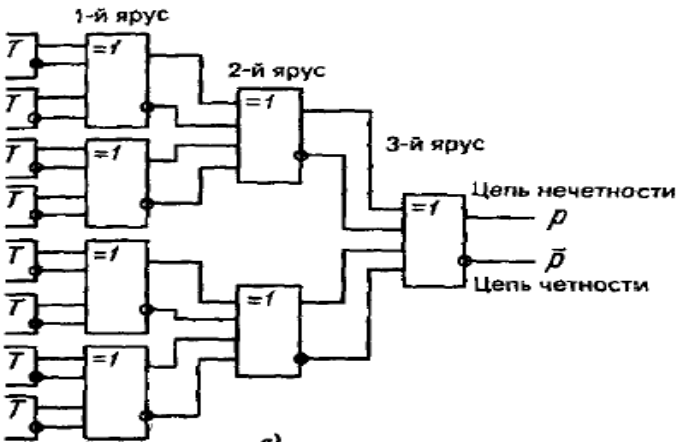



Рис.2. Схема проверки на четность.

Цифры разрядов (и их инверсии) поступают на входы элементов (на рис.2 обозначены =1) первого яруса схемы, в которых они попарно суммируются по модулю 2. Полученные результаты попарно суммируются в элементах второго яруса и т.д.

Результат проверки на четность образуется на выходе элемента старшего яруса. Каждый из элементов схемы реализует следующую логическую функцию:

$$y = x_1 \oplus x_2 = x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 = \overline{\overline{x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2}} = (x_1 \mid \bar{x}_2) \mid (\bar{x}_1 \mid x_2).$$

Построенная по данному выражению схема элемента, выполняющего операцию суммирования по модулю 2, приведена на рис.3.

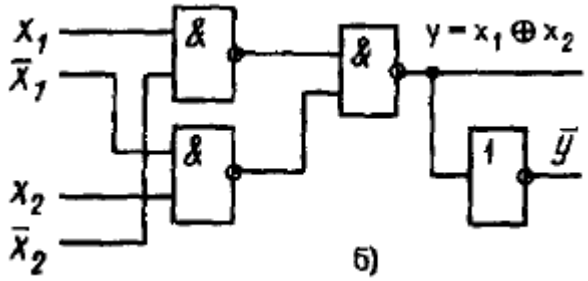



Рис.3.Схема элемента, выполняющего операцию суммирования по модулю 2.

Определим число ярусов и число элементов в этой схеме проверки на четность. Пусть число разрядов ***n*** контролируемого слова составляет целую степень двух. Число элементов в отдельных ярусах $a^i$ составляет геометрическую прогрессию 1,2,4,8,..., ***n***/2, знаменатель которой $q = 2$. Для последнего ***k***-то яруса $a^k = 1$, для первого яруса $a^1 = a^k q^{k-1}$ откуда $2^{k-1} = n/2$ или $2^k = n$.

Из этого соотношения можно найти число ярусов $k$. Число суммирующих элементов в схеме равно сумме членов приведенной выше геометрической прогрессии:

$$N_{эл} = \frac{a_k(q^{k-1} - 1)}{q - 1} = \frac{1 \cdot (2^k - 1)}{2 - 1} = 2^k - 1 = n - 1.$$

**Контроль арифметических операций**

В устройствах хранения и передачи информации одиночная ошибка вызывала искажение цифры лишь одного разряда слова. При выполнении арифметических операций одиночная ошибка в получаемом результате может вызвать искажение одновременно группы разрядов. Пусть в суммирующем счетчике хранится число $N^1$ = 10111 и на вход поступает очередная единица. Произойдет сложение: $N^2 = N^1+1$. При этом

| | |
|---|---|
| **Переносы** | 1 1 1 |
| $N_1$ | 1 0 1 1 1 |
| + | 1 |
| $N_2$ | 1 1 0 0 0 |

Пусть в процессе суммирования из-за ошибочной работы устройства не будет передан перенос из 2-го разряда в 3-й. Такая одиночная ошибка приведет к следующему результату:

| | |
|---|---|
| Переносы | 1 |
| $N_1$ | 1 0 1 1 1 |
| + | 1 |
| $N_{2ош}$ | 1 0 1 0 0 |

Сравнивая ошибочный результат $N^{2ош}$ с правильным $N^2$, видим, что они различаются в двух разрядах. Тем не менее считаем, что в $N^{2ош}$ содержится одиночная ошибка. Во всех случаях, когда ошибочный результат связан с арифметическим прибавлением (или вычитанием) ошибочной единицы к одному из разрядов, имеет место одиночная ошибка. И если ошибочный результат может быть получен из правильного результата путем арифметического суммирования (или вычитания) единицы не менее чем в $k$ разрядах, кратность ошибки равна $k$.

Для контроля арифметических операций чаще всего используется контроль по модулю $q$. Этот метод более универсален и годится также для контроля устройств хранения и передачи информации. Сущность метода состоит в следующем.

Контролируемое число $N$ арифметически делится на $q$, и выделяется остаток $r^N$. Остаток вписывается в контрольные разряды числа $N$ вслед за его информационными разрядами. Принятое число $N^*$ делится на $q$ и выделяется остаток $r^{N*}$. Эту операцию выполняет устройство свертки по модулю $q$ (рис.4).

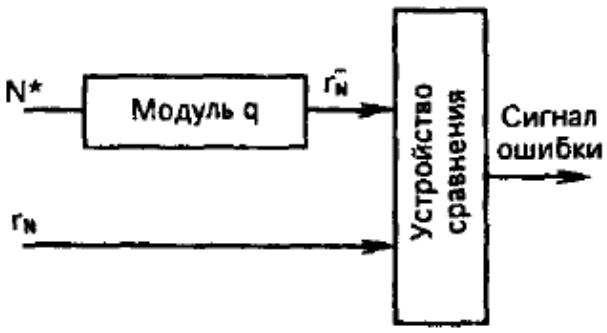


Рис.4. Устройство свертки по модулю *q*.

Устройство сравнения сравнивает $r^N$ и $r^N$*, в случае их несовпадения выносит решение о наличии ошибки в принятом слове. Схема на рис. 5 иллюстрирует принцип контроля суммирующего устройства.


Рис.5. Схема, иллюстрирующая принцип контроля суммирующего устройства.

Пусть в результате суммирования чисел $N^1$ и $N^2$ получено $N$*. Остатки $r^{N1}$ и $r^{N2}$ также суммируются с выделением остатка $r^N$. Если остаток $r^N$*, полученный от деления числа $N$* на модуль $q$, не совпадает с $r^N$, то элемент сравнения сигнализирует о наличии ошибок в работе устройства.

Чаще всего используется $q$ = 3, иногда выбирается $q$ = 7. При увеличении значения $q$ возрастает способность метода к обнаружению ошибок, но одновременно увеличивается объем контролирующего оборудования.

Рассмотрим пример применительно к схеме на рис. 5. Пусть $q$ = 3, $N^1 = 32^{10} = 100000^2$, $N^2 = 29^{10} = 011101^2$. Соответствующие этим числам остатки равны $r^{N1} = 2^{10} = 10^2$, $r^{N2} = 2^{10} = 10^2$ (при q = 3 остатки могут принимать значения 0, 1, 2 и для их представления в двоичной форме достаточно двух контрольных разрядов). При отсутствии ошибок в работе устройства результат суммирования чисел $N$* = $N^1$ + $N^2$ = $61^{10}$ = = $111101^2$, значение свертки по модулю 3 равно $r^N$* = $01^2$. Суммируя $r^{N1}$ и $r^{N2}$ и выделяя остаток по модулю 3, получаем $r^N$ = $01^2$. Совпадение $r^N$* = $r^N$ указывает на отсутствие ошибок. При наличии ошибок не имело бы места совпадение остатков $r^N$* и $r^N$.

Эффективность контроля по модулю характеризуется данными, приведенными в табл. 2.

Таблица 2

| Значение модуля | Доля необнаруживаемых ошибок | | |
|---|---|---|---|
| | 1-й кратности | 2-й кратности | 3-й кратности |
| 3 | 0 | 1/2 | 1/4 |
| 7 | 0 | 1/6 | 1/7 |

В таблице указано, какую часть всех возможных комбинаций ошибок составляют ошибки, которые не обнаруживаются при контроле по модулю. Как видно из приведенных данных, обнаруживаются все однократные ошибки; доля ошибок высокой кратности, оказывающихся необнаруженными, при модуле 7 меньше, чем при модуле 3. Тем самым эффективность контроля по модулю 7 выше, чем при модуле 3. Однако при контроле по модулю 7 контрольная часть слов содержит три двоичных разряда (вместо двух разрядов при модуле 3) и, кроме того, сложнее схемы формирования остатков (схемы свертки).

В заключение рассмотрим построение схем свертки по модулю 3. Общим для этих схем является следующий метод получения остатка. Каждый разряд числа вносит определенный вклад в формируемый остаток. В табл. 3 приведены остатки от деления на 3 значений, выражаемых единицами отдельных разрядов (т.е. весовых коэффициентов разрядов). Эти остатки для единиц нечетных разрядов равны 1, для четных разрядов они равны 2. Следовательно, для получения остатка от деления на 3 всего числа достаточно просуммировать остатки для единиц отдельных его разрядов и затем для получения суммы найти остаток от деления на 3.

Таблица 3

| Номер разряда | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
|---|---|---|---|---|---|---|---|---|---|
| Весовой коэффициент разряда | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | ... |
| Остаток от деления на 3 значения, выражаемого единицей разряда | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | ... |

Например, пусть $N = 11001011_2$; сумма остатков, создаваемых отдельными разрядами, $S=1\cdot2+1\cdot1 + 0\cdot2 + 0\cdot1 + 1\cdot2+0\cdot1+1\cdot2+1\cdot1 = 8$; далее, деля 8 на 3, получаем остаток $r_N = 2$.

**Схема свертки по модулю 3 для последовательной формы передачи чисел**

Схема может быть выполнена в виде двухразрядного счетчика с циклом 3, построенного таким образом, что единицы нечетных разрядов поступающего на вход числа вызывают увеличение содержимого счетчика на единицу, а единицы четных разрядов вызывают увеличение числа в счетчике на два. Функционирование такого счетчика описывается табл. 4.

Здесь $b$— код, определяющий четность номера очередного разряда числа, поступающего на вход счетчика; примем для четных разрядов $b = 0$, для нечетных разрядов $b = 1$.

Таблица 4

| Четность номера в разряде $b$ | Текущее состояние счетчика | | Следующее состояние счетчика | |
|---|---|---|---|---|
| | $a_2$ | $a_1$ | $a_2^*$ | $a_1^*$ |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |

Таблица 5

| Вид перехода триггера | Сигналы на входах | | Вид перехода триггера | Сигналы на входах | |
|---|---|---|---|---|---|
| | $J$ | $K$ | | $J$ | $K$ |
| $0 \to 0$ | 0 | — | $1 \to 0$ | — | 1 |
| $0 \to 1$ | 1 | — | $1 \to 1$ | — | 0 |

По этой таблице и таблице переходов JK-триггера (табл. 5) построены приведенные на рис. 6 карты, по которым находят логические выражения для входов триггеров ТТ1 и ТТ2 счетчика:

$$J_1 = a_2 \cdot \bar{b} \vee \bar{a}_2 \cdot b, \quad K_1 = 1;$$

$$J_2 = \bar{a}_1 \cdot \bar{b} \vee a_1 \cdot b, \quad K_2 = 1.$$

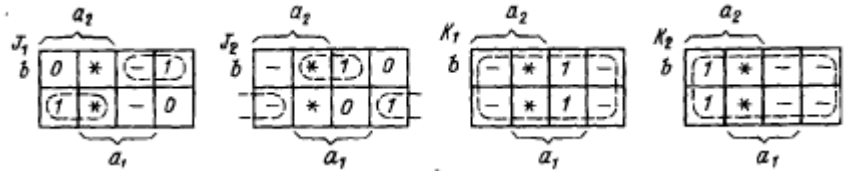



Рис.6. Карты, по которым находят логические выражения для входов триггеров ТТ1 и ТТ2 счетчика.

Представив выражения для $J1$ и $J2$ в базисе И-НЕ:

$$J_1 = (a_2 \mid \bar{b}) \mid (\bar{a}_2 \mid b),$$

$$J_2 = (\bar{a}_1 \mid \bar{b}) \mid (a_1 \mid b),$$

получим схему межтриггерных связей на рис.7. Логическая переменная $b$ формируется триггером 3.

Этот триггер переключается тактовыми импульсами (ТИ), следующими с частотой поступления разрядов числа на вход счетчика. Таким образом, в моменты поступления нечетных разрядов триггер 3 устанавливается в состояние 1 и $b$ = 1, в моменты поступления четных разрядов $b$= 0.

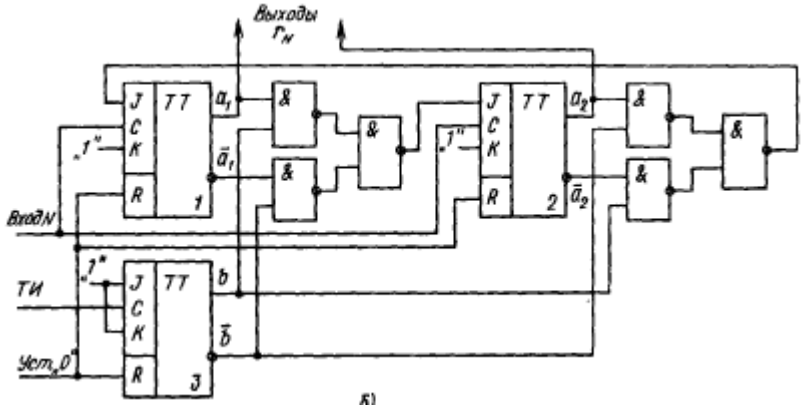
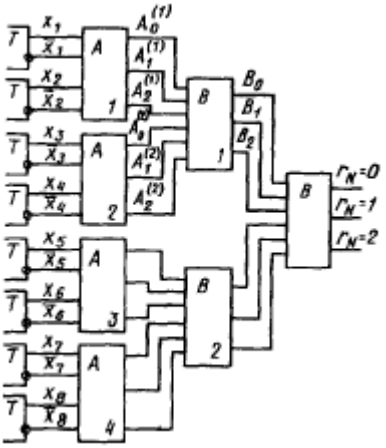



Рис.7. Схема межтриггерных связей.

**Схема свертки для параллельной формы представления числа**

При параллельной форме представления числа обычно используется пирамидальный способ построения схемы свертки, показанный на рис. 8.




Элементы **А** первого яруса формируют остатки для пар разрядов числа, выдавая уровень *лог.* 1 на один из выходов $A^0, A^1, A^2$ в зависимости от значения остатка (0,1,2). В последующих ярусах используются однотипные элементы **В**, которые формируют остатки по результатам, выдаваемым парой элементов предыдущего яруса. На выходе элемента последнего яруса образуется остаток для всего числа.

Выходы элементов определяются следующими логическими выражениями: для первого яруса

$$A_0 = x_1 \cdot x_2 \vee \bar{x}_1 \cdot \bar{x}_2, \quad A_1 = x_1 \cdot \bar{x}_2,$$

$$A_2 = \bar{x}_1 \cdot x_2;$$

для остальных ярусов

$$B_0 = A'_0 \cdot A''_0 \vee A'_1 \cdot A''_2 \vee A'_2 \cdot A''_1, \; B_1 = A'_0 \cdot A''_1 \vee A'_1 \cdot A''_0 \vee A'_2 \cdot A''_2,$$

$$B_2 = A'_0 \cdot A''_2 \vee A''_0 \cdot A'_2 \vee A'_1 \cdot A''_1.$$

Если число разрядов $n = 2^k$, то число ярусов в схеме свертки равно $k$, а число элементов составляет $(n - 1)$.

### 4.4 Понятие качества корректирующего кода

Одной из основных характеристик корректирующего кода является **избыточность кода**, указывающая степень удлинения кодовой комбинации для достижения определенной корректирующей способности.

Если на каждые m символов выходной последовательности кодера канала приходится k информационных и (m-k) проверочных, то относительная избыточность кода может быть выражена одним из соотношений: $R_m = (m-k)/m$ или $R_k = (m-k)/k$.

Величина $R_k$, изменяющаяся от 0 до ∞, предпочтительнее, так как лучше отвечает смыслу понятия избыточности. Коды, обеспечивающие заданную корректирующую способность при минимально возможной избыточности, называют ***оптимальными***.

В связи с нахождением оптимальных кодов оценим, например, возможное наибольшее число Q разрешенных комбинаций m-значного двоичного кода, обладающего способностью исправлять взаимно независимые ошибки кратности до s включительно. Это равносильно отысканию числа комбинаций, кодовое расстояние между которыми не менее d=2s+1.

Общее число различных исправляемых ошибок для каждой разрешающей комбинации составляет

$$\sum_{i=1}^{s} C_m^i,$$

где $C_m^i$ – число ошибок кратности i.

Каждая из таких ошибок должна приводить к запрещенной комбинации, относящейся к подмножеству данной разрешенной комбинации. Совместно с этой комбинацией подмножество включает комбинаций.

$$1+\sum_{i=1}^{s} C_m^i$$

Однозначное декодирование возможно только в том случае, когда названные подмножества не пересекаются. Так как общее число различных комбинаций m-значного двоичного кода составляет $2^m$, число разрешенных комбинаций не может превышать

$$2^m \Big/ (1+\sum_{i=1}^{s} C_m^i)$$

Или $$Q \le 2^m \Big/ \sum_{i=0}^{s} C_m^i .$$

Эта верхняя оценка найдена **Хэммингом**. Для некоторых конкретных значений кодового расстояния d, соответствующие Q укажем в таблице:

Таблица 4.3.

| d | Q | d | Q |
|---|---|---|---|
| 1 | $2^m$ | 5 | $\le \frac{2^{m+1}}{m^2+1}$ |
| 2 | $\le 2^{m-1}$ | … | ... |
| 3 | $\le \frac{2^m}{m+1}$ | … | ... |
| 4 | $\le \frac{2^{m-1}}{m}$ | ... | $\le \frac{2^m}{1+c_m^1+c_m^2+...+c_m^k}$ |

Коды, для которых в приведенном соотношении достигается равенство, называют также ***плотноупакованными***.

