Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС
для студентов специальности 1-39 01 01 «Радиотехника»

Составители
Р.П. Богуш, А.В. Курилович
Под общей редакцией Р.П. Богуша

Новополоцк 2005

УДК 621.391 (075.8)
ББК 32.811.4 я73
П 75

РЕЦЕНЗЕНТЫ:

В.К. КОНОПЕЛЬКО, доктор техн. наук, профессор, зав. кафедрой «Системы и устройства телекоммуникаций» Белорусского государственного университета информатики и радиоэлектроники;

С.В. МАЛЬЦЕВ, канд. техн. наук, зав. кафедрой радиоэлектроники;

Д.А. ДОВГЯЛО, канд. техн. наук, доцент кафедры конструирования и технологии радиоэлектронных средств

Рекомендованы к изданию методической комиссией радиотехнического факультета

П 75 **Прикладная теория кодирования:** Учеб.-метод. комплекс для студ. спец. 1-39 01 01 «Радиотехника»/ Сост. Р.П. Богуш, А.В. Курилович; Под общей редакцией Р.П. Богуша. – Новополоцк: ПГУ, 2005. – 256 с.
ISBN 985-418-348-3

Представлен курс лекций по прикладной теории кодирования: рассмотрены математические основы кодирования информации, приведены методы и алгоритмы эффективного, криптографического и помехоустойчивого кодирования. Представлены методические указания к выполнению практических работ. Предлагается система оценки знаний студентов.

Предназначен для студентов и преподавателей радиотехнического факультета.

УДК 621.391 (075.8)
ББК 32.811.4 я73

ISBN 985-418-348-3

СОДЕРЖАНИЕ

Введение в курс «Прикладная теория кодирования»

Цель и задачи дисциплины

Основная цель курса «Прикладная теория кодирования» – изучение студентами теории кодирования, включая методы и средства эффективного, криптографического и помехоустойчивого кодирования и декодирования информации.

В результате изучения дисциплины студенты должны

знать:

- математические основы современных методов обработки кодированной информации;
- алгебраические принципы построения и свойства кодов,
- методы эффективного, криптографического и помехоустойчивого кодирования и декодирования;
- принципы построения кодеров и декодеров;

уметь:

- обосновывать и выбирать параметры кодов для решения радиотехнических задач;
- разрабатывать эффективные алгоритмы кодирования-декодирования при использовании разных классов кодов;
- реализовывать синтез схем кодеров и декодеров;

иметь представление:

- о вычислительной сложности алгоритмов декодирования, современных методах кодирования, о методах реализации специализированных кодеров и декодеров;
- о направлениях, перспективах и проблемах развития теории кодирования для решения современных информационных задач.

Структура дисциплины

Согласно учебному плану специальности 1-39 01 01 «Радиотехника» курс «Прикладная теория кодирования» изучается студентами на 4 курсе (7 семестр), рассчитан на 115 ч и включает в себя следующие виды аудиторных занятий:

- 48 ч лекций;
- 16 ч практических работ;
- 16 ч лабораторных работ.

Ниже представлено распределение курса по видам аудиторных занятий по разделам и темам.

Лекционный курс

Наименование разделов и тем лекций, их содержание	Количество часов
Введение в курс «Прикладная теория кодирования»	
Содержание дисциплины и ее взаимосвязь с другими дисциплинами	2
Раздел 1. Введение в теорию кодирования информации	
1.1. Модели каналов передачи, обработки и хранения информации	2
Раздел 2. Математические основы теории кодирования	
2.1. Основы теории чисел, основы матричного анализа	2
2.2. Элементы теории групп	2
Раздел 3. Статистическое кодирование информации	
3.1. Элементы теории информации и основы статистического кодирования	2
3.2. Эффективные коды	2
3.3. Кодирование изображений и сжатие информации с помощью спектральных преобразований	2
Раздел 4. Криптографическое кодирование	
4.1 Методы криптографического кодирования, обеспечивающие секретность информации	2
4.2. Современные симметричные криптосистемы	2
4.3. Современные асимметричные криптосистемы	2
Раздел 5. Основные понятия теории помехоустойчивого кодирования	
5.1. Ошибки в каналах передачи информации и их модели	2
5.2. Основные параметры кодов. Границы кодов, исправляющих ошибки.	2
Раздел 6. Линейные коды	
6.1. Способы задания линейных кодов и кодирование информации	2
6.2. Основные методы декодирования линейных кодов	2
6.3. Коды с проверкой на четность и коды Хэмминга	2
6.4. Коды Рида-Маллера и БЧХ-коды	2
6.5. Задание циклических кодов. Линейные переключательные схемы для умножения и деления многочленов	2
6.6. Построение кодирующих и декодирующих устройств циклических кодов	2
6.7. Кодирование для исправления зависимых ошибок	2
Раздел 7. Сверточные коды	
7.1. Структура и описание сверточных кодов	2
7.2. Кодирование и декодирование сверточными кодами	2
Раздел 8. Низкоскоростные коды	
8.1. Построение низкоскоростных кодов	2
8.2. Декодирование низкоскоростных кодов	2
Раздел 9. Кодовые методы повышения надежности цифровых устройств	
9.1. Коды, исправляющие дефекты	2

Лабораторные занятия

Наименование лабораторной работы	Количество часов
1. Эффективное кодирование информации	4
2. Кодирование и декодирование информации кодами Хэмминга	4
3. Кодирование и декодирование информации кодами Рида-Маллера	4
4. Декодирование низкоскоростных кодов методом максимального правдоподобия	4

Практические занятия

Наименование лабораторной работы	Количество часов
1. Алгебраические операции на множестве целых чисел и операции с матрицами	2
2. Арифметика полей Галуа	2
3. Классические шифры, ассиметричные криптосистемы	2
4. Современные симметричные криптосистемы	2
5. Линейные переключательные схемы для умножения и деления многочленов по модулю два	2
6. Задание, кодирование и декодирование циклических кодов	2
7. Помехоустойчивое кодирование информации сверточными кодами	2
8. Формирование и декодирование кодов максимальной длины	2

Оценка знаний студентов

Для оценки работы и знаний студентов в рамках курса используется накопительная система. Результирующая оценка выставляется по сумме баллов, которые студент набирает в течение всего учебного семестра, а также в результате выходного итогового контроля – экзамена.

Для получения аттестации необходимо:

- выполнение всех предшествующих аттестации лабораторных работ;
- по практическим работам должно быть набрано не менее 1/3 от максимально возможного количества баллов на момент аттестации по данному виду занятий;
- написание контрольной работы – два теоретических вопроса-(максимальное количество баллов за коллоквиум – 50, для аттестации – 30 и более).

Распределение баллов по видам занятий

Вид занятий	Форма оценки учебной активности студента	Максимальное количество баллов по каждой форме оценки	Максимальное количество баллов по каждому виду занятий
Практические работы	Устные ответы на вопросы	5	20×8=160
	Решение задач	10	
	Выполнение тестовых заданий	5	
Лабораторные занятия	Выполнение в срок	5	30×4=120
	Грамотное составление отчета с письменными ответами на контрольные вопросы	10	
	Защита на следующем занятии	15	
Экзамен	Ответы на экзаменационные вопросы	120	120

Примечание. Аттестации – 100 баллов максимум

Дополнительные баллы предусматриваются:

- за выполнение научно-исследовательской работы по прикладной теории кодирования (до 200 баллов);
- за подготовку докладов по методам и алгоритмам кодирования информации, не включенным в рабочую программу курса «Прикладная теория кодирования» (до 100 баллов).

Итоговая оценка выставляется по шкале:

Оценка	1	2	3	4	5	6	7	8	9	10
Сумма баллов	0 – 89	90 – 169	170 – 219	220 – 269	270 – 319	320 – 359	360 – 399	400 – 439	440 – 479	480 и более

Очевидно, что для получения минимальной положительной оценки – 4 балла, студент должен набрать 220 баллов, а для этого необходимо:

- выполнить все лабораторные работы – 20 баллов;
- получить две аттестации – минимум 60 баллов + 80 баллов за практические занятия;
- получить 60 баллов при сдаче экзамена.

Для получения высшей оценки – 10 баллов – студент должен: проявлять самостоятельную творческую работу на практических, лабораторных занятиях; выступать с самостоятельно подготовленными докладами по тематике учебного курса; иметь необходимые знания и способность само-

стоятельно и творчески решать задачи кодирования информации в нестандартной ситуации.

Применение кодирования информации в радиотехнических системах

Становление и развитие кодов первоначально стимулировалось задачами связи, поэтому терминология теории кодирования проистекает из теории связи. Однако коды имеют много других приложений. Коды используются для защиты данных в памяти вычислительных устройств и на цифровых лентах и дисках; для защиты от неправильного функционирования или шумов в цифровых логических цепях. Коды используются также для сжатия данных, и теория кодирования тесно связана с теорией планирования статистических экспериментов.

Приложения к задачам связи носят самый различный характер. Двоичные данные обычно передаются между вычислительными терминалами, между летательными аппаратами и между спутниками. Коды могут быть использованы для получения надежной связи даже тогда, когда мощность принимаемого сигнала близка к мощности тепловых шумов. В военных приложениях такие коды часто используются для защиты против намеренно организованной противником интерференции. И поскольку электромагнитный спектр все больше и больше заполняется создаваемым человеком сигналами, коды, контролирующие ошибки, обеспечивающие скорость и защиту передачи информации, становятся еще более важным инструментом не только в настоящее время, но и в будущем. Так во многих системах связи имеется ограничение на передаваемую мощность. Например, в системах ретрансляции через спутники увеличение мощности обходится очень дорого. Коды, контролирующие ошибки, являются замечательным средством снижения необходимой мощности, т.к. с их помощью можно правильно восстановить полученные ослабленные сообщения.

Передача в вычислительных системах обычно чувствительна даже к очень малой доле ошибок – даже одиночная ошибка может нарушить программу вычисления. Кодирование, контролирующее ошибки, становится в этих приложениях весьма важным. Для некоторых носителей вычислительной памяти использование кодов, контролирующих ошибки, позволяет добиться более плотной упаковки битов. Другим типом систем связи является система с многими пользователями и разделением по времени, в которой каждому из данного числа пользователей заранее предписаны некоторые временные окна (интервалы), где разрешается передача. Длинные двоичные сообщения разделяются на пакеты, и один пакет передается в отведенное временное окно. Из-за нарушения синхронизации или дисциплины

обслуживания некоторые пакеты могут быть утеряны. Подходящие коды, контролирующие ошибки, защищают от таких потерь, т.к. утерянные пакеты можно восстановить по известным пакетам, а эффективные коды позволяют уменьшить время передачи сообщений.

Связь важна также внутри одной системы. В современных сложных цифровых системах могут возникнуть большие потоки данных между подсистемами. Цифровые автопилоты, цифровые системы управления процессами, цифровые переключательные системы и цифровые системы обработки сигналов – все это системы, содержащие большие массивы цифровых данных, которые должны быть распределены между многими взаимосвязанными подсистемами. Эти данные должны быть переданы или по специально предназначенным для этого линиям, или посредством более сложных систем с шинами передачи данных и с разделением по времени.

Поэтому в настоящее время кодирование информации широко используется в радиоэлектронике и информатике:

- представление первичной информации в определенном виде, позволяющем упростить дальнейшую обработку;
- сжатие информации (сигналов и изображений) с целью повышения скорости передачи информации, более эффективного использования частотного ресурса, что нашло широчайшее применение в аудио- и видеотехнике, технике связи (форматы MP3, JPEG, MPEG и др.);
- контроль (обнаружение и исправление) ошибок в передаваемых, хранимых, обрабатываемых сообщениях телекоммуникационных, вычислительных, радиотехнических систем; для передачи сообщений в случае сильных непреднамеренных (типа белого шума) и преднамеренных помех (сосредоточенных в спектре сигнала – узкополосных или широкополосных с кодовыми видами модуляции) в телекоммуникационных и радиотехнических системах (системах дальнего космоса, спутниковых навигационных системах типа GPS "Navstar" или "ГЛАНАС", системах скрытой связи, радиолокационных системах дальнего обнаружения целей, системах точного наведения на цель с повышенной точностью, системах мобильной связи с кодовым разделением каналов CDMA), в качестве диагностических, сигнатурных последовательностей для контроля работоспособности и обеспечения отказоустойчивости цифровых устройств;
- защита от случайного и несанкционированного доступа к государственной, военной, коммерческой и личной информации (разработан государственный стандарт шифрования данных ГОСТ28147-89).

Модуль 1

ВВЕДЕНИЕ В ТЕОРИЮ КОДИРОВАНИЯ ИНФОРМАЦИИ

Цель модуля – изучение основ теории кодирования информации, включая модели каналов передачи, обработки, хранения информации и взаимосвязи открытых систем.

В результате изучения модуля студенты должны:

- знать структурные схемы моделей передачи, обработки, хранения информации и назначение их составляющих элементов;
- знать структурную схему модели взаимосвязи открытых систем и назначение всех составляющих ее уровней;
- знать коды, используемые для первичного кодирования информации и уметь выполнять первичное кодирование;
- иметь представление о системах счисления и их качественных характеристиках.

Содержание модуля

1.1. Модели каналов передачи, обработки и хранения информации

1.1.1. Обобщенная модель канала передачи информации

1.1.2. Эталонная модель взаимосвязи открытых систем

1.1.3. Первичное кодирование информации

1.2. Вопросы и задания для самопроверки

Кодирование – преобразование дискретного сообщения в дискретный сигнал, осуществляемое по определенному правилу. Восстановление дискретного сообщения по сигналу на выходе дискретного канала, осуществляемое с учетом правил кодирования, называется *декодированием.*

Код (от лат.codex – свод законов) есть совокупность условных сигналов, обозначающих дискретные сообщения.

Кодовая последовательность (комбинация) – представление дискретного сигнала.

Целью кодирования сообщений обычно являются:

- передача по общему каналу связи нескольких или многих сообщений для кодового разделения сигналов;
- повышение помехоустойчивости и достоверности передачи сообщений;
- более экономное использование полосы частот канала связи, т.е. уменьшение избыточности;

- уменьшение стоимости передачи и хранения сообщений;
- обеспечение скрытности передачи и хранения информации;
- преобразование любой информации независимо от ее происхождения и назначения в единую систему символов;
- приведение исходных символов в соответствие с характеристиками канала связи.

1.1. Модели каналов передачи, обработки и хранения информации

1.1.1. Обобщенная модель канала передачи информации

В самом общем виде модель канала передачи информации можно представить следующим образом (рис.1.1).

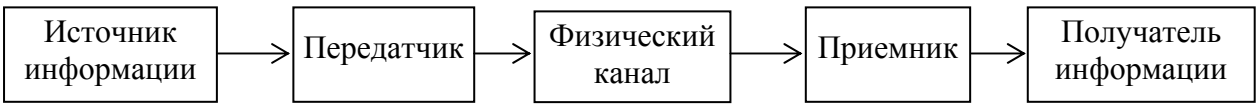


Рис. 1.1. Общее представление модели канала передачи информации

Такая модель содержит лишь основные элементы, присущие любой системе передачи информации, однако она не отражает тех действий, которые должны выполняться над информацией в процессе ее передачи.

Значительно более полной в этом смысле является классическая модель канала передачи (хранения, обработки, распределения) информации, подобная приведенной на рис. 1.2.

Кратко охарактеризуем назначение и функции элементов этой модели.

Источник информации или сообщения – это физический объект, система или явление, формирующие передаваемое сообщение в виде двоичных символов. Само сообщение – это значение или изменение некоторой физической величины, отражающие состояние объекта (системы или явления).

Кодер первичного кода преобразует двоичный код с выхода источника в первичный код, который более удобен для дальнейших преобразований последующими устройствами.

Кодер источника обеспечивает сокращение объема (сжатие) информации с целью повышения скорости ее передачи или сокращения полосы частот, требуемых для передачи. Кодирование источника иногда называют экономным, безызбыточным или эффективным кодированием, а также сжатием данных. Под эффективностью в данном случае понимается степень сокращения объема данных, обеспечиваемая кодированием.

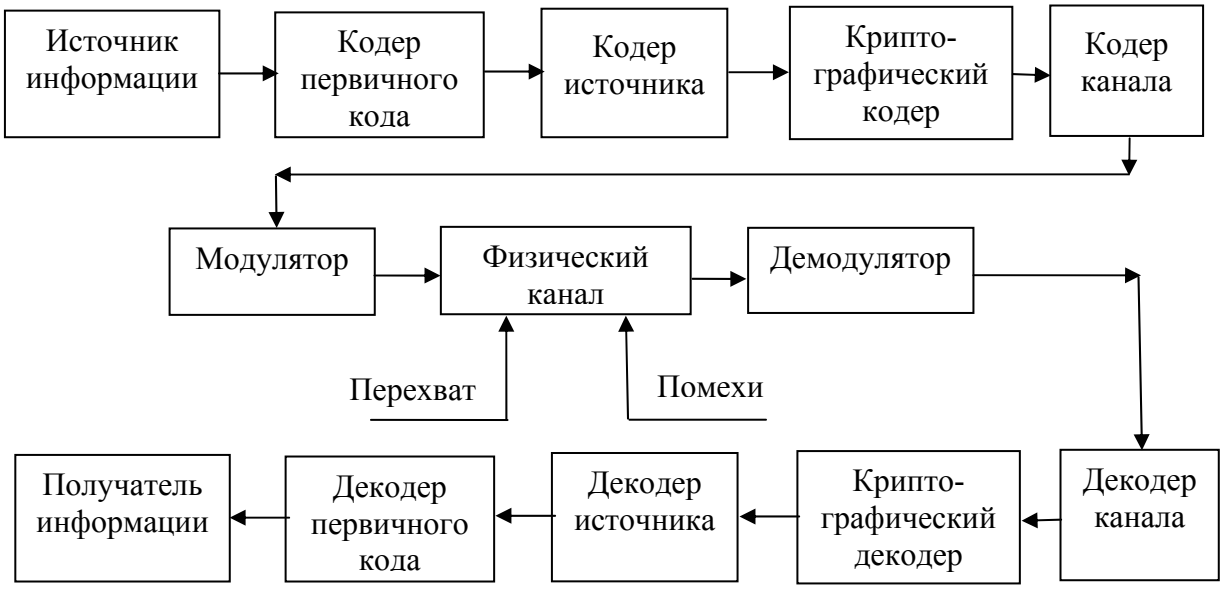


Рис. 1.2. Классическая модель канала передачи, хранения, обработки и распределения информации

Криптографический кодер выполняет криптографическое кодирование (шифрование) для обеспечения секретности передачи информации.

Кодер канала осуществляет помехоустойчивое кодирование, которое представляет собой способ обработки передаваемых данных, обеспечивающий уменьшение количества ошибок, возникающих в процессе передачи по каналу с помехами. Существует большое число различных методов помехоустойчивого кодирования информации, но все они основаны на следующем: при помехоустойчивом кодировании в передаваемые сообщения вносится специальным образом организованная избыточность (в передаваемые кодовые последовательности добавляются избыточные символы), позволяющая на приемной стороне обнаруживать и исправлять возникающие ошибки. Таким образом, если при кодировании источника производится устранение естественной избыточности, имеющей место в сообщении, то при кодировании в канале избыточность в передаваемое сообщение сознательно вносится.

Модулятор порождает множество непрерывных сигналов конечной длительности и реализует отображение выходных последовательностей кодера в это множество сигналов.

Физический канал – это вся аппаратура и вся физическая среда, через которую проходит сигнал на пути от выхода модулятора до входа демодулятора. Физический канал не обязательно представляет собой систему связи, работающую в режиме реального времени; он может быть системой хранения, обработки или распределения информации. Обычно выходной

сигнал канала является суммой входного сигнала, умноженного на коэффициент передачи, и случайного шума.

Демодулятор – это устройство, которое на основе наблюдения принятого сигнала оценивает, какой из возможных символов был передан. Вероятность того, что эта оценка окажется правильной, зависит от отношения мощности сигнала к мощности шума в используемой полосе частот, от искажения сигнала, вызываемого фильтрацией и нелинейными эффектами, и от используемой схемы демодулятора. Кроме этого, демодулятор часто выполняет еще одну функцию – передачу декодеру информации о степени надежности оценки каждого символа.

Декодер канала. Принятые последовательности в общем случае могут отличаться от переданных кодовых слов, т.е. содержать ошибки. Количество таких ошибок зависит от уровня помех в канале связи, скорости передачи, выбранного для передачи сигнала и способа модуляции, а также от способа приема (демодуляции) колебания. Задача декодера канала – обнаружить и, по возможности, исправить эти ошибки. Процедура обнаружения и исправления ошибок в принятой последовательности называется *декодированием канала.* Результатом декодирования является оценка информационной последовательности. Выбор помехоустойчивого кода, способа кодирования, а также метода декодирования должен производиться так, чтобы на выходе декодера канала осталось как можно меньше неисправленных ошибок.

Криптографический декодер выполняет декодирование зашифрованной информации по специальным алгоритмам.

Декодер источника. Поскольку информация источника в процессе передачи подвергалась кодированию с целью ее более компактного представления, необходимо восстановить ее к исходному (или почти исходному) виду по принятой последовательности. Такая процедура называется декодированием источника и может быть обратна операции кодирования (неразрушающее кодирование/декодирование).

Нужно сказать, что в последнее время экономное кодирование занимает все более заметное место в системах передачи информации, поскольку, вместе с помехоустойчивым кодированием, это оказалось самым эффективным способом увеличения скорости и качества ее передачи.

Отметим, что для построения эффективных кодеков для каналов с различной физической средой необходимо использовать более сложные модели каналов, чем модель рис. 1.2. Эти модели обязаны отражать специфику реальных каналов и в этом случае позволят синтезировать эффективные алгоритмы кодирования и конструкции систем. В моделях должны быть уста-

новлены связи кодеков с другими компонентами системы, например, с модулятором/демодулятором, учтены состояние канала в различные промежутки времени и ограничения, накладываемые системой на процессы кодирования информации. Все это приводит к задачам поиска эффективных решений не только по кодированию информации, но и по решению системных, конструкторско-технологических, экономических, организационных и других разнообразных задач.

1.1.2. Эталонная модель взаимосвязи открытых систем

Архитектуру эталонной модели взаимосвязи открытых систем (модель ВОС) предложила в конце 70-х годов с целью поддержки совместимости при проектировании сетей Международная организация по стандартизации (ISO) (рис.1.3). Модель ВОС представляет собой семиуровневую архитектуру.

Физический уровень реализует цифровой канал, который всегда ненадежен как с точки зрения помех, так и перехвата сообщений. Этот канал доставляет информацию от источника сообщений к приемнику и совместно используется многими источниками и приемниками (широковещательный канал).

На канальном уровне осуществляются любые из рассмотренных выше процедур кодирования по сжатию, шифрованию и помехоустойчивому кодированию информации, формированию и доставке пакетов данных между узлами того же физического канала таким образом, чтобы приемник мог извлечь их из потока бит.

Сетевой уровень направляет пакеты по пути, который может состоять из нескольких каналов, от источника к адресату. Типичные способы передачи для данного уровня – передача пакетов по виртуальным каналам с промежуточным накоплением или передача дейтаграмм.

Транспортный уровень модели ВОС управляет сквозной передачей пакетов. На этом уровне может осуществляться как коррекция ошибок в пакетах, так и повторная передача пакетов, ранее принятых с ошибкой. На транспортном уровне контролируется скорость передачи пакетов для того, чтобы избежать перегрузки отдельных участков сети.

Сеансовый уровень использует услуги транспортного уровня для установления и контроля соединения между оконечными системами.

На уровне представления в модели ВОС выполняются операции сжатия данных, обеспечения защиты, преобразования форматов таким обра-

зом, чтобы узлы сети, использующие разное представление информации, могли взаимодействовать эффективно и безопасно.

Последний, прикладной, уровень предоставляет такие популярные сетевые службы как службу каталога, перенос файлов, виртуальный терминал.

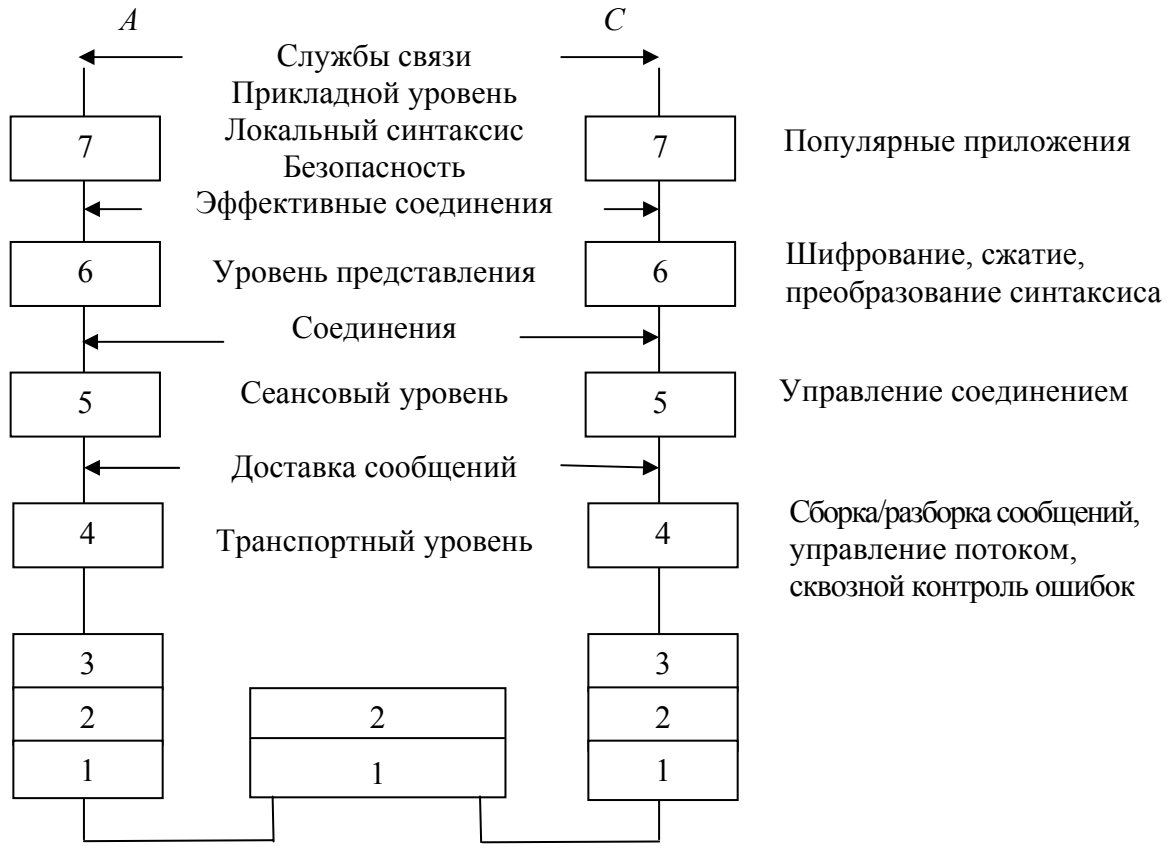


Рис. 1.3. Функции семи уровней модели ВОС
(Уровни 1, 2, 3 являются соответственно
физическим, канальным и сетевым уровнями)

1.1.3. Первичное кодирование информации

Дискретный поток двоичных символов, сформированный аналого-цифровым преобразователем (источником), представленный в виде двоичного кода (последовательности логических единиц и нулей), в большинстве случаев по ряду причин (недопустимо больших частотных и амплитудных искажений характеристик каналов связи, магнитных носителей информации, колебаний скорости записи и воспроизведения информации, а также из-за возникновения определенных комбинаций двоичных символов и т.д.) непригоден для передачи, хранения и обработки информации. Поэтому двоичный код с выхода источника, как правило, преобразуется в

первичный код, который удобен для дальнейшей обработки последующими устройствами. При длине n исходного кода число кодовых слов равно 2^n. Следовательно, для кодирования первичным кодом M сообщений необходимо, чтобы $M \le 2^n$.

Известно, что любое число можно представить с помощью многочлена,

$$Q = a_n \cdot q^{n-1} + a_{n-1}q^{n-2} + \ldots + a_1 q^0 = \sum_{i=1}^{n} a_i q^{i-1},$$

где q – основание системы счисления, q = 2, 3 …;

n – длина кодового слова;

i – номер разряда данного числа;

a_i – множитель, принимающий любые целочисленные значения от 0 до q-1, показывающий, сколько единиц i-го разряда содержится в числе.