Однако не всегда целесообразно стремиться к использованию кодов, близких к оптимальным. Необходимо учитывать другой, не менее важный показатель качества корректирующего кода – сложность технической реализации процессов кодирования и декодирования.

Если информация должна передаваться по медленно действующей и дорогостоящей линии связи, а кодирующее и декодирующее устройства предполагается выполнить на высоконадежных и быстродействующих элементах, то сложность этих устройств не играет существенной роли. Решающим фактором в этом случае является повышение эффективности пользования линией связи, поэтому желательно применение корректирующих кодов с минимальной избыточностью.

Если же корректирующий код должен быть применен в системе, выполненной на элементах, надежность и быстродействие которых равны или близки надежности и быстродействию элементов кодирующей и декодирующей аппаратуры. Это возможно, например, для повышения достоверности воспроизведения информации с запоминающего устройства ЭВМ. Тогда критерием качества корректирующего кода является надежность системы в целом, то есть с учетом возможных искажений и отказов в устройствах кодирования и декодирования. В этом случае часто более целесообразны коды с большей избыточностью, но простые в технической реализации.

## 4.5 Линейные коды

Самый большой класс разделимых кодов составляют ***линейные коды***, у которых значения проверочных символов определяются в результате проведения линейных операций над

определенными информационными символами. Для случая двоичных кодов каждый проверочный символ выбирают таким образом, чтобы его сумма с определенными информационными символами была равна 0. Символ проверочной позиции имеет значение 1, если число единиц информационных разрядов, входящих в данное проверочное равенство, нечетно, и 0, если оно четно. Число проверочных равенств (а следовательно, и число проверочных символов) и номера конкретных информационных разрядов, входящих в каждое из равенств, определяется тем, какие и сколько ошибок должен исправлять или обнаруживать данный код. Проверочные символы могут располагаться на любом месте кодовой комбинации. При декодировании определяется справедливость проверочных равенств. В случае двоичных кодов такое определение сводится к проверкам на четность числа единиц среди символов, входящих в каждое из равенств (включая проверочные). Совокупность проверок дает информацию о том, имеется ли ошибка, а в случае необходимости и о том, на каких позициях символы искажены.

Любой двоичный линейный код является групповым, так как совокупность входящих в него кодовых комбинаций образует группу. Уточнение понятий линейного и группового кода требует ознакомления с основами линейной алгебры.

### 4.6 Математическое введение к линейным кодам

Основой математического описания линейных кодов является ***линейная алгебра*** (теория векторных пространств, теория матриц, теория групп). Кодовые комбинации рассматривают как элементы множества, например, кодовые комбинации двоичного кода принадлежат множеству положительных двоичных чисел.

Множества, для которых определены некоторые алгебраические операции, называют ***алгебраическими системами***. Под ***алгебраической операцией*** понимают однозначные сопоставление двум элементам некоторого третьего элемента по определенным правилам. Обычно основную операцию называют ***сложением*** (обозначают a+b=c) или ***умножением*** (обозначают a*b=c), а обратную ей – ***вычитанием*** или ***делением***, даже, если эти операции проводятся не над числами и не идентичны соответствующим арифметическим операциям.

Рассмотрим кратко основные алгебраические системы, которые широко используют в теории корректирующих кодов.

***Группой*** множество элементов, в котором определена одна основная операция и выполняются следующие аксиомы:

1. В результате применения операции к любым двум элементам группы образуется элемент этой же группы (требование замкнутости).

2. Для любых трех элементов группы a,b,c удовлетворяется равенство (a+b)+c=a+(b+c), если основная операция – сложение, и равенство a(bc)=(ab)c, если основная операция – умножение.

3. В любой группе $G_n$ существует однозначно определенный элемент, удовлетворяющий при всех значениях a из $G_n$ условию a+0=0+a, если основная операция – сложение, или условию a*1=1*a=a, если основная операция – умножение. В первом случае этот элемент называют ***нулем*** и обозначают символом 0, а во втором – ***единицей*** и обозначают символом 1.

4. Всякий элемент a группы обладает элементом, однозначно определенным уравнением a+(-a)=-a+a=0, если основная операция – сложение, или уравнением $a*a^{-1}= a^{-1}*a=1$, если основная операция – умножение.

В первом случае этот элемент называют ***противоположным*** и обозначают (-a), а во втором – ***обратным*** и обозначают $a^{-1}$.

Если операция, определенная в группе, коммутативна, то есть справедливо равенство a+b=b+a (для группы по сложению) или равенство a*b=b*a (для группы по умножению), то группу называют ***коммутативной*** или ***абелевой***.

Группу, состоящую из конечного числа элементов, называют ***порядком*** группы.

Чтобы рассматриваемое нами множество n-разрядных кодовых комбинаций было конечной группой, при выполнении основной операции число разрядов в результирующей кодовой комбинации не должно увеличиваться. Этому условию удовлетворяет операция символического поразрядного сложения по заданному модулю q (q – простое число), при которой цифры одинаковых разрядов элементов группы складываются обычным порядком, а результатом сложения считается остаток от деления полученного числа по модулю q.

При рассмотрении двоичных кодов используется операция сложения по модулю 2. Результатом сложения цифр данного разряда является0, если сумма единиц в нем четная, и 1, если сумма единиц в нем нечетная, например,

$$\begin{array}{l} 1011101 \\ 0111101 \\ \underline{0001110} \\ \oplus\ 1101110 \end{array}$$

Выбранная нами операция коммутативна, поэтому рассматриваемые группы будут абелевыми.

Нулевым элементом является комбинация, состоящая из одних нулей. Противоположным элементом при сложении по модулю 2 будет сам заданный элемент. Следовательно, операция вычитания по модулю 2 тождественна операции сложения.

Пример24. Определить, являются ли группами следующие множества кодовых комбинаций:

1) 0001, 0110, 0111, 0011;

2) 0000, 1101, 1110, 0111;

3) 000, 001, 010, 011, 100, 101, 110, 111.

Решение: Первое множество не является группой, так как не содержит нулевого элемента.

Второе множество не является группой, так как не выполняется условие замкнутости, например, сумма по модулю 2 комбинаций 1101 и 1110 дает комбинацию 0011, не принадлежащую исходному множеству.

Третье множество удовлетворяет всем перечисленным условиям и является группой.

Подмножества группы, являющиеся сами по себе группами относительно операции, определенной в группе, называют ***подгруппами***. Например, подмножество трехразрядных кодовых комбинаций: 000, 001, 010, 011 образуют подгруппу указанной в примере группы трехразрядных кодовых комбинаций.

Пусть в абелевой группе $G_n$ задана определенная подгруппа А. Если В – любой, не входящий в А элемент из $G_n$, то суммы (по модулю 2) элементов В с каждым из элементов подгруппы А образуют определенный класс группы $G_n$ по подгруппе А, порождаемый элементом В.

Элемент В, естественно, содержится в этом смежном классе, так как любая подгруппа содержит нулевой элемент. Взяв последовательно некоторые элементы $B_j$ группы, не вошедшие в уже образованные смежные классы, можно разложить всю группу на смежные классы по подгруппе А.

Элементы $B_j$ называют ***образующими элементами смежных классов по подгруппам***.

В таблице разложения, иногда называемой ***групповой таблицей***, образующие элементы обычно располагают в крайнем левом столбце, причем крайним левым элементом подгруппы является нулевой элемент.

Пример25: Разложить группу трехразрядных двоичных кодовых комбинаций по подгруппе двухразрядных кодовых комбинаций.

Решение: Разложение выполняют в соответствии с таблицей:

Таблица 4.4.

| $A_1=0$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|
| 000 | 001 | 010 | 011 |
| $B_1$ | $A_2 \oplus B_1$ | $A_3 \oplus B_1$ | $A_4 \oplus B_1$ |
| 100 | 101 | 110 | 111 |

Пример26: Разложить группу четырехразрядных двоичных кодовых комбинаций по подгруппе двухразрядных кодовых комбинаций.

Решение: Существует много вариантов разложения в зависимости от того, какие элементы выбраны в качестве образующих смежных классов.

Один из вариантов:

Таблица 4.5.

| $A_1=0$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|
| 0000 | 0001 | 0110 | 0111 |
| $B_1$<br>0100 | $A_2 \oplus B_1$<br>0101 | $A_3 \oplus B_1$<br>0110 | $A_4 \oplus B_1$<br>0111 |
| $B_2$<br>1010 | $A_2 \oplus B_2$<br>1011 | $A_3 \oplus B_2$<br>1000 | $A_4 \oplus B_2$<br>1001 |
| $B_3$<br>1100 | $A_2 \oplus B_3$<br>1101 | $A_3 \oplus B_3$<br>1110 | $A_4 \oplus B_3$<br>1111 |

***Кольцом*** называют множество элементов R, на котором определены две операции (сложение и умножение), такие, что

1) множество R является коммутативной группой по отношению;

2) произведение элементов $a \in R$ и $b \in R$ есть элемент R (замкнутость по отношению и умножению);

3) для любых трех элементов a,b,c из R справедливо равенство a(bc)=(ab)c (ассоциативный закон для умножения);

4) для любых трех элементов a,b,c из R выполняются соотношения a(b+c)=ab+ac и (b+c)a=ba+ca (дистрибутивные законы);

Если для любых двух элементов кольца справедливо соотношение ab=ba, то кольцо называют ***коммутативным***.

Кольцо может не иметь единичного элемента по умножению и обратных элементов.

Примером кольца может служить множество действительных четных целых чисел относительно обычных операций сложения и умножения.

***Полем F*** называют множество, по крайней мере, двух элементов, в котором определены две операции – сложение и умножение, и выполняются следующие аксиомы:

1) множество элементов образуют коммутативную группу по сложению;

2) множество ненулевых элементов образуют коммутативную группу по умножению;

3) для любых трех элементов множества a,b,c выполняется соотношение (дистрибутивный закон) a(b+c)=ab+ac.

Поле F является, следовательно, коммутативным кольцом с единичным элементом по умножению, в котором каждый ненулевой элемент обладает обратным элементом. Примером поля может служить множество всех действительных чисел.

Поле GF(P), состоящее из конечного числа элементов P, называют ***конечным полем*** или ***полем Галуа***. Для любого числа P, являющегося степенью простого числа q, существует поле, насчитывающее p элементов. Например, совокупность чисел по модулю q, если q - простое число, является полем.

Поле не может содержать менее двух элементов, поскольку в нем должны быть по крайней мере единичный элемент относительно операции сложения (0) и единичный элемент относительно операции умножения (1). Поле, включающее только 0 и 1, обозначим GF(2). Правила сложения и умножения в поле с двумя элементами следующие:

| + | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| × | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Рис. 4.5 Правила сложения и умножения в поле с двумя элементами

Двоичные кодовые комбинации, являющиеся упорядоченными последовательностями из n Элементов поля GF(2), рассматриваются в теории кодирования как частный случай последовательностей из n элементов поля GF(P). Такой подход позволяет строить и анализировать коды с основанием, равным степени простого числа. В общем случае суммой кодовых комбинаций $A_j$ и $A_i$ называют комбинацию $A_f = A_i + A_j$, в которой любой символ $A_k$ (k=1,2,…,n) представляет собой сумму k-х символов комбинаций, причем суммирование производится по правилам поля GF(P). При этом вся совокупность n-разрядных кодовых комбинаций оказывается абелевой группой.

В частном случае, когда основанием кода является простое число q, правило сложения в поле GF(q) совпадает с правилом сложения по заданному модулю q.

### 4.7 Линейный код как пространство линейного векторного пространства

В рассмотренных алгебраических системах (группа, кольцо, поле) операции относились к единому классу математических объектов (элементов). Такие операции называют внутренними законами композиции элементов.

В теории кодирования широко используются модели, охватывающие два класса математических объектов (например, L и Ω). Помимо внутренних законов композиции в них задаются внешние законы композиции элементов, по которым любым двум элементам $\omega \in \Omega$ и $a \in L$ ставится в соответствие элемент $c \in L$.

***Линейным векторным пространством над полем элементов F*** (скаляров) называют множество элементов V (векторов), если для него выполняются следующие аксиомы:

1) множество V является коммутативной группой относительно операции сложения;

2) для любого вектора v изV и любого скаляра c из F определено произведение cv, которое содержится в V (замкнутость по отношению умножения на скаляр);

3) если u и v из V векторы, а c и d из F скаляры, то справедливо c(c+v)=cu+cv; (c+d)v=cv+dv (дистрибутивные законы);

4) если v-вектор, а c и d-скаляры, то (cd)v=c(dv) и 1*v=v

(ассоциативный закон для умножения на скаляр).

Выше было определено правило поразрядного сложения кодовых комбинаций, при котором вся их совокупность образует абелеву группу. Определим теперь операцию умножения последовательности из n элементов поля GF(P) (кодовой комбинации) на элемент поля $a_i$ GF(P)аналогично правилу умножения вектора на скаляр: $a_i(a_1, a_2, \dots, a_n)= (a_i a_1, a_i a_2, \dots, a_i a_n)$

(умножение элементов производится по правилам поля GF(P)).

Поскольку при выбранных операциях дистрибутивные законы и ассоциативный закон (п.п.3.4) выполняются, все множество n-разрядных кодовых комбинаций можно рассматривать как векторное линейное пространство над полем GF(2) (т.е. 0 и 1). Сложение производят поразрядно по модулю 2. При умножении вектора на один элемент поля (1) он не изменяется, а умножение на другой (0) превращает его в единичный элемент векторного пространства, обозначаемый символом 0=(0 0…0).

Если в линейном пространстве последовательностей из n элементов поля GF(P) дополнительно задать операцию умножения векторов, удовлетворяющую определенным условиям (ассоциативности, замкнутости, билинейности по отношению к умножению на скаляр), то вся совокупность n-разрядных кодовых комбинаций превращается в линейную коммутативную алгебру над полем коэффициентов GF(P).

Подмножество элементов векторного пространства, которое удовлетворяет аксиомам векторного пространства, называют ***подпространством***.

***Линейным кодом*** называют множество векторов, образующих подпространства векторного пространства всех n-разрядных кодовых комбинаций над полем GF(P).

В случае двоичного кодирования такого подпространства комбинаций над полем GF(2) образует любая совокупность двоичных кодовых комбинаций, являющаяся подгруппой группы всех n-разрядных двоичных кодовых комбинаций. Поэтому любой двоичный линейный код является групповым.

### 4.8 Построение двоичного группового кода

Построение конкретного корректирующего кода производят, исходя из требуемого объема кода Q, т. е. необходимого числа передаваемых команд или дискретных значений измеряемой

величины и статистических данных о наиболее вероятных векторах ошибок в используемом канале связи.

*Вектором ошибки* называют n-разрядную двоичную последовательность, имеющую единицы в разрядах, подвергшихся искажению, и нули во всех остальных разрядах. Любую искаженную кодовую комбинацию можно рассматривать теперь как сумму (или разность) но модулю 2 исходной разрешенной кодовой комбинации и вектора ошибки.

Исходя из неравенства $2^k - 1 \geq Q$ (нулевая комбинация часто не используется, так как не меняет состояния канала связи), определяем число информационных разрядов *k,* необходимое для передачи заданного числа команд обычным двоичным кодом.

Каждой из $2^k$ - 1 ненулевых комбинаций *k* -разрядного безызбыточного кода нам необходимо поставить в соответствие комбинацию из *n* символов. Значения символов в $n - k$ проверочных разрядах такой комбинации устанавливаются в результате суммирования по модулю 2 значений символов в определенных информационных разрядах.

Поскольку множество $2^k$ комбинаций информационных символов (включая нулевую) образует подгруппу группы всех n-разрядных комбинаций, то и множество $2^k$ n-разрядных комбинаций, полученных по указанному правилу, тоже является подгруппой группы n-разрядных кодовых комбинаций. Это множество разрешенных кодовых комбинаций и будет групповым кодом.

Нам надлежит определить число проверочных разрядов и номера информационных разрядов, входящих в каждое из равенств для определения символов в проверочных разрядах.

Разложим группу $2^n$ всех n-разрядных комбинаций на смежные классы по подгруппе $2^k$ разрешенных n-разрядных кодовых комбинаций, проверочные разряды в которых еще не заполнены. Помимо самой подгруппы кода в разложении насчитывается $2^{n-k} - 1$ смежных классов. Элементы каждого класса представляют собой суммы по модулю 2 комбинаций кода и образующих элементов данного класса. Если за образующие элементы каждого класса принять те наиболее вероятные для заданного канала связи вектора ошибок, которые должны быть исправлены, то в каждом смежном классе сгруппируются кодовые комбинации, получающиеся в результате воздействия на все разрешенные комбинации определенного вектора ошибки. Для исправления любой полученной на выходе канала связи кодовой комбинации теперь достаточно определить, к какому классу смежности она относится. Складывая ее затем (по модулю 2) с образующим элементом этого смежного класса, получаем истинную комбинацию кода.

Ясно, что из общего числа $2^n - 1$ возможных ошибок групповой код может исправить всего $2^{n-k} - 1$ разновидностей ошибок по числу смежных классов.

Чтобы иметь возможность получить информацию о том, к какому смежному классу относится полученная комбинация, каждому смежному классу должна быть поставлена в

соответствие некоторая контрольная последовательность символов, называемая *опознавателем (синдромом).*

Каждый символ опознавателя определяют в результате проверки на приемной стороне справедливости одного из равенств, которые мы составим для определения значений проверочных символов при кодировании.

Ранее указывалось, что в двоичном линейном коде значения проверочных символов подбирают так, чтобы сумма по модулю 2 всех символов (включая проверочный), входящих в каждое из равенств, равнялась нулю. В таком случае число единиц среди этих символов четное. Поэтому операции определения символов опознавателя называют *проверками на четность.* При отсутствии ошибок в результате всех проверок на четность образуется опознаватель, состоящий из одних нулей. Если проверочное равенство не удовлетворяется, то в соответствующем разряде опознавателя появляется единица. Исправление ошибок возможно лишь при наличии взаимно однозначного соответствия между множеством опознавателей и множеством смежных классов, а следовательно, и множеством подлежащих исправлению векторов ошибок.

Таким образом, количество подлежащих исправлению ошибок является определяющим для выбора числа избыточных символов $n - k$ . Их должно быть достаточно для того, чтобы обеспечить необходимое число опознавателей. Если, например, необходимо исправить все одиночные независимые ошибки, то исправлению подлежат *n* ошибок:

000…01

000…10

…….

010…00

100…00

Различных ненулевых опознавателей должно быть не менее *n.* Необходимое число проверочных разрядов, следовательно, должно определяться из соотношения

$2^{n-k}-1 \geq n$ или $2^{n-k}-1 \geq C_n^1$

Если необходимо исправить не только все единичные, но и все двойные независимые ошибки, соответствующее неравенство принимает вид

$2^{n-k}-1 \geq C_n^1 + C_n^2$

В общем случае для исправления всех независимых ошибок кратности до s включительно получаем

$2^{n-k}-1 \geq C_n^1 + C_n^2 + \ldots + C_n^s$

Стоит подчеркнуть, что в приведенных соотношениях указывается теоретический предел минимально возможного числа проверочных символов, который далеко не во всех случаях

можно реализовать практически. Часто проверочных символов требуется больше, чем следует из соответствующего равенства.

Одна из причин этого выяснится при рассмотрении процесса сопоставления каждой подлежащей исправлению ошибки с ее опознавателем.

### 4.8.1 Составление таблицы опознавателей

Начнем для простоты с установления опознавателей для случая исправления одиночных ошибок. Допустим, что необходимо закодировать 15 команд. Тогда требуемое число информационных разрядов равно четырем. Пользуясь соотношением $2^{n-k} - 1 = n$, определяем общее число разрядов кода, а следовательно, и число ошибок, подлежащих исправлению (n = 7). Три избыточных разряда позволяют использовать в качестве опознавателей трехразрядные двоичные последовательности.

В данном случае ненулевые последовательности в принципе могут быть сопоставлены с подлежащими исправлению ошибками в любом порядке. Однако более целесообразно сопоставлять их с ошибками в разрядах, начиная с младшего, в порядке возрастания двоичных чисел (табл. 4.6).

Таблица 4.6.

| Векторы ошибок | Опознаватели | Векторы ошибок | Опознаватели |
|---|---|---|---|
| 0000001 | 001 | 0010000 | 101 |
| 0000010 | 010 | 0100000 | 110 |
| 0000100 | 011 | 1000000 | 111 |
| 0001000 | 100 | | |

При таком сопоставлении каждый опознаватель представляет собой двоичное число, указывающее номер разряда, в котором произошла ошибка.

Коды, в которых опознаватели устанавливаются по указанному принципу, известны как коды Хэмминга.

Возьмем теперь более сложный случай исправления одиночных и двойных независимых ошибок. В качестве опознавателей одиночных ошибок в первом и втором разрядах можно принять, как и ранее, комбинации 0...001 и 0...010.

Однако в качестве опознавателя одиночной ошибки в третьем разряде комбинацию 0...011 взять нельзя. Такая комбинация соответствует ошибке одновременно в первом и во втором разрядах, а она также подлежит исправлению и, следовательно, ей должен соответствовать свой опознаватель 0...011.

В качестве опознавателя одиночной ошибки в третьем разряде можно взять только трехразрядную комбинацию 0...0100, так как множество двухразрядных комбинаций уже исчерпано. Подлежащий исправлению вектор ошибки 0...0101 также можно рассматривать как результат суммарного воздействия двух векторов ошибок 0...0100 и 0...001 и, следовательно, ему должен быть поставлен в соответствие опознаватель, представляющий собой сумму по модулю 2 опознавателей этих ошибок, т.е. 0...0101.

Аналогично находим, что опознавателем вектора ошибки 0...0110 является комбинация 0...0110.

Определяя опознаватель для одиночной ошибки в четвертом разряде, замечаем, что еще не использована одна из трехразрядных комбинаций, а именно 0...0111. Однако, выбирая в качестве опознавателя единичной ошибки в i-м разряде комбинацию с числом разрядов, меньшим i, необходимо убедиться в том, что для всех остальных подлежащих исправлению векторов ошибок, имеющих единицы в i-м и более младших разрядах, получатся опознаватели, отличные от уже использованных. В нашем случае подлежащими исправлению векторами ошибок с единицами в четвертом и более младших разрядах являются: 0...01001, 0...01010, 0...01100.

Если одиночной ошибке в четвертом разряде поставить в соответствие опознаватель 0...0111, то для указанных векторов опознавателями должны были бы быть соответственно

| | | |
|---|---|---|
| $\oplus$ 0…0111 | $\oplus$ 0…0111 | $\oplus$ 0…0111 |
| 0…0001 | 0…0010 | 0…0100 |
| _____ | _____ | _____ |
| 0…0110 | 0…0101 | 0…0011 |

Однако эти комбинации уже использованы в качестве опознавателей других векторов ошибок, а именно: 0...0110, 0...0101, 0...0011.

Следовательно, во избежание неоднозначности при декодировании в качестве опознавателя одиночной ошибки в четвертом разряде следует взять четырехразрядную комбинацию 1000. Тогда для векторов ошибок

0...01001, 0...01010, 0...01100

опознавателями соответственно будут:

0...01001, 0...01010, 0...01100.

Аналогично можно установить, что в качестве опознавателя одиночной ошибки в пятом разряде может быть выбрана не использованная ранее четырехразрядная комбинация 01111.

Действительно, для всех остальных подлежащих исправлению векторов ошибок с единицей в пятом и более младших разрядах получаем опознаватели, отличающиеся от ранее установленных:

| Векторы ошибок | Опознаватели |
|---|---|
| 0...010001 | 0…01110 |
| 0...010010 | 0…01101 |
| 0...010100 | 0…01011 |
| 0...011000 | 0…00111 |

Продолжая сопоставление, можно получить таблицу опознавателей для векторов ошибок данного типа с любым числом разрядов. Так как опознаватели векторов ошибок с единицами в нескольких разрядах устанавливаются как суммы по модулю 2 опознавателей одиночных ошибок в этих разрядах, то для определения правил построения кода и составления проверочных равенств достаточно знать только опознаватели одиночных ошибок в каждом из разрядов. Для построения кодов, исправляющих двойные независимые ошибки, таблица таких опознавателей определена с помощью вычислительной машины вплоть до 29-го разряда [Теория кодирования. Сборник. – М.:Мир, 1964]. Опознаватели одиночных ошибок в первых пятнадцати разрядах приведены в табл. 4.7.

По тому же принципу аналогичные таблицы определены и для ошибок других типов, например для тройных независимых ошибок, пачек ошибок в два и три символа.

### 4.8.2 Определение проверочных равенств

Итак, для любого кода, имеющего целью исправлять наиболее вероятные векторы ошибок заданного канала связи (взаимно независимые ошибки или пачки ошибок), можно составить таблицу опознавателей одиночных ошибок в каждом из разрядов. Пользуясь этой таблицей, нетрудно определить, символы каких разрядов должны входить в каждую из проверок на четность.

Таблица 4.7.

| Номер разряда | Опознаватель | Номер разряда | Опознаватель | Номер разряда | Опознаватель |
|---|---|---|---|---|---|
| 1 | 00000001 | 6 | 00010000 | 11 | 01101010 |
| 2 | 00000010 | 7 | 00100000 | 12 | 10000000 |
| 3 | 00000100 | 8 | 00110011 | 13 | 10010110 |
| 4 | 00001000 | 9 | 01000000 | 14 | 10110101 |
| 5 | 00001111 | 10 | 01010101 | 15 | 11011011 |

Рассмотрим в качестве примера опознаватели для кодов, предназначенных исправлять единичные ошибки (табл. 4.8).

Таблица 4.8.

| Номер разрядов | Опознаватель | Номер разрядов | Опознаватель | Номер разрядов | Опознаватель |
|---|---|---|---|---|---|
| 1 | 0001 | 7 | 0111 | 12 | 1100 |
| 2 | 0010 | 8 | 1000 | 13 | 1101 |
| 3 | 0011 | 9 | 1001 | 14 | 1110 |
| 4 | 0100 | 10 | 1010 | 15 | 1111 |
| 5 | 0101 | 11 | 1011 | 16 | 10000 |
| 6 | 0110 | | | | |

В принципе можно построить код, усекая эту таблицу на любом уровне. Однако из таблицы видно, что оптимальными будут коды (7, 4), (15, 11), где первое число равно n, а второе k, и другие, которые среди кодов, имеющих одно и то же число проверочных символов, допускают наибольшее число информационных символов.

Усечем эту таблицу на седьмом разряде и найдем номера разрядов, символы которых должны войти в каждое из проверочных равенств.

Предположим, что в результате первой проверки на четность для младшего разряда опознавателя будет получена единица. Очевидно, это может быть следствием ошибки в одном из разрядов, опознаватели которых в младшем разряде имеют единицу. Следовательно, первое проверочное равенство должно включать символы 1, 3, 5 и 7-го разрядов: $a_1 \oplus a_3 \oplus a_5 \oplus a_7=0$.

Единица во втором разряде опознавателя может быть следствием ошибки в разрядах, опознаватели которых имеют единицу во втором разряде. Отсюда второе проверочное равенство должно иметь вид $a_2 \oplus a_3 \oplus a_6 \oplus a_7= 0$ .

Аналогично находим и третье равенство: $a_4 \oplus a_5 \oplus a_6 \oplus a_7= 0$ .

Чтобы эти равенства при отсутствии ошибок удовлетворялись для любых значений информационных символов в кодовой комбинации, в нашем распоряжении имеется три проверочных разряда. Мы должны так выбрать номера этих разрядов, чтобы каждый из них входил только в одно из равенств. Это обеспечит однозначное определение значений символов в проверочных разрядах при кодировании. Указанному условию удовлетворяют разряды, опознаватели которых имеют по одной единице. В нашем случае это будут первый, второй и четвертый разряды.

Таким образом, для кода (7, 4), исправляющего одиночные ошибки, искомые правила построения кода, т. е. соотношения, реализуемые в процессе кодирования, принимают вид:

$a_1=a_3 \oplus a_5 \oplus a_7$,

$a_2=a_3 \oplus a_6 \oplus a_7$, (4.1)

$a_4=a_5 \oplus a_6 \oplus a_7$ .

Поскольку построенный код имеет минимальное хэммингово расстояние $d_{min}$ = 3, он может использоваться с целью обнаружения единичных и двойных ошибок. Обращаясь к табл. 4.8, легко убедиться, что сумма любых двух опознавателей единичных ошибок дает ненулевой опознаватель, что и является признаком наличия ошибки.

Пример 27. Построим групповой код объемом 15 слов, способный исправлять единичные и обнаруживать двойные ошибки.

В соответствии с $d_{и\ 0\ min} \geq r+s+1$код должен обладать минимальным хэмминговым расстоянием, равным 4. Такой код можно построить в два этапа. Сначала строим код заданного объема, способный исправлять единичные ошибки. Это код Хэмминга (7, 4). Затем добавляем еще один проверочный разряд, который обеспечивает четность числа единиц в разрешенных комбинациях.

Таким образом, получаем код (8, 4). В процессе кодирования реализуются соотношения:

$a_1=a_3 \oplus a_5 \oplus a_7$,

$a_2=a_3 \oplus a_6 \oplus a_7$,

$a_4=a_5 \oplus a_6 \oplus a_7$ .

$a_8=a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7$,

Обозначив синдром кода (7, 4) через $S_1$, результат общей проверки на четность — через $S_2$($S_2 = \sum_{i=1}^{8} a_i$) и пренебрегая возможностью возникновения ошибок кратности 3 и выше, запишем алгоритм декодирования:

при $S_1$= 0 и $S_2$= 0 ошибок нет;

при $S_1$= 0 и $S_2$= 1 ошибка в восьмом разряде;

при $S_1 \neq 0$ и $S_2$= 0 двойная ошибка (коррекция блокируется, посылается запрос повторной передачи);

при $S_1 \neq 0$ и $S_2$= 1 одиночная ошибка (осуществляется ее исправление).

Пример 28. Используя табл. 4.8, составим правила построения кода (8,2), исправляющего все одиночные и двойные ошибки.

Усекая табл. 4.7 на восьмом разряде, найдем следующие проверочные равенства:

$a_1 \oplus a_5 \oplus a_8=0$,

$a_2 \oplus a_5 \oplus a_8=0$,

$a_3 \oplus a_5=0$,

$a_4 \oplus a_5=0$,

$a_6 \oplus a_8=0$,

$a_7 \oplus a_8=0$.

Соответственно правила построения кода выразим соотношениями

$a_1=a_5 \oplus a_8$ (4.2 а)

$a_2=a_5 \oplus a_8$ (4.2 б)

$a_3=a_5$ (4.2 в)

$a_4=a_5$ (4.2 г)

$a_6=a_8$ (4.2 д)

$a_7=a_8$ (4.2 е)

Отметим, что для построенного кода $d_{min} = 5$, и, следовательно, он может использоваться для обнаружения ошибок кратности от 1 до 4.

Соотношения, отражающие процессы кодирования и декодирования двоичных линейных кодов, могут быть реализованы непосредственно с использованием сумматоров по модулю два. Однако декодирующие устройства, построенные таким путем для кодов, предназначенных исправлять многократные ошибки, чрезвычайно громоздки. В этом случае более эффективны другие принципы декодирования.

### 4.8.3 Мажоритарное декодирование групповых кодов

Для линейных кодов, рассчитанных на исправление многократных ошибок, часто более простыми оказываются декодирующие устройства, построенные по мажоритарному принципу. Это метод декодирования называют также принципом голосования или способом декодирования по большинству проверок. В настоящее время известно значительное число кодов, допускающих мажоритарную схему декодирования, а также сформулированы некоторые подходы при конструировании таких кодов.

Мажоритарное декодирование тоже базируется на системе проверочных равенств. Система последовательно может быть разрешена относительно каждой из независимых переменных, причем в силу избыточности это можно сделать не единственным способом.

Любой символ $a_i$ выражается d (минимальное кодовое расстояние) различными независимыми способами в виде линейных комбинаций других символов. При этом может использоваться тривиальная проверка $a_i = a_i$. Результаты вычислений подаются на соответствующий этому символу мажоритарный элемент. Последний представляет собой схему, имеющую d входов и один выход, на котором появляется единица, когда возбуждается больше половины его входов, и нуль, когда возбуждается число таких входов меньше половины. Если ошибки отсутствуют, то проверочные равенства не нарушаются и на выходе мажоритарного элемента получаем истинное значение символа. Если число проверок $d \geq 2s+ 1$ и появление ошибки кратности s и менее не приводит к нарушению более s проверок, то правильное решение

может быть принято по большинству неискаженных проверок. Чтобы указанное условие выполнялось, любой другой символ $a_j (j \neq i)$ не должен входить более чем в одно проверочное равенство. В этом случае мы имеем дело с системой *разделенных проверок.*

Пример 29. Построим систему разделенных проверок для декодирования информационных символов рассмотренного ранее группового кода (8,2).

Поскольку код рассчитан на исправление любых единичных и двойных ошибок, число проверочных равенств для определения каждого символа должно быть не менее 5. Подставив в равенства (4.2 а) и (4.2 б) значения $a_8$, полученные из равенств (4.2 д) и (4.2 е) и записав их относительно $a_5$ совместно с равенствами (4.2 в) и (4.2 г) и тривиальным равенством $a_5 = a_5$, получим следующую систему разделенных проверок для символа $a_5$:

$a_5 = a_6 \oplus a_1$,

$a_5 = a_7 \oplus a_2$,

$a_5 = a_3$,

$a_5 = a_4$,

$a_5 = a_5$.

Для символа $a_8$ систему разделенных проверок строим аналогично:

$a_8 = a_3 \oplus a_1$,

$a_8 = a_4 \oplus a_2$,

$a_8 = a_6$,

$a_8 = a_7$,

$a_8 = a_8$.