Очевидно, что чем больше основание системы счисления q, тем меньшее число разрядов требуется для представления (кодирования) числа. Однако с увеличением q логические элементы, реализующие кодирование информации, должны иметь большее число устойчивых состояний. Это приводит к усложнению требований к аппаратуре формирования и распознавания различных символов.

Учитывая эти обстоятельства, целесообразно выбирать основания систем счисления q таким образом, чтобы обеспечивалось минимальное значение произведения $q \cdot n$ ($\min\ q \cdot n$) при кодовом представлении (выражении) любого числа. Зависимости q от n при представлении числа Q = 60 000 приведены в табл. 1.1.

Таблица 1.1

Зависимость q от n

Q	n	$q \cdot n$
1	60 000	60 000
2	16	32
3	10	30
4	8	32
16	4	64

Очевидно, что наиболее эффективной системой счисления является троичная. Незначительно уступает ей двоичная система счисления, которая обеспечивает следующие достоинства: минимальную сложность при реализации логических и арифметических операций и при переводе из одной системы счисления в другую; разделение символов сводится лишь к задаче обнаружения импульса (есть ли импульс или нет) и др.

Поэтому двоичная система счисления широко используется в различных информационных, радиотехнических и телекоммуникационных системах.

Однако представление информации в двоичном коде не всегда удобно, например, при вводе-выводе информации в ЭВМ, при записи и считывании информации с магнитных носителей информации и т.д. Поэтому в этих случаях требуется перекодирование исходной последовательности двоичных символов, для чего используются первичные коды. На практике широкое применение получили коды, которые, с одной стороны, легко сводятся к двоичной и десятичной системам счисления, а с другой – обеспечивают их более компактное представление. К таким системам счисления относятся восьмеричная, шестнадцатеричная и двоично-десятичная.

Для сохранения достоинств как двоичной, так и десятичной систем счисления используют двоично-десятичные коды с различными весами (весовые коды): 8-4-2-1, 5-1-2-1, 2-4-2-1 и др. Цифры в названии кода обозначают вес единиц в соответствующих двоичных разрядах. Двоично-десятичные коды чаще всего используют как промежуточные коды при вводе в ЭВМ данных, представленных в десятичном коде. Двоично-десятичные коды с весами 5-1-2-1 и 2-4-2-1 широко используются при поразрядном (посимвольном) уравновешивании в измерительной технике.

В двоично-десятичном коде каждую цифру десятичного числа записывают в виде четырехразрядного числа.

Например, числа 3, 7 и 10 в двоично-десятичном коде могут быть представлены следующим образом (табл. 1.2).

Таблица 1.2

Преобразование чисел в двоично-десятичный код

Число в десятичном коде	Двоично-десятичный код		
	8-4-2-1	5-1-2-1	2-4-2-1
1	0000 | 0001	0000 | 0001	0000 | 0001
3	0000 | 0011	0000 | 0011	0000 | 0011
7	0000 | 0111	0000 | 1010	0000 | 1101
10	0001 | 0000	0001 | 0000	0001 | 0000

Кроме рассмотренных весовых кодов используются невесовые коды, которые получили название *рефлексных* кодов. Наиболее типичным представителем данных кодов является код Грея. Основным свойством данного кода является то, что разряды кодовых слов не имеют весов и любые два соседних кодовых слова различаются только в одном разряде. Отсюда сле-

дует, что при ошибочной регистрации (воспроизведении) соседнего кодового слова ошибочно будет принят только один разряд в дешифрируемом кодовом слове. Код Грея имеет кодовое расстояние, равное единице, и не обладает свойствами обнаружения ошибок, а только минимизирует ошибки, связанные с аппаратурной неисправностью блока распознавания уровней соседних сигналов.

Правила перевода двоичного кода в код Грея и кода Грея в двоичный код поясняются соответственно рис. 1.4 и 1.5. Перевод числа из кода Грея в двоичный код выполняется по следующему правилу: первая единица со стороны старших разрядов остается без изменения, а последующие цифры (0 и 1) остаются без изменения, если число единиц, им предшествующих, четно, и инвертируются, если число единиц нечетно.

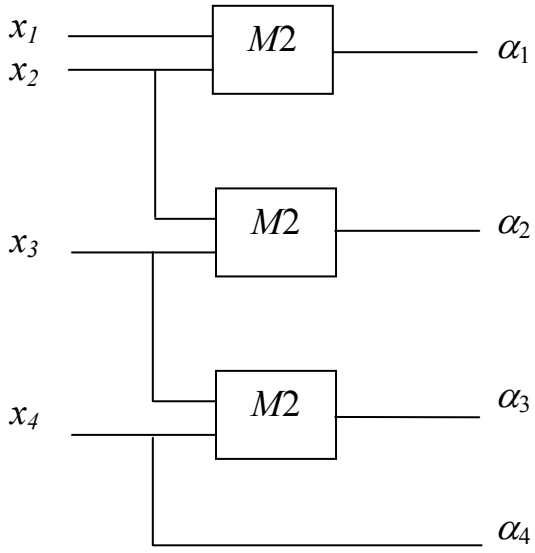


Рис. 1.4. Схема преобразования двоичного кода в код Грея

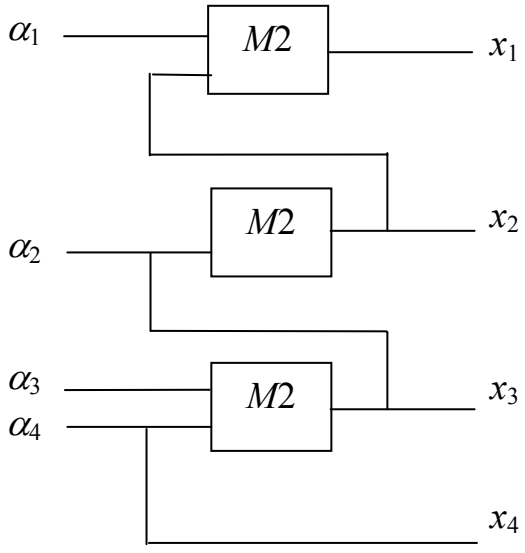


Рис. 1.5. Схема преобразования кода Грея в двоичный код

Код Грея широко применяется в телекоммуникационных системах с многократной относительной фазовой модуляцией, в аналого-цифровых преобразователях, в матричных кодерах телевизионных сигналов с использованием ЭЛТ и т.д.

1.2. Вопросы и задания для самопроверки

1.1. Дайте определение кодированию и декодированию.

1.2. Для каких целей используется кодирование информации?

1.3. Проверьте, сможете ли Вы представить обобщенную модель ка-

нала передачи, обработки и хранения информации.

1.4. Для каких целей используются: первичный кодер, кодер источника, криптографический кодер, кодер канала, модулятор, демодулятор, декодер канала, криптографический декодер, декодер источника, декодер первичного кода?

1.5. Знаете ли Вы расшифровку аббревиатуры ВОС?

1.6. Проверьте, сможете ли Вы представить эталонную модель взаимосвязи открытых систем?

1.7. Для решения каких задач предназначен соответствующий уровень модели ВОС?

1.8. Что понимают под основанием системы счисления?

1.9. Какая система счисления является наиболее эффективной?

1.10. Почему двоичная система счисления наиболее широко используется в радиотехнических и информационных системах?

1.11. Для решения какой задачи используются первичные коды?

1.12. Какие коды называются весовыми? Приведите примеры.

1.13. Представьте число 8 в известных Вам двоично-десятичных кодах.

1.14. Назовите основное свойство кода Грея.

1.15. Сколько ошибок может обнаружить код Грея?

1.16. Проверьте, сможете ли Вы представить схему преобразования двоичного кода в код Грея и схему обратного преобразования.

Модуль 2

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ТЕОРИИ КОДИРОВАНИЯ

Цель модуля – изучение студентами математических основ теории кодирования информации, включая элементы теории чисел и арифметику конечных полей.

В результате изучения модуля студенты должнызнать:

- знать алгоритм Евклида и уметь его использовать для нахождения наибольшего общего делителя целых чисел (многочленов);
- знать сравнение чисел (многочленов) и основные свойства сравнений;
- уметь выполнять алгебраические операции с матрицами;
- знать правила и уметь выполнять арифметические операции над целыми числами (многочленами) в конечных полях;
- иметь представление о группах и кольцах и знать их основные свойства.

Содержание модуля

2.1. Основы теории чисел, основы матричного анализа

2.1.1. Алгебраические операции на множестве целых чисел

Множество – это любая определенная совокупность объектов. Объекты, из которых составлено множество, называются его *элементами*. Элементы множества различны и отличны друг от друга.

Например, множество N натуральных чисел 1, 2, 3… . Множество Z целых чисел. Множество R вещественных чисел.

Если объект x является элементом множества M, то говорят, что x принадлежит M. Обозначение: $x \in M$. В противном случае говорят, что x не принадлежит M. Обозначается: $x \notin M$.

Рассмотрим множество целых чисел. Данное множество Z – счетное и состоит из элементов $0;\pm1;\pm2;\ \ldots;\pm n,\ \ldots$. На множестве Z определены две алгебраические операции – сложение и умножение. Эти операции обладают следующими общими свойствами (для любых $a,b,c \in Z$):

- ассоциативность: $a+(b+c)=(a+b)+c$; $a\cdot(b\cdot c)=a\cdot(b\cdot c)$;
- коммутативность: $b+a=a+b$; $a\cdot b=b\cdot a$;
- дистрибутивность $(a+b)\cdot c=a\cdot c+b\cdot c$;
- существование нейтрального элемента – 0 и 1 соответственно: $a+0=0+a=a$; $a\cdot 1=1\cdot a=a$.

Кроме того, операция сложения обладает следующим свойством: для каждого целого $a \in Z$ существует единственное целое b, такое, что $a+b=b+a=0$. Ясно, что здесь $b=-a$ – противоположное целое. Это свойство позволяет ввести вспомогательную операцию вычитания ($a-b=a+(-b)=c$ – целое число, разность чисел a и b (получаемое вычитанием b из a)).

Для каждого целого числа $a \in Z$ существует обратное (то есть такое число b, что $a\cdot b=1$), но оно является рациональным, а не целым числом. Следовательно, результат операции деления целого числа a на целое число $b\neq 0$ есть число рациональное и в редких случаях является целым.

Для любых целых чисел a и b, $b\neq 0$, существуют единственные целые числа q и r, $0\le r<|b|$, такие, что $a=b\cdot q+r$.

В этом равенстве r называют остатком, а q – частным (неполным частным при $r\neq 0$) от деления a на b.

Пример. а) $a=-21, b=4;$ тогда $q=-5, r=1;$

б) $a=-17, b=-5;$ тогда $q=4, r=3.$

Если $r=0$, то есть $a=b\cdot q$, то говорят, что a делится на b и на q или a является кратным чисел b и q или b и q делят a; b и q называют также делителями или множителями числа a.

Если в равенстве $a_1+a_2+\ldots+a_n=b_1+b_2+\ldots+b_m$ все слагаемые – целые числа и все, кроме, может быть, одного, делятся на целое d, то и это исключенное слагаемое делится на d.

2.1.2. Наибольший общий делитель чисел. Алгоритм Евклида

Если целые числа $a_1,a_2,\ldots a_n$ делятся на целое d, то d называют их *общим делителем*. В дальнейшем речь идет только о положительных целых делителях.

Максимальный из общих делителей целых чисел $a_1,a_2,\ldots a_n$ называется их *наибольшим общим делителем* и обозначается через НОД $(a_1,a_2,\ldots a_n)$.

Теорема 2.1. Если $a = b \cdot q + c$, то НОД (a,b) = НОД (b,c).

Эта теорема позволила Евклиду обосновать свой алгоритм.

*Теорема 2.2***.** Наибольший общий делитель целых чисел a и b $(a > b)$ равен последнему отличному от нуля остатку цепочки равенств:

$$a = b \cdot q_1 + r_1; \quad b = b \cdot q_2 + r_2;$$

$$r_{n-2} = r_{n-1} \cdot q_n + r_n \text{ , т. е. } r_n = \text{НОД } (a, b);$$

$$r_{n-1} = r_n \cdot q_{n+1}.$$

Пример. Найти НОД (72,26).

Решение. $72 = 26 \cdot 2 + 20;\ 26 = 20 \cdot 1 + 6;\ 20 = 6 \cdot 3 + 2;\ 6 = 2 \cdot 3.$

Следовательно, НОД (72,26) = 2 .

Теорема 2.2 формулирует алгоритм Евклида нахождения наибольшего общего делителя целых чисел. Он легко преобразуется в алгоритм нахождения наибольшего общего делителя не только двух, но и большего количества целых чисел. Алгоритм Евклида остается в классе самых быстрых алгоритмов нахождения наибольшего общего делителя целых чисел.

2.1.3. Сравнения и вычеты

Натуральное число $p>1$ называется *простым,* если оно делится только на 1 и на себя. Всякое натуральное число $n>1$ либо является простым числом, либо имеет простой делитель. На конец XX века наибольшим известным простым числом было число $2^{6972593}$-1, открытое 1.06.1999 г. Оно принадлежит классу так называемых чисел Мерсенна (имеющих вид $2^p - 1$, где p – простое).

Целые числа a и b называются *взаимно простыми*, если их наибольший общий делитель равен единице: НОД (a,b) =1.

Теорема 2.3 (основная теорема арифметики). Всякое целое число $n>1$ однозначно раскладывается в произведение простых сомножителей

$$n = p_1 \cdot p_2 \cdot \ldots . p_s .$$

Если в этом равенстве собрать одинаковые множители, то получим каноническое разложение целого числа

$$n = p_1^{r_1} \cdot p_2^{r_2} \cdot \ldots p_t^{r_t}.$$

Пример. а) $700 = 2 \cdot 350 = 2 \cdot 2 \cdot 175 = 2^2 \cdot 5^2 \cdot 7$;

По каноническому разложению целых чисел легко находится их наибольший общий делитель, наименьшее общее кратное, решаются иные задачи.

*Теорема 2.4***.** Пусть m – натуральное число, $m>1$. Для любых целых чисел a и b следующие условия равносильны:

a и b имеют одинаковые остатки от деления на m;

a - b делится на m, т.е. $a-b=mq$, для подходящего целого q;

$a=b+mq$ для некоторого целого q.

Целые числа a и b называются *сравнимыми по модулю m*, если они удовлетворяют одному из условий теоремы 2.4. Этот факт обозначают формулой $a=b(\mathrm{mod}\ m)$, или $a=b(m)$.

Пример. $(-7) \equiv 13(\mathrm{mod}\ 3) \equiv 19(\mathrm{mod}\ 3) \equiv 1(\mathrm{mod}\ 3)$.

Основные свойства сравнений:

1. Пусть $a = b\ (\mathrm{mod}\ m)$. Тогда $(a \pm c) \equiv (b \pm c)\ (\mathrm{mod}\ m)$ для всякого целого c, то есть к обеим частям сравнения можно добавить (или вычесть) одно и то же число.

2. Сравнения можно почленно складывать и вычитать. Если $a=b(m)$, $c=d(m)$, то $(a \pm c) = (b \pm d)(m)$.

3. Сравнения можно почленно перемножать: если $a=b(m)$, $c=d(m)$ то $ac=bd(m)$.

4. Сравнения можно почленно возводить в любую натуральную степень. Если $a=b(m)$, то $a^n = b^n(m)$.

5. Если в сравнении $a=b(m)$ числа a,b,m имеют общий множитель d, то на него сравнение можно сократить: $a/d=b/d(m/d)$.

6. Сравнение можно сократить на общий множитель, взаимно простой с модулем: если $a=a_1d$, $b=b_1d$, НОД (d,m) =1, то из сравнения $a_1d=b_1d(m)$ следует $a_1=b_1(m)$.

7. Рефлексивность: для любого целого a и всякого натурального $m>1$, $a=a(m)$.

8. Симметричность: если $a=b(m)$, то $b=a(m)$.

9. Транзитивность: если $a=b(m)$, $b=c(m)$, то $a=c(m)$.

При делении целых чисел на натуральное целое $m>1$ существует m различных остатков: 0,1,2, …, m-1. Соответственно этим остаткам Z разбивается на m непересекающихся классов сравнимых друг с другом чисел, имеющих один и тот же остаток. В соответствии с остатками от

деления на m эти классы обозначаются через $\bar{0},\bar{1},\ldots,\overline{m-1}$. Таким образом, $\bar{i}=\{mq+i \mid q\in Z\}$ для каждого i =0,1,2 …m-1. Любой представитель класса однозначно определяет свой класс, т.е. для каждого $mq+i$ класс $\overline{mq+i}=\bar{i}$. Поскольку остаток (по-латински residu) переводится на русский язык как вычет, то множество всех классов по данному модулю сравнимых друг с другом чисел называют множеством классов вычетов по модулю m и обозначают через Z/mZ. Таким образом, $Z/mZ=\{\bar{0},\bar{1},...,\overline{m-1}\}$ – множество из m элементов.

Малая теорема Ферма

Если n – простое и НОД(a, n)=1, то

$$a^{n-1}\equiv 1 \bmod n .$$

Функция Эйлера

Количество положительных целых, меньших n, которые взаимно просты с n, определяется с помощью функции Эйлера $\varphi(n)$:

Модуль	n простое	n^2	n^m	$p\cdot q$ (p и q простые)
$\varphi(n)$	n-1	$n(n-1)$	$n^{m-1}(n-1)$	$(p-1)\cdot(q-1)$

Обобщение Эйлера малой теоремы Ферма:

если НОД(a, n)=1, то

$$a^{\varphi(n)}\equiv 1 \bmod n .$$

Нахождение обратных величин

Если задано уравнение $(a\cdot x)\bmod n=1$, то величина $a^{-1}\equiv x \bmod n$ называется обратной величиной a по модулю n.

Обратная величина существует, если a и n – взаимно простые числа.

Способы нахождения обратных чисел

1. Перебором возможных значений.

Подставляя вместо x числа $1,2,\ldots,n-1$, добиваемся выполнения исходного уравнения.

Пример. $(5\cdot x)\bmod 7=1$, $x=\left(5^{-1}\right)\bmod 7=3$, т.к.

$$(5\cdot 3)\bmod 7=15\bmod 7=(15-7\cdot 2)\bmod 7=1 .$$

2. С помощью функции Эйлера $\varphi(n)$.

$$\left(a^{-1}\right)\bmod n=\left(a^{\varphi(n)-1}\right)\bmod n.$$

Пример.

$$x=\left(5^{-1}\right)\bmod 7=\left(5^{6-1}\right)\bmod 7=\left(\left(5^{2}\right)\bmod 7\cdot\left(5^{3}\right)\bmod 7\right)\bmod 7=\left(4\cdot 6\right)\bmod 7=3$$

3. С помощью алгоритма Евклида.

Алгоритм Евклида применяется для нахождения НОД чисел *a* и *b*. Однако его расширенный вариант можно использовать и для вычисления обратной величины.

Доказано, что при неотрицательных *a* и *b* можно найти такие целые числа u_1, u_2, u_3, что будет выполняться

$$a\cdot u_1+b\cdot u_2=u_3=\left(a,b\right).$$

Если выбрать $b=n$ и *a*, *n* – взаимно простые числа, тогда

$$a\cdot u_1+n\cdot u_2=1,$$
$$\left(a\cdot u_1+n\cdot u_2\right)\bmod n=\left(a\cdot u_1\right)\bmod n=1,$$
$$\left(a^{-1}\right)\bmod n=u_1\bmod n.$$

Т.е. для нахождения обратной величины необходимо вычислить $u_1\bmod n$. Эта задача решается в ходе вычисления НОД(*a*, *n*) в соответствии с алгоритмом Евклида. Дополнительно на каждом шаге вычисляются координаты двух векторов

$$\vec{u}=(u_1,u_2,u_3),\ \vec{v}(v_1,v_2,v_3).$$

Алгоритм вычисления u_1 имеет следующий вид

1. Начальные установки

$\vec{u}_0=(0,1,n)$, т.е. $u_1=0$, $u_2=1$, $u_3=n$. При этом $a\cdot 0+b\cdot 1=n$, т.е. $b=n$,

$\vec{v}_0=(1,0,a)$, т.е. $v_1=1$, $v_2=0$, $v_3=a$. При этом $a\cdot 1+n\cdot 0=a$.

2. Проверяем, выполняется ли $u_3 = 1$, если да, то алгоритм заканчивается.

3. Делим *n* на *a* (u_3 на v_3) и определяем

$q_1=\left[\dfrac{u_3}{v_3}\right]$ и значения векторов: $\vec{u}_1=\vec{v}_0;\ \vec{v}_1=\vec{u}_0-q_1\cdot\vec{v}_0$.

4. Вернуться к шагу 2.

На каждом шаге при расчетах используются результаты предыдущего

$$q_i = \left[\frac{u_3}{v_3}\right]_{i-1},\ \vec{u}_i = \vec{v}_{i\text{-}1},\ \vec{v}_i = \vec{u}_{i\text{-}1} - q_i \cdot \vec{v}_{i\text{-}1}.$$

При $u_3 = 1$ вычисления заканчиваются $\left(a^{-1}\right) \bmod n = u_1 \bmod n$, где u_1 – значение u_1, полученное на последнем шаге.

Пример. Пусть $n = 23$ и $a = 5$. Найти число *x*, обратное числу *a* по модулю *n*, т.е. найти $5^{-1} \bmod 23$.

Используя расширенный алгоритм Евклида, выполним вычисления.

q	u_1	u_2	u_3	v_1	v_2	v_3
–	0	1	n=23	1	0	a=5
4	1	0	5	-4	1	3
1	-4	1	3	5	-1	2
1	5	-1	2	-9	2	1
–	-9	2	1			

При $u_1 = -9$, $u_2 = 2$, $u_3 = 1$ выполняется уравнение $a \cdot u_1 + n \cdot u_2 = 1$, и $a^{-1} \bmod n = 5^{-1} \bmod 23 = (-9) \bmod 23 = 14$.

Итак, $x = 5^{-1} \bmod 23 = (-9) \bmod 23 = 14$.

2.1.4. Основы матричного анализа

Матрицей A размером $(n \times m)$ называется прямоугольная таблица, состоящая из n строк и m столбцов и содержащая nm элементов,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2m} \\ \cdot & & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nm} \end{bmatrix} = \left[a_{ij} \right],$$

Матрицы обычно обозначаются прописными буквами (*A*, *B*, *C*), если у матрицы *n* строк и *m* столбцов, то она имеет обозначение *A* $n \times m$.

Матрица называется *числовой*, если ее элементы a_{ij} числа; *функционной*, если a_{ij} – функции; *векторной*, если a_{ij} – вектора. Матрица, которая содержит один столбец или одну строку, называется *вектором.*

Множество элементов a_{ij}, для которых номер строки совпадает с номером столбца, называется *главной диагональю*. Если n равно m, то матрица называется *квадратной матрицей*. Если матрица размером $(n \times n)$

и a_{ij}=1 для $i=j$, остальные элементы равны нулю, то она называется *единичной* $(n\times n)$- *матрицей.* Единичная матрица обозначается через I (может обозначаться E). Примером единичной матрицы являются

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{и} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Две $(n\times m)$-матрицы A и B можно складывать по правилу

$$A+B=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \ldots & a_{1m}+b_{1m} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \ldots & a_{nm}+b_{nm} \end{bmatrix}.$$

$(n\times m)$- матрицу A можно умножать на элемент поля β по правилу

$$\beta A=\begin{bmatrix} \beta a_{11} & \beta a_{12} & \ldots & \beta a_{1m} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ \beta a_{n1} & \beta a_{n2} & \ldots & \beta a_{nm} \end{bmatrix}.$$

$(l\times n)$-матрицу A можно умножить на $(n\times m)$- матрицу B, получив результирующую $(l\times m)$- матрицу C, по правилу

$$c_{ij}=\sum_{k=1}^{n} a_{ik}\cdot b_{kj}\,, \qquad i=1,\ldots,l\,, \qquad j=1,\ldots,m.$$

Это произведение матриц обозначается через

$$C=AB\,.$$

Операция умножения двух матриц существует только для случая, когда число столбцов первой матрицы равно числу строк второй. Произведение матриц в общем случае не обладает свойством коммутативности, т.е. не всегда $AB=BA$. Если $AB=BA$, то матрицы A и B называются *коммутативными* или *перестановочными*.

Таким образом, результатом умножения двух матриц является матрица, а результатом векторно-матричного умножения является вектор-столбец, имеющий столько координат, сколько строк у матрицы A.

$$A \times x = y \text{, где} - \ y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}$$

Используя операции сложения матриц (векторов) и умножения матрицы на число, вектор $y=A\times x$ представим в виде

$$y = A \times x = \begin{pmatrix} a_{11} \\ a_{21} \\ \dots \\ a_{n1} \end{pmatrix} x_1 + \begin{pmatrix} a_{12} \\ a_{22} \\ \dots \\ a_{n2} \end{pmatrix} x_2 + \dots + \begin{pmatrix} a_{1m} \\ a_{2m} \\ \dots \\ a_{nm} \end{pmatrix} x_m = \begin{pmatrix} a_{11}x_1 + a_{21}x_2 + \dots + a_{1m}x_m \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2m}x_m \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nm}x_m \end{pmatrix}.$$

Таким образом, вектор $y=A\times x$ равен линейной комбинации столбцов матрицы A, а коэффициентами этой комбинации являются координаты вектора x.

Матрицу можно разбить на блоки по правилу

$$A = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right],$$

где A_{11}, A_{12}, A_{21} и A_{22} – меньшие матрицы, размеры которых очевидным образом дополняют друг друга до размеров матрицы A. А именно сумма числа строк матрицы A_{21}(или A_{22}) равна числу строк матрицы A; аналогичное утверждение выполняется для столбцов. Матрицу можно перемножить поблочно, а именно если

$$A = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right], \quad B = \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right], \quad C = AB,$$

то при условии корректного выбора размеров блоков (корректного в том смысле, что все произведения и суммы матриц определены)

$$C = \left[\begin{array}{c|c} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{21} \end{array}\right].$$

Транспонированной к $(n\times m)$- матрице A называется $(m\times n)$-матрица A^T, такая, что ${a_{ij}}^T = a_{ji}$. Таким образом, строками матрицы A^T служат столбцы матрицы A, а столбцами матрицы A^T служат строки матрицы A.

Обратной к квадратной матрице A называется квадратная матрица A^{-1} (если она существует), такая что $A^{-1}A = AA^{-1} = I$. Как можно сразу проверить, множество всех обратимых $(n \times n)$- матриц образует группу относительно операции умножения. Матрица, имеющая обратную, называется *невырожденной*; в противном случае она называется *вырожденной*. Если $C = AB$, то при условии, что A и B обратимы, $C^{-1} = A^{-1}B^{-1}$, так как $(B^{-1}A^{-1})C = I = C(B^{-1}A^{-1})$. Если у A или у B нет обратной матрицы, то и у C нет обратной матрицы.

Строки $(n \times m)$- матрицы A над полем $GF(q)$ можно рассматривать как множество m-мерных векторов над $GF(q)$. Пространство строк матрицы A определяется как множество всех линейных комбинаций векторов-строк матрицы A. Размерность пространства строк называется *рангом матрицы по строкам.* Аналогично столбцы матрицы A можно рассматривать как множество n-мерных векторов над $GF(q)$. Пространство столбцов матрицы A определяется как множество всех линейных комбинаций векторов-столбцов матрицы A, а размерность пространства столбцов называется *рангом матрицы по столбцам.* Множество всех векторов v, таких, что $Av^T = 0$, называется *нулевым пространством матрицы* A. Ясно, что нулевое пространство является подпространством в $GF^n(q)$. В частности, нулевое пространство является ортогональным дополнением пространства строк матрицы A, так как нулевое пространство можно задать как множеством всех векторов, ортогональных ко всем векторам пространства строк.

Элементарными операциями над строками матрицы называются следующие действия:

- перестановка двух произвольных строк;
- умножение произвольной строки на нулевой элемент поля;
- замена произвольной строки на сумму ее самой и некоторого кратного любой другой строки.