### 4.8.4 Матричное представление линейных кодов

*Матрицей* размерности $l \times n$ называют упорядоченное множество $l \times n$ элементов, расположенных в виде прямоугольной таблицы с l строками и n столбцами:

$$A \equiv \begin{bmatrix} a_{11}a_{12}.....a_{1n} \\ a_{21}a_{22}....a_{2n} \\ ..... \\ a_{l1}a_{l2}.....a_{\ln} \end{bmatrix}.$$

*Транспонированной* матрицей к матрице **A** называют матрицу, строками которой являются столбцы, а столбцами строки матрицы **A**:

$$A^T \equiv \begin{bmatrix} a_{11}a_{12}.....a_{l1} \\ a_{21}a_{22}....a_{l2} \\ ..... \\ a_{1n}a_{2n}.....a_{\ln} \end{bmatrix}.$$

Матрицу размерности n×n называют *квадратной матрицей порядка n.* Квадратную матрицу, у которой по одной из диагоналей расположены только единицы, а все остальные элементы равны нулю, называют *единичной матрицей* **I**. Суммой двух матриц $\mathbf{A} \equiv |a_{ij}|$ и $\mathbf{B} \equiv |b_{ij}|$ размерности l×n называют матрицу размерности l×n:

$\mathbf{A} + \mathbf{B} \equiv |a_{ij}| + |b_{ij}| \equiv |a_{ij} + b_{ij}|$.

Умножение матрицы $\mathbf{A} \equiv |a_{ij}|$ размерности l×n на скаляр *c* дает матрицу размерности l×n: $c\mathbf{A} \equiv c\,|a_{ij}| \equiv |c\,a_{ij}|$.

Матрицы $\mathbf{A} \equiv |a_{ij}|$ размерности l×n и $\mathbf{B} \equiv |b_{jk}|$ размерности n×m могут быть перемножены, причем элементами $c_{ik}$ матрицы — произведения размерности l×m являются суммы произведений элементов l-й строки матрицы **A** на соответствующие элементы k-го столбца матрицы **B:**

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}.$$

В теории кодирования элементами матрицы являются элементы некоторого поля GF(P), а строки и столбцы матрицы рассматриваются как векторы. Сложение и умножение элементов матриц осуществляется по правилам поля GF(P).

Пример 30. Вычислим произведение матриц с элементами из поля GF (2):

$$M_1 = \begin{bmatrix} 011 \\ 100 \\ 001 \end{bmatrix} \quad M_2 = \begin{bmatrix} 111 \\ 010 \\ 100 \end{bmatrix}$$

Элементы $c_{ik}$ матрицы произведения $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$ будут равны:

$c_{11} = (011)\,(101) = 0 + 0 + 1 = 1$

$c_{12} = (011)\,(110) = 0 + 1 + 0 = 1$

$c_{13} = (011)\,(100) = 0 + 0 + 0 = 0$

$c_{21} = (100)\,(101) = 1 + 0 + 0 = 1$

$c_{22} = (100)\,(110) = 1 + 0 + 0 = 1$

$c_{23} = (100)\,(100) = 1 + 0 + 0 = 1$

$c_{31} = (001)\,(101) = 0 + 0 + 1 = 1$

$c_{32} = (001)\,(110) = 0 + 0 + 0 = 0$

$c_{33} = (001)\,(100) = 0 + 0 + 0 = 0$

Следовательно, $M = \begin{bmatrix} 110 \\ 111 \\ 100 \end{bmatrix}$

Зная закон построения кода, определим все множество разрешенных кодовых комбинаций. Расположив их друг под другом, получим матрицу, совокупность строк которой

является подпространством векторного пространства n-разрядных кодовых комбинаций (векторов) из элементов поля GF(P). В случае двоичного (n, k)-кода матрица насчитывает n столбцов и $2^k$—1 строк (исключая нулевую). Например, для рассмотренного ранее кода (8,2), исправляющего все одиночные и двойные ошибки, матрица имеет вид

$$\begin{bmatrix} a_5 a_8 a_1 a_2 a_3 a_4 a_6 a_7 \\ 0\ 1\ 1\ 1\ \ 0\ 0\ 1\ 1 \\ 1\ 0\ 1\ 1\ \ 1\ 1\ 0\ 0 \\ 1\ 1\ 0\ 0\ \ 1\ 1\ 1\ 1 \end{bmatrix}$$

При больших n и k матрица, включающая все векторы кода, слишком громоздка. Однако совокупность векторов, составляющих линейное пространство разрешенных кодовых комбинаций, является линейно зависимей, так как часть векторов может быть представлена в виде линейной комбинации некоторой ограниченной совокупности векторов, называемой *базисом пространства.*

Совокупность векторов $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3, ..., \mathbf{V}_n$ называют *линейно зависимой*, когда существуют скаляры $c_1,..c_n$ (не все равные нулю), при которых

$c_1\mathbf{V}_1 + c_2\mathbf{V}_2 + \ldots + c_n\mathbf{V}_n = \mathbf{0}$

В приведенной матрице, например, третья строка представляет собой суммы по модулю два первых двух строк. Для полного определения пространства разрешенных кодовых комбинаций линейного кода достаточно записать в виде матрицы только совокупность линейно независимых векторов. Их число называют *размерностью векторного пространства.*

Среди $2^k$ – 1 ненулевых двоичных кодовых комбинаций — векторов их только k. Например, для кода (8,2)

$$M_{8,2} = \begin{bmatrix} a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 \\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ \ 1 \\ 1\ 1\ 1\ \ 1\ 1\ 0\ 0\ \ 0 \end{bmatrix}$$

Матрицу, составленную из любой совокупности векторов линейного кода, образующей базис пространства, называют *порождающей (образующей)* матрицей кода.

Если порождающая матрица содержит k строк по n элементов поля GF(q), то код называют (n, k)-кодом. В каждой комбинации (n, k)-кода k информационных символов и n – k проверочных. Общее число разрешенных кодовых комбинаций (исключая нулевую) $Q = q^k - 1$.

Зная порождающую матрицу кода, легко найти разрешенную кодовую комбинацию, соответствующую любой последовательности $\mathbf{A}_{ki}$ из k информационных символов.

Она получается в результате умножения вектора $\mathbf{A}_{ki}$ на порождающую матрицу $\mathbf{M}_{n,k}$:

$\mathbf{A}_{ni} = \mathbf{A}_{ki} \bullet \mathbf{M}_{n,k}$ .

Найдем, например, разрешенную комбинацию кода (8,2), соответствующую информационным символам $a_5$=l, $a_8$ = 1:

$$M_{8,2} = |11| \begin{bmatrix} 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \end{bmatrix} = 0\;0\;1\;1\;1\;1\;1\;1 .$$

Пространство строк матрицы остается неизменным при выполнении следующих элементарных операций над строками: 1) перестановка любых двух строк; 2) умножение любой строки на ненулевой элемент поля; 3) сложение какой-либо строки с произведением другой строки на ненулевой элемент поля, а также при перестановке столбцов.

Если образующая матрица кода $\mathbf{M}_2$ получена из образующей матрицы кода $\mathbf{M}_1$ с помощью элементарных операций над строками, то обе матрицы порождают один и тот же код. Перестановка столбцов образующей матрицы кода приводит к образующей матрице *эквивалентного* кода. Эквивалентные коды весьма близки по своим свойствам. Корректирующая способность таких кодов одинакова.

Для анализа возможностей линейного (n, k)-кода, а также для упрощения процесса кодирования удобно, чтобы порождающая матрица ($\mathbf{M}_{n,k}$) состояла из двух матриц: единичной матрицы размерности k × k и дописываемой справа *матрицы-дополнения* (контрольной подматрицы) размерности k • (n – k), которая соответствует n – k проверочным разрядам:

$$M_{n,k} = \left[I_k P_{k,n-k}\right] = \begin{bmatrix} 1\;0\ldots0\,p_{1,k+1}p_{1,k+2}\ldots p_{1,j}\ldots p_{1,n} \\ 0\;1\ldots0\,p_{2,k+1}p_{2,k+2}\ldots p_{2,j}\ldots p_{2,n} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0\;1\ldots0\,p_{i,k+1}p_{i,k+2}\ldots p_{i,j}\ldots p_{i,n} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0\;0\ldots1\,p_{k,k+1}p_{k,k+2}\ldots p_{k,j}\ldots p_{k,n} \end{bmatrix} \quad (4.3)$$

Разрешенные кодовые комбинации кода с такой порождающей матрицей отличаются тем, что первые k символов в них совпадают с исходными информационными, а проверочными оказываются (n - k) последних символов. Действительно, если умножим вектор-строку $\mathbf{A}_{k,i}$ = = ($a_1$ $a_2$…$a_i$...$a_k$) на матрицу $\mathbf{M}_{n,k}$= [$\mathbf{I}_k\mathbf{P}_{k,n-k}$], получим вектор

$\mathbf{A}_{n,i}$ = ($a_1a_2$...$a_i$...$a_k$...$a_{k+1}$...$a_j$...$a_n$),

где проверочные символы $a_j$($k+1 \le j \le n$) являются линейными комбинациями информационных:

$$a_j = \sum_{i=1}^{k} a_i p_{ij} . \quad (4.4)$$

Коды, удовлетворяющие этому условию, называют *систематическими.* Для каждого линейного кода существует эквивалентный систематический код. Как следует из (4.3), (4.4), информацию о способе построения такого кода содержит матрица-дополнение. Если правила построения кода (уравнения кодирования) известны, то значения символов любой строки матрицы-дополнения получим, применяя эти правила к символам соответствующей строки единичной матрицы.

Пример 31. Запишем матрицы $\mathbf{I}_k$, $\mathbf{P}_{k,n-k}$ и $\mathbf{M}_{n,k}$ для двоичного кода (7,4).

Единичная матрица на четыре разряда имеет вид

$$I_4 = \begin{bmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \end{bmatrix}.$$

Один из вариантов матрицы дополнения можно записать, используя соотношения (4.1)

$$P_{4,3} = \begin{bmatrix} 110 \\ 101 \\ 011 \\ 111 \end{bmatrix}.$$

Тогда для двоичного кода Хэммнига имеем:

$$M_{7,4} = \begin{bmatrix} a_3 a_5 a_6 a_7 a_1 a_2 a_4 \\ 1\,0\,0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,1\,0\,1 \\ 0\,0\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1 \end{bmatrix}$$

Запишем также матрицу для систематического кода (7,4):

$$M_{7,4} = \begin{bmatrix} a_1 a_2 a_3 a_4 a_5 a_6 a_7 \\ 1\,0\,0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,1\,0\,1 \\ 0\,0\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1\,1 \end{bmatrix}$$

В свою очередь, по заданной матрице-дополнению $\mathbf{P}_{k,n-k}$ можно определить равенства, задающие правила построения кода. Единица в первой строке каждого столбца указывает на то, что в образовании соответствующего столбцу проверочного разряда участвовал первый информационный разряд. Единица в следующей строке любого столбца говорит об участии в образовании проверочного разряда второго информационного разряда и т, д.

Так как матрица-дополнение содержит всю информацию о правилах построения кода, то систематический код с заданными свойствами можно синтезировать путем построения соответствующей матрицы-дополнения.

Так как минимальное кодовое расстояние d для линейного кода равно минимальному весу его ненулевых векторов, то в матрицу-дополнение должны быть включены такие k строк, которые удовлетворяли бы следующему общему условию: вектор-строка образующей матрицы, получающаяся при суммировании любых l ($1 \le l \le k$) строк, должна содержать не менее d – l отличных от нуля символов.

Действительно, при выполнении указанного условия любая разрешенная кодовая комбинация, полученная суммированием l строк образующей матрицы, имеет не менее d ненулевых символов, так как l ненулевых символов она всегда содержит в результате суммирования строк единичной матрицы. Синтезируем таким путем образующую матрицу двоичного систематического кода (7,4) с минимальным кодовым расстоянием d = 3. В каждой вектор-строке матрицы-дополнения согласно сформулированному условию (при l=1) должно быть не менее двух единиц. Среди трехразрядных векторов таких имеется четыре: 011, 110, 101, 111.

Эти векторы могут быть сопоставлены со строками единичной матрицы в любом порядке. В результате получим матрицы систематических кодов, эквивалентных коду Хэмминга, например:

$$M_{7,4} = \begin{bmatrix} 1\,0\,0\,0 & 0\,1\,1 \\ 0\,1\,0\,0 & 1\,1\,1 \\ 0\,0\,1\,0 & 1\,0\,1 \\ 0\,0\,0\,1 & 1\,1\,0 \end{bmatrix}$$

Нетрудно убедиться, что при суммировании нескольких строк такой матрицы (l>1) получим вектор-строку, содержащую не менее d=3 ненулевых символов.

Имея образующую матрицу систематического кода $\mathbf{M}_{n,k} = [\mathbf{I}_k \mathbf{P}_{k,n-k}]$, можно построить так называемую проверочную (контрольную) матрицу **H** размерности $(n - k) \times n$:

$$H = \left[ -P^{T}{}_{k,n-k} I_{n-k} \right] = \begin{bmatrix} p_{1,k+1} p_{2,k+1} \cdots p_{k,k+1} & -1 & 0 \ldots\ldots 0 \\ p_{1,k+2} p_{2,k+2} \cdots p_{k,k+2} & 0 & -1 \ldots\ldots 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ p_{1,n} \quad p_{2,n} \ldots\ldots p_{k,n} & 0 & 0 \ldots -1 \end{bmatrix}$$

При умножении неискаженного кодового вектора $\mathbf{A}_{ni}$ на матрицу, транспонированную к матрице **H**, получим вектор, все компоненты которого равны нулю:

$$A_{n,i}H^T = \left|a_1, a_2, ..., a_{k,} a_{k+1,} ..., a_j, ..., a_n\right| \cdot \begin{bmatrix} p_{1,k+1} \cdots p_{1,j} \cdots p_{1,n} \\ p_{2,k+1} \cdots p_{2,j} \cdots p_{2,n} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ p_{k,k+1} \cdots p_{k,j} \cdots p_{k,n} \\ -1 \ldots\ldots 0 \ldots\ldots 0 \\ 0 \ldots\ldots -1 \ldots\ldots 0 \\ 0 \ldots\ldots 0 \ldots\ldots -1 \end{bmatrix} =$$

$$= \left|S_{k+1}, S_{k+2}, ..., S_j, ...S_n\right| = \left|0,0,..,0,..,0\right|.$$

Каждая компонента $S_j$ является результатом проверки справедливости соответствующего уравнения декодирования:

$$S_j = \sum_{i=1}^{k} a_i P_{ij} - a_j = 0.$$

В общем случае, когда кодовый вектор $\mathbf{A}_{ni} = (a_1, a_2, \ldots, a_i, \ldots, a_k, a_{k+1}, \ldots, a_j, \ldots, a_n)$ искажен вектором ошибки $\xi_{ni} = (\xi_1, \xi_2, \ldots, \xi_i, \ldots, \xi_k, a\xi_{k+1}, \ldots, \xi_j, \ldots, \xi_n)$, умножение вектора $(\mathbf{A}_{ni} + \xi_{ni})$ на матрицу $\mathbf{H}^T$ дает ненулевые компоненты:

$$S_j = \sum_{i=1}^{k} \xi_i P_{ij} - \xi_j = 0.$$

Отсюда видно, что $S_j (k+1 \le j \le n)$ представляют собой символы, зависящие только от вектора ошибки, а вектор $\mathbf{S} = (S_{k+1}, S_{k+2}, ..., S_j, ..., S_n)$ является не чем иным, как опознавателем ошибки (синдромом).

Для двоичных кодов (операция сложения тождественна операции вычитания) проверочная матрица имеет вид

$$H = \left[P^T{}_{k,n-k} I_{n-k}\right] = \begin{bmatrix} p_{1,k+1} p_{2,k+1} \cdots p_{k,k+1} & 1\ 0 \ldots\ldots 0 \\ p_{1,k+2} p_{2,k+2} \cdots p_{k,k+2} & 0\ 1 \ldots\ldots 0 \\ \cdots\cdots\cdots\cdots\cdots\cdots & \\ p_{1,n} \quad p_{2,n} \ldots\ldots p_{k,n} & 0\ 0 \ldots . 1 \end{bmatrix}$$

Пример 32. Найдем проверочную матрицу **H** для кода (7,4) с образующей матрицей **M**:

$$M_{7,4} = \begin{bmatrix} 1\ 0\ 0\ 0\ 1\ 1\ 0 \\ 0\ 1\ 0\ 0\ 1\ 0\ 1 \\ 0\ 0\ 1\ 0\ 0\ 1\ 1 \\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \end{bmatrix}$$

Определим синдромы в случаях отсутствия и наличия ошибки в кодовом векторе 1100011.

Выполним транспонирование матрицы $\mathbf{P}_{4,3}$

$$P_{4,3} = \begin{bmatrix} 1101 \\ 1011 \\ 0111 \end{bmatrix}.$$

Запишем проверочную матрицу:

$$H = \begin{bmatrix} 1101100 \\ 1011010 \\ 0111001 \end{bmatrix}.$$

Умножение на $\mathbf{H}^{\mathrm{T}}$ неискаженного кодового вектора 1100011 дает нулевой синдром:

$$[1100011] \begin{bmatrix} 110 \\ 101 \\ 011 \\ 111 \\ 100 \\ 010 \\ 001 \end{bmatrix} = [000].$$

При наличии в кодовом векторе ошибки, например, в 4-м разряде (1101011) получим:

$$[1101011] \begin{bmatrix} 110 \\ 101 \\ 011 \\ 111 \\ 100 \\ 010 \\ 001 \end{bmatrix} = [111].$$

Следовательно, вектор-строка 111 в данном коде является опознавателем (синдромом) ошибки в четвертом разряде. Аналогично можно найти и синдромы других ошибок. Множество всех опознавателей идентично множеству опознавателей кода Хэмминга (7,4), но сопоставлены они конкретным векторам ошибок по-иному, в соответствии с образующей матрицей данного (эквивалентного) кода.

### 4.8.5 Технические средства кодирования и декодирования для групповых кодов

Кодирующее устройство строится на основании совокупности равенств, отражающих правила построения кода. Определение значений символов в каждом из n – k проверочных разрядов в кодирующем устройстве осуществляется посредством сумматоров по модулю два.