Каждая элементарная операция над строками обратима, и обратная операция имеет такой же вид. Каждая элементарная операция над строками $(n \times n)$- матрицы A может быть выполнена путем левого умножения A на соответствующим образом подобранную так называемую элементарную $(n \times n)$- матрицу F.

Элементарные операции над строками используются для приведения матрицы к стандартному виду, называемому *каноническим ступенчатым видом* и определяемому следующим образом:

1. ведущий ненулевой элемент каждой ненулевой строки равен единице;

2. все остальные элементы каждого столбца, содержащего такой ведущий элемент, равны нулю;

3. ведущий элемент любой строки находится правее любой расположенной выше строки. Нулевые строки расположены ниже всех ненулевых строк.

Примером матрицы, приведенной к каноническому ступенчатому виду, является

$$A=\begin{bmatrix}1 & 1 & 0 & 1 & 3 & 0\\ 0 & 0 & 1 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}.$$

Нулевая строка расположена снизу, и если удалить последнюю строку, то все столбцы единичной (3×3)- матрицы появятся среди столбцов матрицы, но в разбросанном виде. В общем случае если имеется k ненулевых строк и по меньшей мере такое же количество столбцов, то матрица в каноническом ступенчатом виде всегда будет содержать все столбцы единичной матрицы размера k. Частным случаем канонической ступенчатой формы матрицы является матрица вида

$$A=\left[I \mid P\right],$$

где I – единичная матрица.

С помощью элементарных операций над строками каждая матрица, содержащая по меньшей мере столько же столбцов, сколько и строк, может быть приведена к каноническому ступенчатому виду, но не к указанному выше частному случаю.

2.2. Элементы теории групп

2.2.1. Понятие алгебраической системы

Алгебраической структурой называется множество вместе с операциями (замкнутыми) на этом множестве. Структуру вместе со всеми теоремами, правилами вычислений и вывода называют *алгебраической системой.*

Бинарной алгебраической операцией на непустом множестве X называется всякое правило, по которому каждой упорядоченной паре (x,y) эле-

ментов из множества X ставится в соответствие один вполне определенный элемент z из X.

Обычно операции обозначаются знаками $*$, $\times$, $\bullet$, $\circ$, $+$, $\cdot$ и т.п. Воспользуемся первым из обозначений операции. Тогда $z=x*y$.

Если на множестве X заданы одна или несколько алгебраических операций, то говорят, что X есть алгебраическая система с данными операциями. Алгебраические системы различают по операциям на них и свойствам этих операций. Например $(Z,+,\cdot)$ – алгебраическая система целых чисел с операциями сложения и умножения, которую принято называть кольцом целых чисел.

2.2.2. Группы и их основные свойства

Группой называется непустое множество G с определенной на нем бинарной алгебраической операцией, которая обладает свойствами:

1. ассоциативность $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, для любых $a, b, c \in G$;
2. существует нейтральный элемент (единица), т.е. такой элемент e, что $g \cdot e = e \cdot g = g$ для каждого $g \in G$;
3. каждый элемент $g \in G$ имеет обратный, т.е. такой элемент $h \in G$, что $g \cdot h = h \cdot g = e$.

В любой группе единица и обратный элемент определяются однозначно в силу ассоциативности операции.

Абелевыми или *коммутативными* называют группы $(G, \cdot)$

4. со свойством $a \cdot b = b \cdot a$ для произвольных $a, b \in G$.

Например, $(Q,+)$; $(R,+)$; $(C,+)$ – множества рациональных, вещественных или комплексных чисел с операцией сложения. Это так называемые аддитивные группы (т.е. группы относительно сложения). Исторически сложилось, что все аддитивные группы коммутативны. Нейтральный элемент аддитивной группы называют нулем и обозначают символом 0, а обратный к a – противоположным и обозначают через $-a$. К *аддитивным* относятся группы $(M_{m\times n}(R),+)$ – множество прямоугольных $m \times n$ матриц с вещественными коэффициентами с операцией сложения матриц. Всякий линейный код является группой относительно операции сложения. *Мультипликативные группы* – группы с операцией умножения: $(Q^*,\bullet);(R^*,\bullet);(C^*,\bullet)$ и т.д., где $Q^* = Q \setminus \{0\}$, $R^* = R \setminus \{0\}$, $C^* = C \setminus \{0\}$.

Подгруппа в группе $(G, \cdot)$ – это подмножество H в группе G, которое в свою очередь является группой относительно той же операции. Этот

факт отмечают так: $H \le G$ или $H < G$, если включение $H \subset G$ строгое. Например, $(Z, +) < (Q, +) < (R, +) < (C, +)$, $C_n < C^*$, $(Z, +) > (2Z, +) > (4Z, +) > \ldots$, где $nZ = \{nq \mid q \in Z\}$.

Непустое подмножество H группы $(G, \cdot)$ является подгруппой тогда и только тогда, когда для произвольных $a, b \in H$ произведение $a \cdot b^{-1} \in H$.

В силу критерия подгруппы в любой группе G подмножество $\{e\}$ из одного нейтрального элемента e этой группы является подгруппой.

Пусть a – фиксированный элемент группы G. Пусть $H = \{a^0 = e, a, a^2, \ldots, a^{-1}, a^{-2}, \ldots\}$ - множество всевозможных степеней элемента a. Тогда H – подгруппа группы G, причем абелева.

Подгруппа $H = \{a^0 = e, a, a^2, \ldots, a^{-1}, a^{-2}, \ldots\}$ называется *циклической* подгруппой, порождённой элементом a, и обозначается через $\langle a \rangle$. Если найдется такой элемент $b \in G$, что $G = \langle b \rangle$, то такую группу называют циклической. Например $(Z, +) = \langle 1 \rangle$.

Смежные классы

Для заданных конечной группы G и подгруппы H существует операция, которая устанавливает некоторые взаимосвязи между G и H. Она называется разложением группы G на смежные классы по H. Обозначим через $h_1, h_2, h_3, \ldots$ элементы из H; через h_1 – единичный элемент. Построим таблицу следующим образом. Первая строка состоит из элементов подгруппы H, причем первым слева выписан единичный элемент h_1 и каждый элемент из H записан в строке один и только один раз. Выберем произвольный элемент группы G, не содержащийся в первой строке. Назовем его g_2 и используем в качестве первого элемента второй строки. Остальные элементы второй строки получаются умножением слева элементов подгруппы на этот элемент. Аналогично строим третью, четвертую и пятую строки: каждый раз в качестве элемента первого столбца выбираем не использованный на предыдущих шагах элемент группы G. Построение заканчивается тогда, когда после некоторого шага оказывается, что каждый элемент группы записан в некотором месте таблицы. Процесс обрывается в силу конечности G.

Первый элемент слева в каждой строке называется *лидером смежного класса.* Каждая строка таблицы называется *левым смежным классом*, а в случае абелевой группы – просто *смежным классом*. Если при построении разложения группы на смежные классы использовать правое умножение на элементы группы G вместо левого, то строки называются *правыми смежными классами*. В силу указанных выше правил построения разложение на смежные классы всегда представляется прямоугольной таблицей, все строки кото-

рой полностью заполнены. В разложении группы G на смежные классы каждый элемент из G встречается один раз и только один раз.

$$
\begin{array}{cccccccccc}
 & h_1 = 1 & & h_2 & & h_3 & \ldots & & h_n \\
g_2 * & h_1 = g_2 & g_2 * & h_2 & g_2 * & h_3 & \ldots & g_2 * & h_n \\
g_3 * & h_1 = g_3 & g_3 * & h_2 & g_3 * & h_3 & \ldots & g_3 * & h_n \\
. & & . & & . & & . & & \\
. & & . & & . & & . & & \\
. & & . & & . & & . & & \\
g_m * & h_1 = g_m & g_m * & h_2 & g_m * & h_3 & \ldots & g_m * & h_n
\end{array}
$$

Пример. Пусть $G = M_{1\times 4}(Z/2Z)$ – строки-матрицы с четырьмя координатами из $Z/2Z$. Это группа по сложению, состоящая из 16 элементов. Пусть H – подгруппа группы G, состоящая из четырех элементов $\overline{0} = (0000), \overline{e}_1 = (1011), \overline{e}_2 = (0101), \overline{e}_1 + \overline{e}_2 = (1110)$. Следовательно, индекс подгруппы H в группе G равен 16:4 = 4.

Выпишем таблицу смежных классов группы G по подгруппе H (табл. 2.1).

Таблица 2.1

Смежные классы группы G по подгруппе H

№	Класс $\overline{a} + H$	$\overline{a} + \overline{0}$	$\overline{a} + \overline{e}_1$	$\overline{a} + \overline{e}_2$	$\overline{a} + (\overline{e}_1 + \overline{e}_2)$
1	$\overline{0} + H = H$	(0000)	(1011)	(0101)	(1110)
2	(1000)+H	(1000)	(0011)	(1101)	(0110)
3	(0100)+H	(0100)	(1111)	(1001)	(1010)
4	(0010)+H	(0010)	(1001)	(0111)	(1100)

2.2.3. Кольца и поля, их свойства. Поля Галуа

Кольца

Кольцо представляет собой абстрактное множество, которое является абелевой группой и наделено дополнительной операцией.

Кольцом R называется множество с двумя определенными на нем операциями: первая называется сложением, вторая называется умножением, причем выполняются следующие аксиомы:

1. Относительно сложения (+) R является абелевой группой.

2. Замкнутость. Произведение $(a \cdot b)$ принадлежит R для любых a и b из R;

3. Закон ассоциативности

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c;$$

4. Закон дистрибутивности:

$$a \cdot (b + c) = a \cdot b + a \cdot c,$$
$$(b + c) \cdot a = b \cdot a + c \cdot a.$$

Сложение в кольце всегда коммутативно, а умножение не обязательно должно быть коммутативным. Коммутативное кольцо – это кольцо с коммутативным умножением: $a \cdot b = b \cdot a$ для всех a и b из R. Дистрибутивный закон в определении кольца связывает операции сложения и умножения.

Важнейшими примерами колец являются кольцо Z целых чисел и кольцо Z/qZ целых чисел по модулю q.

Для произвольных элементов a и b из кольца R выполняются равенства

$$a \cdot 0 = 0 \cdot a = 0;$$
$$a \cdot (-b) = (-a) \cdot b = -a \cdot b.$$

Относительно операции сложения кольцо содержит единицу, называемую «нулем». Относительно операции умножения кольцо не обязательно имеет единицу, но если единица есть, то она единственна. Кольцо, обладающее единицей относительно умножения, называется кольцом с единицей. Эта единица называется «единицей» и обозначается символом 1. Тогда для всех R имеет место равенство

$$1 \cdot a = a \cdot 1 = a.$$

Относительно операции сложения каждый элемент кольца имеет обратный. Относительно операции умножения обратные к данному элементу не обязательно существуют, но в кольце с единицей обратные могут существовать. Это означает, что для заданного элемента a может существовать элемент b, такой, что $a \cdot b = 1$.

Если кольцо содержит единицу, то можно произвольное число раз сложить ее саму с собой или вычесть из себя самой, получив, таким образом, бесконечную в обе стороны последовательность

$$\ldots, -(1 + 1 + 1), -(1 + 1), -1, 0, 1, 1 + 1, 1 + 1 + 1, \ldots$$

Эти элементы называются целыми кольца и иногда обозначаются просто как 0, ±1, ±2, ±3, …. Кольцо может содержать как конечное, так и бесконечное множество целых. Число целых в кольце с единицей называет-

ся его *характеристикой*. Если характеристика кольца равна конечному целому числа q, то целые этого кольца могут быть записаны в виде множества

$$(1, 1 + 1, 1 + 1 + 1, \ldots.),$$

или в более простом, используемом для обычных целых чисел, обозначении

$$\{1, 2, 3, \ldots, q - 1, 0\}$$

Это подмножество является подгруппой аддитивной группы кольца, на самом деле это циклическая подгруппа, порождаемая элементом 1. Следовательно, если характеристика кольца – конечное целое число, то сложение в кольце является сложением по модулю q. Если характеристика бесконечна, то целые кольца складываются как целые числа. Следовательно, каждое кольцо с единицей содержит подмножество, которое ведет себя относительно сложения либо как Z, либо как Z/qZ.

Конечные поля

Конечное поле, называемое также полем Галуа и обозначаемое $GF(q)$, – это конечное множество с двумя определенными на нем операциями – сложением и умножением, состоящее из q элементов, в котором определены правила для выполнения арифметических операций и обладают свойствами:

1. множество образует абелеву группу по сложению;
2. поле замкнуто относительно умножения, и множество ненулевых элементов образует абелеву группу по умножению;
3. выполняются правила ассоциативности, коммутативности и дистрибутивности для любых a, b, c из поля;
4. поле всегда содержит мультипликативную единицу и аддитивную единицу для любого элемента поля;
5. для любого элемента поля существует обратный элемент по сложению и обратный элемент по умножению.

Конечные поля существую в случае, если число элементов поля является простым число или степенью простого числа.

Наименьшее поле $GF(2)$ обязательно содержит нулевой элемент и единичный элемент. Не существует другого поля с двумя элементами.

Поле $GF(3) = \{0,1,2\}$ с операциями

+	0 1 2
0	0 1 2
1	1 2 0
2	2 0 1

•	0 1 2
0	0 0 0
1	0 1 2
2	0 2 1

Для выполнения вычитания или деления необходимо найти соответствующий обратный элемент, а затем выполнить сложение или умножение. Например, в поле $GF(5)$: 3-4=3+1=4; $3/4=3\cdot 4^{-1}=3\cdot 4=2$.

Поле обладает всеми свойствами кольца, а также важным дополнительным свойством – в нем всегда возможно сокращение. Сокращение представляет собой слабую форму деления и означает, что если $ab = ac$, то $b = c$.

Некоторые кольца могут также удовлетворять этому условию сокращения, но все-таки не быть полями.

Примитивным элементом поля $GF(q)$ называется такой элемент α, что все элементы поля, за исключением нуля, могут быть представлены в виде степени элемента α. Например, в поле $GF(5)$ $2^1 = 2$, $2^2 = 4$, $2^3 = 3$, $2^4 = 1$, т.к. 2 является примитивным элементом поля $GF(5)$.

В каждом поле Галуа имеется примитивный элемент. В поле $GF(8)$ порядок каждого ненулевого элемента делит 7. Т.к. 7 – простое число, то каждый элемент, за исключением нуля и единицы, имеет порядок, равный 7, и, следовательно, примитивен. Поле $GF(8)$ можно построить с помощью многочлена $p(x) = x^3 + x + 1$. Основываясь на примитивном элементе $\alpha = x$,

$$\alpha = x,\ \alpha^2 = x^2,\ \alpha^3 = x + 1,\ \alpha^4 = x^2 + x,\ \alpha^5 = x^2 + x + 1,\ \alpha^6 = x^2 + 1,$$
$$\alpha^7 = 1 - \alpha^0.$$

2.2.4. Основы полиномиальных вычислений

Многочленом над полем $GF(q)$ называется математическое выражение

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + ... + f^1x_1 + f_0,$$

где символ x – называется неопределенной переменной, коэффициенты $f_{n-1},...,f_0$ – принадлежат полю $GF(q)$, а индексы и показатели степени являются целыми числами. *Нулевым многочленом* называется многочлен $f(x)=0$. *Приведенным многочленом* называется многочлен, старший коэффициент которого f_{n-1} равен 1. Два многочлена равны, если равны их коэффициенты f_i при всех соответствующих степенях.

Степенью ненулевого многочлена $f(x)$ называется индекс старшего коэффициента f_{n-1}; степень многочлена $f(x)$ обозначается через $\deg f(x)$. Степень ненулевого многочлена всегда конечна, а степень нулевого многочлена по соглашению полагается равной бесконечности.

Множество всех многочленов над полем $GF(q)$ образует кольцо относительно сложения и умножения, определяемых по обычным правилам сложения и умножения многочленов. Такое полиномиальное кольцо можно определить для каждого поля Галуа $GF(q)$. Это кольцо обозначается через $GF(q)[x]$.

Суммой двух многочленов $f(x)$ и $g(x)$ из $GF(q)[x]$ называется многочлен из $GF(q)[x]$**,** определяемый следующим образом,

$$f(x) + g(x) = \sum_{i=0}^{n-1} \left(f_i + g_i \right) \cdot x^i .$$

Например, над $GF(2)$, сложению двоичных многочленов соответствует сложение по mod 2 коэффициентов при одинаковых степенях x.

$$\begin{array}{r} + \quad X^3+X^2+0\cdot X+1 \\ \underline{\qquad\qquad X+1} \\ X^3+X^2+X+0 \end{array} = X^3+X^2+X ,$$

Произведение двух многочленов из $GF(q)[x]$ называется многочлен из $GF(q)[x]$, определяемый равенством,

$$f(x)\cdot g(x) = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{i} f_j \cdot g_{i-j} \right) \cdot x^i .$$

Например, над $GF(2)$ умножение выполняется следующим образом

$$\begin{array}{r} X^3 + X^2 + 0 + 1 \\ \underline{\qquad\qquad X + 1} \\ X^3 + X^2 + 0 + 1 \end{array}$$

$$\begin{array}{l} \underline{X^4 + X^3 + 0 + X} \\ X^4 + 0 + X^2 + X + 1 \end{array} = X^4 + X^2 + X + 1 ,$$

то есть произведение получается по обычному правилу перемножения степенных функций, однако получаемые коэффициенты при данной степени X складываются по mod 2.

Кольцо многочленов во многом аналогично кольцу целых чисел. Говорят, что многочлен $s(x)$ делится на многочлен $r(x)$ или что $r(x)$ делит $s(x)$, если существует многочлен $a(x)$, такой, что $r(x) \cdot a(x) = s(x)$. Многочлен $p(x)$, делящийся только на многочлены $\alpha p(x)$ или α, где α – произвольный ненулевой элемент поля $GF(q)$, называется *неприводимым многочленом*. Приведенный неприводимый многочлен называется *простым многочленом*.

Наибольший общий делитель двух многочленов $r(x)$ и $s(x)$ обозначается через НОД($r(x)$,$s(x)$) и определяется как приведенный многочлен наибольшей степени, делящий одновременно оба из них. Если НОД($r(x)$, $s(x)$)=1, то многочлены $r(x)$ $s(x)$ называют взаимно простыми.

Для каждой пары многочленов $c(x)$ и $d(x)$, $d(x)\neq 0$ существует единственная пара многочленов $Q(x)$ (частное) и $s(x)$ (остаток), таких, что

$$c(x)=Q(x)d(x)+s(x)$$
$$\deg s(x)<\deg d(x).$$

Практическое вычисление частного и остатка выполняется с помощью простого правила деления многочленов «уголком».

Иными словами, деление многочленов производится по правилам деления степенных функций, при этом операция вычитания заменяется суммированием по mod 2 в поле $GF(2)$.

$$\begin{array}{rl|l} & X^4+0\cdot X^3+X^2+X+1 & X+1 \\ \hline \oplus & X^4+X^3 & X^3+X^2+1 \\ \hline & \oplus\ X^3+X^2+X+1 & \\ & X^3+X^2 & \\ \hline & \oplus\ X+1 & \\ & X+1 & \\ \hline & 0 & \end{array}$$

Для теории кодирования больший интерес представляет остаток. Часто остаток называют вычетом многочлена $c(x)$ по модулю многочлена $d(x)$.

Ненулевой многочлен $p(x)$ над некоторым полем однозначно (с точностью до порядка следования множителей) разлагается в произведение элемента поля и простых многочленов над этим полем.

Теорема 2.5. (алгоритм Евклида для многочленов).

Наибольший общий делитель двух многочленов $r(x)$ и $s(x)$ над полем $GF(q)$ можно вычислить с помощью итеративного применения алгоритма деления. Если $\deg s(x) \geq \deg r(x) \geq 0$, то последовательность вычислений такова

$$\begin{aligned} s(x) &= Q_1(x)\cdot r(x) + r_1(x), \\ r(x) &= Q_2(x)\cdot r_1(x) + r_2(x), \\ r_1(x) &= Q_3(x)\cdot r_2(x) + r_3(x), \\ &\vdots \\ r_{n-1}(x) &= Q_{n+1}(x)\cdot r_n(x), \end{aligned}$$

и процесс обрабатывается, как только получается равный нулю остаток. Тогда $r_n(x) = \alpha\,\text{НОД}[r(x), s(x)]$, где α – некоторый скаляр.

Для произвольного элемента β поля $GF(q)$ можно вычислить значение многочлена над $GF(q)$ в этой точке, подставив элемент β вместо неопределенной переменной. Например, пусть над $GF(3)$

$$p(x)=2x^5+x^4+x^2+2$$

Тогда $p(0)=2\cdot 0^5+0^4+0^2+2=2$; $p(1)=2\cdot 1^5+1^4+1^2+2=0$; $p(2)=2\cdot 2^5+2^4+2^2+2=2$.

Если $p(\beta)=0$, то элемент β называется *корнем многочлена* $p(x)$ или *корнем уравнения* $p(x)=0$. Многочлен не обязательно имеет корни в своем собственном поле.

Конечные поля можно построить из колец многочленов таким же образом, каким были построены поля из колец целых чисел. Ограничимся рассмотрением только приведенных многочленов, так как это ограничение снимает ненужную неопределенность построения.

Для произвольного приведенного многочлена $p(x)$ ненулевой степени над полем F кольцом многочленов по модулю $p(x)$ называется множество всех многочленов над F, степень которых не превосходит степени многочлена $p(x)$, с операциями сложения и умножения многочленов по модулю $p(x)$. Это принято обозначать через $F(x)/(p(x))$.

Произвольный элемент $r(x)$ кольца $F[x]$ можно отобразить в элемент кольца $F[x]/(p(x))$ с помощью соответствия $r(x) \to R_{p(x)}[r(x)]$. Два элемента $a(x)$ и $b(x)$ из $F[x]$, отображаемые в один и тот же элемент из $F[x]/(p(x))$, называются *сравнимыми*,

$$a(x) \equiv b(x) (\text{mod}\, p(x)).$$

Тогда $b(x) = a(x) + Q(x)\cdot p(x)$ для некоторого многочлена $Q(x)$.

Множество $F[x]/(p(x))$ является кольцом. Выберем в кольце многочленов над $GF(2)$, например, многочлен $p(x) = x^3 + 1$. Тогда кольцо многочленов по модулю $p(x)$ равно $GF(2)[x]/(x^3+1)$. Оно состоит из элементов $\{0,1,x,x+1,x^2,x^2+1,x^2+x,x^2+x+1\}$. В этом кольце умножение выполняется, например, следующим образом:

$$(x^2+1)\cdot(x^2) = R_{x^3+1}[(x^2+1)\cdot x^2] = R_{x^3+1}[x\cdot(x^3+1)+x^2+x] = x^2+x,$$

где использована редукция по правилу $x^4 = x(x^3+1)+x$.

Кольцо многочленов по модулю приведенного многочлена $p(x)$ является полем тогда и только тогда, когда многочлен $p(x)$ прост.

Если над полем $GF(q)$ найден простой многочлен степени n, то можно построить поле Галуа, содержащее q^n элементов. В этом построении элементы поля представляются многочленами над $GF(q)$ степени не выше $n-1$. Всего существует q^n таких многочленов, и, следовательно, их столько же, сколько элементов в поле.

Одним из способов проверки простоты многочленов является метод проб и ошибок, т.е. непосредственная проверка всех возможных разложений, хотя для многочленов высоких степеней для этого потребуется ЭВМ.

2.3. Вопросы и задания для самопроверки

2.1. Какими свойствами обладают алгебраические операции на множестве целых чисел?

2.2. Какой делитель называется наибольшим общим делителем целых чисел?

2.3. Является ли число 1 простым?

2.4. Сформулируйте основную теорему арифметики.

2.5. Назовите условия, на основании которых можно утверждать, что числа сравнимы по модулю m.

2.6. Назовите элементарные операции, которые можно производить над строками матрицы.

2.7. В каком случае матрица называется транспонированной?

2.8. Что Вы понимаете под множеством классов вычетов?

2.9. Что означает формула $a=b(\mathrm{mod}\ m)$?

2.10.Запишите правила перемножения элементов двух матриц.

2.11. Приведите пример алгебраической системы.

2.12. В каком случае группа называется абелевой?

2.13. Сформулируйте алгоритм разложения группы на смежные классы?

2.14. Назовите все аксиомы, которыми характеризуется кольцо и приведите примеры колец.

2.15. Что Вы понимаете под полем Галуа?

2.16. Какими свойствами обладает конечное поле?

2.17. Какой многочлен называется приведенным?

2.18. Дайте определение примитивному элементу поля Галуа.

2.19. Запишите последовательность вычислений для нахождения НОД двух многочленов.

2.20. Запишите формулы для сложения и умножения многочленов.

2.4. Практическое занятие №1

Алгебраические операции на множестве целых чисел и операции с матрицами

Теория и методы решений для практического занятия представлены в разделе 2.1.

Тестовые задания

1. Какие алгебраические операции определены на множестве целых чисел?

a) сложение;
b) умножение;
c) сложение и умножение;
d) сложение, умножение, вычитание и деление.

2. Какие числа называются взаимно простыми?

a) числа, у которых наибольший общий делитель равен 1;
b) числа, являющиеся простыми;
c) числа, у которых наибольший общий делитель равен одному из чисел;
d) числа, которые делятся друг на друга без остатка.

3. Какие числа являются сравнимыми по модулю 7?

a) 5 и 12; b) 13 и 22; c) 9 и 44; d) 19 и –2.

4. Матрица, которая содержит один столбец или одну строку называется

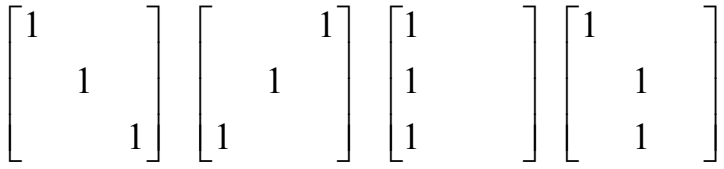
a) диагональной;
b) единичной;
c) нулевой;
d) вектором.

5. Операция умножения двух матриц существует только для случая:

a) число столбцов первой матрицы равно числу строк второй;
b) число строк первой матрицы равно числу строк второй;
c) число столбцов первой матрицы равно числу столбцов второй;
d) число строк первой матрицы равно числу столбцов второй.

6. Выберите правильный ответ на вопрос: какая матрица называется единичной?

a) 1; b) 1 и 2; c) 3; d) 4; e) 2 и 3



1 2 3 4

7. Достоинством алгоритма Евклида является

a) точность;

b) быстродействие;

c) простота программной реализации.

8. Множество классов вычетов целых чисел по модулю m обозначают

a) Z/mZ; b) mZ/Z; c) mZ; d) Zm/Z

9. Какое выражение позволяет определить необходимое число операций типа сложение-вычитание при умножении вектора на матрицу размером $m \times n$?

a) $m \cdot n$; b) $m \cdot (n\text{-}1)$; c) $n \cdot (m\text{-}1)$ e) $m^2 \cdot (n\text{-}1)$; d) $(m\text{-}1) \cdot (n\text{-}1)$; e) $m^2 \cdot (n\text{-}1)$.

10. Какое число операций типа умножение потребуется при умножении матрицы A размером 5×10 на матрицу B размером 10×20?

a) 100; b) 500; c) 1000; d) 1500; e) 2000.

Задачи

1. По алгоритму Евклида найдите наибольший общий делитель чисел

a)1056 и 84; b) 5046 и 104; c) 8075456 и 6896466; в) 501173314 и 821084428.

2. Получите каноническое разложение чисел: 450, 1026, 88200, 645678.

3. Найти сумму матриц $A+B$, $A = \begin{bmatrix} 2 & 3 & -1 \\ 4 & -3 & 2 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 4 & -3 \\ 5 & -2 & 1 \end{bmatrix}$.

4. Пусть $A = \begin{bmatrix} 3 & -1 & 4 \\ 2 & 0 & 6 \end{bmatrix}$, $x = \begin{pmatrix} -2 \\ 1 \\ 3 \end{pmatrix}$. Найти $y = A \times x$.

5. Существуют ли произведения матриц $A = \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}$ и $B = \begin{bmatrix} -1 & 0 & 3 \\ 1 & 2 & -9 \end{bmatrix}$?

Если – да, то найти $A \times B$.

6. Найти произведения матриц $A_{4\times4} \times I_{4\times4}$ и $I_{4\times4} \times A_{4\times4}$.

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 \end{bmatrix}$$

7. Определить, являются ли матрицы A и B коммутативными,

$$A=\begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix},\ B=\begin{bmatrix}2 & -2\\ -1 & 1\end{bmatrix}.$$

8. Перемножить две матрицы $A\times B$ в поле $GF(2)$, $A=\begin{bmatrix}1 & 0 & 0 & 1\\ 0 & 1 & 1 & 0\\ 1 & 1 & 1 & 0\end{bmatrix}$ и $B=\begin{bmatrix}0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\\ 1 & 0 & 1 & 0\\ 1 & 1 & 1 & 1\end{bmatrix}$. В данном примере правила сложения и умножения элементов матриц необходимо выполнять в соответствии с арифметикой конечных полей (см. раздел 2.2.3.).

9. Пусть $A=\begin{bmatrix}1 & 2 & 1\\ 1 & 2 & 0\end{bmatrix}$, $x=\begin{pmatrix}2\\ 2\\ 1\end{pmatrix}$. Найти $y=A\times x$ в поле $GF(3)$.

10. Приведите матрицу

$$A=\begin{bmatrix}1 & 1 & 0 & 0 & 0 & 1 & 1\\ 0 & 1 & 0 & 1 & 1 & 0 & 1\\ 0 & 1 & 1 & 0 & 1 & 1 & 0\end{bmatrix}$$

к каноническому виду.

2.5. Практическое занятие №2

Арифметика полей Галуа

Теория и методы решений для практического занятия представлены в разделе 2.2.

Тестовые задания

1. Группа, в которой каждый элемент может быть записан в виде степени некоторого фиксированного элемента, называется

a) периодической;

b) циклической;

c) степенной;

d) переменной.

2. Пусть G – группа и H – подмножество в G. Тогда H называется подгруппой группы G, если

a) в данном подмножестве H выполняются те же свойства, что и в заданной группе G;
b) если H само является группой относительно ограничения групповой операции на H;
c) если H является подмножеством G.

3. В разложении группы на смежные классы каждый элемент группы встречается
a) с некоторой последовательностью;
b) один и только один раз;
c) один раз в каждой строке;
d) один раз в каждом столбце.

4. Относительно какой операции кольцо содержит имеет единичный элемент, называемый «нулем»?
a) сложения;
b) умножения;
c) вычитания;
d) деления.

5. Что представляет собой наименьшее поле?
a) нулевой элемент;
b) единичный элемент;
c) нулевой и единичный элементы;
d) содержащее наименьшее количество элементов.

6. Некоторое подмножество поля будет подполем, если данное подмножество
a) незамкнуто относительно умножения;
b) содержит нулевой элемент и замкнуто относительно умножения;
c) содержит ненулевой элемент, замкнуто относительно сложения и умножения;
d) содержит ненулевой элемент и замкнуто относительно сложения.

7. Число элементов любого поля Галуа равно
a) степени простого числа;
b) сумме простых чисел;
c) ограничений нет;
d) простому числу.

8. В каком случае два многочлена равны
a) если равны все их соответствующие показатели степени;
b) если равны все соответствующие коэффициенты f_i являются целыми числами;

c) если равны все их соответствующие коэффициенты f_i;
d) если равны все их соответствующие неопределенные переменные x_i;

9. Как обозначается степень многочлена $f(x)$?

a) ged $f(x)$; b) deg $f(x)$; c) dec $f(x)$; d) ded $f(x)$.

10. Известно, что если над полем $GF(q)$ найден простой многочлен степени n, то можно построить поле Галуа. Сколько многочленов существует в этом поле?

a) q^{n-1}; b) q^n; c) n^{q-1}; d) n^q.

Задачи

1. Построить группу для множества чисел {0, 1, 2, 3, 4} с групповой операцией сложения по модулю 5 и проверить выполняются ли все свойства группы.

2. Постройте таблицы умножения и сложения для поля $GF(7)$ и найдите обратные по умножению элементы для каждого ненулевого элемента поля.

3. В поле $GF(11)$ выполните следующие операции:
a) вычитание: 3-5; 7-8; 8-4; 4-2;
b) деление: 3/5; 7/8; 8/4; 4/2.

4. Найти примитивный элемент поля $GF(7)$.

5. Над $GF(2)$ выполнить умножение многочленов: $p_1(x)=x^5+1$ и $p_2(x)=x^4+x^3+x+1$.

6. Над $GF(2)$ выполнить деление многочленов $p_1(x)/p_2(x)$, $p_1(x)=x^6+1$, $p_2(x)=x^3+x+1$.

7. Сколько различных приведенных многочленов над $GF(2)$ делят многочлен x^6-1.

8. Пусть над $GF(2)$ $p_1(x)=x^3+1$ и $p_2(x)=x^4+x^3+x+1$, найти НОД[$p_1(x)$, $p_2(x)$].

9. Выяснить, являются ли многочлены $p_1(x)$, $p_2(x)$, $p_3(x)$ неприводимыми над полем $GF(2)$, $p_1(x)=x^3+1$, $p_2(x)=x^4+x^2+1$, $p_1(x)=x^2-2$.

10. Построить таблицы сложения и умножения поля $GF(2^3)$, используя неприводимый многочлен x^3+x^2+1 над $GF(2)$.

Модуль 3

СТАТИСТИЧЕСКОЕ КОДИРОВАНИЕ ИНФОРМАЦИИ

Цель модуля – изучение методов и алгоритмов экономного кодирования информации, приобретение навыков по эффективному кодированию.

В результате изучения модуля студенты должны:

- знать основные информационные характеристики источника сообщений;
- знать алгоритмы Шеннона – Фано и Хаффмена и уметь использовать их для кодирования информации;
- знать алгоритмы кодирования длин повторений и уметь их использовать для сжатия изображений;
- иметь представление о количестве информации и энтропии, о методах спектрального сжатия информации.

Содержание модуля

3.1. Элементы теории информации и основы статистического кодирования

3.1.1. Количественная оценка информации, энтропия источника сообщений

3.1.2. Энтропия сложных сообщений

3.1.3. Основы статистического кодирования

3.2. Эффективные коды

3.2.1. Коды Шеннона – Фано

3.2.2. Коды Хаффмена

3.2.3. Недостатки системы эффективного кодирования

3.3. Кодирование изображений и сжатие информации с помощью спектральных преобразований

3.3.1. Кодирование длин повторений

3.3.2. Сжатие информации с помощью спектральных преобразований

3.3.3.Кодирование изображений посредством преобразований

3.4. Вопросы и задания для самопроверки

3.5. Лабораторная работа №1 «Эффективное кодирование информации» (Методические указания к выполнению лабораторных работ по курсу «Прикладная теория кодирования» для студентов специальности 1-39 01 01 «Радиотехника»// Сост. Богуш Р.П. – УО «ПГУ»: Новополоцк, 2005 – 48 с.)

3.1. Элементы теории информации и основы статистического кодирования

3.1.1. Количественная оценка информации, энтропия источника сообщений

Статистическое кодирование информации (кодирование источника сообщений или сжатие информации) связано с преобразованием выходной информации дискретного источника в последовательность букв заданного кодового алфавита.

Всякая информация получается потребителем после приема сообщения, т.е. в результате опыта. Сообщение, получаемое на приемной стороне, несет полезную информацию лишь в том случае, если имеется неопределенность относительно состояния источника. Если опыт может закончиться только одним исходом и наблюдатель заранее знает исход опыта, то по его результату он не получает никакой информации. Информация появится лишь тогда, когда источник будет иметь по крайней мере более одного возможного состояния.

Рассмотрим источник, выдающий последовательность независимых дискретных сообщений $\{\lambda_i\}$, каждое из которых случайным образом выбирают из алфавита сообщения $A\ (\lambda_i\ ) = \lambda_1, \lambda_2, \lambda_3, \ldots \lambda_K$, где K – размер алфавита источника. Такой источник будем называть источником без памяти с конечным дискретным алфавитом. Сообщения, вырабатываемые таким источником, называются простыми сообщениями. В каждом элементарном сообщении λ_i для его получателя содержится некоторая информация. Количество информации, содержащейся в элементарном сообщении λ_i, является некоторой функцией от вероятности передачи этого сообщения $P(\lambda_i)$ и определяется

$$J\ (\lambda_i) = a \log P(\lambda_i),$$

при этом как коэффициент a, так и основание логарифма могут быть выбраны произвольно. Указанная мера была предложена Р. Хартли в 1928 г. для количественной оценки способности системы хранить или передавать информацию. Однако для удобства (чтобы количественная мера информации была положительной) принимают $a = -\ 1$. Если основание логарифма равно двум, то количество информации, содержащееся в элементарном сообщении, определяется

$$J\ (\lambda_i) = -\ \log_2 P(\lambda_i).$$

Определенная таким образом единица измерения информации называется *двоичной единицей* или *битом информации.*

Количество информации, содержащееся в одном элементарном сообщении λ_i, еще никак не характеризует источник. Одни элементарные сообщения могут нести много информации, но передаваться очень редко, другие – передаваться чаще, но нести меньше информации. Поэтому источник может быть охарактеризован средним количеством информации, приходящимся на одно элементарное сообщение, носящим название «энтропия источника» и определяемым следующим образом

$$H(\bar{\lambda}) = -\sum_{i=1}^{K} P(\lambda_i) \cdot \log P(\lambda_i), \qquad i = 1, K.$$

Энтропия как количественная мера информативности источника обладает следующими свойствами:

1. Энтропия есть величина вещественная, ограниченная и неотрицательная. Эти ее свойства вытекают из вида выражения для $H(\lambda)$, а также с учетом того, что $0 < P(\lambda_i) < 1$.

2. Энтропия детерминированных сообщений равна нулю $H(\lambda) = 0$, если хотя бы одно из сообщений имеет вероятность, равную единице.

3. Энтропия максимальна, если сообщения λ_i равновероятны

$$P(\lambda_1) = P(\lambda_2) = .. \ldots\ldots .. P(\lambda_k) = 1/K ,$$

тогда

$$H(\lambda) = -\,{}^{1}\!/_{K} \sum_{i=1}^{K} \log {}^{1}\!/_{K} = \log K.$$

Как видно из последнего выражения, в случае равновероятных сообщений энтропия растет с увеличением объема алфавита источника (ростом числа сообщений). При неравновероятных элементарных сообщениях λ_i энтропия, соответственно, уменьшается.

4. Энтропия двоичного источника ($K = 2$) может изменяться от нуля до единицы. Действительно, энтропия системы из двух сообщений λ_1 и λ_2

$$H(\lambda) = -P(\lambda_1) \cdot \log P(\lambda_1) - P(\lambda_2) \cdot \log P(\lambda_2) =$$
$$= -P(\lambda_1) \cdot \log P(\lambda_1) - \{1 - P(\lambda_1)\} \cdot \log \{1 - P(\lambda_1)\}.$$

Из последнего выражения видно, что энтропия равна нулю при $P(\lambda_1)=0$; $P(\lambda_2)=1$ или $P(\lambda_1) = 1$; $P(\lambda_2) = 0$; при этом максимум энтропии будет иметь место, когда $P(\lambda_1)=P(\lambda_2)=1/2$ и ее максимальное значение будет равно 1 бит.

3.1.2. Энтропия сложных сообщений

Энтропию сложного сообщения, вырабатываемого двумя зависимыми источниками X и Y, определяют следующим образом

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

где $P(x_i,y_j)$ – вероятность совместного появления сообщений x_i и y_j . Подобным образом определяется энтропия сложного сообщения, вырабатываемого одним источником с памятью.

Поскольку совместная вероятность $P(x_i,y_j)$ по формуле Байеса определяется как

$$P(x_i, y_j) = P(x_i)\cdot P(y_j / x_i) = P(y_j)\cdot P(x_i / y_j),$$

то выражение для совместной энтропии можно записать в следующем виде

$$H(X,Y) = -\sum_{i=1}^{K}\sum_{j=1}^{M} P(x_i)\cdot P(y_j / x_i)\cdot \log\{P(x_i)\cdot P(y_j / x_i)\} =$$

$$= -\sum_{i=1}^{K} P(x_i)\cdot \log P(x_i)\cdot \sum_{j=1}^{M} P(y_j / x_i) - \sum_{i=1}^{K} P(x_i)\cdot \sum_{j=1}^{M} P(y_j / x_i)\cdot \log P(y_j / x_i).$$

Т.к. передаче сообщения x_i обязательно соответствует передача одного из сообщений (любого) из ансамбля Y , то

$$\sum_{j=1}^{M} P(y_j / x_i) = 1$$

и совместная энтропия $H(X,Y)$ определится как

$$H(X,Y) = -\sum_{i=1}^{k} P(xi)\cdot \log P(xi) - \sum_{i=1}^{K} P(x_i)\cdot \sum_{j=1}^{M} P(y_j / x_i)\cdot \log P(y_j / x_i) =$$

$$= H(X) + \sum_{i=1}^{K} P(x_i)\cdot H(Y / x_i),$$

где $H\,(Y/x_i\,)$ – так называемая частная условная энтропия, отражающая энтропию сообщения Y при условии, что имело место сообщение x_i. Второе слагаемое в последнем выражении представляет собой усреднение $H\,(Y/x_i\,)$ по всем сообщениям x_i и называется средней условной энтропией источника Y при условии передачи сообщения X. И окончательно

$$H\,(X,Y\,) = H\,(X) + H\,(Y/X)\,.$$

Таким образом, совместная энтропия двух сообщений равна сумме безусловной энтропии одного из них и условной энтропии второго.

Основные свойства энтропии сложных сообщений:

1. При статистически независимых сообщениях X и Y совместная энтропия равна сумме энтропий каждого из источников

$$H(X,Y) = H(X) + H(Y),$$

т.к. $H(Y/X) = H(Y)$.

2. При полной статистической зависимости сообщений X и Y совместная энтропия равна безусловной энтропии одного из сообщений. Второе сообщение при этом информации не добавляет.

3.Условная энтропия изменяется в пределах

$$0 < H(Y/X) < H(Y).$$

4. Для совместной энтропии двух источников всегда справедливо соотношение

$$H(X,Y) \le H(X) + H(Y),$$

при этом условие равенства выполняется только для независимых источников сообщений.

Следовательно, при наличии связи между элементарными сообщениями энтропия источника снижается, причем тем более, чем сильнее связь между элементами сообщения.

Значит, можно сделать следующие выводы относительно степени информативности источников сообщений:

1. Энтропия источника и количество информации тем больше, чем больше размер алфавита источника.

2. Энтропия источника зависит от статистических свойств сообщений. Энтропия максимальна, если сообщения источника равновероятны и статистически независимы.

3. Энтропия источника, вырабатывающего неравновероятные сообщения, всегда меньше максимально достижимой.

4. При наличии статистических связей между элементарными сообщениями (памяти источника) его энтропия уменьшается.

Известно, что к основными информационными характеристиками источника относятся

– энтропия источника $H\left(\overline{\lambda}\right) = -\sum_{i=1}^{K} P(\lambda_i)\log P(\lambda_i)$;

– избыточность источника $\rho_u = 1 - \dfrac{H(\bar{\lambda})}{\log K}$;

– среднее число символов в коде $\bar{n} = \sum_{i=1}^{K} n(\lambda_i) \cdot P(\lambda_i)$;

где $n(\lambda_i)$ – число символов кода *i*- го сообщения;

– избыточность кода $\rho_k = 1 - \dfrac{H(\bar{\lambda})}{\bar{n}}$.

3.1.3. Основы статистического кодирования

В реальных условиях независимость элементарных сообщений, вырабатываемых источником, – явление довольно редкое. Чаще бывает как раз обратное – сильная детерминированная или статистическая связь между элементами сообщения одного или нескольких источников.

Например, при передаче текста вероятности появления отдельных букв зависят от того, какие буквы им предшествовали. Для русского текста, например, если передана буква «*П*», вероятность того, что следующей будет «*А*», гораздо выше, чем «*Н*», после буквы «*Ъ*» никогда не встречается «*Н*» и т.д. Подобная же картина наблюдается при передаче изображений – соседние элементы изображения имеют обычно почти одинаковые яркость и цвет.

Пример.

Пусть вероятности появления в тексте различных букв будут разными (табл. 3.1).

Таблица 3.1

Вероятность появления букв в сообщении

а	*б*	*в*	*г*	*д*	*е*	*ж*	*з*
Ра=0,6	*Рб*=0,2	*Рв*=0,1	*Рг*=0,04	*Рд*=0,025	*Ре*=0,015	*Рж*=0,01	*Рз*=0,01

Энтропия источника в этом случае составит $H(\bar{\lambda}) = 1,781$

Среднее число символов на одно сообщение при использовании равномерного трехразрядного кода $\bar{n} = \sum_{i=1}^{K} n(\lambda_i) P(\lambda_i) = n \sum_{i=1}^{K} P(\lambda_i) = n = 3.$

Избыточность кода в этом случае будет

$$\rho_k = 1 - \frac{H(\bar{\lambda})}{\bar{n}} = 1 - \frac{1,781}{3} \approx 0,41,$$

или довольно значительной величиной (в среднем 4 символа из 10 не несут никакой информации). Иначе говоря, 4 символа являются избыточными и могут безо всяких потерь просто не передаваться.

В связи с тем, что при кодировании неравновероятных сообщений равномерные коды обладают большой избыточностью, было предложено использовать для кодирования неравномерные коды, длительность кодовых комбинаций которых была бы согласована с вероятностью выпадения различных букв. Впервые эта простая идея была реализована американским инженером Морзе в предложенном им коде. Предполагают, что создавая свой код, Морзе отправился в ближайшую типографию и подсчитал число литер в наборных кассах. Буквам и знакам, для которых литер в этих кассах было припасено больше, он сопоставил более короткие кодовые обозначения (ведь эти буквы встречаются чаще). Так, например, в русском варианте азбуки Морзе буква «е» передается одной точкой, а редко встречающаяся буква «ц» — набором из четырех символов.

Существует много методов эффективного кодирования для различных источников. Почти все они основаны на тех же принципах: укрупнения сообщений или предсказания для уменьшения избыточности, вызванной корреляцией между сообщениями, и применения неравномерного кода для уменьшения избыточности, вызванной неравной вероятностью появления сообщений. Следует, однако, помнить про технические трудности декодирования неравномерного кода. Если источник производит буквы с фиксированной во времени скоростью и если необходимо передавать закодированные символы с фиксированной во времени скоростью, то неравномерный код приводит к проблеме ожидающей очереди. Когда источник выдает редкую букву, производится длинное кодовое слово и ожидающая очередь увеличивается. Наоборот, часто встречающиеся буквы порождают короткие кодовые слова, сокращая ожидающую очередь.

Шенноном сформулирована *следующая теорема*. При кодировании сообщений $\{\lambda_i\}$ в алфавите, насчитывающем K символов, при условии отсутствия шумов средняя длина кодового слова не может быть меньше, чем энтропия $H(\lambda_i)$.

Данная теорема не показывает путей для нахождения кодовых слов с минимально возможной средней длиной, а поэтому она является теоремой существования. Важность этой теоремы состоит в том, что она определяет предельно возможную эффективность кода, позволяет оценить, насколько тот или иной конкретный код близок к самому экономному.

Согласно теореме Шеннона для кодирования сообщений из табл. 3.1. может использоваться код, средняя длина которого $\bar{n} \geq 1,781$.

В табл. 3.2 представлен один из возможных вариантов кодирования, где наиболее вероятным сообщениям присвоены наиболее короткие кодовые слова.

Таблица 3.2

Произвольное кодирование сообщений

Буква	$P(\lambda_i)$	Код
а	0.6	1
б	0.2	011
в	0.1	110
г	0.04	101
д	0.025	1001
е	0.015	10001
ж	0.01	100001
з	0.01	110000

Однако если, например, принята последовательность (11011), то ее в соответствии с данным кодом можно декодировать как «ага» или «ваа», т.е. в данном случае хотя и гарантирует сокращение избыточности, но не обеспечивает однозначность декодирования (следовательно, выбранный код не пригоден для передачи сообщения).

Однозначность декодирования при данном коде можно обеспечить, если после каждого сообщения передавать некоторый символ, разделяющий сообщения. В этом случае уже будет не двоичный код, а троичный, это используется в коде Морзе, где кроме точки и тире используется третий символ – пробел. Очевидно, что введение разделительного символа снижает эффективность кодирования.

Однозначность декодирования можно обеспечить, не вводя разделительного символа, если строить код так, чтобы он удовлетворял условию, известному под названием «свойство префикса». Оно заключается в том, что ни одна комбинация более кратного кода не должна совпадать с началом («префиксом») другого кодового слова. Это свойство не выполнено в рассмотренном коде, т.к. например, слово, соответствующее сообщению *а*, является началом слова, соответствующего сообщению в, г, д и т.п. Коды, удовлетворяющие этому условию, называют префиксными кодами. Эти коды обеспечивают однозначное декодирование принятых кодовых слов без введения дополнительной информации для их разделения, т.е. всякая последовательность кодовых символов должна быть единственным образом разделена на кодовые слова. Коды, в которых это требование выпол-

нимо, называются однозначно декодируемыми или кодами без запятой. Префиксный код называют полным, если добавление к нему нового кодового слова (в данном алфавите) нарушает свойство префиксности. Например, для двоичного неполного префиксного кода 0, 10, 111 можно добавить слово 110 и новый код также будет префиксным.

Если код префиксный, то, читая принятую последовательность подряд с начала до конца, можно установить, где кончается одно кодовое слово и начинается следующее. Например, если код префиксный и в последовательности встретился код 110, то очевидно, в коде не должно содержаться слов (1), (11). Существует несколько алгоритмов построения префиксных кодов. Среди них коды Шеннона – Фано и Хаффмана более всего позволяют приблизиться к границе, определяемой энтропией.

Следует, еще раз отметить, что эффективное кодирование широко применяется в цифровой сотовой связи стандартов GSM и CDMA, в системах цифрового спутникового телевидения, сети Internet и др. цифровых радиотехнических системах.

3.2. Эффективные коды

3.2.1. Коды Шеннона – Фано

Простейшим способом статистического кодирования является кодирование по методу Шеннона – Фано. Кодирование в соответствии с этим алгоритмом производится так:

1. все буквы из алфавита сообщения записывают в порядке убывания их вероятностей;

2. затем всю совокупность букв разбивают на две примерно равные по сумме вероятностей группы; одной из них (в группе может быть любое число символов, в том числе – один) присваивают символ «1(0)», другой – «0(1)»;

3. каждую из этих групп снова разбивают (если это возможно) на две части и каждой из частей присваивают «1(0)» и «0(1)» и т.д., пока в каждой группе не останется по одной букве.

Пример. Если заданы четыре сообщения A_1, A_2, A_3, A_4 с вероятностями $P(A_1) = l/2$, $P(A_2) = l/4$, $P(A_3)$=$P(A_4) = 1/8$, то это означает, что среди, например, 1000 переданных сообщений около 500 раз появляется сообщение A_1, около 250 – сообщение A_2 и примерно по 125 раз – каждое из сообщений A_3 и A_4.

Используя алгоритм Шеннона – Фано, сообщение разбивается на две равновероятные группы: в первую попадает сообщение A_1, во вторую – сообщения A_2, A_3, A_4. Первой группе на каждом этапе разбиения можно сопоставить символ 0, второй – символ 1, затем вторая группа разбивается на две равновероятные – A_2 и A_3, A_4, и последний шаг – разбиваем A_3 и A_4.

A_1	1/2	0		
A_2	1/4	1	0	
A_3	1/8		1	0
A_4	1/8			1

В итоге сообщения кодируются следующим образом: A_1-0, A_2-10, A_3-110, A_4-111.

Понятно, что чем более вероятно сообщение, тем быстрее оно образует «самостоятельную» группу и тем более коротким словом оно будет закодировано. Это обстоятельство и обеспечивает высокую экономность кода Фано. Такой метод кодирования можно применять и в случае произвольного алфавита из q символов с той лишь разницей, что на каждом шаге следует производить разбиение на q равновероятных групп.

Алгоритм кодирования Фано имеет очень простую графическую иллюстрацию в виде множества точек (вершин) на плоскости, соединенных отрезками (ребрами) по определенному правилу (такие фигуры называют графами). Граф для кода Шеннона – Фано строится следующим образом. Из нижней (корневой) вершины графа исходят два ребра, одно из которых (любое) помечено символом 0, другое – символом 1. Эти два ребра соответствуют разбиению множества сообщений на две равновероятные группы, одной из которых сопоставляется символ 0, а другой – символ 1. Ребра, исходящие из вершин следующего «этажа», соответствуют разбиению получившихся групп снова на равновероятные подгруппы и т.д. Построение графа заканчивается, когда множество сообщений будет разбито на одноэлементные подмножества. Каждая концевая вершина графа, т.е. вершина, из которой уже не исходят ребра, соответствует некоторому кодовому слову. Чтобы указать это слово, надо пройти путь от корневой вершины до соответствующей концевой, выписывая в порядке следования по этому пути символы проходимых ребер.

Граф для рассмотренного выше примера представлен на рис.3.1.

Кодовые деревья дают удобное геометрическое представление для многих важных понятий и облегчают решение различных задач, возникающих при построении экономных кодов.

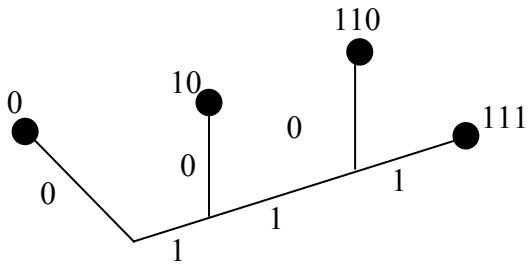


Рис. 3.1. Граф кода Шеннона – Фано для примера 3.2

Пример. Кодирование сообщений из табл. 3.1. по методу Шеннона – Фано представлено в табл. 3.3.

Таблица 3.3

Пример статистического кодирования методом Шеннона – Фано

Буква	$P(\lambda_i)$	I	II	III	IV	V	V I	Код
а	0.6	1						1
б	0.2	0	1	1				011
в	0.1			0				010
г	0.04		0	1				001
д	0.025			0	1			0001
е	0.015				0	1		00001
ж	0.01					0	1	000001
з	0.01						0	000000

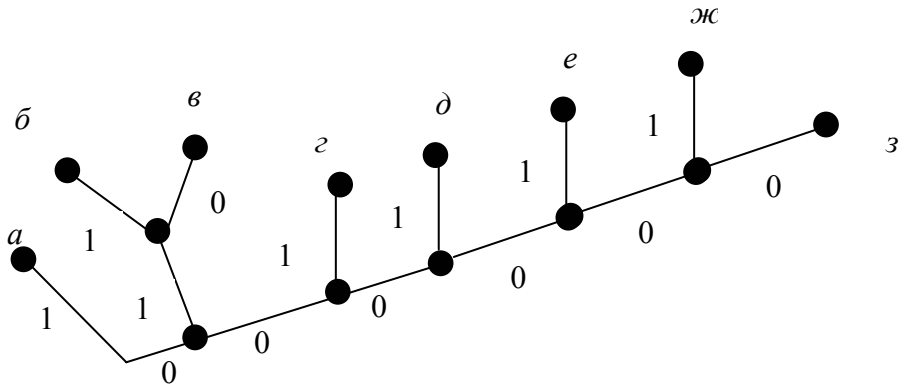


Рис. 3.2. Граф кода Шеннона – Фано для таблицы 3.3.

Для полученного таким образом кода среднее число двоичных символов, приходящихся на одну букву, равно

$$\overline{n} = \sum_{i=0}^{7} n_i P_i \approx 1.9,$$

а избыточность кода составит

$$\rho_k = 1 - \frac{1.781}{1.9} \approx 0.06,$$

т.е. также существенно меньшую величину, нежели для равномерного кода.

Коды Шеннона – Фано являются префиксными кодами. Эти коды обеспечивают однозначное декодирование принятых кодовых слов без введения дополнительной информации для их разделения.