На каждый разряд сумматора (кроме первого) используется четыре элемента **И** (вентиля) и два элемента **ИЛИ**.

Сумматор по модулю два выпускают в виде отдельного логического элемента, который на схеме изображается прямоугольником с надписью внутри - **М2**. Приведем несколько примеров реализации кодирующих и декодирующих устройств групповых кодов.

Пример 33. Рассмотрим техническую реализацию кода (7,4), имеющего целью исправление одиночных ошибок.

Правила построения кода определяются равенствами

$a_1=a_3 \oplus a_5 \oplus a_7$,

$a_2=a_3 \oplus a_6 \oplus a_7$,

$a_4=a_5 \oplus a_6 \oplus a_7$.

Схема кодирующего устройства приведена на рис. 4.6.

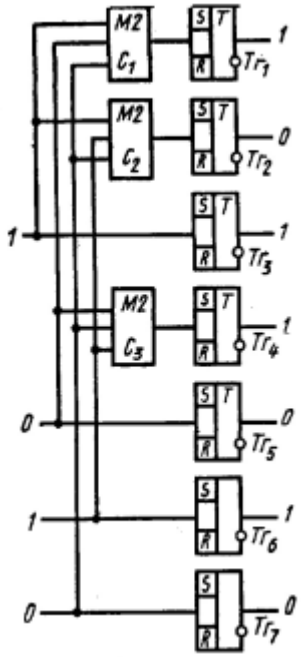



Рис. 4.6.

При поступлении импульса синхронизации со схемы управления подлежащая кодированию k-разрядная комбинация неизбыточного кода переписывается, например, с аналого-кодового преобразователя в информационные разряды n-разрядного регистра. Предположим, что в результате этой операции триггеры регистра установились в состояния, указанные в табл. 4.9.

С некоторой задержкой формируются выходные импульсы сумматоров $C_1$, $C_2$ $C_3$, которые устанавливают триггеры проверочных разрядов в положение 0 или 1 в соответствии с приведенными выше равенствами. Например, в нашем случае ко входам сумматора $C_1$ подводится информация, записанная в 3, 5 и 7-разрядах и, следовательно, триггер $Тг_1$ первого проверочного разряда устанавливается в положение 1, аналогично триггер $Тг_2$ устанавливается в положение 0, а триггер $Тг_4$ — в положение 1.

Сформированная в регистре разрешенная комбинация (табл. 4.10) импульсом, поступающим с блока управления, последовательно или параллельно считывается в линию связи. Далее начинается кодирование следующей комбинации.

Рассмотрим теперь схему декодирования и коррекции ошибок (рис. 4.7), строящуюся на основе совокупности проверочных равенств. Для кода (7, 4) они имеют вид:

$a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 0$ ,

$a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 0$,

$a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 0$ .

Таблица 4.9.

| $Tr_1$ | $Tr_2$ | $Tr_3$ | $Tr_4$ | $Tr_5$ | $Tr_6$ | $Tr_7$ |
|---|---|---|---|---|---|---|
| — | — | 1 | — | 0 | 1 | 0 |

Кодовая комбинация, возможно содержащая ошибку, поступает на n-разрядный приемный регистр (на рис. 4.7 триггеры $Tr_1$ –$Tr_7$). По окончании переходного процесса в триггерах с блока управления на каждый из сумматоров ($C_1$ – $C_3$) поступает импульс опроса.

Таблица 4.10.

| $Tr_1$ | $Tr_2$ | $Tr_3$ | $Tr_4$ | $Tr_5$ | $Tr_6$ | $Tr_7$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |

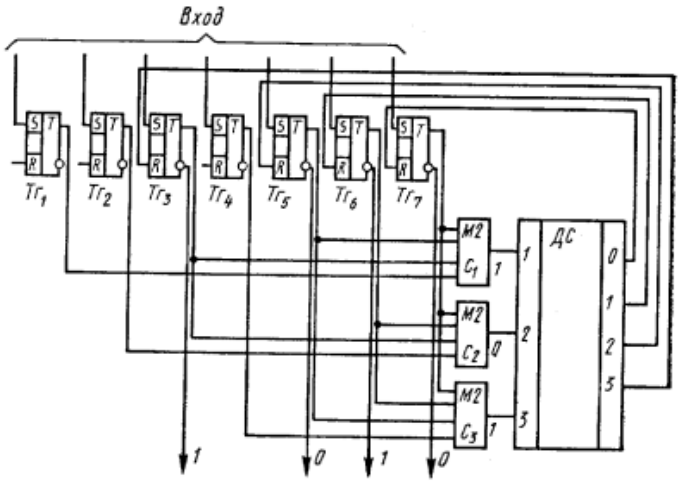



Рис. 4.7.

Выходные импульсы сумматоров устанавливают в положение 0 или 1 триггеры регистра опознавателей. Если проверочные равенства выполняются, все триггеры регистра опознавателей устанавливаются в положение 0, что соответствует отсутствию ошибок. При наличии ошибки в регистр опознавателей запишется опознаватель этого вектора ошибки. Дешифратор ошибки **DC** ставит в соответствие множеству опознавателей множество векторов ошибок. При опросе выходных вентилей дешифратора сигналы коррекции поступают только на те разряды, в которых вектор ошибки, соответствующий записанному на входе опознавателю, имеет единицы. Сигналы

коррекции воздействуют на счетные входы триггеров. Последние изменяют свое состояние, и, таким образом, ошибка исправлена. На триггеры проверочных разрядов импульсы коррекции можно не посылать, если после коррекции информация списывается только с информационных разрядов. Для кода Хэмминга (7,4) любой опознаватель представляет собой двоичное трехразрядное число, равное номеру разряда приемного регистра, в котором записан ошибочный символ.

Предположим, что сформированная ранее в кодирующем устройстве комбинация при передаче исказилась и на приемном регистре была зафиксирована в виде, записанном в табл. 4.11.

Таблица 4.11.

| $Тг_1$ | $Тг_2$ | $Тг_3$ | $Тг_4$ | $Тг_5$ | $Тг_6$ | $Тг_7$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |

По результатам опроса сумматоров получаем:

на выходе $C_1$ $a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 1+1+1+0=1$,

на выходе $C_2$ $a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 0+1+1+0=0$,

на выходе $C_3$ $a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 1+1+1+0=1$ .

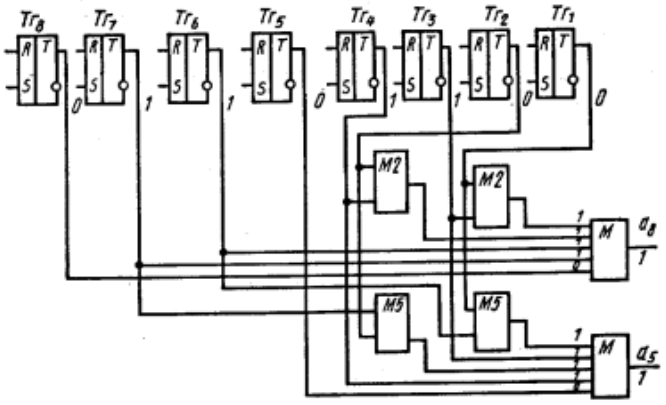



Рис. 4.8.

Следовательно, номер разряда, в котором произошло искажение, 101 или 5. Импульс коррекции поступит на счетный вход триггера $Тг_5$, и ошибка будет исправлена.

Пример 34. Реализуем мажоритарное декодирование для группового кода (8,2), рассчитанного на исправление двойных ошибок.

В случае мажоритарного декодирования сигналы с триггеров приемного регистра поступают непосредственно или после сложения по модулю два в соответствии с уравнениями системы разделённых проверок на мажоритарные элементы *M*, формирующие скорректированные информационные символы.

Схема декодирования представлена на рис. 4.8. На входах мажоритарных элементов указаны сигналы, соответствующие случаю поступления из канала связи кодовой информации,

искаженной в обоих информационных разрядах (5-м и 8-м). Реализуя принцип решения «по большинству», мажоритарные элементы восстанавливают на выходе правильные значения информационных символов.

## 4.9 Построение циклических кодов

### 4.9.1 Общие понятия и определения

Любой групповой код *(n, k)* может быть записан в виде матрицы, включающей *k* линейно независимых строк по *n* символов и, наоборот, любая совокупность *k* линейно независимых n-разрядных кодовых комбинаций может рассматриваться как образующая матрица некоторого группового кода. Среди всего многообразия таких кодов можно выделить коды, у которых строки образующих матриц связаны дополнительным условием цикличности.

Все строки образующей матрицы такого кода могут быть получены циклическим сдвигом одной комбинации, называемой образующей для данного кода. Коды, удовлетворяющие этому условию, получили название *циклических кодов.*

Сдвиг осуществляется справа налево, причем крайний левый символ каждый раз переносится в конец комбинации. Запишем, например, совокупность кодовых комбинаций, получающихся циклическим сдвигом комбинации 001011:

$$G = \begin{bmatrix} 0\,0\,1\,0\,1\,1 \\ 0\,1\,0\,1\,1\,0 \\ 1\,0\,1\,1\,0\,0 \\ 0\,1\,1\,0\,0\,1 \\ 1\,1\,0\,0\,1\,0 \\ 1\,0\,0\,1\,0\,1 \end{bmatrix}.$$

Число возможных циклических (n, k)-кодов значительно меньше числа различных групповых (n, k)-кодов.

При описании циклических кодов n-разрядные кодовые комбинации представляются в виде многочленов фиктивной переменной *x.* Показатели степени у *x* соответствуют номерам разрядов (начиная с нулевого), а коэффициентами при *x* в общем случае являются элементы поля *GF(q).* При этом наименьшему разряду числа соответствует фиктивная переменная $x^0 = 1$. Многочлен с коэффициентами из поля *GF(q)* называют многочленом над полем *GF(q).* Так как мы ограничиваемся рассмотрением только двоичных кодов, то коэффициентами при *x* будут только цифры 0 и 1. Иначе говоря, будем оперировать с многочленами над полем *GF(2).* Запишем, например, в виде многочлена образующую кодовую комбинацию 01011:

$G(x) = 0 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 1$

Поскольку члены с нулевыми коэффициентами при записи многочлена опускаются, образующий многочлен:

$G(x) = x^3 + x + 1$

Наибольшую степень $x$ в слагаемом с ненулевым коэффициентом называют *степенью многочлена.* Теперь действия над кодовыми комбинациями сводятся к действиям над многочленами. Суммирование многочленов осуществляется с приведением коэффициентов по модулю два.

Указанный циклический сдвиг некоторого образующего многочлена степени n - k без переноса единицы в конец кодовой комбинации соответствует простому умножению на $x$. Умножив, например, первую строку матрицы (001011), соответствующую многочлену $g_0(x) = x^3 + x + 1$, на $x$, получим вторую строку матрицы (010110), соответствующую многочлену $x \cdot g_0(x)$.

Нетрудно убедиться, что кодовая комбинация, получающаяся при сложении этих двух комбинаций, также будет соответствовать результату умножения многочлена $x^3 + x + 1$ на многочлен x+1. Действительно,

$$\begin{array}{r} 001011 \\ \oplus\ 010110 \\ \hline 011101 \end{array} \qquad \begin{array}{r} x^3+0+x+1 \\ x+1 \\ \hline x^3+0+x+1 \\ \oplus\ x^4+0+x^2+x \\ \hline x^4+x^3+x^2+0+1 \end{array}$$

Циклический сдвиг строки матрицы с единицей в старшем (n-м) разряде (слева) равносилен умножению соответствующего строке многочлена на $x$ с одновременным вычитанием из результата многочлена $x^n + 1 = x^n - 1$, т. е. с приведением по модулю $x^n + 1$.

Отсюда ясно, что любая разрешенная кодовая комбинация циклического кода может быть получена в результате умножения образующего многочлена на некоторый другой многочлен с приведением результата по модулю $x^n + 1$. Иными словами, при соответствующем выборе образующего многочлена любой многочлен циклического кода будет делиться на него без остатка.

Ни один многочлен, соответствующий запрещенной кодовой комбинации, на образующий многочлен без остатка не делится. Это свойство позволяет обнаружить ошибку. По виду остатка можно определить и вектор ошибки.

Умножение и деление многочленов весьма просто осуществляется на регистрах сдвига с обратными связями, что и явилось причиной широкого применения циклических кодов.

### 4.9.2 Математическое введение к циклическим кодам

Так как каждая разрешенная комбинация n-разрядного циклического кода есть произведение двух многочленов, один из которых является образующим, то эти комбинации можно рассматривать как подмножества всех произведений многочленов степени не выше n-1. Это наталкивает на мысль использовать для построения этих кодов еще одну ветвь теории алгебраических систем, а именно теорию колец.

Как следует из приведенного ранее определения, для образования кольца на множестве n-разрядных кодовых комбинаций необходимо задать две операции: сложение и умножение.

Операция сложения многочленов уже выбрана нами с приведением коэффициентов по модулю два.

Определим теперь операцию умножения. Нетрудно видеть, что операция умножения многочленов по обычным правилам с приведением подобных членов по модулю два может привести к нарушению условия замкнутости. Действительно, в результате умножения могут быть получены многочлены более высокой степени, чем n-1, вплоть до 2(n-1), а соответствующие им кодовые комбинации будут иметь число разрядов, превышающее n и, следовательно, не относятся к рассматриваемому множеству. Поэтому операция символического умножения задается так:

1) многочлены перемножаются по обычным правилам, но с приведением подобных членов по модулю два;

2) если старшая степень произведения не превышает n-1, то оно и является результатом символического умножения;

3) если старшая степень произведения больше или равна n, то многочлен произведения делится на заранее определенный многочлен степени n и результатом символического умножения считается остаток от деления.

Степень остатка не превышает n-1, и, следовательно, этот многочлен принадлежит к рассматриваемому множеству k-разрядных кодовых комбинаций.

При анализе циклического сдвига с перенесением единицы в конец кодовой комбинации установлено, что таким многочленом n-й степени является многочлен $x^n + 1$.

Действительно, в результате умножения многочлена степени n-1 на *x* получим

$$G(x) = (x^{n-1} + x^{n-2} + \ldots + x + 1)x = x^n + x^{n-1} + \ldots + x$$

Следовательно, чтобы результат умножения и теперь соответствовал кодовой комбинации, образующейся путем циклического сдвига исходной кодовой комбинации, в нем необходимо заменить $x^n$ на 1. Такая замена эквивалентна делению полученного при умножении многочлена на $x^n + 1$ с записью в качестве результата остатка от деления, что обычно называют *взятием остатка* или *приведением по модулю* $x^n + 1$ (сам остаток при этом называют вычетом).

Выделим теперь в нашем кольце подмножество всех многочленов, кратных некоторому многочлену *g(x)*. Такое подмножество называют *идеалом,* а *многочлен g(x)-порождающим многочленом идеала.*

Количество различных элементов в идеале определяется видом его порождающего многочлена. Если на порождающий многочлен взять 0, то весь идеал будет составлять только этот многочлен, так как умножение его на любой другой многочлен дает 0.

Если за порождающий многочлен принять 1[g(x) = 1], то в идеал войдут все многочлены кольца. В общем случае число элементов идеала, порожденного простым многочленом степени n-k, составляет $2^k$.

Теперь становится понятным, что циклический двоичный код в построенном нами кольце n-разрядных двоичных кодовых комбинаций является идеалом. Остается выяснить, как выбрать многочлен *g(x),* способный породить циклический код с заданными свойствами.

### 4.9.3 Требования, предъявляемые к образующему многочлену

Согласно определению циклического кода все многочлены, соответствующие его кодовым комбинациям, должны делиться на *g(x)* без остатка. Для этого достаточно, чтобы на *g(x)* делились без остатка многочлены, составляющие образующую матрицу кода. Последние получаются циклическим сдвигом, что соответствует последовательному умножению *g(x)* на *x* с приведением по модулю $x^n + 1$.

Следовательно, в общем случае многочлен $g_i(x)$ является остатком от деления произведения $g(x) \cdot x^i$ на многочлен $x^n + 1$ и может быть записан так:

$g_i(x)=g(x)x^i + c(x^n + 1)$

где c =1, если степень $g(x)\ x^i$ превышает $n$-1; $c = 0$, если степень $g(x)\ x^i$ не превышает $n$-1.

Отсюда следует, что все многочлены матрицы, а поэтому и все многочлены кода будут делиться на *g(x)* без остатка только в том случае, если на *g(x)* будет делиться без остатка многочлен $x^n + 1$.

Таким образом, чтобы *g(x)* мог породить идеал, а, следовательно, и циклический код, он должен быть делителем многочлена $x^n + 1$.

Поскольку для кольца справедливы все свойства группы, а для идеала - все свойства подгруппы, кольцо можно разложить на смежные классы, называемые в этом случае *классами вычетов по идеалу.*

Первую строку разложения образует идеал, причем нулевой элемент располагается крайним слева. В качестве образующего первого класса вычетов можно выбрать любой многочлен, не принадлежащий идеалу. Остальные элементы данного класса вычетов образуются путем суммирования образующего многочлена с каждым многочленом идеала.

Если многочлен *g(x)* степени m = n-k является делителем $x^n + 1$, то любой элемент кольца либо делится на *g(x)* без остатка (тогда он является элементом идеала), либо в результате деления появляется остаток *r(x),* представляющий собой многочлен степени не выше m-1.

Элементы кольца, дающие в остатке один и тот же многочлен $r_i(x)$, относятся к одному классу вычетов. Приняв многочлены $r(x)$ за образующие элементы классов вычетов, разложение кольца по идеалу с образующим многочленом $g(x)$ степени $m$ можно представить табл. 4.12, где $f(x)$-произвольный многочлен степени не выше n-m-1.