3.2.2. Коды Хаффмена

Методика Шеннона – Фано не всегда приводит к однозначному построению кода, поскольку при разбиении на части можно сделать больше по вероятности как верхнюю, так и нижнюю части. Кроме того, методика не обеспечивает отыскания оптимального множества кодовых слов для кодирования данного множества сообщений. (Под оптимальностью подразумевается то, что никакое другое однозначно декодируемое множество кодовых слов не имеет меньшую среднюю длину кодового слова, чем заданное множество).

Если S=$\{ w_1, w_2, \ldots , w_k \}$ – префиксное множество, то можно определить некоторый вектор $v(S) = (L_1, L_2, \ldots , L_k)$, состоящий из чисел, являющихся длинами соответствующих префиксных последовательностей (L_i – длина w_i).

Вектор $(L_1, L_2, \ldots , L_k)$, состоящий из неуменьшающихся положительных целых чисел, называется *вектором Крафта*. Для него выполняется неравенство

$$2^{-L_1} + 2^{-L_2} + \ldots + 2^{-L_k} \le 1.$$

Это неравенство называется *неравенством Крафта*. Для него справедливо следующее утверждение: если S – какое-либо префиксное множество, то $v(S)$ – вектор Крафта. Иными словами, длины двоичных последовательностей в префиксном множестве удовлетворяют неравенству Крафта.

Если неравенство Крафта переходит в строгое равенство, то такой код называется *компактным* и обладает наименьшей среди кодов с данным алфавитом длиной, т.е. является оптимальным.

Ниже приведены примеры простейших префиксных множеств и соответствующие им векторы Крафта

$S_1 = \{0, 10, 11\}$ и $v(S1) = (1, 2, 2)$;

$S_2 = \{0, 10, 110, 111\}$ и $v(S2) = (1, 2, 3, 3)$;

$S_3 = \{0, 10, 1100, 1101, 1110, 11110, 111110, 111111\}$ и $v(S3) = (1, 2, 4, 4, 4, 5, 6, 6)$.

Допустим, необходимо разработать код без памяти для сжатия вектора данных X=(x_1, x_2,… x_n) с алфавитом A размером в k символов. Введем в рассмотрение так называемый вектор частот F=(F_1, F_2, ... , F_k), где F_i – количество появлений i- го символа из A в X. Закодируем X кодом без памяти, для которого вектор Крафта L = (L_1, L_2, ... , L_k).

Тогда длина двоичной кодовой последовательности $B(X)$ на выходе кодера составит

$$B(X) = L_1 \cdot F_1 + L_2 \cdot F_2 + ... + L_k \cdot F_k .$$

Лучшим кодом без памяти является код, для которого длина $B(X)$ – минимальна. Для разработки такого кода необходимо найти вектор Крафта L, для которого сумма произведений $L_i \cdot F_i$ была бы минимальной. Простой перебор возможных вариантов не всегда позволяет найти такой вектор Крафта, особенно для большого k.

Эффективный способ поиска оптимального вектора Крафта L разработал Дэвида Хаффмен. Алгоритм Хаффмена, названный в честь его изобретателя, дает код, полученный с использованием оптимального вектора Крафта, который называют *кодом Хаффмена*.

Алгоритм Хаффмена требует выполнения следующих шагов:

1. Выписываем в ряд все символы алфавита в порядке возрастания или убывания вероятности их появления в тексте.

2. Последовательно объединяем два символа с наименьшими вероятностями появления в новый составной символ, вероятность появления которого полагаем равной сумме вероятностей составляющих его символов. В конце концов построим дерево, каждый узел которого имеет суммарную вероятность всех узлов, находящихся ниже него.

3. Прослеживаем путь к каждому листу дерева, помечая направление к каждому узлу (например, направо – 1, налево – 0) . Полученная последовательность дает кодовое слово, соответствующее каждому символу.

Пример. Кодирование сообщений из табл. 3.1. по алгоритму Хаффмена представлено в табл. 3.4.

Среднее число символов для такого кода составит

$$\overline{n} = \sum_{i=1}^{8} n_i P_i = 1.825,$$

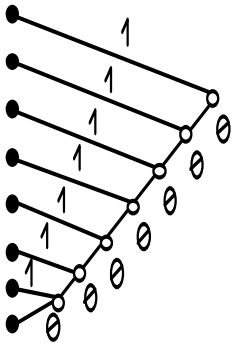
а избыточность кода

$$\rho_k = 1 - \frac{H(\overline{\lambda})}{\overline{n}} = 1 - \frac{1.781}{1.825} \approx 0.03,$$

т.е. на порядок меньше, чем при равномерном кодировании.

Таблица 3.4

Пример статистического кодирования алгоритмом Хаффмена

Буква	Вероятность $P(\lambda_i)$	Кодовое дерево	Код
а	0.6		1
б	0.2		01
в	0.1		001
г	0.04		0001
д	0.025		00001
е	0.015		000001
ж	0.01		0000001
з	0.01		00000001

3.2.3. Недостатки системы эффективного кодирования

Причиной одного из недостатков является различие в длине кодовых комбинаций. Если моменты снятия информации с источника неуправляемы, кодирующее устройство через равные промежутки времени выдает комбинации различной длины. Т.к. линия связи используется эффективно только в том случае, когда символы поступают в нее с постоянной скоростью, то на выходе кодирующего устройства должно быть предусмотрено буферное устройство. Оно запасает символы по мере поступления и выдает их в линию связи с постоянной скоростью. Аналогичное устройство необходимо и на приемной стороне.

Второй недостаток связан с возникновением задержки в передаче информации. Наибольший эффект достигается при кодировании длинными блоками, а это приводит к необходимости накапливать знаки, прежде чем поставить им в соответствие определенную последовательность символов. При декодировании задержка возникает снова. Общее время задержки может быть велико, особенно при появлении блока, вероятность которого мала. Это следует учитывать при выборе длины кодируемого блока.

Еще один недостаток заключается в специфическом влиянии помех на достоверность приема. Одиночная ошибка может перевести передаваемую кодовую комбинацию в другую, не равную ей по длительности. Это повлечет за собой неправильное декодирование ряда последующих комбинаций, которые называют треком ошибки. Специальными методами построения эффективного кода трек ошибки стараются свести к минимуму.

Следует отметить относительную сложность технической реализации систем эффективного кодирования.

Методы эффективного кодирования Шеннона – Фано и Хаффмана, рассмотренные выше, позволяют производить кодирование, если известна статистика входных сообщений, т.е. известна вероятность их появления.

3.3. Кодирование изображений и сжатие информации с помощью спектральных преобразований

3.3.1. Кодирование длин повторений

Кодирование длин участков (или повторений) может быть достаточно эффективным при сжатии двоичных данных, например, черно-белых факсимильных изображений, черно-белых изображений, содержащих множество прямых линий и однородных участков, схем и т.п. Кодирование длин повторений является одним из элементов известного алгоритма сжатия изображений JPEG.

Идея сжатия данных на основе кодирования длин повторений состоит в том, что вместо кодирования собственно данных подвергаются кодированию числа, соответствующие длинам участков, на которых данные сохраняют неизменное значение.

Пример. Предположим, что нужно закодировать двоичное (двухцветное) изображение размером 8×8 элементов, приведенное на рис. 3.3.

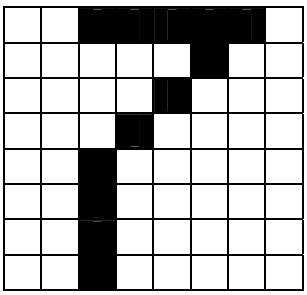


Рис.3.3. Кодируемое изображение

Просканируем это изображение по строкам (двум цветам на изображении будут соответствовать 0 и 1), в результате получим двоичный вектор данных

X=(0011111000000100000010000001000000100000001000000010000000100000)

длиной 64 бита (скорость исходного кода составляет 1 бит на элемент изображения).

Выделим в векторе *X* участки, на которых данные сохраняют неизменное значение, и определим их длины. Результирующая последовательность длин участков – положительных целых чисел, соответствующих исходному вектору данных *X***,** – будет иметь вид

$$r = (2, 5, 6, 1, 6, 1, 6, 1, 6, 1, 7, 1, 7, 1, 7, 1, 5).$$

Теперь эту последовательность, в которой заметна определенная повторяемость (единиц больше, чем других символов), можно закодировать каким-либо статистическим кодом, например, кодом Хаффмена, имеющим кодовое дерево (рис.3.4).

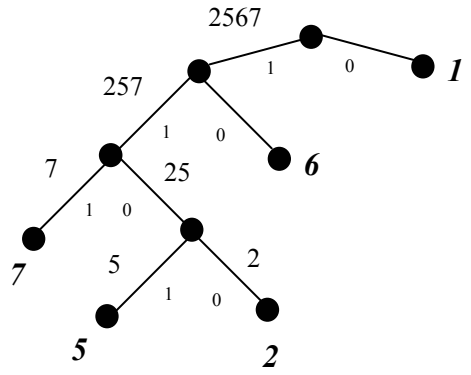


Рис.3.4. Кодовое дерево кода Хаффмена

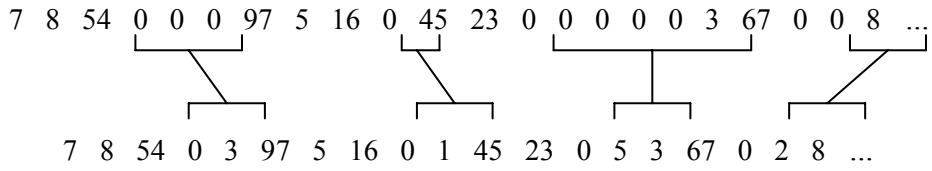
Для того чтобы указать, что кодируемая последовательность начинается с нуля, добавим в начале кодового слова префиксный символ 0. В результате получим кодовое слово

$$B(r)=(0|1100|1101|10|0|10|0|10|0|10|0|111|0|111|0|111|0|1101)$$

длиной в 37 битов, т.е. результирующая скорость кода *R* составит 37/64 или примерно 0,58 бита на элемент изображения. При сжатии изображений большего размера и содержащих множество повторяющихся элементов эффективность сжатия может оказаться существенно более высокой. Подобный алгоритм используется при передаче факсимальных изображений.

Ниже приведен другой пример использования кодирования длин повторений, когда в цифровых данных встречаются участки с большим количеством нулевых значений. Всякий раз, когда в потоке данных встречается «ноль», он кодируется двумя числами. Первое – 0, являющееся флагом начала кодирования длины потока нулей, и второе – количество нулей в очередной группе. Если среднее число нулей в группе больше двух, будет иметь место сжатие. С другой стороны, большое число отдельных нулей

может привести даже к увеличению размера кодируемого файла. Пример кодирования приведен ниже.



Такой алгоритм и алгоритм Хаффмена используется в стандарте сжатия изображений JPEG.

3.3.2. Сжатие информации с помощью спектральных преобразований

Спектральные методы кодирования изображений основаны на том, что цифровой эквивалент аналогового сигнала путем соответствующего линейного ортогонального преобразования может быть приведен к виду, который наиболее удобен с точки зрения сокращения избыточности информации. В этом случае повышение эффективности кодирования связано с тремя факторами:

- в процессе преобразования ряд спектральных коэффициентов (трансформант) становится настолько малым по величине, что их можно отбросить без заметного ухудшения качества восстанавливаемых изображений;
- в процессе преобразования осуществляется декорреляция данных, обеспечивающая повышение эффективности статистического кодирования;
- различное нелинейное квантование коэффициентов преобразования позволяет существенно сократить объем передаваемой информации без заметного ухудшения качества передаваемого изображения при его восстановлении.

Сущность спектральных методов кодирования заключается в том, что кодируется и передается не само изображение, а значения трансформант, получающихся при ортогональном преобразовании этого изображения. В процессе преобразований изображения, имеющего сильные корреляционные связи между смежными элементами, происходит процесс декорреляции, так что значения коэффициентов преобразования оказываются практически некоррелированными. В отличие от исходного изображения, для которого характерно в среднем равномерное распределение энергии

между элементами, распределение энергии между спектральными коэффициентами резко неравномерно. Основная доля энергии приходится на коэффициенты с малыми номерами (низкие пространственные частоты), и лишь небольшая ее часть – на прочие. В дальнейшем коэффициенты, имеющие малую амплитуду, опускаются либо квантуются на малое число уровней, что позволяет использовать меньшее число разрядов кода для передачи. При приеме выполняются обратные преобразования. Предварительно по принятому коду восстанавливается матрица трансформант, а затем путем обратного ортогонального преобразования изображение восстанавливается.

В общем случае кодирование посредством преобразований реализуется путем двух последовательных операций. Первая операция состоит в линейном преобразовании статистически зависимых элементов изображения. С помощью второй операции осуществляется кодирование определенным способом выбранных коэффициентов преобразования.

3.3.3. Кодирование изображений посредством преобразований

При кодировании источника изображения после аналого-цифрового преобразования изображение предоставляется множеством отсчетов интенсивности светового потока, распределенным на двухмерной прямоугольной плоскости с координатами (n_i, n_j). Величина отсчета (например, яркости X) в точке с координатами (n_i, n_j) характеризуется неким целым числом $X(n_i, n_j)$. Такой элемент цифрового изображения называется *пикселом*. Пусть изображение создается совокупностью отсчетов, образующих матрицу из ($N\times N$) пикселов, т.е. исходное изображение будет состоять из N строк, каждая из которых содержит N элементов изображения. Причем $N = 2WT$, где W – верхняя частота спектра обрабатываемого сигнала изображения; T – время обработки сигнала.

Если дискретный сигнал X содержит N^2 отсчетов, то его можно представить в виде вектора с координатами (n_i, n_j) в N^2-мерном векторном пространстве. Тогда первая операция эффективного кодирования N^2-мерного вектора выражается

$$Y=G\times X,$$

где $Y=[y_0, y_1, \ldots, y_{N^2-1}]^T$, $X=[x_0, x_1, \ldots, x_{N^2-1}]^T$ – векторы-столбцы отсчетов с N^2 координатами коэффициентов преобразования (трансформант) и сигнала соответственно;

$G=[g_0, g_1, \ldots, g_{N^2-1}]^T$ – матрица кодирования (преобразования), столбцами которой являются N^2-мерные порождающие векторы g.

Двумерное прямое преобразование случайных величин $[X(n_1,n_2)]$ можно записать как

$$[Y(k_1,k_2)] = [G(k, n)][X(n_1,n_2)][G(k, n)]^T,$$

где $[Y(k_1,k_2)]$ – матрица коэффициентов преобразования (трансформант);

$[G(k,n)]$ – матрица кодирования; n_1, n_2 и k_1, $k_2 \in \{0, 1, \ldots, N\text{-}1\}$.

Если элементами матрицы G являются действительные числа, то

$$GG^T=E,$$

где E – единичная матрица.

Обратное ортогональное двухмерное преобразование коэффициентов $[Y(k_1, k_2)]$ определяется как

$$[X(n_1,n_2)] = [G(k, n)]^T[Y(k_1,k_2)][G(k, n)].$$

Матрица $[G(k, n)]$, обеспечивающая некоррелированные трансформанты, может быть вычислена по ковариационной матрице K_x. Эта матрица также представляется с помощью собственных векторов и соответствующих им собственных чисел в виде дискретного разложения Карунена – Лоэва

$$K_x = \Phi\lambda\Phi^T = \sum_{i=0}^{N^2-1} \lambda_i \Phi_i \Phi_i^T$$

где Φ – матрица порядка $\left(N^2 \times N^2\right)$, столбцы которой являются собственными векторами,

λ – диагональная матрица, образованная собственными числами.

В преобразованном пространстве небольшое число M координат представляют с минимальной среднеквадратической ошибкой большую часть статистической информации о кодируемых элементах изображения. Упорядочим собственные числа и собственные векторы в соответствии с

$$\lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N^2-1}.$$

Кодирование изображения осуществим через преобразование

$$Y_1=\Phi \cdot X.$$

Тогда вектор-столбец Y_1, состоящий из N^2 трансформант, будет иметь некоррелированные компоненты с упорядоченными дисперсиями так, что эти компоненты сосредоточивают максимум дисперсии в первых M координатах. Отобрав M компонент вектора Y_1 с наибольшими диспер-

сиями, получим вектор длиной $M < N^2$. Далее M коэффициентов вместе с информацией о координатах отобранных векторов передаются по каналу связи. В результате потребуется только $(M \times K)$ бит для представления всего изображения. Тем самым достигается сокращение (сжатие) полосы частот канала передачи. Коэффициент сжатия определяется отношением длины всех кодовых слов исходного изображения к суммарной длине слов, предназначенных к передаче,

$$K_{сж} = \frac{N^2 K}{MK} = \frac{N^2}{M}.$$

На приемной стороне вектор M декодируется в вектор Y_2 длиной N^2 пикселов, причем вместо переданных трансформант вставляются нули. Выполняя обратное преобразование по отношению к Φ, можно восстановить изображение

$$X' = \Phi^T Y_2.$$

Подобный алгоритм кодирования изображений требует выполнить N^4 вычислительных операций. Для того чтобы уменьшить вычислительную сложность кодирования, используют идею разбиения массива данных на меньшие размеры (фрагменты), а затем для каждого из фрагментов производится процедура преобразования. Экспериментальные исследования показали, что корреляция между двумя пикселями быстро стремится к нулю с ростом расстояния между ними. Поэтому можно кодировать изображение, разбив его на отдельные равные блоки L_1, L_2, …, L_i. Вычисление ковариационных матриц и кодирование проводится в пределах L_i-го блока. При $L << N$ достигается существенное сокращение временных затрат на операцию кодирования. Но даже при использовании блоковой структуры кодирования изображения применение разложения по собственным векторам приводит к значительному объему вычислений. Поэтому оно заменяется известными ортогональными преобразованиями, которым присущи алгоритмы быстрых вычислений.

Эффективное кодирование изображений на основе быстрых ортогональных преобразований вместо преобразования Карунена – Лоэва ПКЛ приводит к определенным качественным потерям, которые можно оценить, сравнивая с ошибками, полученными при сжатии на основе ПКЛ. Среднеквадратическая ошибка восстановленного изображения по отношению к исходному посредством ПКЛ будет минимальной по сравнению с ошибками при всех других возможных дискретных ортогональных преоб-

разованиях. В этом смысле кодирование с помощью ПКЛ достигает предельной эффективности, являясь оптимальным.

Для кодирования и сжатия изображений можно использовать такие двухмерные линейные преобразования как дискретное преобразование Фурье (ДПФ), преобразование Адамара, дискретное косинусное преобразование (ДКП), дискретные преобразования Хартли, Хаара и ряд других. Большинство из названных одномерных преобразований позволяют снизить вычислительную сложность до величины порядка $N\log_2 N$ и даже N. Поскольку двухмерные преобразования разделяются на одномерные операции по строкам и столбцам, то для преобразования всего изображения потребуется порядка $2N^2\log_2 N$ операций. Кодирование через снижение размерности преобразования до $(n' \times n'')$ пикселов приводит к еще большему практическому выигрышу в числе операций.

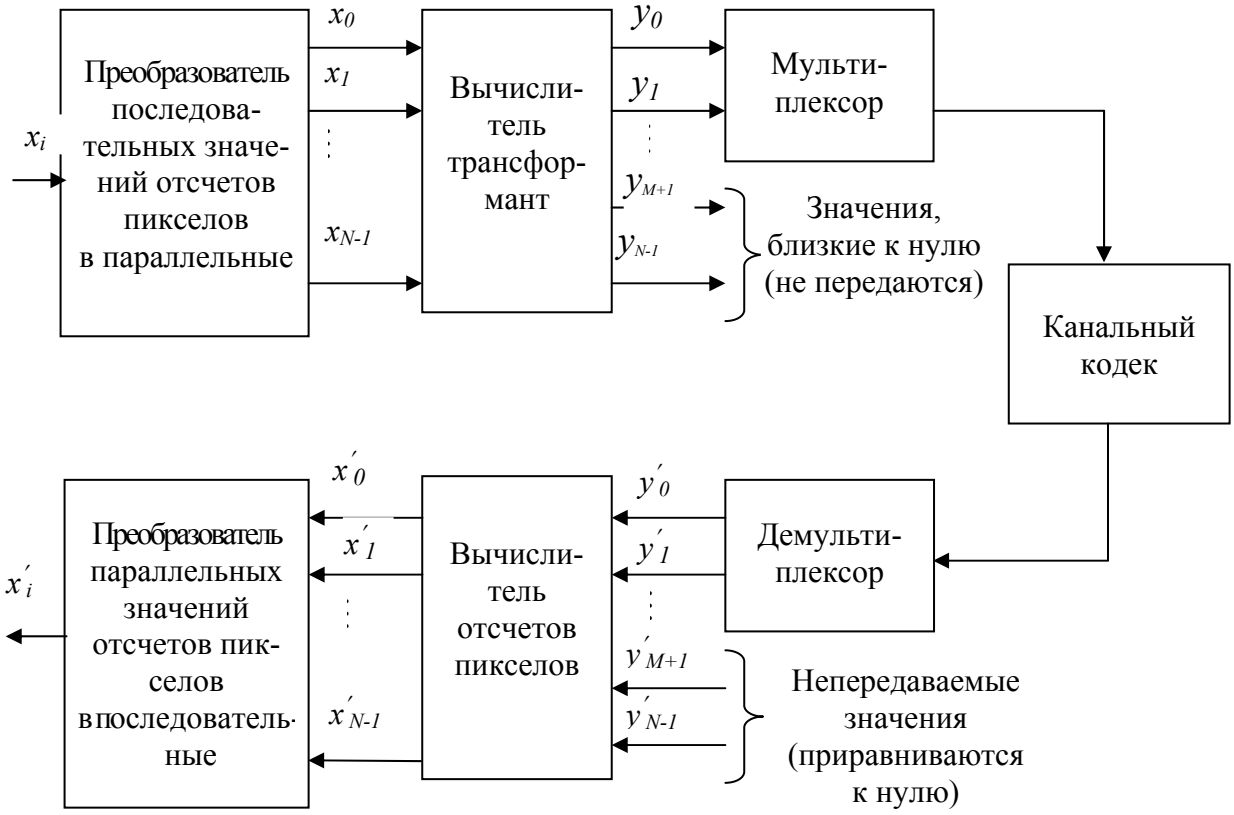


Рис. 3.5. Структурная схема кодирования изображения на основе дискретного ортогонального преобразования

Примером алгоритмов эффективного кодирования изображений (сжатия) на основе дискретных ортогональных преобразований являются

широко распространенные алгоритмы JPEG и JPEG2000, использующие дискретное косинусное преобразование и вейвлет-преобразование соответственно.

3.4. Вопросы и задания для самопроверки

3.1. Запишите выражение для определения количества информации по Хартли, содержащейся в элементарном сообщении?

3.2. Чему равна энтропия источника дискретных сообщений при равновероятных и независимых сообщениях?

3.3. Назовите основные свойства энтропии.

3.4. На основе какого выражения можно определить энтропию сложного сообщения, вырабатываемого двумя зависимыми источниками?

3.5. Каковы основные свойства энтропии сложных сообщений?

3.6. Назовите основные информационные характеристики источника.

3.7. Как изменяется энтропия при увеличении (уменьшении) объема алфавита?

3.8. Для какой цели используется эффективное кодирование информации?

3.9. Какой основной принцип лежит в основе методов эффективного кодирования информации?

3.10. До какого предела может быть уменьшена средняя длина кодового слова при эффективном кодировании префиксными кодами?

3.11. Какие коды являются префиксными?

3.12. В каком случае префиксный код является полным? Приведите пример неполного префиксного кода.

3.13. Сформулируйте алгоритм кодирования сообщений по методу Шеннона – Фано. Приведите пример.

3.14. Для кода Шеннона – Фано из предыдущего задания постройте кодовое дерево.

3.15. Являются ли коды Шеннона – Фано префиксными? Подтвердите ответ примером.

3.16. В чем недостаток метода Шеннона – Фано? Покажите на примере.

3.17. Дайте определение вектору Крафта. Приведите примеры префиксных множеств и соответствующих им векторов Крафта.

3.18. Для решения какой задачи используется вектор Крафта при эффективном кодировании информации?

3.19. Сформулируйте алгоритм кодирования информации методом Хаффмена. В чем преимущество методики Хаффмана по сравнению с методикой Шеннона – Фано?

3.20. Приведите пример кодирования информации алгоритмом Хаффмена и постройте кодовое дерево.

3.21. Назовите алгоритмы сжатия информации, в которых используется алгоритм Хаффмена.

3.22. Каковы основные недостатки методов эффективного кодирования?

3.23. Поясните работу известных Вам алгоритмов кодирования длин повторений.

3.24. Что лежит в основе спектральных методов сжатия информации?

3.25. На основании каких факторов достигается повышение эффективности кодирования информации при использовании спектральных методов?

3.26. Что представляет собой изображение в цифровом виде?

3.27. Запишите общее выражение для прямого двумерного преобразования.

3.28. Какими способами достигается уменьшение вычислительных затрат при спектральном кодировании изображений?

3.29. Назовите двумерные преобразования, которые используются при спектральном кодировании изображений.

3.30. Представьте обобщенную структурную схему кодирования изображений на основе двумерных ортогональных преобразований.

Модуль 4

КРИПТОГРАФИЧЕСКОЕ КОДИРОВАНИЕ

Цель модуля – изучение моделей каналов с криптографическим кодированием информации, основных традиционных методов шифрования передаваемой информации, современных симметричных и асимметричных криптосистем.

В результате изучения модуля студенты должны :

- знать основные классические шифры;
- иметь представление о различиях моделей каналов с криптографическим кодированием информации;
- знать процедуры зашифровывания и расшифровывания информации симметричными алгоритмами DES и ГОСТ 28147-89;
- знать процедуры зашифровывания и расшифровывания информации асимметричными алгоритмами RSA и Эль Гамаля;
- знать достоинства и недостатки симметричных и асимметричных криптосистем;
- иметь представление о размере ключа в симметричных и модуля в асимметричных криптосистемах для надежного шифрования информации на современном уровне развития техники.

Содержание модуля

4.1. Методы криптографического кодирования, обеспечивающие секретность информации

4.1.1. Модели каналов с криптографическим кодированием информации

4.1.2. Шифры перестановки

4.1.3. Шифры простой замены

4.1.4. Шифры сложной замены

4.1.5. Одноразовые шифры

4.1.6. Шифрование методом гаммирования

4.2. Современные симметричные криптосистемы

4.2.1. Американский стандарт шифрования данных DES

4.2.2. Основные режимы работы алгоритма DES

4.2.3. Стандарт шифрования данных ГОСТ 28147-89

4.3. Современные асимметричные криптосистемы

4.3.1. Криптосистема шифрования данных RSA

4.3.2. Схема шифрования данных Эль Гамаля

4.4. Вопросы и задания для самопроверки
4.5. Практическое занятие № 3
4.6. Практическое занятие № 4

4.1. Методы криптографического кодирования, обеспечивающие секретность информации

4.1.1. Модели каналов передачи с криптографическим кодированием информации

Криптография представляет собой совокупность методов кодирования данных, направленных на то, чтобы сделать эти данные бесполезными для противника. Такое кодирование позволяет решить две главные проблемы защиты данных: проблему секретности – лишение противника возможности извлечь информацию из канала передачи и проблему имитостойкости – лишение противника возможности ввести ложную информацию в канал передачи или изменить сообщение так, чтобы изменился его смысл.

Проблемы секретности и имитостойкости тесно связаны между собой, поэтому методы решения одной из них часто применимы для решения другой. Из двух названных проблем секретность рассматривается первой, как наиболее исследованная на протяжении веков. На рис. 4.1 представлена модель канала передачи данных, обеспечивающая секретность благодаря криптографическому кодированию.

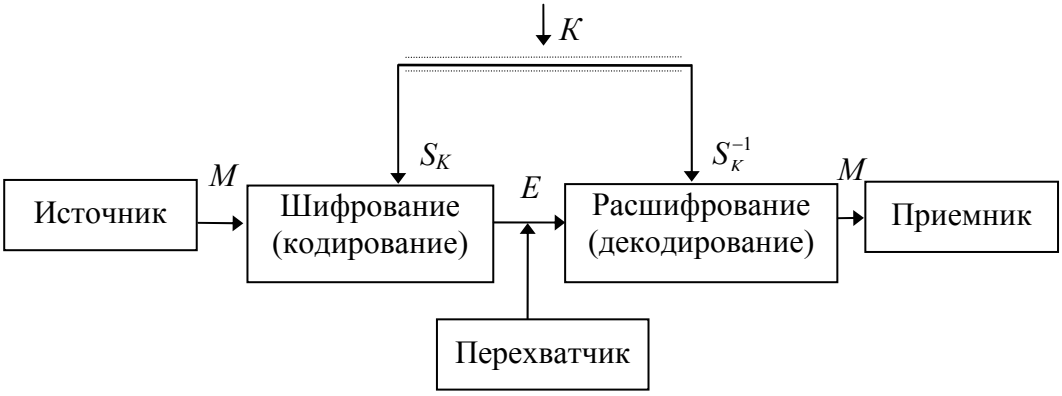


Рис. 4.1. Модель канала передачи с криптографическим кодированием

Источник информации генерирует открытый текст или незашифрованное сообщение *M*, которое должно быть передано соответствующему получателю по незащищенному каналу, за которым следит перехватчик. Для того чтобы перехватчик не смог распознать сообщение *M*, отправитель шифрует (кодирует) его с помощью обратимого кодирования (преоб-

разования) S_K и получает криптограмму или шифрованный текст $E = S_K(M)$, который отправляет получателю.

Законный получатель, приняв криптограмму E, декодирует (расшифровывает) ее с помощью обратного преобразования S_K^{-1} и получает исходное сообщение в виде открытого сообщения

$$M: S_K(E) = S_K^{-1}(S_K(M)) = M.$$

Преобразование S_K выбирается из семейства криптографических преобразований, называемых *криптографической системой* или *общей системой.*

Параметр, выбираемый в качестве отдельного преобразования, называется *криптографическим ключом* или просто *ключом*. Общая система – это набор инструкций (часть аппаратуры или программа ЭВМ), с помощью которой можно закодировать открытый текст и декодировать шифрованный текст различными способами, один из которых выбирается с помощью конкретного ключа. Формально, криптографическая система − это однопараметрическое семейство обратимых преобразований кодирования *Sк*: $M \rightarrow E$ из пространства M сообщений открытых данных в пространство E (закодированных шифрованных сообщений). Причем ключ K_i выбирается из конечного множества K, называемого *пространством ключей*.

Обычно общая система, являющаяся семейством преобразований, рассматривается как общедоступная система. С одной стороны, то, что открытая для всех часть называется общей системой, отражает очень важное правило техники криптографического кодирования: защищенность системы не должна зависеть от секретности чего-либо такого, что нельзя быстро изменить в случае утечки секретной информации. Обычно общая система является некоторой совокупностью аппаратуры, которую можно изменить только со значительными затратами времени и средств, тогда как ключ является легко и просто изменяемым объектом. Криптографическая система подобна кодовому замку − структура замка известна любому, кто его приобрел, однако конкретная используемая комбинация неизвестна и может быть изменена всякий раз, когда есть подозрение, что она стала известна постороннему лицу. Даже если противник знает множество всех возможных комбинаций, он может все же оказаться не в состоянии определить, какая из них правильная.

Поскольку вся секретность сосредоточена в секретности ключа, то его надо передавать отправителю и получателю по защищенному каналу распространения ключей. На рис. 4.1 этот канал показан экранированной линией.

Криптографическое кодирование может быть симметричным или асимметричным относительно преобразования расшифровывания. Это важное свойство функции преобразования определяет два класса криптосистем:

- симметричные (одноключевые) криптосистемы;
- асимметричные (двухключевые) криптосистемы (с открытым ключом).

Схема симметричной криптосистемы с одним секретным ключом была показана на рис. 4.1. В ней используются одинаковые секретные ключи в блоке шифрования и блоке расшифровывания.

На рис. 4.2 представлена модель канала передачи данных, обеспечивающая секретность с использованием асимметричного криптографического кодирования. В этой криптосистеме один из ключей является открытым, а другой – секретным.

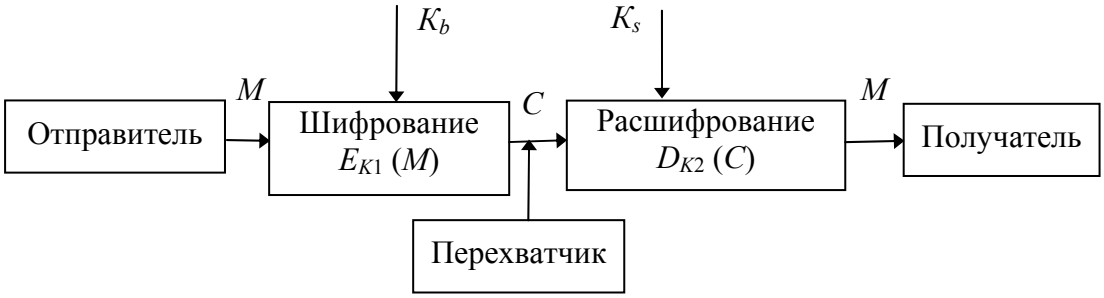


Рис. 4.2. Обобщенная схема асимметричной криптосистемы

В асимметричной криптосистеме для зашифровывания данных используется один ключ, а для расшифровывания – другой ключ (отсюда и название – асимметричная). Первый ключ является открытым и может быть опубликован для использования всеми пользователями системы, которые зашифровывают данные. Расшифровывание данных с помощью открытого ключа невозможно.

Для расшифровывания данных получатель зашифрованной информации использует второй ключ, который является секретным. Разумеется, ключ расшифровывания не может быть определен из ключа зашифровывания.

В асимметричных системах нет необходимости в защищенном канале для передачи ключей, т.к. открытый ключ передается по незащищенному каналу.

Любая попытка со стороны перехватчика расшифровать криптограмму E для получения открытого текста M или зашифровать свой собственный текст M' для получения приемлемой криптограммы E' без получения ключа из канала распространения ключей называется *криптоанализом.*

Если криптоанализ невозможен и криптоаналитик не может вывести M из E или E' из M' без предварительного получения ключа, то такая криптографическая система называется *криптостойкой*.

4.1.2. Шифры перестановки

При шифровании перестановкой символы шифруемого текста переставляются по определенному правилу в пределах блока этого текста. Шифры перестановки являются самыми простыми и, вероятно, самыми древними шифрами.

Шифрующие таблицы

В качестве ключа в шифрующих таблицах используются: размер таблицы, слово или фраза, задающие перестановку, особенности структуры таблицы.

Одним из самых примитивных табличных шифров перестановки является простая перестановка, для которой ключом служит размер таблицы. Например, сообщение ГОРОД МИНСК СТОЛИЦА РЕСПУБЛИКИ БЕЛАРУСЬ записывается в таблицу поочередно по столбцам. Результат заполнения таблицы из 5 строк и 7 столбцов показан на рис. 4.3. После заполнения таблицы текстом сообщения по столбцам для формирования шифротекста считывают содержимое таблицы по строкам.

Г	М	С	Ц	П	К	А
О	И	Т	А	У	И	Р
Р	Н	О	Р	Б	Б	У
О	С	Л	Е	Л	Е	С
Д	К	И	С	И	Л	Ь

Рис. 4.3. Заполнение сообщения в таблице из 5 строк и 7 столбцов

Если шифротекст записывать группами по пять букв, получается такое шифрованное сообщение: ГМСЦП КАОИТ АУИРР НОРББ УОСЛЕ ЛЕСДК ИСИЛЬ.

Естественно, отправитель и получатель сообщения должны заранее условиться об общем ключе в виде размера таблицы. Следует заметить, что объединение букв шифротекста в 5-буквенные группы не входит в ключ шифра и осуществляется для удобства записи не смыслового текста. При расшифровывании действия выполняют в обратном порядке.

Несколько большей стойкостью к раскрытию обладает метод шифрования, называемый одиночной перестановкой по ключу. Этот метод от-

личается от предыдущего тем, что столбцы таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы. Применим в качестве ключа, например, слово РАДОСТЬ, а текст сообщения возьмем из предыдущего примера. На рис. 4.4 показаны две таблицы, заполненные текстом сообщения и ключевым словом, при этом левая таблица соответствует заполнению до перестановки, а правая таблица – заполнению после перестановки.

Ключ →

Р	А	Д	О	С	Т	Ь
4	1	2	3	5	6	7
Г	М	С	Ц	П	К	О
О	И	Т	А	У	И	Р
Р	Н	О	Р	Б	Б	У
О	С	Л	Е	Л	Е	С
Д	К	И	С	И	Л	Ь

До перестановки

А	Д	О	Р	С	Т	Ь
1	2	3	4	5	6	7
М	С	Ц	Г	П	К	О
И	Т	А	О	У	И	Р
Н	О	Р	Р	Б	Б	У
С	Л	Е	О	Л	Е	С
К	И	С	Д	И	Л	Ь

После перестановки

Рис. 4.4. Таблицы, заполненные ключевым словом и текстом сообщения

В верхней строке левой таблицы записан ключ, а номера под буквами ключа определены в соответствии с естественным порядком соответствующих букв ключа в алфавите. Если бы в ключе встретились одинаковые буквы, они были бы пронумерованы слева направо. В правой таблице столбцы переставлены в соответствии с упорядоченными номерами букв ключа. При считывании содержимого правой таблицы по строкам и записи шифротекста группами по пять букв получим шифрованное сообщение

МСЦГП КОИТА ОУИРН ОРРББ УСЛЕО ЛЕСКИ СДИЛЬ

Для обеспечения дополнительной скрытности можно повторно зашифровать сообщение, которое уже было зашифровано. Такой метод шифрования называется двойной перестановкой. В случае двойной перестановки столбцов и строк таблицы перестановки определяются отдельно для столбцов и отдельно для строк. Сначала в таблицу записывается текст сообщения, а потом поочередно переставляются столбцы, а затем строки. При расшифровывании порядок перестановок должен быть обратным. Пример выполнения шифрования методом двойной перестановки показан на рис. 4.5. Если считывать шифротекст из правой таблицы построчно блоками по четыре буквы, то получится следующее

ИАЙК ВДОА ИСНМ ТЗОВ

Ключом к шифру двойной перестановки служит последовательность номеров столбцов и номеров строк исходной таблицы (в нашем примере последовательности 4132 и 3142 соответственно).

Исходная таблица

	4	1	3	2
3	М	И	Н	С
1	К	И	Й	А
4	В	Т	О	З
2	А	В	О	Д

Перестановка столбцов

	1	2	3	4
3	И	С	Н	М
1	И	А	Й	К
4	Т	З	О	В
2	В	Д	О	А

Перестановка строк

	1	2	3	4
1	И	А	Й	К
2	В	Д	О	А
3	И	С	Н	М
4	Т	З	О	В

Рис. 4.5. Шифрование методом двойной перестановки

Число вариантов двойной перестановки быстро возрастает при увеличении размера таблицы: для таблицы 3×3 – 36 вариантов, для таблицы 4×4 – 576 вариантов, для таблицы 5×5 – 14400 вариантов.

Однако двойная перестановка не отличается высокой стойкостью и сравнительно просто «взламывается» при любом размере таблицы шифрования.

4.1.3. Шифры простой замены

При шифровании заменой (подстановкой) символы шифруемого текста заменяются символами того же или другого алфавита с заранее установленным правилом замены. В шифре простой замены каждый символ исходного текста заменяется символами того же алфавита по одному правилу на всем протяжении текста. Часто шифры простой замены называют шифрами одноалфавитной подстановки.

Система шифрования Цезаря

Шифр Цезаря является частным случаем шифра простой замены (одноалфавитной подстановки). Свое название этот шифр получил по имени римского императора Гая Юлия Цезаря, который использовал этот шифр при переписке.

При шифровании исходного текста каждая буква заменялась на другую букву того же алфавита по следующему правилу. Заменяющая буква определялась путем смещения по алфавиту от исходной буквы на *K* букв. При достижении конца алфавита выполнялся циклический переход к его началу. Цезарь использовал шифр замены при смещении *K*=3. Такой шифр замены можно задать таблицей подстановок, содержащей соответствующие пары букв открытого текста и шифротекста. Совокупность возможных подстановок для *K*=3 показана в табл. 4.1.

Таблица 4.1

Одноалфавитные подстановки (K=3, m=26)

A→D	J→M	S→V
B→E	K→N	T→W
C→F	L→O	U→X
D→G	M→P	V→Y
E→H	N→Q	W→Z
F→I	O→R	X→A
G→J	P→S	Y→B
H→K	Q→T	Z→C
I→L	R→U	

Например, известное послание Цезаря VENI VIDI VICI (в переводе на русский означает «Пришел, увидел, победил») выглядело бы в зашифрованном виде так: YHQL YLGL YLFL.

Достоинством системы шифрования Цезаря является простота шифрования и расшифровывания. К недостаткам системы Цезаря можно отнести следующее:

- подстановки, выполняемые в соответствии с системой Цезаря, не маскируют частот появления различных букв исходного открытого текста;
- сохраняется алфавитный порядок в последовательности заменяющих букв; при изменении значения K изменяются только начальные позиции такой последовательности;
- число возможных ключей K мало;
- шифр Цезаря легко вскрывается на основе анализа частот появления букв в шифротексте.

Криптоаналитическая атака против системы одноалфавитной замены начинается с подсчета частот появления символов: определяется число появлений каждой буквы в шифротексте. Затем полученное распределение частот букв в шифротексте сравнивается с распределением частот букв в алфавите исходных сообщений, например, в английском языке. Буква с наивысшей частотой появления в шифротексте заменяется на букву с наивысшей частотой появления в английском языке и т.д. Вероятность успешного вскрытия системы шифрования повышается с увеличением длины шифротекста. Вместе с тем идеи, заложенные в системе шифрования Цезаря, оказались весьма плодотворными, о чем свидетельствуют многочисленные модификации.

Аффинная система подстановок Цезаря

В системе шифрования Цезаря использовались только аддитивные свойства множества чисел. Однако символы можно также умножать по

модулю m. Применяя одновременно операции сложения и умножения по модулю m над буквами алфавита, можно получить систему подстановок, которую называют *аффинной* системой подстановок Цезаря.

В данном преобразовании буква, соответствующая числу t, заменяется на букву, соответствующую числовому значению $(at+b)$ по модулю m. Следует заметить, что данное преобразование является взаимно однозначным отображением тогда и только тогда, когда НОД (a, m) – наибольший общий делитель чисел a и m равен единице, т.е. если a и m – взаимно простые числа.

Пример.

Пусть $m = 26$, $a = 3$, $b = 5$. Тогда, очевидно, НОД (3,26) = 1, и мы получаем следующее соответствие между числовыми кодами букв

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
3t+5	5	8	11	14	17	20	23	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2

Преобразуя числа в буквы английского языка, получаем следующее соответствие для букв открытого текста и шифротекста:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	I	L	O	R	U	X	A	D	G	J	M	P	S	V	Y	B	E	H	K	N	Q	T	W	Z	C

Исходное сообщение HOPE преобразуется в шифротекст AVYR.

Достоинством аффинной системы является удобное управление ключами – ключи шифрования и дешифрования представляются в компактной форме в виде пары чисел (a, b). Недостатки аффинной системы аналогичны недостаткам системы шифрования Цезаря. Аффинная система использовалась на практике несколько веков назад, а сегодня ее применение ограничивается большей частью иллюстрациями основных криптологических положений.

4.1.4. Шифры сложной замены

Шифры сложной замены называют многоалфавитными, т.к. для шифрования каждого символа исходного сообщения применяют свой шифр простой замены. Многоалфавитная подстановка последовательно и циклически меняет используемые алфавиты. При r-алфавитной подстановке символ x_0 исходного сообщения заменяется символом y_0 из алфавита B_0, символ x_1 – символом y_1 из алфавита B_1 , и т.д., символ x_{r-1} заменяется символом y_{r-1} из алфавита B_{r-1} , символ x_r заменяется символом y_r снова из ал-

фавита B_0, и т.д. Общая схема многоалфавитной подстановки для случая $r = 4$ показана на рис. 4.6.

Входной символ	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
Алфавит подстановки	B_0	B_1	B_2	B_3	B_0	B_1	B_2	B_3	B_0	B_1

Рис. 4.6. Схема *r*-алфавитной подстановки для случая $r = 4$

Эффект использования многоалфавитной подстановки заключается в том, что обеспечивается маскировка естественной статистики исходного языка, т.к. конкретный символ из исходного алфавита A может быть преобразован в несколько различных символов шифровальных алфавитов B_j. Степень обеспечиваемой защиты теоретически пропорциональна длине периода r в последовательности используемых алфавитов B_j.

Система шифрования Вижинера

Система Вижинера подобна такой системе шифрования Цезаря, у которой ключ подстановки меняется от буквы к букве. Этот шифр многоалфавитной замены можно описать таблицей шифрования, называемой таблицей (квадратом) Вижинера. На рис. 4.7 и 4.8 показаны таблицы Вижинера для русского и английского алфавитов соответственно. Таблица Вижинера используется как для зашифровывания, так и для расшифровывания. Она имеет два входа:

- верхнюю строку подчеркнутых символов, используемую для считывания очередной буквы исходного открытого текста;
- крайний левый столбец ключа.

Последовательность ключей обычно получают из числовых значений букв ключевого слова. При шифровании исходного сообщения его выписывают в строку, а под ним записывают ключевое слово (или фразу).

Если ключ оказался короче сообщения, то его циклически повторяют. В процессе шифрования находят в верхней строке таблицы очередную букву исходного текста и в левом столбце очередное значение ключа. Очередная буква шифротекста находится на пересечении столбца, определяемого шифруемой буквой, и строки, определяемой числовым значением ключа.

Рассмотрим пример получения шифротекста с помощью таблицы Вижинера. Пусть выбрано ключевое слово АМБРОЗИЯ. Необходимо зашифровать сообщение ПРИЛЕТАЮ СЕДЬМОГО. Выпишем исходное сообщение в строку и запишем под ним ключевое слово с повторением. В

третью строку будем выписывать буквы шифротекста, определяемые из таблицы Вижинера

Сообщение	П	Р	И	Л	Е	Т	А	Ю	С	Е	Д	Ь	М	О	Г	О
Ключ	А	М	Б	Р	О	З	И	Я	А	М	Б	Р	О	З	И	Я
Шифротекст	П	Ъ	Й	Ы	У	Щ	И	Э	С	С	Е	К	Ь	Х	Л	Н

Ключ	**а**	**б**	**в**	**г**	**д**	**е**	**ж**	**з**	**и**	**й**	**к**	**л**	**м**	**н**	**о**	**п**	**р**	**с**	**т**	**у**	**ф**	**х**	**ц**	**ч**	**ш**	**щ**	**ь**	**ы**	**ъ**	**э**	**ю**	**я**
0	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я
1	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а
2	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б
3	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в
4	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г
5	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д
6	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е
7	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж
8	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з
9	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и
10	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й
11	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к
12	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л
13	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м
14	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н
15	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о
16	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
17	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р
18	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с
19	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т
20	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у
21	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф
22	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х
23	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц
24	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч
25	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш
26	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ
27	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь
28	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы
29	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ
30	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э
31	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю

Рис.4.7. Таблица Вижинера для русского алфавита

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

Рис. 4.8. Таблица Вижинера для английского алфавита

4.1.5. Одноразовые шифры

Почти все применяемые на практике шифры характеризуются как условно надежные, поскольку они могут быть в принципе раскрыты при наличии неограниченных вычислительных возможностей. Абсолютно надежные шифры нельзя разрушить даже при использовании неограниченных вычислительных возможностей. Существует единственный такой шифр, применяемый на практике, – одноразовая система шифрования. Характерной особенностью одноразовой системы шифрования является одноразовое использование ключевой последовательности.

Одноразовая система шифрует исходный открытый текст $\overline{X} = (X_0, X_1,..., X_{n-1})$ в шифротекст $\overline{Y} = (Y_0, Y_1, \ldots, Y_{n-1})$ посредством подстановки Цезаря $Y_i = (X_i + K_i) \bmod m$, $0 \le i < n$, где $K_i - i$ -й элемент случайной ключевой последовательности.

Ключевое пространство $\overline{K}$ одноразовой системы представляет собой набор дискретных случайных величин из $\overline{Z}_m$ и содержит m^n значений.

Процедура расшифровывания описывается соотношением

$$X_i = (Y_i - K_i) \bmod m,$$

где $K_i - i$-й элемент той же самой случайной ключевой последовательности.

Этот шифр абсолютно надежен, если набор ключей K_i действительно случаен и непредсказуем. Если криптоаналитик попытается использовать для заданного шифротекста все возможные наборы ключей и восстановить все возможные варианты исходного текста, то они все окажутся равновероятными. Не существует способа выбрать исходный текст, который был действительно послан. Теоретически доказано, что одноразовые системы являются недешифрируемыми системами, поскольку их шифротекст не содержит достаточной информации для восстановления открытого текста.

Возможности применения одноразовой системы ограничены чисто практическими аспектами. Существенным моментом является требование одноразового использования случайной ключевой последовательности. Ключевая последовательность с длиной, не меньшей длины сообщения, должна передаваться получателю сообщения заранее или отдельно по некоторому секретному каналу. Такое требование практически сложно осуществимо для современных систем обработки информации, где требуется шифровать многие миллионы символов, однако в обоснованных случаях построение систем с одноразовыми шифрами является наиболее целесообразным.

4.1.6. Шифрование методом гаммирования

Под гаммированием понимают процесс наложения по определенному закону гаммы шифра на открытые данные. Гамма шифра – это псевдослучайная последовательность, выработанная по заданному алгоритму для зашифровывания открытых данных и расшифровывания принятых данных.

Процесс зашифровывания заключается в генерации гаммы шифра и наложении полученной гаммы на исходный открытый текст обратимым образом, например, с использованием операции сложения по модулю 2.

Следует отметить, что перед зашифровыванием открытые данные разбивают на блоки $T_0^{(i)}$ одинаковой длины, обычно по 64 бита. Гамма шифра вырабатывается в виде последовательности блоков $Г_ш^{(i)}$ аналогичной длины. Уравнение шифрования можно записать в виде

$$T_ш^{(i)} = Г_ш^{(i)} \oplus T_0^{(i)}, \quad i = 1 \ldots M,$$

где $T_ш^{(i)}$ – i-й блок шифротекста;

$Г_ш^{(i)}$ – i-й блок гаммы шифра;

$T_0^{(i)}$ – i-й блок открытого текста;

M – количество блоков открытого текста.

Процесс расшифровывания сводится к повторной генерации гаммы шифра и наложению этой гаммы на принятые данные. Уравнение расшифровывания имеет вид

$$T_0^{(i)} = Г_ш^{(i)} \oplus T_ш^{(i)}.$$

Получаемый этим методом шифротекст достаточно труден для раскрытия, поскольку теперь ключ является переменным. По сути дела гамма шифра должна изменяться случайным образом для каждого шифруемого блока. Если период гаммы превышает длину всего шифруемого текста и злоумышленнику неизвестна никакая часть исходного текста, то такой шифр можно раскрыть только прямым перебором всех вариантов ключа. В этом случае криптостойкость шифра определяется длиной ключа.

4.2. Современные симметричные криптосистемы

4.2.1. Американский стандарт шифрования данных DES

Алгоритм DES использует комбинацию подстановок и перестановок. DES осуществляет зашифровывание 64-битовых блоков данных с помощью 64-битого ключа, в котором значащими являются 56 бит (остальные 8 бит - проверочные биты для контроля на четность). Расшифровывание в DES является операцией, обратной шифрованию, и выполняется путем повторения операций зашифровывания в обратной последовательности.

Обобщенная схема процесса шифрования в алгоритме DES (рис. 4.9) заключается в начальной перестановке бит 64-битого блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке бит.

Следует отметить, что все приводимые таблицы являются стандартными и должны включаться в реализацию алгоритма DES в неизменном виде.

Все перестановки и коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс взлома шифра.

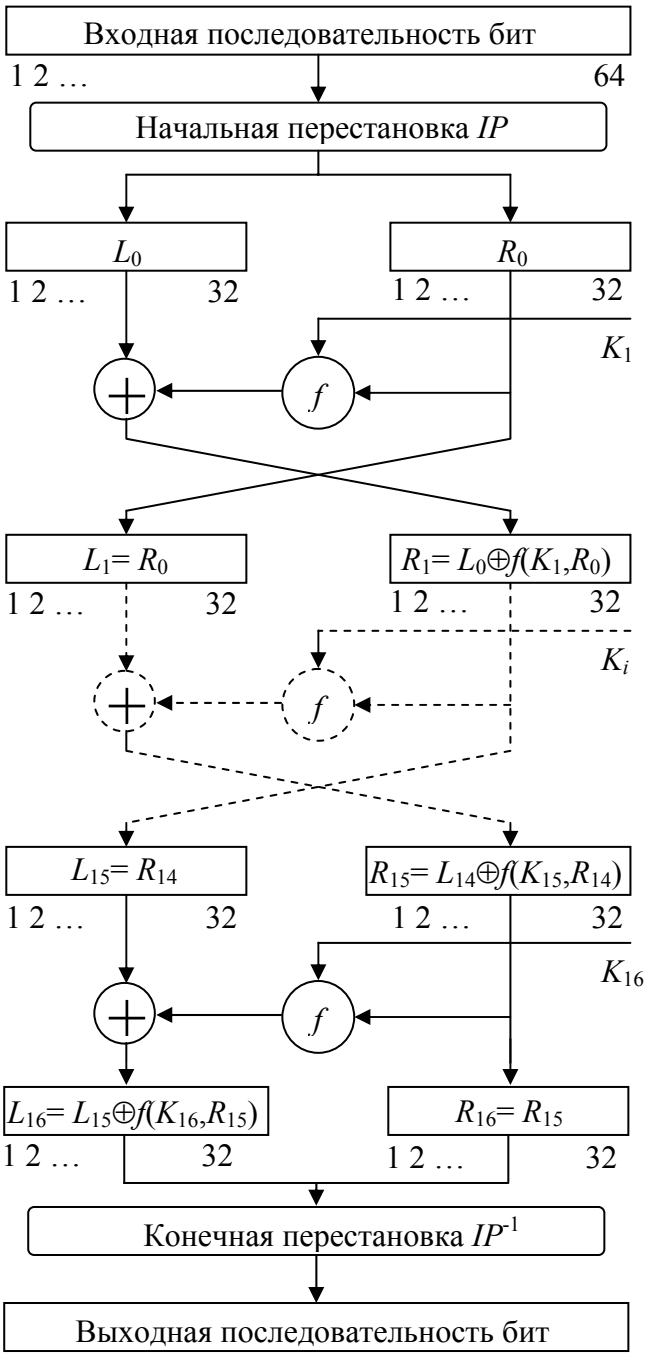


Рис. 4.9. Обобщенная схема зашифровывания в алгоритме DES

Пусть из файла исходного текста считан очередной 64-битовый блок T_0. Этот блок преобразуется с помощью матрицы начальной перестановки *IP* (табл. 4.2).

Биты входного блока T_0 (64 бита) переставляются в соответствии с матрицей IP: бит 58 входного блока T_0 становится битом 1, бит 50 - битом 2 и т.д. Эту перестановку можно описать выражением T_0=IP(T). Полученная последовательность бит T_0 разделяется на две последовательности: L_0 – левые или старшие биты, R_0 - правые или младшие биты, каждая из которых содержит 32 бита.

Таблица 4.2

Начальная перестановка *IP*							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Затем выполняется итеративный процесс шифрования, состоящий из 16 циклов. Пусть T_i - результат i-й итерации

$$T_i = L_i R_i,$$

где $L_i = t_1 t_2 \ldots t_{32}$ (первые 32 бита);

$R_i = t_{33} t_{34} \ldots t_{64}$ (последние 32 бита).

Тогда результат i-й итерации описывается следующими формулами

$$L_i = R_{i-1},\ i = 1,2,\ldots,16;$$
$$R_i = L_{i-1} \oplus f\left(R_{i-1}, K_i\right),\ i = 1,2,\ldots,16.$$

Функция f называется функцией шифрования. Ее аргументами являются последовательность R_{i-1}, получаемая на предыдущем шаге итерации, и 48-битовый ключ K_i, который является результатом преобразования 64-битого ключа шифра K. (Подробнее функция шифрования f и алгоритм получения ключа K описаны ниже.)