Таблица 4.12.

| | | | | | |
|---|---|---|---|---|---|
| 0 | g(x) | x·g(x) | (x+1)·g(x) | … | f(x)·g(x) |
| $r_1(x)$ | $g(x) + r_1(x)$ | $x·g(x) + r_1(x)$ | $(x+1)·g(x) + r_1(x)$ | … | $f(x)·g(x) + r_1(x)$ |
| $r_2(x)$ | $g(x) + r_2(x)$ | $x·g(x) + r_2(x)$ | $(x+1)·g(x) + r_2(x)$ | … | $f(x)·g(x) + r_2(x)$ |
| … | … | … | … | … | … |
| … | … | … | … | … | … |
| $r_n(x)$ | $g(x) + r_n(x)$ | $x·g(x) + r_n(x)$ | $(x+1)·g(x) + r_n(x)$ | … | $f(x)·g(x) + r_n(x)$ |

Как отмечалось, групповой код способен исправить столько разновидностей ошибок, сколько различных классов насчитывается в приведенном разложении. Следовательно, корректирующая способность циклического кода будет тем выше, чем больше остатков может быть образовано при делении многочлена, соответствующего искаженной кодовой комбинации, на образующий многочлен кода.

Наибольшее число остатков, равное $2^m - 1$ (исключая нулевой), может обеспечить только неприводимый (простой) многочлен, который делится сам на себя и не делится ни на какой другой многочлен (кроме 1).

## 4.10 Выбор образующего многочлена по заданному объему кода и заданной корректирующей способности

По заданному объему кода однозначно определяется число информационных разрядов $k$. Далее необходимо найти наименьшее $n$, обеспечивающее обнаружение или исправление ошибок заданной кратности. В случае циклического кода эта проблема сводится к нахождению нужного многочлена $g(x)$.

Начнем рассмотрение с простейшего циклического кода, обнаруживающего все одиночные ошибки.

### 4.10.1 Обнаружение одиночных ошибок

Любая принятая по каналу связи кодовая комбинация $h(x)$, возможно содержащая ошибку, может быть представлена в виде суммы по модулю два неискаженной комбинации кода $f(x)$ и вектора ошибки $\xi(x)$:

$h(x) = f(x) \oplus \xi(x)$

При делении *h(x)* на образующий многочлен *g(x)* остаток, указывающий на наличие ошибки, обнаруживается только в том случае, если многочлен, соответствующий вектору ошибки, не делится на *g(x): f(x)*-неискаженная комбинация кода и, следовательно, на *g(x)* делится без остатка.

Вектор одиночной ошибки имеет единицу в искаженном разряде и нули во всех остальных разрядах. Ему соответствует многочлен $\xi(x) = x^i$. Последний не должен делиться на *g(x)*. Среди неприводимых многочленов, входящих в разложении $x^n+1$, многочленом наименьшей степени, удовлетворяющим указанному условию, является x + 1. Остаток от деления любого многочлена на x + 1 представляет собой многочлен нулевой степени и может принимать только два значения: 0 или 1. Все кольцо в данном случае состоит из идеала, содержащего многочлены с четным числом членов, и одного класса вычетов, соответствующего единственному остатку, равному 1. Таким образом, при любом числе информационных разрядов необходим только один проверочный разряд. Значение символа этого разряда как раз и обеспечивает четность числа единиц в любой разрешенной кодовой комбинации, а, следовательно, и делимость ее на $x^n + 1$.

Полученный циклический код с проверкой на четность способен обнаруживать не только одиночные ошибки в отдельных разрядах, но и ошибки в любом нечетном числе разрядов.

### 4.10.2 Исправление одиночных или обнаружение двойных ошибок

Прежде чем исправить одиночную ошибку в принятой комбинации из *n* разрядов, необходимо определить, какой из разрядов был искажен. Это можно сделать только в том случае, если каждой одиночной ошибке в определенном разряде соответствуют свой класс вычетов и свой опознаватель. Так как в циклическом коде опознавателями ошибок являются остатки от деления многочленов ошибок на образующий многочлен кода *g(x),* то *g(x)* должно обеспечить требуемое число различных остатков при делении векторов ошибок с единицей в искаженном разряде. Как отмечалось, наибольшее число остатков дает неприводимый многочлен. При степени многочлена m = n-k он может дать $2^{n-k} - 1$ ненулевых остатков (нулевой остаток является опознавателем безошибочной передачи).

Следовательно, необходимым условием исправления любой одиночной ошибки является выполнение неравенства

$$2^{n-k} - 1 \geq C_n^1 = n,$$

где $C_n^1$ - общее число разновидностей одиночных ошибок в кодовой комбинации из *n* символов; отсюда находим степень образующего многочлена кода

$$m = n - k \geq \log_2(n+1)$$

и общее число символов в кодовой комбинации. Наибольшие значения *k* и *n* для различных m можно найти пользуясь табл. 4.13.

Таблица 4.13.

| M | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| N | 1 | 3 | 7 | 15 | 31 | 63 | 127 | 255 | 511 | 1023 |
| K | 0 | 1 | 4 | 11 | 26 | 57 | 120 | 247 | 502 | 1013 |

Как указывалось, образующий многочлен *g(x)* должен быть делителем двучлена $x^n+1$. Доказано, что любой двучлен типа $x^{2m-1}+ 1 = x^n+1$ может быть представлен произведением всех неприводимых многочленов, степени которых являются делителями числа *m* (от 1 до *m* включительно). Следовательно, для любого *m* существует по крайней мере один неприводимый многочлен степени *m,* входящий сомножителем в разложение двучлена $x^n+1$.

Пользуясь этим свойством, а также имеющимися в ряде книг таблицами многочленов, неприводимых при двоичных коэффициентах, выбрать образующий многочлен при известных n и m несложно. Определив образующий многочлен, необходимо убедиться в том, что он обеспечивает заданное число остатков.

Пример 35. Выберем образующий многочлен для случая n = 15 и m = 4.

Двучлен $x^{15} + 1$ можно записать в виде произведения всех неприводимых многочленов, степени которых являются делителями числа 4. Последнее делится на 1, 2, 4.

В таблице неприводимых многочленов находим один многочлен первой степени, а именно x+1, один многочлен второй степени $x^2 + x + 1$ и три многочлена четвертой степени: $x^4 + x + 1$, $x^4 + x^3 + 1$, $x^4 + x^3 + x^2 + x + 1$. Перемножив все многочлены, убедимся в справедливости соотношения $(x + 1)(x^2 + x + 1)(x^4 + x + 1)(x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1) = x^{15} + 1$

Один из сомножителей четвертой степени может быть принят за образующий многочлен кода. Возьмем, например, многочлен $x^4 + x^3 + 1$, или в виде двоичной последовательности 11001.

Чтобы убедиться, что каждому вектору ошибки соответствует отличный от других остаток, необходимо поделить каждый из этих векторов на 11001.Векторы ошибок m младших разрядов имеют вид: 00…000, 00…0010, 00…0100, 00…1000. Степени соответствующих им многочленов меньше степени образующего многочлена g(x). Поэтому они сами являются остатками при нулевой целой части. Остаток, соответствующий вектору ошибки в следующем старшем разряде, получаем при делении 00...10000 на 11001, т.е.

$$\begin{array}{l} 10000 \,\big|\, 11001 \\ \oplus\ 11001 \\ \hline 1001 \end{array}$$

Аналогично могут быть найдены и остальные остатки. Однако их можно получить проще, деля на g(x) комбинацию в виде единицы с рядом нулей и выписывая все промежуточные остатки:

```
                                        Остатки
⊕ 10000000000...|11001
  11001                                 1001
  10010
⊕ 11001                                 1011
  10110
⊕ 11001                                 1111
  11110
⊕ 11001                                 0111
  011100
⊕ 11001                                 1110
  010100
⊕ 11001                                 0101
  11010
⊕ 11001                                 1010
  0011000
⊕ 11001                                 1101
  0001000                               0011
                                        0110
                                        1100
                                        0001
                                        0010
                                        0100
                                        1000
```

При последующем делении остатки повторяются.

Таким образом, мы убедились в том, что число различных остатков при выбранном g(x) равно n = 15, и, следовательно, код, образованный таким g(x), способен исправить любую одиночную ошибку. С тем же успехом за образующий многочлен кода мог быть принят и многочлен $x^4 + x + 1$. При этом был бы получен код, эквивалентный выбранному.

Однако использовать для тех же целей многочлен $x^4 + x^3 + x^2 + x + 1$ нельзя. При проверке числа различных остатков обнаруживается, что их у него не 15, а только 5. Действительно,

```
⊕ 100000000...|11111        Остатки
  11111                       1111
  11110
⊕ 11111                       0001
  00010000                      0010
                                0100
                               1000
```

Это объясняется тем, что многочлен $x^4 + x^3 + x^2 + x + 1$ входит в разложение не только двучлена $x^{15} + 1$, но и двучлена $x^5 + 1$.

Из приведенного примера следует, что в качестве образующего следует выбирать такой неприводимый многочлен *g(x)* (или произведение таких многочленов), который, являясь делителем двучлена $x^n + 1$, не входит в разложение ни одного двучлена типа $x^{\lambda} + 1$, степень которого $\lambda$ меньше *n*. В этом случае говорят, что многочлен *g(x)* принадлежит показателю степени *n*.

В табл. 4.14 приведены основные характеристики некоторых кодов, способных исправлять одиночные ошибки или обнаруживать все одиночные и двойные ошибки.

Таблица 4.14.

| Показатель неприводимого многочлена | Образующий многочлен | Число остатков | Длина кода |
|---|---|---|---|
| 2 | $x^2 + x + 1$ | 3 | 3 |
| 3 | $x^3 + x + 1$ | 7 | 7 |
| 3 | $x^3 + x^2 + 1$ | 7 | 7 |
| 4 | $x^4 + x^3 + 1$ | 15 | 15 |
| 4 | $x^4 + x + 1$ | 15 | 15 |
| 5 | $x^5 + x^2 + 1$ | 31 | 31 |
| 5 | $x^5 + x^3 + 1$ | 31 | 31 |

Это циклические коды Хэмминга для исправления одной ошибки, в которых в отличие от групповых кодов Хэмминга все проверочные разряды размещаются в конце кодовой комбинации.

Эти коды могут быть использованы для обнаружения любых двойных ошибок. Многочлен, соответствующий вектору двойной ошибки, имеет вид $\xi(x) = x^i - x^j$, или $\xi(x) = x^i(x^{j-i} + 1)$ при $j>i$. Так как $j - i<n$, а $g(x)$ не кратен $x$ и принадлежит показателю степени $n$, то $\xi(x)$ не делится на $g(x)$, что и позволяет обнаружить двойные ошибки.

### 4.10.3 Обнаружение ошибок кратности три и ниже

Образующие многочлены кодов, способных обнаруживать одиночные, двойные и тройные ошибки, можно определить, базируясь на следующем указании Хэмминга. Если известен образующий многочлен $p(x^m)$ кода длины $n$, позволяющего обнаруживать ошибки некоторой кратности $z$, то образующий многочлен $g(x)$ кода, способного обнаруживать ошибки следующей кратности $(z + 1)$, может быть получен умножением многочлена $p(x^m)$ на многочлен $x + 1$, что соответствует введению дополнительной проверки на четность. При этом число символов в комбинациях кода за счет добавления еще одного проверочного символа увеличивается до $n + 1$.

В табл. 4.15 приведены основные характеристики некоторых кодов, способных обнаруживать ошибки кратности три и менее.

Таблица 4.15.

| Показатель неприводимого многочлена | Образующий многочлен | Число информационных символов | Длина кода |
|---|---|---|---|
| 3 | $(x+1)(x^3 + x + 1)$ | 4 | 8 |
| 4 | $(x+1)(x^4+ x + 1)$ | 11 | 16 |
| 5 | $(x+1)(x^5+ x + 1)$ | 26 | 32 |

### 4.10.4 Обнаружение и исправление независимых ошибок произвольной кратности

Важнейшим классом кодов, используемых в каналах, где ошибки в последовательностях символов возникают независимо, являются коды Боуза-Чоудхури-Хоквингема. Доказано, что для любых целых положительных чисел m и s<n/2 существует двоичный код этого класса длины $n = 2^m$-1 с числом проверочных символов не более *ms,* который способен обнаруживать ошибки кратности 2s или исправлять ошибки кратности s. Для понимания теоретических аспектов этих кодов необходимо ознакомиться с рядом новых понятий высшей алгебры.

### 4.10.5 Обнаружение и исправление пачек ошибок

Для произвольного линейного блокового *(n,* k)-кода, рассчитанного на исправление пакетов ошибок длины *b* или менее, основным соотношением, устанавливающим связь корректирующей способности с числом избыточных символов, является граница Рейджера: $n - k \geq 2b$

При исправлении линейным кодом пакетов длины *b* или менее с одновременным обнаружением пакетов длины $l \geq b$ или менее требуется по крайней мере $b + l$ проверочных символов.

Из циклических кодов, предназначенных для исправления пакетов ошибок, широко известны коды Бартона, Файра и Рида-Соломона.

Первые две разновидности кодов служат для исправления одного пакета ошибок в блоке. Коды Рида-Соломона способны исправлять несколько пачек ошибок.

Особенности декодирования циклических кодов, исправляющих пакеты ошибок, рассмотрены далее на примере кодов Файра.

### 4.10.6 Методы образования циклического кода

Существует несколько различных способов кодирования. Принципиально наиболее просто комбинации циклического кода можно получить, умножая многочлены *a(x),* соответствующие комбинациям безызбыточного кода (информационным символам), на образующий многочлен кода *g(x).* Такой способ легко реализуется. Однако он имеет тот существенный недостаток, что получающиеся в результате умножения комбинации кода не содержат информационные символы в явном виде. После исправления ошибок такие комбинации для выделения информационных символов приходится делить на образующий многочлен кода. Применительно к циклическим кодам принято (хотя это и не обязательно) отводить под информационные *k* символов, соответствующих старшим степеням многочлена кода, а под проверочные *n-k* символов низших разрядов. Чтобы получить такой систематический код, применяется следующая процедура кодирования. Многочлен *a(x),* соответствующий k-разрядной комбинации безызбыточного кода, умножаем на $x^m$, где m = *n-k.* Степень каждого одночлена,

входящего в $a(x)$, увеличивается, что по отношению к комбинации кода означает необходимость приписать со стороны младших разрядов m нулей. Произведение $a(x)x^m$ делим на образующий многочлен $g(x)$. В общем случае при этом получаем некоторое частное $q(x)$ той же степени, что и $a(x)$ и остаток $r(x)$. Последний прибавляем к $a(x)x^m$. В результате получаем многочлен

$$f(x) = a(x)x^m + r(x)$$

Поскольку степень $g(x)$ выбираем равной $m$, степень остатка $r(x)$ не превышает m – 1. В комбинации, соответствующей многочлену $a(x)x^m$, $m$ младших разрядов нулевые, и, следовательно, указанная операция сложения равносильна приписыванию $r(x)$ к $a(x)$ со стороны младших разрядов.

Покажем, что $f(x)$ делится на $g(x)$ без остатка, т. е. является многочленом, соответствующим комбинации кода. Действительно, запишем многочлен $a(x)x^m$ в виде

$$a(x)x^m = q(x)g(x) + r(x)$$

Так как операции сложения и вычитания по модулю два идентичны, $r(x)$ можно перенести влево, тогда что и требовалось доказать.

$$a(x)x^m + r(x) = f(x) = q(x)g(x)$$

Таким образом, циклический код можно строить, приписывая к каждой комбинации безызбыточного кода остаток от деления соответствующего этой комбинации многочлена на образующий многочлен кода. Для кодов, число информационных символов в которых больше числа проверочных, рассмотренный способ реализуется наиболее просто.

Следует указать еще на один способ кодирования. Так как циклический код является разновидностью группового кода, то его проверочные символы должны выражаться через суммы по модулю два определенных информационных символов.

Равенства для определения проверочных символов могут быть получены путем решения рекуррентных соотношений:

$$a_{i+k} = \sum_{j=0}^{k-1} h_j a_{i+j} \qquad (4.5)$$

где $h$-двоичные коэффициенты так называемого генераторного многочлена $h(x)$, определяемого так

$$h(x) = (x^n + 1)/g(x) = h_0 + h_1 x + \ldots + h_k x^k$$

Соотношение (4.5) позволяет по заданной последовательности информационных сигналов $a_0, a_1, \ldots, a_{k-1}$ вычислить $n$-$k$ проверочных символов $a_k, a_{k+1}, \ldots \ldots, a_{n-1}$. Проверочные символы, как и ранее, размещаются на местах младших разрядов. При одних и тех же информационных символах комбинации кода, получающиеся таким путем, полностью совпадают с комбинациями, получающимися при использовании предыдущего способа кодирования. Применение данного

способа целесообразно для кодов с числом проверочных символов, превышающим число информационных, например для кодов Боуза-Чоудхури-Хоквингема.

### 4.10.7 Матричная запись циклического кода

Полная образующая матрица циклического кода $M_{n,k}$ составляется из двух матриц: единичной $I_k$ (соответствующей $k$ информационным разрядам) и дополнительной $C_{k,n-k}$ (соответствующей проверочным разрядам):

$$\mathbf{M}_{n,k} = \| \mathbf{I}_k \mathbf{C}_{k,n-k} \|$$

Построение матрицы $I_k$ трудностей не представляет. Если образование циклического кода производится на основе решения рекуррентных соотношений, то его дополнительную матрицу можно определить, воспользовавшись правилами, указанными ранее. Однако обычно строки дополнительной матрицы циклического кода $\mathbf{C}_{k,n-k}$ определяются путем вычисления многочленов $r(x)$. Для каждой строки матрицы $I_k$ соответствующий $r(x)$ находят делением информационного многочлена $a(x)x^m$ этой строки на образующий многочлен кода $g(x)$.