На последнем шаге итерации получают последовательности R_{16} и L_{16} (без перестановки местами), которые конкатенируются в 64-битую последовательность $R_{16}L_{16}$.

По окончании шифрования осуществляется восстановление позиций бит с помощью матрицы обратной перестановки IP^{-1} (табл. 4.3).

Процесс расшифровывания данных является инверсным по отношению к процессу шифрования. Все действия должны быть выполнены в обратном порядке. Это означает, что расшифровываемые данные сначала переставляются в соответствии с матрицей IP^{-1}, а затем над последовательностью бит $R_{16}L_{16}$ выполняются те же действия, что и в процессе зашифровывания, но в обратном порядке.

Таблица 4.3

Матрица обратной перестановки

Обратная перестановка IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Итеративный процесс расшифровывания может быть описан следующими формулами

$$R_{i-1} = L_i,\ i = 1,2,\ldots,16;$$

$$L_{i-1} = R_i \oplus f\left(L_i, K_i\right),\ i = 1,2,\ldots,16.$$

Таким образом, для процесса расшифровывания с переставленным входным блоком $R_{16}L_{16}$ на первой итерации используется ключ K_{16}, на второй итерации - K_{15} и т.д. На 16-й итерации используется ключ K_1. На последнем шаге итерации будут получены последовательности L_0 и R_0, которые конкатенируются в 64-битую последовательность L_0R_0. Затем в этой последовательности 64 бита переставляются в соответствии с матрицей IP. Результат такого преобразования - исходная последовательность бит (расшифрованное 64-битовое значение).

Реализация функции шифрования

Схема вычисления функции шифрования $f\left(R_{i-1}, K_{\mathrm{i}}\right)$ показана на рис. 4.10.

Для вычисления значения функции *f* используются:

- функция E (расширение 32 бит до 48);
- функция $S_1, S_2, \ldots, S_8$ (преобразование 6-битового числа в 4-битовое);

- функция P (перестановка бит в 32 битовой последовательности).

Приведем определения этих функций.

Аргументами функции шифрования f являются $R_{i\text{-}1}$ (32 бита) и K_i (48 бит). Результат функции $E\left(R_{i\text{-}1}\right)$ есть 48-битовое число. Функция расширения E, выполняющая расширение 32 бит до 48 (принимает блок из 32 бит и порождает блок из 48 бит), определяется табл. 4.4.

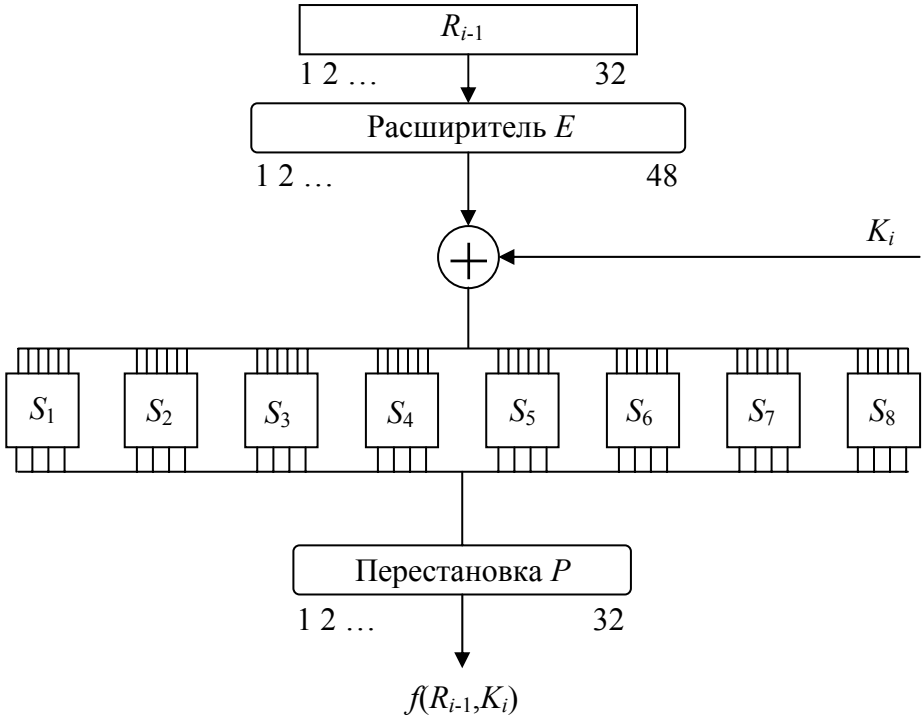


Рис. 4.10. Схема вычисления функции шифрования f

Таблица 4.4

Функция E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

В соответствии с табл. 4.4 первые три бита $E\left(R_{i\text{-}1}\right)$ – это биты 32, 1 и 2, а последние – 31, 32 и 1. Полученный результат (обозначим его $E\left(R_{i\text{-}1}\right)$) складывается по модулю 2 с текущим значением ключа K_i и затем разбивается на

восемь 6-битовых блоков $B_1, B_2, \ldots, B_8 = E\left(R_{i-1}\right) \oplus K_i$. Далее каждый из этих блоков используется как номер элемента в функциях - матрицах $S_1, S_2, \ldots, S_8$, содержащих 4-битовые значения (табл. 4.5).

Следует отметить, что выбор элемента в матрице S осуществляется достаточно оригинальным образом. Пусть на вход матрицы S поступает 6-битовый блок $B_j = b_1 b_2 b_3 b_4 b_5 b_6$, тогда 2-битовое число $b_1 b_6$ указывает номер строки матрицы, а 4-битовое число $b_2 b_3 b_4 b_5$ - номер столбца.

Таблица 4.5

Функции S

		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
S_1	**0**	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	4	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	**0**	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	**0**	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	**0**	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	**0**	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	**0**	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	**0**	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	**0**	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Например, если на вход матрицы S_1 поступает 6-битовый блок $B_1 = b_1b_2b_3b_4b_5b_6 = 100110_{(2)}$, то 2-битовое число $b_1b_6 = 10_{(2)} = 2_{(2)}$ указывает строку с номером 2 матрицы S_1, а 4-битовое число $b_2b_3b_4b_5 = 0011_{(2)} = 3_{(10)}$ указывает столбец с номером 3 матрицы S_1. Это означает, что в матрице S_1 блок $B_1 = 100110$ выбирает элемент на пересечении строки с номером 2 и столбца с номером 3, т.е. элемент $8_{(10)} = 1000_{(2)}$. Совокупность 6-битовых блоков $B_1, B_2, \ldots, B_8$ обеспечивает выбор 4-битового элемента в каждой из матриц $B_1, B_2, \ldots, B_8$. В результате получаем $S_1(B_1), S_2(B_1), \ldots, S_8(B_1)$, т.е. 32-битовый блок (поскольку матрицы S, содержат 4-битовые элементы). Этот 32-битовый блок преобразуется с помощью функции перестановки бит P (табл. 4.6).

Таблица 4.6

Функция *P*

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Таким образом, функция шифрования

$$f(R_{i-1}, K_i) = P(S_1(B_1), \ldots, S_8(B_1)).$$

Алгоритм вычисления ключей

Как нетрудно заметить, на каждой итерации используется новое значение ключа K_i (длиной 48 бит). Новое значение ключа K_i вычисляется из начального ключа K (рис. 4.11).

Ключ K представляет собой 64-битовый блок с 8 битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных бит и подготовки ключа к работе используется функция G первоначальной подготовки ключа (табл. 4.7).

Табл. 4.7 разделена на две части. Результат преобразования $G(\mathrm{K})$ разбивается на две половины C_0 и D_0, по 28 бит каждая. Первые четыре строки матрицы G определяют, как выбираются биты последовательности C (первым битом C_0 будет бит 57 ключа шифра, затем бит 49 и т.д., а последними битами - биты 44 и 36 ключа).

Следующие четыре строки матрицы G определяют, как выбираются биты последовательности D_0 (т.е. последовательность D_0 будет состоять из бит 63, 55, 47,...,12, 4 ключа шифра).

Таблица 4.7

Функция G

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

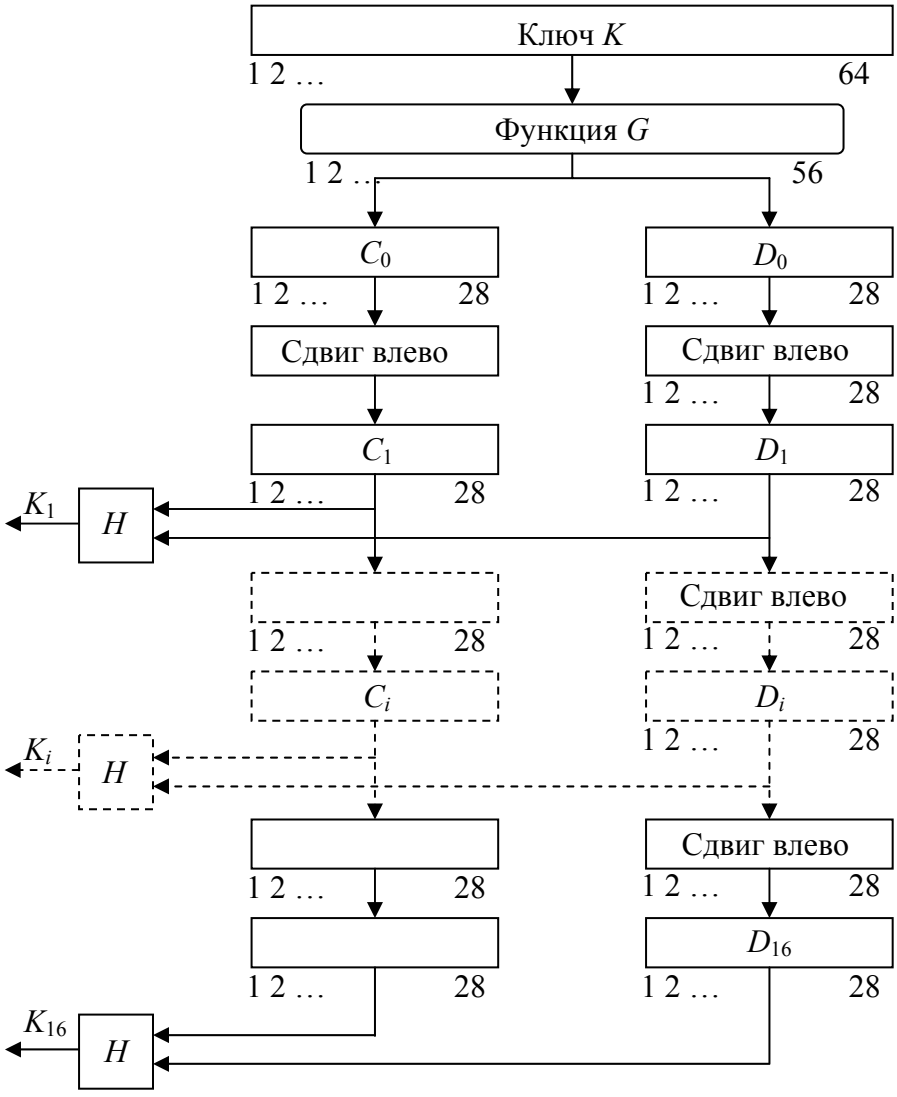


Рис. 4.11. Схема алгоритма вычисления ключей K_i

Как видно из табл. 4.7, для генерации последовательностей C_0 и D_0 не используются биты 8, 16, 24, 32, 40, 48, 56 и 64 ключа шифра. Эти биты не влияют на шифрование и могут служить для других целей (например, для контроля по четности). Таким образом, в действительности ключ шифра является 56-битовым.

После определения C_0 и D_0 рекурсивно определяются C_i и D_i, $i = 1,2,\ldots,16$. Для этого применяются операции циклического сдвига влево на один или два бита в зависимости от номера шага итерации, как показано в табл. 4.8.

Таблица 4.8

Таблица сдвигов для вычисления ключа

Итерация	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Сдвиг влево	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Операции сдвига выполняются для последовательностей C_i и D_i независимо. Например, последовательность C_3 получается посредством циклического сдвига влево на две позиции последовательности C_2, а последовательность D_3 - посредством сдвига влево на две позиции последовательности D_2, C_{16} и D_{16} получаются из C_{15} и D_{15} посредством сдвига влево на одну позицию.

Ключ K_i, определяемый на каждом шаге итерации, есть результат выбора конкретных бит из 56 – битовой последовательности C_iD_i и их перестановки. Другими словами, ключ $K_i = H\left(C_iD_i\right)$, где функция H определяется матрицей, завершающей обработку ключа (табл. 4.9).

Таблица 4.9

Функция H

14	17	11	24	1	5
3	28	15	6	21	10
23	19	22	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Как следует из табл. 4.9, первым битом ключа K_i будет 14-й бит последовательности C_iD_i, вторым – 17-й бит, 47-м битом ключа K_i будет 29-й бит C_iD_i, а 48-м битом – 32-й бит C_iD_i.

4.2.2. Основные режимы работы алгоритма DES

Чтобы воспользоваться алгоритмом DES для решения разнообразных криптографических задач, разработаны четыре рабочих режима:

- электронная кодовая книга ECB (Electronic Code Book);
- сцепление блоков шифра CBC (Cipher Block Chaining);
- обратная связь по шифротексту CPB (Cipher FeedBack);
- обратная связь по выходу OFB (Output FeedBack).

Режим Электронная кодовая книга

Длинный файл разбивают на 64-битые отрезки (блоки) по 8 байт. Каждый из этих блоков шифруют независимо с использованием одного и того же ключа шифрования (рис. 4.12).

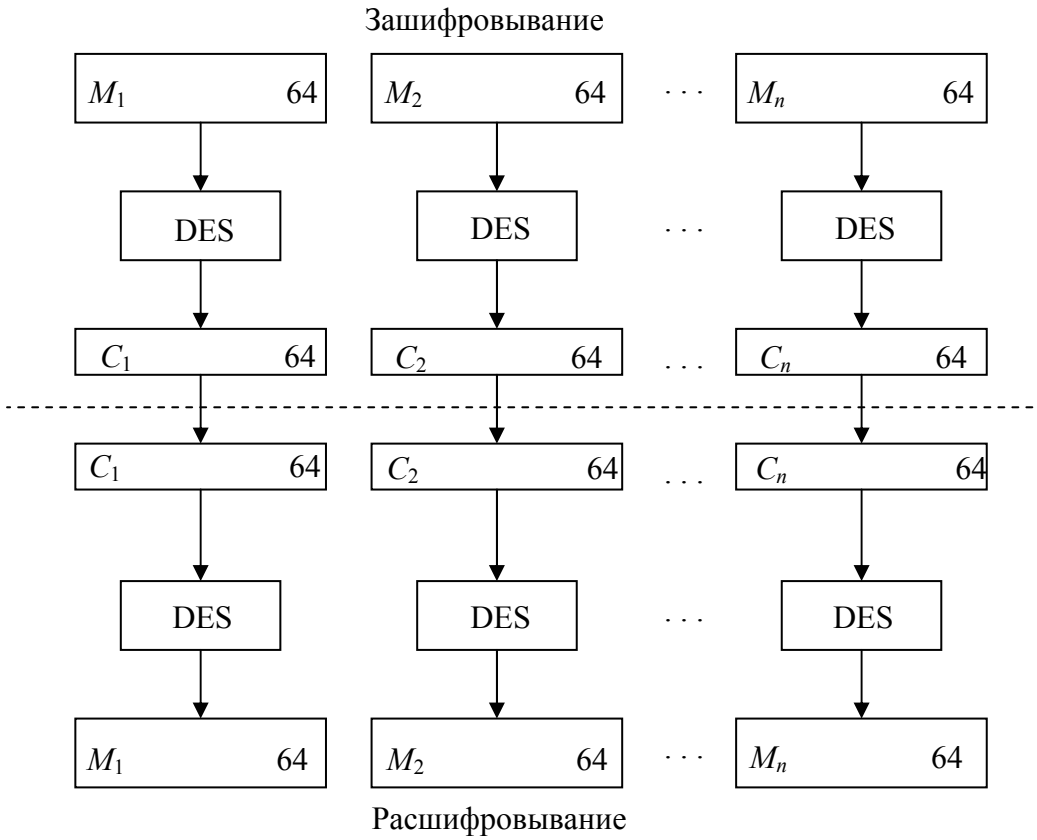


Рис. 4.12. Схема алгоритма DES в режиме электронной кодовой книги

Основное достоинство - простота реализации. Недостаток – относительно слабая устойчивость против квалифицированных криптоаналитиков. Из-за фиксированного характера шифрования при ограниченной длине блока 64 бита возможно проведение криптоанализа «со словарем». Блок такого размера может повториться в сообщении вследствие большой избы-

точности в тексте на естественном языке. Это приводит к тому, что идентичные блоки открытого текста в сообщении будут представлены идентичными блоками шифротекста, что дает криптоаналитику некоторую информацию о содержании сообщения.

Режим Сцепление блоков шифра

В этом режиме исходный файл M разбивается на 64-битые блоки: $M = M_1 M_2 ... M_n$. Первый блок M_1 складывается по модулю 2 с 64-битовым начальным вектором IV, который меняется ежедневно и держится в секрете (рис. 4.13).

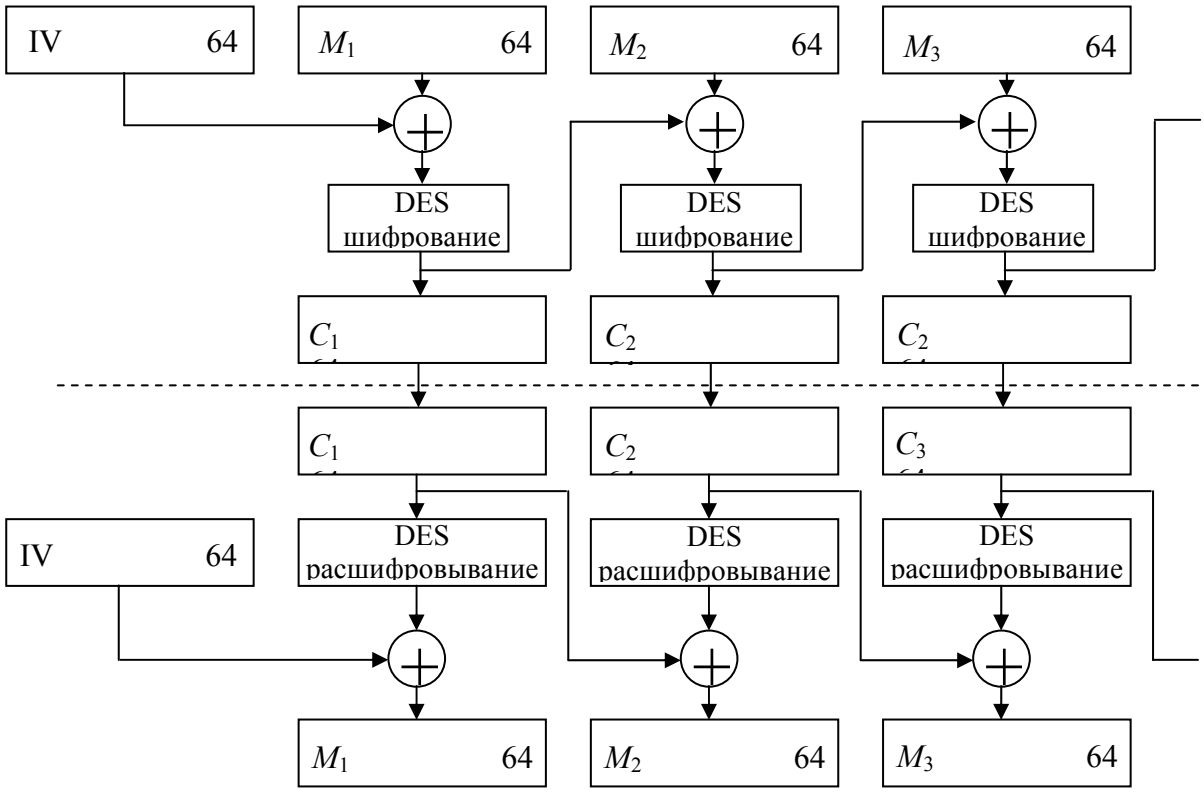


Рис. 4.13. Схема алгоритма DES в режиме сцепления блоков шифра

Полученная сумма затем шифруется с использованием ключа DES, известного и отправителю, и получателю информации. Полученный 64-битовый шифр C_1 складывается по модулю 2 со вторым блоком текста, результат шифруется и получается второй 64-битовый шифр C_2 и т.д. Процедура повторяется до тех пор, пока не будут обработаны все блоки текста.

Таким образом, для всех $i = 1 \ldots n$ (n - число блоков) результат шифрования C определяется следующим образом: $C_i = \mathrm{DES}\left(M_i \oplus C_{i-1}\right)$, где

$C_0 = \text{IV}$ - начальное значение шифра, равное начальному вектору (вектору инициализации).

Очевидно, что последний 64-битовый блок шифротекста является функцией секретного ключа, начального вектора и каждого бита открытого текста независимо от его длины. Этот блок шифротекста называют *кодом аутентификации сообщения (КАС).*

Код КАС может быть легко проверен получателем, владеющим секретным ключом и начальным вектором, путем повторения процедуры, выполненной отправителем. Посторонний, однако, не может осуществить генерацию КАС, который воспринялся бы получателем как подлинный, чтобы добавить его к ложному сообщению, либо отделить КАС от истинного сообщения для использования его с измененным или ложным сообщением.

Достоинство данного режима в том, что он не позволяет накапливаться ошибкам при передаче.

Блок M_i является функцией только C_{i-1} и C_i. Поэтому ошибка при передаче приведет к потере только двух блоков исходного текста.

Режим Обратная связь по шифротексту

В этом режиме размер блока может отличаться от 64 бит (рис 4.14).

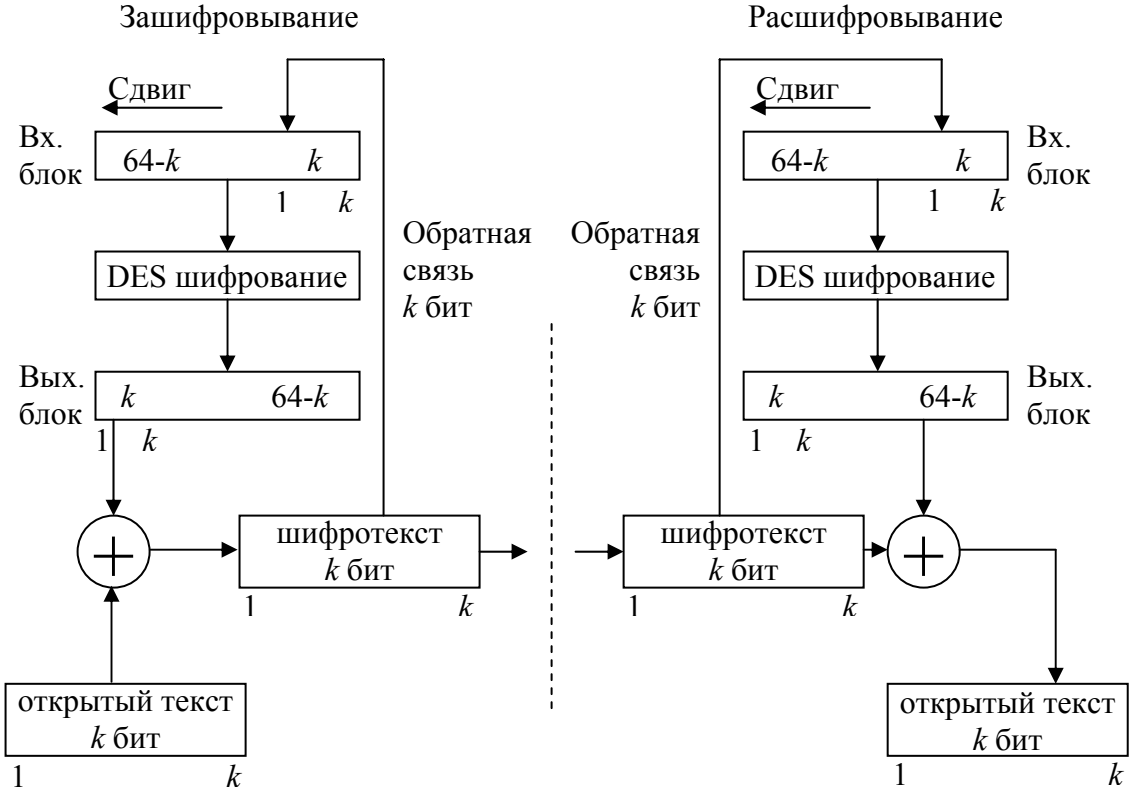


Рис. 4.14. Схема алгоритма DES в режиме обратной связи по шифротексту

Файл, подлежащий шифрованию (расшифровыванию), считывается последовательными блоками длиной k бит ($k = 1 \ldots 64$).

Входной блок (64-битовый регистр сдвига) вначале содержит вектор инициализации, выровненный по правому краю. Предположим, что в результате разбиения на блоки мы получили n блоков длиной k бит каждый (остаток дописывается нулями или пробелами). Тогда для любого $i = 1 \ldots n$ блок шифротекста $C_i = M_i \oplus P_{i-1}$, где P_{i-1} обозначает k старших бит предыдущего зашифрованного блока.

Обновление сдвигового регистра осуществляется путем удаления его старших k бит и записи C_i в регистр. Восстановление зашифрованных данных также выполняется относительно просто: P_{i-1} и C_i вычисляются аналогичным образом и $M_i = C_i \oplus P_{i-1}$.

Режим Обратная связь по выходу

Этот режим тоже использует переменный размер блока и сдвиговый регистр, инициализируемый так же, как в режиме СРВ, а именно - входной блок вначале содержит вектор инициализации IV, выровненный по правому краю (рис.4.15). При этом для каждого сеанса шифрования данных необходимо использовать новое начальное состояние регистра, которое должно пересылаться по каналу открытым текстом.

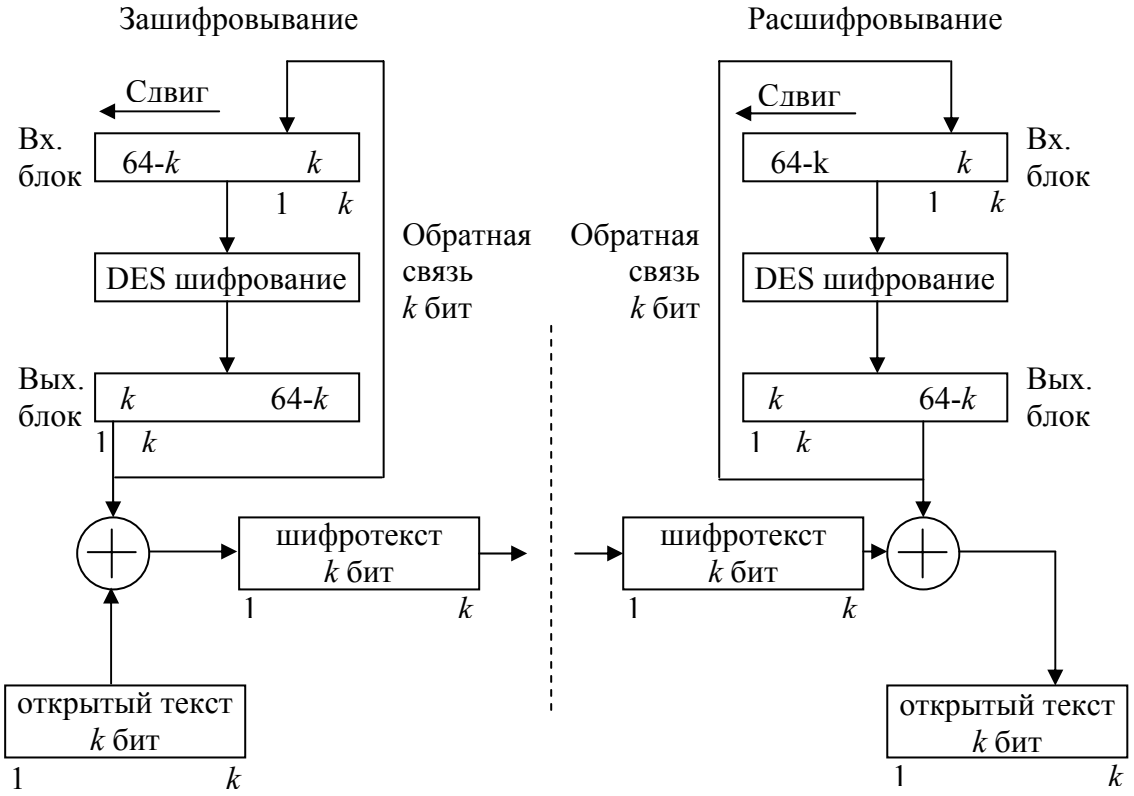


Рис. 4.15. Схема алгоритма DES в режиме обратной связи по выходу

Положим $M = M_1 M_2 ... M_n$ для всех $i = 1 \ldots n$ $C_i = M_i \oplus P_i$, где P_i - старшие k бит операции $\mathrm{DES}(C_{i-1})$. Отличие от режима обратной связи по шифротексту состоит в методе обновления сдвигового регистра.