Дополнительную матрицу можно определить и не строя $I_k$. Для этого достаточно делить на $g(x)$ комбинацию в виде единицы с рядом нулей и получающиеся остатки записывать в качестве строк дополнительной матрицы. При этом если степень какого-либо $r(x)$ оказывается меньше n – k – 1, то следующие за этим остатком строки матрицы получают путем циклического сдвига предыдущей строки влево до тех пор, пока степень $r(x)$ не станет равной $n$-$k$— 1. Деление производится до получения $k$ строк дополнительной матрицы.

Пример 36. Запишем образующую матрицу для циклического кода (15,11) с порождающим многочленом $g(x) = x^4 + x^3 + 1$.

Воспользовавшись результатами ранее проведенного деления, получим

$$\mathbf{M}_{15,11} = \begin{bmatrix} 00000000001 & 1001 \\ 00000000010 & 1011 \\ 00000000100 & 1111 \\ 00000001000 & 0111 \\ 00000010000 & 1110 \\ 00000100000 & 0101 \\ 00001000000 & 1010 \\ 00010000000 & 1101 \\ 00100000000 & 0011 \\ 01000000000 & 0110 \\ 10000000000 & 1100 \end{bmatrix}$$

Существует другой способ построения образующей матрицы, базирующийся на основной особенности циклического (n, k)-кода. Он проще описанного, но получающаяся матрица менее удобна. Матричная запись кодов достаточно широко распространена.

### 4.10.8 Укороченные циклические коды

Корректирующие возможности циклических кодов определяются степенью *m* образующего многочлена. В то время как необходимое число информационных символов может быть любым целым числом, возможности в выборе разрядности кода весьма ограничены.

Если, например, необходимо исправить единичные ошибки при $k = 5$, то нельзя взять образующий многочлен третьей степени, поскольку он даст только семь остатков, а общее число разрядов получится равным 8.

Следовательно, необходимо брать многочлен четвертой степени и тогда n= 15. Такой код рассчитан на 11 информационных разрядов.

Однако можно построить код минимальной разрядности, заменив в (n, k) -коде j первых информационных символов нулями и исключив их из кодовых комбинаций. Код уже не будет циклическим, поскольку циклический сдвиг одной разрешенной кодовой комбинации не всегда приводит к другой разрешенной комбинации того же кода. Получаемый таким путем линейный *(n-j, k*-j)-код называют *укороченным циклическим кодом.* Минимальное расстояние этого кода не менее, чем минимальное кодовое расстояние *(n, k)*-кода, из которого он получен. Матрица укороченного кода получается из образующей матрицы *(n, k)*-кода исключением j строк и столбцов, соответствующих старшим разрядам. Например, образующая матрица кода (9,5), полученная из матрицы кода (15,11), имеет вид

$$\mathbf{M}_{9,5} = \begin{bmatrix} 00001 & 1001 \\ 00010 & 1011 \\ 00100 & 1111 \\ 01000 & 0111 \\ 10000 & 1110 \end{bmatrix}.$$

## 4.11 Технические средства кодирования и декодирования для циклических кодов

### 4.11.1 Линейные переключательные схемы

Основу кодирующих и декодирующих устройств циклических кодов составляют регистры сдвига с обратными связями, позволяющие осуществлять как умножение, так и деление многочленов с приведением коэффициентов по модулю два. Такие регистры также называют многотактными линейными переключательными схемами и линейными кодовыми фильтрами Хаффмена. Они состоят из ячеек памяти, сумматоров по модулю два и устройств умножения на коэффициенты многочленов множителя или делителя. В случае двоичных кодов для умножения на коэффициент, равный 1, требуется только наличие связи в схеме. Если коэффициент равен 0, то связь отсутствует. Сдвиг информации в регистре осуществляется импульсами, поступающими с генератора продвигающих импульсов, который на схеме, как правило, не указывается. На вход

устройств поступают только коэффициенты многочленов, причем начиная с коэффициента при переменной в старшей степени.

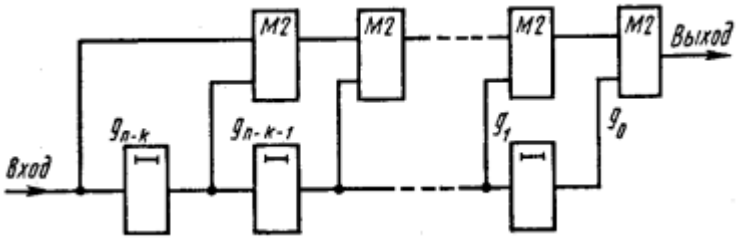



Рис. 4.9.

На рис. 4.9 представлена схема, выполняющая умножение произвольного (например, информационного) многочлена

$a(x) = a_0 + a_1x + \ldots + a_{k-1}x^{k-1}$

на некоторый фиксированный (например, образующий) многочлен

$g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}$.

Произведение этих многочленов равно

$a(x)g(x)=a_0g_0 + (a_0g_1 + a_1g_0)x + \ldots + (a_{k-2}g_{n-k} + a_{k-1}g_{n-k-1})x^{n-2} + a_{k-1}g_{n-k}x^{n-1}$

Предполагаем, что первоначально ячейки памяти находятся в нулевом состоянии и что за коэффициентами множимого следует $n-k$ нулей.

На первом такте на вход схемы поступает первый коэффициент $a_{k-1}$ многочлена $a(x)$ и на выходе появляется первый коэффициент произведения, равный

$a_{k-1}g_{n-k}$

На следующем такте на выход поступит сумма

$a_{k-2}g_{n-k} + a_{k-1}g_{n-k-1}$ ,

т.е. второй коэффициент произведения, и т. д. На n-м такте все ячейки, кроме последней, будут в нулевом состоянии и на выходе получим последний коэффициент $a_0g_0$

Используется также схема умножения многочленов при поступлении множимого младшим разрядом вперед (рис. 4.10).

На рис. 4.11 представлена схема, выполняющая деление произвольного многочлена, например

$a(x)x^m = a_0 + a_1x + \ldots + a_{n-1}x^{n-1}$

на некоторый фиксированный (например, образующий) многочлен

$g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}$

Обратные связи регистра соответствуют виду многочлена $g(x)$. Количество включаемых в него сумматоров равно числу отличных от нуля коэффициентов $g(x)$, уменьшенному на единицу. Это объясняется тем, что сумматор сложения коэффициентов старших разрядов многочленов

делимого и делителя в регистр не включается, так как результат сложения заранее известен (он равен 0).

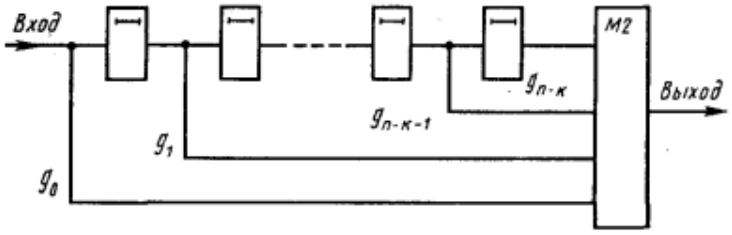



Рис. 4.10.

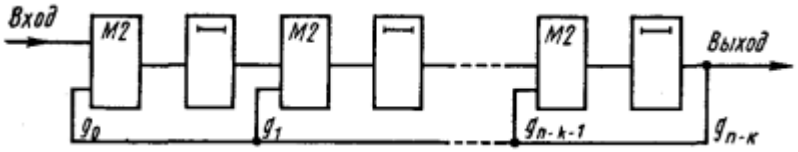



Рис. 4.11.

За первые *n-k* тактов коэффициенты многочлена-делимого заполняют регистр, причем коэффициент при *x* в старшей степени достигает крайней правой ячейки. На следующем такте «единица» делимого, выходящая из крайней ячейки регистра, по цепи обратной связи подается к сумматорам по модулю два, что равносильно вычитанию многочлена-делителя из многочлена-делимого. Если в результате предыдущей операции коэффициент при старшей степени *x* у остатка оказался равным нулю, то на следующем такте делитель не вычитается. Коэффициенты делимого только сдвигаются вперед по регистру на один разряд, что находится в полном соответствии с тем, как это делается при делении многочленов столбиком.

Деление заканчивается с приходом последнего символа многочлена-делимого. При этом разность будет иметь более низкую степень, чем делитель. Эта разность и есть остаток.

Отметим, что если в качестве многочлена-делителя выбран простой многочлен степени m = *n*-k, то, продолжая делить образовавшийся остаток при отключенном входе, будем получать в регистре по одному разу каждое из ненулевых m-разрядных двоичных чисел. Затем эта последовательность чисел повторяется.

Пример 37. Рассмотрим процесс деления многочлена $a(x)x^r = (x^3+1)x^3$ на образующий многочлен $g(x) = x^3 + x^2+1$. Схема для этого случая представлена на рис. 4.12, где 1, 2, 3-ячейки регистра. Работа схемы поясняется табл. 4.16.

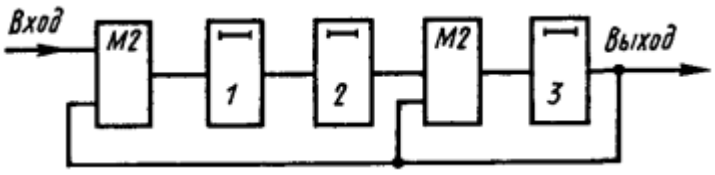



Рис. 4.12.

Вычисление остатка начинается с четвертого такта и заканчивается после седьмого такта. Последующие сдвиги приводят к образованию в регистре последовательности из семи различных

ненулевых трехразрядных чисел. В дальнейшем эта последовательность чисел повторяется. Рассмотренные выше схемы умножения и деления многочленов непосредственно в том виде, в каком они представлены на рис. 4.10, 4.11, в качестве кодирующих устройств циклических кодов на практике не применяются: первая - из-за того, что образующаяся кодовая комбинация в явном виде не содержит информационных символов, а вторая - из-за того, что между информационными и проверочными символами образуется разрыв в n - *k* разрядов.

### 4.11.2 Кодирующие устройства

Все известные кодирующие устройства для любых типов циклических кодов, выполненные на регистрах сдвига, можно свести к двум типам схем согласно рассмотренным ранее методам кодирования.

Таблица 4.16.

| Номер такта | Вход | Состояние ячеек регистра | | | Номер такта | Вход | Состояние ячеек регистра | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | | | 1 | 2 | 3 |
| 1 | 1 | 1 | 0 | 0 | 8 | - | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 9 | - | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 10 | - | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 | 11 | - | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 12 | - | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 1 | 13 | - | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 0 | 14 | - | 1 | 1 | 0 |

Схемы первого типа вычисляют значения проверочных символов путем непосредственного деления многочлена $a(x)x^m$ на образующий многочлен $g(x)$. Это делается с помощью регистра сдвига, содержащего $n$-$k$ разрядов (рис. 4.13).

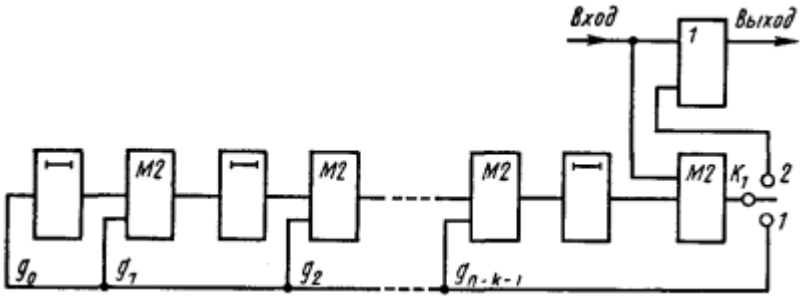



Рис. 4.13.

Схема отличается от ранее рассмотренной тем, что коэффициенты кодируемого многочлена участвуют в обратной связи не через $n$-$k$ сдвигов, а сразу с первого такта. Это позволяет устранить разрыв между информационными и проверочными символами. В исходном

состоянии ключ $K_1$ находится в положении 1. Информационные символы одновременно поступают как в линию связи, так и в регистр сдвига, где за $k$ тактов образуется остаток. Затем ключ $K_1$ переходит в положение *2* и остаток поступает в линию связи.

Пример 38. Рассмотрим процесс деления многочлена $a(x)x^r = (x^3 + 1)x^3$ на многочлен $g(x) = x^3 + x^2 + 1$ за k тактов.

Схема кодирующего устройства для заданного g(x) приведена на рис. 4.14. Процесс формирования кодовой комбинации шаг за шагом представлен в табл. 4.17, где черточками отмечены освобождающиеся ячейки, занимаемые новыми информационными символами.

С помощью схем второго типа вычисляют значения проверочных символов как линейную комбинацию информационных символов, т. е. они построены на использовании основного свойства систематических кодов. Кодирующее устройство строится на основе k-разрядного регистра сдвига (рис. 4.15).

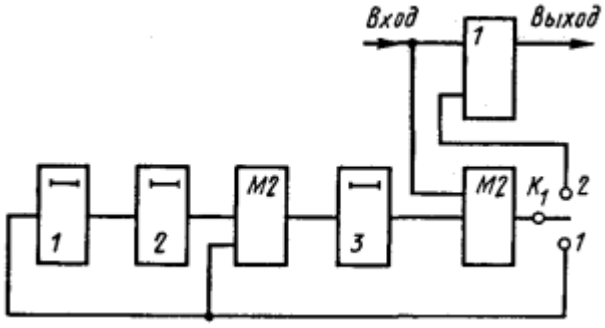



Рис. 4.14.

Таблица 4.17.

| Номер такта | Вход | Состояние ячеек регистра | | | Выход |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | 1 | 1 | 01 |
| 3 | 0 | 1 | 1 | 0 | 001 |
| 4 | 1 | 1 | 1 | 0 | 1001 |
| 5 | 0 | - | 1 | 1 | 01001 |
| 6 | 0 | - | - | 1 | 101001 |
| 7 | 0 | - | - | - | 1101001 |

Выходы ячеек памяти подключаются к сумматору в цепи обратной связи в соответствии с видом генераторного многочлена

$h(x) = (x^n + 1)/g(x) = h_0 + h_1x + \ldots + h_kx^k$

В исходном положении ключ $K_1$ находится в положении 1. За первые $k$ тактов поступающие на вход информационные символы заполняют все ячейки регистра. После этого ключ переводят в положение *2*. На каждом из последующих тактов один из информационных

символов выдается в канал связи и одновременно формируется проверочный символ, который записывается в последнюю ячейку регистра. Через *n-k* тактов процесс формирования проверочных символов заканчивается и ключ $K_1$ снова переводится в положение 1.

В течение последующих *k* тактов содержимое регистра выдается в канал связи с одновременным заполнением ячеек новой последовательности информационных символов.

Пример 39. Рассмотрим процесс формирования кодовой комбинации с использованием генераторного многочлена для случая $g(x) = x^3 + x^2 + 1$ и $a(x) = x^3 + 1$.

Определяем генераторный многочлен:

$$h(x) = \frac{x^7 + 1}{x^3 + x^2 + 1} = x^4 + x^3 + x^2 + 1$$

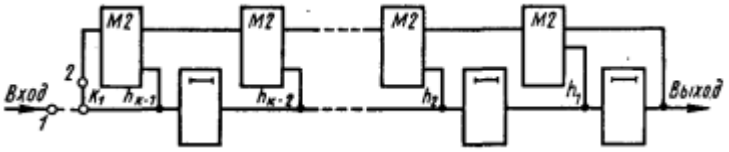



Рис. 4.15.

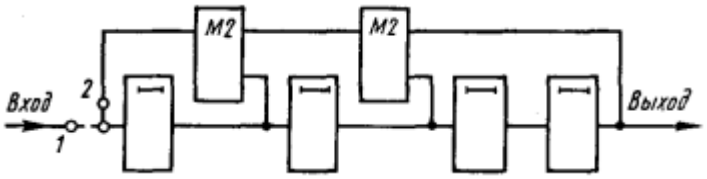



Рис. 4.16.

Соответствующая *h(x)* схема кодирующего устройства приведена на рис. 4.16. Формирование кодовой комбинации поясняется табл. 4.18. Оно начинается после заполнения регистра информационными символами.

### 4.11.3 Декодирующие устройства

Декодирование комбинаций циклического кода можно проводить различными методами. Существуют методы, основанные на использовании рекуррентных соотношений, на мажоритарном принципе, на вычислении остатка от деления принятой комбинации на образующий многочлен кода и др. Целесообразность применения каждого из них зависит от конкретных характеристик используемого кода.

Рассмотрим сначала устройства декодирования, в которых для обнаружения и исправления ошибок производится деление произвольного многочлена *f(x),* соответствующего принятой комбинации, на образующий многочлен кода $g_0(x)$. В этом случае при декодировании могут использоваться те же регистры сдвига, что и при кодировании.

Таблица 4.18.

| Номер такта | Состояние ячеек регистра | Выход |
|---|---|---|

| | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 01 |
| 2 | 1 | 0 | 1 | 0 | 001 |
| 3 | 1 | 1 | 0 | 1 | 1001 |
| 4 | - | 1 | 1 | 0 | 01001 |
| 5 | - | - | 1 | 1 | 101001 |
| 6 | | - | - | 1 | 1101001 |
| 7 | - | - | - | - | |

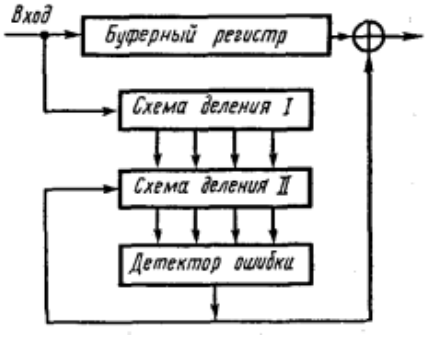



Рис. 4.17.

Декодирующие устройства для кодов, обнаруживающих ошибки, по существу ничем не отличаются от схем кодирующих устройств. В них добавляется лишь буферный регистр для хранения принятого сообщения на время проведения операции деления. Если остатка не обнаружено (случай отсутствия ошибки), то информация с буферного регистра считывается в дешифратор сообщения. Если остаток обнаружен (случай наличия ошибки), то информация в буферном регистре уничтожается и на передающую сторону посылается импульс запроса повторной передачи. В случае исправления ошибок схема несколько усложняется. Информацию о разрядах, в которых произошла ошибка, несет, как и ранее, остаток. Схема декодирующего устройства представлена на рис. 4.17. Символы подлежащей декодированию кодовой комбинации, возможно, содержащей ошибку, последовательно, начиная со старшего разряда, вводятся в n-разрядный буферный регистр сдвига и одновременно в схему деления, где за n тактов определяется остаток, который в случае непрерывной передачи сразу же переписывается в регистр второй аналогичной схемы деления. Начиная с (n + 1)-го такта в буферный регистр и первую схему деления начинают поступать символы следующей кодовой комбинации. Одновременно на каждом такте буферный регистр покидает один символ, а в регистре второй схемы деления появляется новый остаток (синдром). Детектор ошибок, контролирующий состояния ячеек этого регистра, представляет собой комбинаторно-логическую схему, построенную с таким расчетом, чтобы она отмечала все те синдромы *(«выделенные синдромы»),* которые появляются в схеме деления, когда каждый из ошибочных символов занимает крайнюю

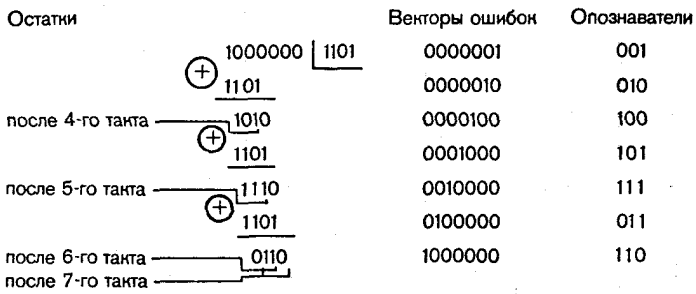
правую ячейку в буферном регистре. При последующем сдвиге детектор формирует сигнал «1», который, воздействуя на сумматор коррекции, исправляет искаженный символ.

Одновременно по цепи обратной связи с выхода детектора подается сигнал «1» на входной сумматор регистра второй схемы деления. Этот сигнал изменяет выделенный синдром так, чтобы он снова соответствовал более простому типу ошибки, которую еще подлежит исправить. Продолжая сдвиги, обнаружим и другие выделенные синдромы. После исправления последней ошибки все ячейки декодирующего регистра должны оказаться в нулевом состоянии. Если в результате автономных сдвигов состояние регистра не окажется нулевым, это означает, что произошла неисправимая ошибка. Для декодирования кодовых комбинаций, разнесенных во времени, достаточно одной схемы деления, осуществляющей декодирование за 2n тактов. Сложность детектора ошибок зависит от числа выделенных синдромом. Простейшие детекторы получаются при реализации кодов, рассчитанных на исправление единичных ошибок. Выделенный синдром появляется в схеме деления раньше всего в случае, когда ошибка имеет место в старшем разряде кодовой комбинации, так как он первым достигает крайней правой ячейки буферного регистра. Поскольку неискаженная кодовая комбинация делится на $g_0(x)$ без остатка, то для определения выделенного синдрома достаточно разделить на $g_0(x)$ вектор ошибки с единицей в старшем разряде. Остаток, получающийся на *n-м* такте, и является искомым выделенным синдромом. В зависимости от номера искаженного разряда после первых тактов будем получать различные остатки (опознаватели соответствующих векторов ошибок). Вследствие этого выделенный синдром будет появляться в регистре схемы деления через различное число последующих тактов, обеспечивая исправление искаженного символа.

В качестве схем деления в декодирующем устройстве могут быть использованы как схемы, определяющие остаток за n тактов (см. рис. 4.11), так и схемы, определяющие остаток за *k* тактов (рис. 4.13). При использовании схемы деления за *k* тактов векторам одиночных ошибок $\xi(x)$ будут соответствовать другие остатки на n-м такте, являющиеся результатом деления на образующий многочлен кода векторов $\xi(x)x^m$, *а* на $\xi(x)$. Поэтому выделенные синдромы, а следовательно, и детекторы ошибок для указанных схем будут различны.

Пример 40. Рассмотрим процесс исправления единичной ошибки при использовании кода (7,4) с образующим многочленом $g(x) = x^3 + x^2 + 1$ и применении в декодирующем устройстве схем деления за n и k тактов.

Определим опознаватели ошибок и выделенный синдром для случая использования схемы деления за n тактов:




Детектор ошибки, обеспечивающий формирование на выходе сигнала только в случае появления в схеме деления остатка 110, можно реализовать посредством двух логических элементов НЕ и одного логического элемента ИЛИ-НЕ.

На рис. 4.18 приведена схема соответствующего декодирующего устройства. В табл. 4.19 представлен процесс исправления ошибки для случая, когда кодовая комбинация 1001011 (см. табл. 4.18) поступила на вход декодирующего устройства с искаженным символом в 4-м разряде (1000011).

После n (в данном случае 7) тактов в схему деления II переписывается опознаватель ошибки 101.

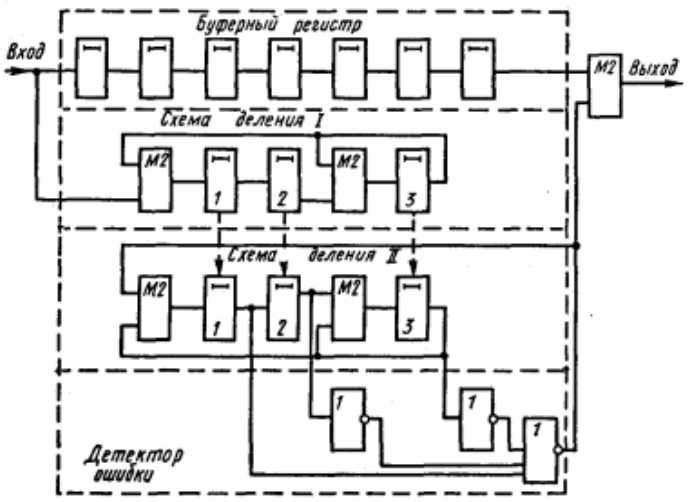


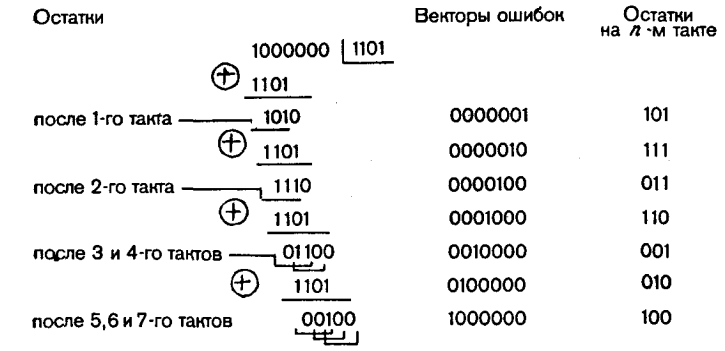

Рис. 4.18.

Таблица 4.19.

| Номер такта | Вход | Состояние ячеек схем деления | | | Выход после коррекции |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | |
| 1 | 1 | 1 | 0 | 0 | |
| 2 | 0 | 0 | 1 | 0 | |
| 3 | 0 | 0 | 0 | 1 | |
| 4 | 0 | 1 | 0 | 1 | |
| 5 | 0 | 1 | 1 | 1 | |
| 6 | 1 | 0 | 1 | 0 | |

| | | | | | |
|---|---|---|---|---|---|
| 7 | 1 | 1 | 0 | 1 | Переписывается в схему деления |
| 8 | 0 | 1 | 1 | 1 | II |
| 9 | 0 | 1 | 1 | 0 | 1 |
| 10 | 0 | 0 | 1 | 1 | 01 |
| 11 | 0 | 0 | 0 | 0 | 001 |
| 12 | 0 | 0 | 0 | 0 | 1001 |
| 13 | 0 | 0 | 0 | 0 | 01001 |
| 14 | 0 | 0 | 0 | 0 | 101001 |
| | | | | | 1101001 |

На каждом последующем такте на выходе буферного регистра появляется неискаженный символ корректируемой кодовой комбинации, а в схеме деления II новый остаток. Выделенный синдром появится в схеме деления на 10-м такте, когда искаженный символ займет крайнюю правую ячейку регистра. На следующем такте он попадет в корректирующий сумматор и будет там исправлен импульсом, поступающим с выхода детектора ошибки. Этот же импульс по цепи обратной связи приводит ячейки схемы деления II в нулевое состояние (корректирует выделенный синдром). При использовании схемы деления за k тактов соответствие между векторами ошибок и остатками на n-м такте иное.



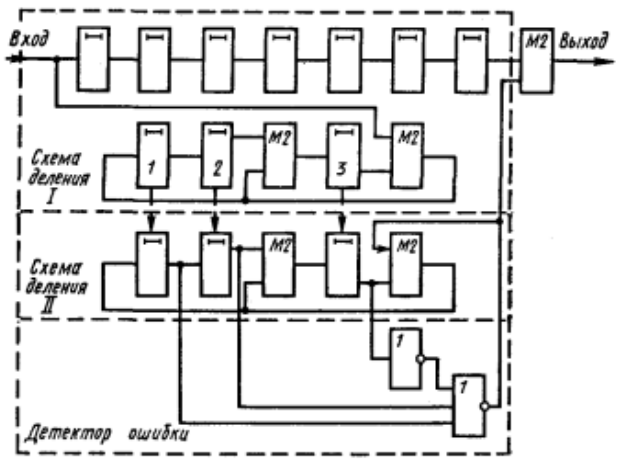




Рис. 4.19.

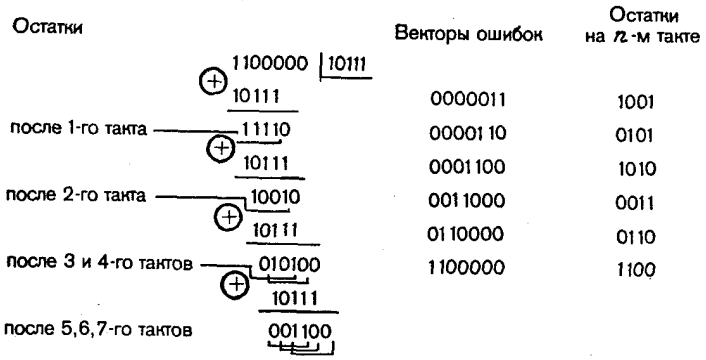
Таблица 4.20.

| Номер такта | Вход | Состояние ячеек схем деления | | | Выход после коррекции |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | |
| 1 | 1 | 1 | 0 | 1 | |
| 2 | 0 | 1 | 1 | 1 | |
| 3 | 0 | 1 | 1 | 0 | |
| 4 | 0 | 0 | 1 | 1 | |
| 5 | 0 | 1 | 0 | 0 | |
| 6 | 1 | 1 | 1 | 1 | |
| 7 | 1 | 0 | 1 | 1 | Переписывается в схему деления II |
| 8 | 0 | 1 | 0 | 0 | 1 |
| 9 | 0 | 0 | 1 | 0 | 01 |
| 10 | 0 | 0 | 0 | 1 | 001 |
| 11 | 0 | 0 | 0 | 0 | 1001 |
| 12 | 0 | 0 | 0 | | 01001 |
| 13 | 0 | 0 | 0 | 0 | 101001 |
| 14 | 0 | 0 | 0 | 0 | 1101001 |

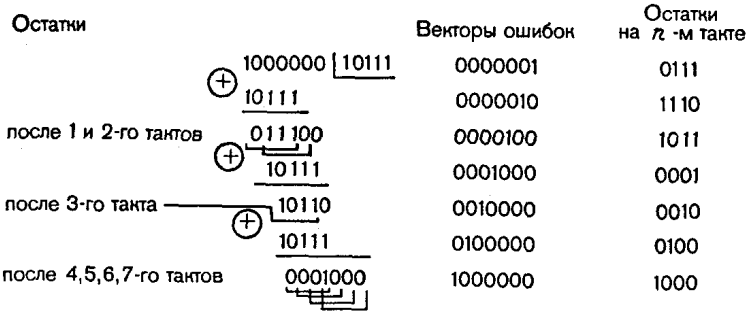
Детектор для выделенного синдрома 100 можно построить из одного логического элемента НЕ и одного элемента ИЛИ-НЕ. На рис. 4.19 представлена схема декодирующего устройства для этого случая. Табл. 4.20 позволяет проследить по тактам процесс исправления ошибки в кодовой комбинации 1000011 (искажен символ в 4-м разряде). Сравнение показывает, что использование в декодирующем устройстве схемы деления за k тактов предпочтительнее, так как выделенный синдром в этом случае при любом объеме кода содержит единицу в старшем и нули во всех остальных разрядах, что приводит к более простому детектору ошибки.

Пример 41. Рассмотрим более сложный случай исправления одиночных и двойных смежных ошибок. Для этой цели может использоваться циклический код (7,3) с образующим многочленом $g(x) = (x + 1)(x^3 + x^2+1)$.

Ориентируясь на схему деления за k тактов, найдем выделенный синдром для двойных смежных ошибок:




Для одиночных ошибок соответственно получим




Детектор ошибок в этом случае должен формировать сигнал коррекции при появлении каждого выделенного синдрома. Схема декодирующего устройства представлена на рис. 4.20. Процесс исправления кодовой комбинации 1000010 с искаженными символами в 4-м и 5-м разрядах поясняется табл. 4.21. На 9-м такте в схеме деления II появляется первый выделенный синдром 1100. На следующем такте на выходе аналогично обозначенного элемента ИЛИ-НЕ детектора ошибок формируется импульс коррекции, который исправляет 5-й разряд кодовой комбинации и одновременно по цепи обратной связи изменяет остаток в схеме деления II, приводя его в соответствие выделенному синдрому еще не исправленной одиночной ошибки в 4-м разряде (1000). На 11-м такте импульс коррекции формирует элемент ИЛИ-НЕ детектора ошибок, соответствующий указанному выделенному синдрому. Этим импульсом обеспечивается исправление 4-го разряда кодовой комбинации и получение нулевого остатка в схеме деления II.

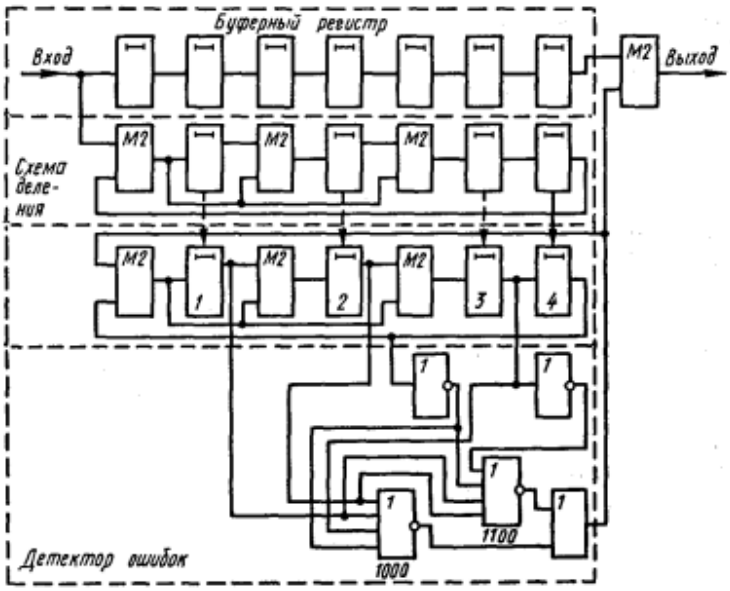



Рис. 4.20.

Таблица 4.21.

| Номер такта | Вход | Состояние ячеек схем деления | | | | Выход после коррекции |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | |
| 1 | 1 | 1 | 1 | 1 | 0 | |
| 2 | 0 | 0 | 1 | 1 | 1 | |
| 3 | 0 | 1 | 1 | 0 | 1 | |
| 4 | 0 | 1 | 0 | 0 | 0 | |
| 5 | 1 | 1 | 0 | 1 | 0 | |
| 6 | 0 | 0 | 1 | 0 | 1 | |
| 7 | 0 | 1 | 1 | 0 | 0 | Переписывается в схему деления II |
| 8 | 0 | 0 | 1 | 1 | 0 | 1 |
| 9 | 0 | 0 | 0 | 1 | 1 | 01 |
| 10 | 0 | 0 | 0 | 0 | 1 | 101 |
| 11 | 0 | 0 | 0 | 0 | 0 | 1101 |
| 12 | 0 | 0 | 0 | 0 | 0 | 11101 |
| 13 | 0 | 0 | 0 | 0 | 0 | 011101 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0011101 |

**Список литературы**

Дмитриев В.И. Прикладная теория информации. Учебник для студентов ВУЗов по специальности «Автоматизированные системы обработки информации и управления». – М.: Высшая школа, 1989 – 320 с.