Это осуществляется путем отбрасывания старших k бит и дописывания справа P_i.

Комбинирование блочных алгоритмов

В настоящее время блочный алгоритм DES считается относительно безопасным алгоритмом шифрования. Он подвергался тщательному криптоанализу в течение 20 лет, и самым практичным способом его взлома является метод перебора всех возможных вариантов ключа. Ключ DES имеет длину 56 бит, поэтому существует 2^{56} возможных вариантов такого ключа. Однако, с учетом вычислительных мощностей современных компьютеров недалеко то время, когда поиск ключа DES методом полного перебора станет возможным для мощных в финансовом отношении государственных и коммерческих организаций.

Возникает естественный вопрос: нельзя ли использовать DES в качестве строительного блока для создания другого алгоритма с более длинным ключом?

В принципе, существует много способов комбинирования блочных алгоритмов для получения новых алгоритмов. Одним из таких способов комбинирования является многократное шифрование, т.е. использование блочного алгоритма несколько раз с разными ключами для шифрования одного и того же блока открытого текста. Двукратное шифрование блока открытого текста одним и тем же ключом не приводит к положительному результату. При использовании одного и того же алгоритма такое шифрование не влияет на сложность криптоаналитической атаки полного перебора.

Рассмотрим эффективность двукратного шифрования блока открытого текста с помощью двух разных ключей. Сначала шифруют блок P ключом K_1, а затем получившийся шифротекст $E_{K_1}(P)$ шифруют ключом K_2. В результате двукратного шифрования получают криптограмму $C = E_{K_2}(E_{K_1}(P))$.

Расшифровывание является обратным процессом: $P = D_{K_1}(D_{K_2}(C))$.

Если блочный алгоритм обладает свойствами группы, то всегда найдется такой ключ K_3, что $C = E_{K_2}(E_{K_1}(P)) = E_{K_3}(P)$.

Если же блочный алгоритм не является группой, то результирующий двукратно шифрованный блок текста окажется намного сложнее для взлома

методом полного перебора вариантов. Вместо 2^n попыток, где n – длина ключа в битах, потребуется 2^{2n} попыток. В частности, если n=64, то двукратно зашифрованный блок текста потребует 2^{128} попыток для нахождения ключа.

Однако Р. Меркль и М. Хеллман показали на примере DES, что, используя метод «обмена времени на память» и криптоаналитическую атаку «встреча посередине», можно взломать такую схему двукратного шифрования за 2^n попыток. Правда, эта атака потребует очень большого объема памяти (для алгоритма с 56-битым ключом потребуется 2^{56} 64-битовых блоков или 10^{17} бит памяти).

Более привлекательную идею предложил У. Тачмен. Суть этой идеи состоит в том, чтобы шифровать блок открытого текста P три раза с помощью двух ключей K_1 и K_2 (рис. 4.16). Процедура шифрования: $C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$, т.е. блок открытого текста P сначала шифруется ключом K_1, затем расшифровывается ключом K_2 и окончательно зашифровывается ключом K_1.

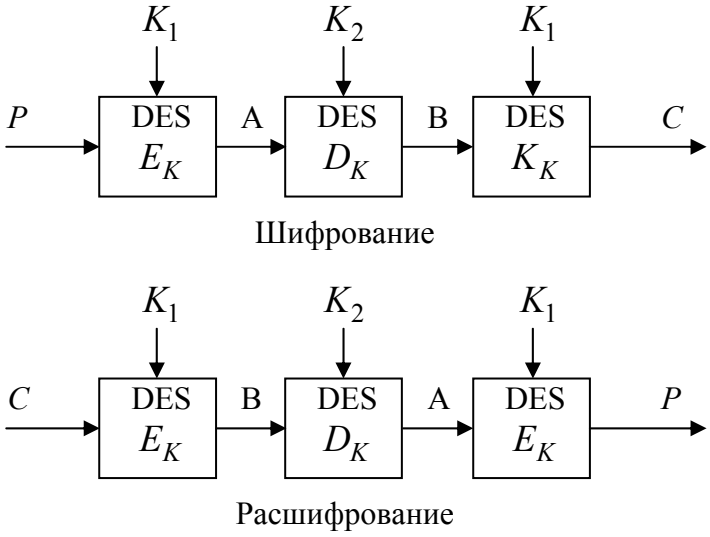


Рис.4.16. Схемы трехкратного применения алгоритма DES с двумя разными ключами

Этот режим иногда называют режимом EDE (encrypt – decrypt – encrypt). Введение в данную схему операции расшифровывания D_{K_2} позволяет обеспечить совместимость этой схемы со схемой однократного использования алгоритма DES. Если в схеме трехкратного использования DES выбрать все ключи одинаковыми, то эта схема превращается в схему однократного использования DES. Процедура расшифровывания выполняется в обратном порядке: $P = D_{K_1}(E_{K_2}(D_{K_1}(C)))$, т.е. блок шифротекста C

сначала расшифровывается ключом K_1, затем зашифровывается ключом K_2 и окончательно расшифровывается ключом K_1.

Если исходный блочный алгоритм имеет *n*-битовый ключ, то схема трехкратного шифрования имеет 2*n*-битовый ключ. Чередование ключей K_1 и K_2 позволяет предотвратить криптоаналитическую атаку «встреча посредине». При трехкратном шифровании можно применить три различных ключа. При этом возрастает общая длина результирующего ключа. Процедуры шифрования и расшифровывания описываются выражениями

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P))),\ P = D_{K_1}(E_{K_2}(D_{K_3}(C))).$$

Трехключевой вариант имеет большую стойкость.

4.2.3. Стандарт шифрования данных ГОСТ 28147-89

В качестве официального алгоритма криптографического преобразования данных для систем обработки информации в Республике Беларусь выбран алгоритм, стандартизованный в ГОСТ 28147-89. Он предназначен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации. Алгоритм шифрования данных представляет собой 64-битовый блочный алгоритм с 256-битовым ключом, который предусматривает четыре режима работы:

- шифрование данных в режиме простой замены;
- шифрование данных в режиме гаммирования;
- шифрование данных в режиме гаммирования с обратной связью;
- выработка имитовставки.

Основными режимами шифрования являются режимы с использованием гаммирования, однако они базируются на использовании шифрования данных в режиме простой замены.

Зашифровывание открытых данных в режиме простой замены

Открытые данные, подлежащие зашифровыванию, разбивают на 64-разрядные блоки T_0. Процедура зашифровывания 64-разрядного блока T_0 в режиме простой замены включает 32 цикла $(j = 1,2,\ldots,32)$. В ключевое запоминающее устройство вводят 256 бит ключа K в виде восьми 32-разрядных подключей (чисел) K_i

$$K = K_7K_6K_5K_4K_3K_2K_1K_0.$$

Последовательность битов блока

$$T_{\text{O}} = \left(a_1(0), a_2(0), \ldots, a_{31}(0), a_{32}(0), b_1(0), b_2(0), \ldots, b_{31}(0), b_{32}(0)\right)$$

разбивают на две последовательности по 32 бита: $b(0)$ и $a(0)$, где $b(0)$ - левые или старшие биты, $a(0)$ - правые или младшие биты.

Работа алгоритма в режиме простой замены изображена на рис. 4.17.

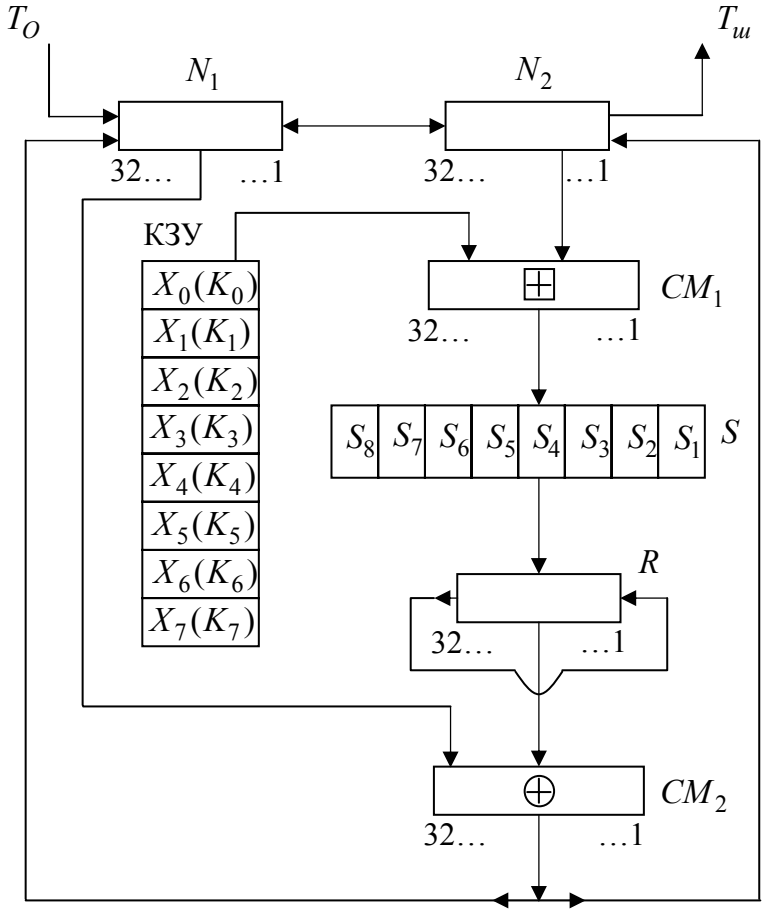


Рис. 4.17. Схема реализации режима простой замены

Обозначения на схеме:

N_1, N_2 – 32-разрядные накопители;

CM_1 – 32-разрядный сумматор по модулю $2^{32}(\boxplus)$;

CM_2 – 32-разрядный сумматор по модулю $2(\oplus)$;

R – 32-разрядный регистр циклического сдвига;

КЗУ – ключевое запоминающее устройство на 256 бит, состоящее из восьми 32-разрядных накопителей $X_0, X_1, X_2, \ldots, X_7$;

S – блок подстановки, состоящий из восьми узлов замены (S-блоков замены) $S_1, S_2, S_3, \ldots, S_8$.

Эти последовательности вводят в накопители N_1 и N_2 перед началом первого цикла зашифровывания. В результате начальное заполнение накопителя N_1

$$\begin{array}{lcl} a(0) & = & \left(a_{32}(0),\ a_{31}(0),\ \ldots,\ a_2(0),\ a_1(0)\right) \\ & & 32,\quad 31,\quad \ldots,\quad 2,\quad 1 \qquad \leftarrow \textit{номер разряда } N_1, \end{array}$$

начальное заполнение накопителя N_2

$$\begin{array}{lcl} b(0) & = & \left(b_{32}(0),\ b_{31}(0),\ \ldots,\ b_2(0),\ b_1(0)\right) \\ & & 32,\quad 31,\quad \ldots,\quad 2,\quad 1 \qquad \leftarrow \textit{номер разряда } N_2. \end{array}$$

Первый цикл $(j=1)$ процедуры зашифровывания 64-разрядного блока открытых данных можно описать уравнениями

$$\begin{cases} a(1) = f\left(a(0) + K_0\right) \oplus b(0), \\ b(1) = a(0). \end{cases}$$

Здесь $a(1)$ - заполнение N_1 после 1-го цикла зашифровывания; $b(1)$ - заполнение N_2 после 1-го цикла зашифровывания; f - функция шифрования.

Аргументом функции f является сумма по модулю 2^{32} числа $a(0)$ (начального заполнения накопителя N_1) и числа K_0 подключа, считываемого из накопителя X_0 КЗУ. Каждое из этих чисел равно 32 битам.

Функция f включает две операции над полученной 32-разрядной суммой $(a(0) + K_0)$.

Первая операция называется подстановкой (заменой) и выполняется блоком подстановки S. Блок подстановки S состоит из восьми узлов замены (S-блоков замены) $S_1, S_2, \ldots, S_8$ с памятью 64 бит каждый. Поступающий из CM_1 на блок подстановки, S 32-разрядный вектор разбивают на восемь последовательно идущих 4-разрядных векторов, каждый из которых преобразуется в четырехразрядный вектор соответствующим узлом замены. Каждый узел замены можно представить в виде таблицы-перестановки шестнадцати четырехразрядных двоичных чисел в диапазоне 0000…1111. Входной вектор указывает адрес строки в таблице, а число в этой строке является выходным вектором. Затем четырехразрядные выходные векторы последовательно объединяют в 32-разрядный вектор. Узлы замены (таблицы-перестановки) представляют собой ключевые элементы, которые являются общими для сетей ТКС и редко изменяются. Эти узлы замены должны сохраняться в секрете.

Вторая операция - циклический сдвиг влево (на 11 разрядов) 32-разрядного вектора, полученного с выхода блока подстановки S. Циклический сдвиг выполняется регистром сдвига R. Затем результат работы функции шифрования f суммируют поразрядно по модулю 2 в сумматоре CM_2 с 32-разрядным начальным заполнением $b(0)$ накопителя N_2. Затем полученный на выходе CM_2 результат (значение $a(1)$) записывают в накопитель N_1, а старое значение N_1 (значение $a(0)$) переписывают в накопитель N_2 (значение $b(1)=a(0)$). Первый цикл завершен. Последующие циклы осуществляются аналогично, при этом во втором цикле из КЗУ считывают заполнение X_1 – подключ K_1, в третьем цикле - подключ K_2 и т.д., в восьмом цикле - подключ K_7. В циклах с 9-го по 16-й, а также в циклах с 17-го по 24-й подключи из КЗУ считываются в том же порядке: $K_0, K_1, K_2, \ldots, K_6, K_7$. В последних восьми циклах с 25-го по 32-й порядок считывания подключей из КЗУ обратный: $K_7, K_6, \ldots, K_2, K_1, K_0$. Таким образом, при зашифровывании в 32 циклах осуществляется следующий порядок выборки из КЗУ подключей:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$
$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

В 32-м цикле результат из сумматора CM_2 вводится в накопитель N_2, а в накопителе N_1 сохраняется прежнее заполнение. Полученные после 32-го цикла зашифровывания заполнения накопителей N_1 и N_2 являются блоком зашифрованных данных $T_{Ш}$, соответствующим блоку открытых данных T_O.

Уравнения зашифровывания в режиме простой замены имеют вид

$$\begin{cases} a(j) = f\left(a(j-1) + K_{(j-1)\bmod 8}\right) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \textit{при } j = 1\ldots 24,$$

$$\begin{cases} a(j) = f\left(a(j-1) + K_{(32-j)\bmod 8}\right) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \textit{при } j = 25\ldots 31,$$

$$\begin{cases} a(32) = a(31) \\ b(32) = f\left(a(31) + K_0\right) \oplus b(31) \end{cases} \quad \textit{при } j = 32.$$

где $a(j) = \left(a_{32}(j), a_{31}(j), \ldots, a_1(j)\right)$ – заполнение N_1 после j-го цикла зашифровывания;

$b(j)=(b_{32}(j),b_{31}(j),\ldots,b_1(j))$ - заполнение N_2 после j-го цикла зашифровывания, $j=1\ldots32$.

Блок зашифрованных данных $T_{Ш}$ (64 разряда) выводится из накопителей N_1, N_2 в следующем порядке: из разрядов 1…32 накопителя N_1, затем из разрядов 1…32 накопителя N_2, т.е. начиная с младших разрядов,

$$T_{Ш}=(a_1(32),a_2(32),\ldots,a_{31}(32),a_{32}(32),b_1(32),b_2(32),\ldots,b_{31}(32),b_{32}(32)).$$

Остальные блоки открытых данных зашифровываются в режиме простой замены аналогично.

Расшифровывание в режиме простой замены

Криптосхема, реализующая алгоритм расшифровывания в режиме простой замены, имеет тот же вид, что и при зашифровывании.

В КЗУ вводят 256 бит ключа, на котором осуществлялось зашифровывание. Зашифрованные данные, подлежащие расшифровыванию, разбиты на блоки $T_{Ш}$, по 64 бита в каждом. Ввод любого блока

$$T_{Ш}=(a_1(32),a_2(32),\ldots,a_{31}(32),a_{32}(32),b_1(32),b_2(32),\ldots,b_{31}(32),b_{32}(32))$$

в накопители N_1 и N_2 производят так, чтобы начальное значение накопителя N_1 имело вид

$$\begin{array}{ccccc} (a_{32}(32), & a_{31}(32), & \ldots, & a_2(32), & a_1(32)) \\ 32, & 31, & \ldots, & 2, & 1 \end{array} \quad \leftarrow \textit{номер разряда } N_1,$$

а начальное заполнение накопителя N_2

$$\begin{array}{ccccc} (b_{32}(32), & b_{31}(32), & \ldots, & b_2(32), & b_1(32)) \\ 32, & 31, & \ldots, & 2, & 1 \end{array} \quad \leftarrow \textit{номер разряда } N_2.$$

Расшифровывание осуществляется по тому же алгоритму, что и зашифровывание, с тем изменением, что заполнения накопителей $X_0,X_1,X_2,\ldots,X_7$ считываются из КЗУ в циклах расшифровывания в следующем порядке:

$$K_0,K_1,K_2,K_3,K_4,K_5,K_6,K_7,K_7,K_6,K_5,K_4,K_3,K_2,K_1,K_0,$$
$$K_7,K_6,K_5,K_4,K_3,K_2,K_1,K_0,K_7,K_6,K_5,K_4,K_3,K_2,K_1,K_0.$$

Уравнения расшифровывания имеют вид

$$\begin{cases} a(32-j) = f\left(a(32-j+1) + K_{j-1}\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при } j = 1\ldots 8,$$

$$\begin{cases} a(32-j) = f\left(a(32-j+1) + K_{(32-j) \bmod 8}\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при } j = 9\ldots 31,$$

$$\begin{cases} a(0) = a(1) \\ b(0) = f\left(a(1) + K_0\right) \oplus b(1) \end{cases} \quad \text{при } j = 32.$$

Полученные после 32 циклов работы заполнения накопителей N_1 и N_2 образуют блок открытых данных

$$T_O = \left(a_1(0), a_2(0), \ldots, a_{31}(0), a_{32}(0), b_1(0), b_2(0), \ldots, b_{31}(0), b_{32}(0)\right),$$

соответствующий блоку зашифрованных данных $T_{Ш}$. При этом состояние накопителя N_1

$$\begin{array}{cccccc} \left(a_{32}(0), & a_{31}(0), & \ldots, & a_2(0), & a_1(0)\right) & \\ 32, & 31, & \ldots, & 2, & 1 & \leftarrow \textit{номер разряда } N_1, \end{array}$$

состояние накопителя N_2

$$\begin{array}{cccccc} \left(b_{32}(0), & b_{31}(0), & \ldots, & b_2(0), & b_1(0)\right) & \\ 32, & 31, & \ldots, & 2, & 1 & \leftarrow \textit{номер разряда } N_2. \end{array}$$

Аналогично расшифровываются остальные блоки зашифрованных данных.

Если алгоритм зашифровывания в режиме простой замены 64-битового блока T_O обозначить через A, то

$$A(T_O) = A\left(a(0), b(0)\right) = \left(a(32), b(32)\right) = T_{Ш}.$$

Следует иметь в виду, что режим простой замены допустимо использовать для шифрования данных только в ограниченных случаях при выработке ключа и зашифровывании его с обеспечением имитозащиты для передачи по каналам связи или для хранения в памяти ЭВМ.

4.3. Современные асимметричные криптосистемы

4.3.1. Криптосистема шифрования данных RSA

Последовательность действий абонентов криптосистемы RSA

Действия получателя криптограммы *B*

1. *B* генерирует два произвольных больших простых числа P и Q.

Эти числа должны быть примерно одинаковыми, размерностью 100 – 150 десятичных разрядов. Они должны быть секретными.

2. *B* вычисляет значение модуля $n = P \cdot Q$ и функции Эйлера $\varphi(n) = (P\text{-}1)\cdot(Q\text{-}1)$ и выбирает значение открытого ключа K_O с соблюдением условий: $1 < K_O \le \varphi(n)$, $(K_O, \varphi(n)) = 1$, т.е. K_O и $\varphi(n)$ должны быть взаимно простыми.($\varphi(n)$ – функция Эйлера, количество положительных целых, меньших n, которые взаимно просты с n).

3. *B* вычисляет значение секретного ключа K_C (обратного числа к числу K_O по модулю $\varphi(n)$),

$$K_C = \left(K_O^{-1}\right) \bmod \varphi(n).$$

4. *B* посылает *A* пару чисел n, K_O по открытому каналу.

Действия отправителя криптограммы *A*

1. Разбивает исходный текст M на блоки M_i, $i = 1,2,\ldots,m$, т.е. $M = M_1, M_2, \ldots, M_m$. Величина $M_i < n$.

2. Шифрует каждое число M_i по формуле $C_i = \left(M_i^{K_O}\right) \bmod n$ и отправляет криптограмму $C = C_1, C_2, \ldots, C_m$.

Получатель *B*, получив криптограмму, расшифровывает каждый блок секретным ключом K_C, $M_i = \left(C_i^{K_C}\right) \bmod n$, и восстанавливает весь текст $M = M_1, M_2, \ldots, M_m$.

Реализуемость и безопасность RSA

Покажем, что при расшифровывании восстанавливается исходный текст. Согласно обобщению Эйлером малой теоремы Ферма: если НОД(*a*,*n*)=1 и $a^{\varphi(n)+1} \equiv a \bmod n$. Открытый K_O и закрытый K_C ключи в алгоритме связаны соотношением $K_O \cdot K_C \equiv 1 \bmod \varphi(n)$, или $K_O \cdot K_C = k \cdot \varphi(n) + 1$ для некоторого целого k. Таким образом, процесс шифрования, а затем расшифровывания некоторого сообщения M_i выглядит следующим образом:

$$\left(\left(M_i^{K_O}\right) \bmod n\right)^{K_C} \bmod n = \left(M_i^{K_O \cdot K_C}\right) \bmod n = \left(M_i^{k\cdot\varphi(n)+1}\right) \bmod n = M_i.$$

В процессе применения RSA злоумышленник может иметь: C_i, K_O, n – и организовать дешифрование двумя способами:

1. По C_i, K_O, n получить M_i. Для этого он решает задачу вычисления M_i из уравнения $C_i = M_i^{K_O} \bmod n$. Эта задача вычислительно трудна.

2. По n вычислить P, Q, затем найти $\varphi(n)$ и вычислить $K_C = \left(K_O^{-1}\right) \bmod \varphi(n)$ и дешифровать сообщение $M_i = C_i^{K_C} \bmod n$.

Однако задача разложения большого числа на простые множители вычислительно сложна.

Пользователи A и B должны быстро осуществлять все вычисления: вычислять K_O, шифровать и расшифровывать.

Вычисление K_O с использованием алгоритма Евклида - довольно быстрый процесс и не представляет трудности. Зашифровывание и расшифровывание – возведение большого числа в большую степень – требует определенных затрат времени, но с учетом наличия быстрых алгоритмов и быстродействия современных компьютеров, это приемлемая процедура.

Пример.

Шифрование сообщения «CAB»

Для простоты вычислений будут использоваться небольшие числа. На практике применяются очень большие.

Действия *B*:

1. Выбирает $P = 3$ и $Q = 11$.

2. Вычисляет модуль $N = P \cdot Q = 3 \cdot 11 = 33$.

3. Вычисляет значение функции Эйлера для $N = 33$

$$\varphi(N) = \varphi(33) = (P - 1)(Q - 1) = 2 \cdot 10 = 20.$$

Выбирает в качестве открытого ключа K_O произвольное число с учетом выполнения условий: $1 < K_O \le 20$, НОД (K_O, 20) = 1. Пусть $K_O = 7$.

4. Вычисляет значение секретного ключа K_C, используя расширенный алгоритм Евклида при сравнении $K_C \equiv 7^{-1} \pmod{20}$. Решение дает

$$K_C = 3.$$

5. Пересылает *A* пару чисел ($N = 33$, $K_O = 7$).

Действия *A*:

6. Представляет шифруемое сообщение как последовательность целых чисел в диапазоне 0 ... 32. Пусть буква A представляется как число 1, буква B – как число 2, буква C – как число 3. Тогда сообщение CAB можно представить как последовательность чисел 312, т.е. $M_1 = 3$, $M_2 = 1$, $M_3 = 2$.

7. Шифрует текст, представленный в виде последовательности чисел M_1, M_2, M_3, используя ключ $K_O = 7$, и $N = 33$, по формуле $C_i = M_i^{K_O} (\bmod N) = M_i^7 (\bmod 33)$.

Получает:

$C_1 = 3^7 \pmod{33} = 2187 \pmod{33} = 9$,

$C_2 = 1^7 \pmod{33} = 1 \pmod{33} = 1$,

$C_3 = 2^7 \pmod{33} = 128 \pmod{33} = 29$.

Отправляет *B* криптограмму $C_1, C_2, C_3 = 9, 1, 29$.

Действия *B*:

8. Расшифровывает принятую криптограмму C_1, C_2, C_3, используя секретный ключ $K_C = 3$, по формуле $M_i = C_i^{K_C} (\mathrm{mod}\, N) = C_i^3 (\mathrm{mod}\, 33)$.

Получает:

$M_1 = 9^3 \pmod{33} = 729 \pmod{33} = 3$,

$M_2 = 1^3 \pmod{33} = 1 \pmod{33} = 1$,

$M_3 = 29^3 \pmod{33} = 24389 \pmod{33} = 2$.

Таким образом, восстановлено исходное сообщение CAB

4.3.2. Схема шифрования данных Эль Гамаля

Безопасность схемы Эль Гамаля обусловлена сложностью вычисления дискретных логарифмов в конечном поле.

Для того чтобы генерировать пару ключей (открытый ключ – секретный ключ), сначала выбирают некоторое большое простое число *P* и большое целое число *G*, причем $G < P$. Числа *P* и *G* могут быть распространены среди группы пользователей. Затем выбирают случайное целое число *X*, причем $X<P$. Число *X* является секретным ключом и должно храниться в секрете. Далее вычисляют $Y = G^X \bmod P$. Число *Y* является открытым ключом.

Для того чтобы зашифровать сообщение *M*, выбирают случайное целое число $1< K< P - 1$ такое, что числа *K* и $(P - 1)$ являются взаимно простыми. Затем вычисляют числа $a = G^K \bmod P$, $b = Y^K M \bmod P$. Пара чисел (a, b) является шифротекстом. Заметим, что длина шифротекста вдвое больше длины исходного открытого текста *M*.

Для того чтобы расшифровать шифротекст (a, b), вычисляют

$$M = b/a^X \bmod P.$$

Поскольку $a^X \equiv G^{KX} \bmod P$, $b/a^X \equiv Y^K M/a^X \equiv G^{KX} M/G^{KX} \equiv M \pmod{P}$, то соотношение справедливо.

Пример.

Выберем $P = 11$, $G = 2$, секретный ключ $X = 8$

Вычисляем $Y = G^X \bmod P = 2^8 \bmod 11 = 256 \bmod 11 = 3$.

Итак, открытый ключ $Y = 3$.

Пусть сообщение $M = 5$. Выберем некоторое случайное число $K = 9$.

Убедимся, что НОД (K, P–1) =1. Действительно, НОД (9, 10) =1. Вычисляем пару чисел a и b:

$$a = G^K \bmod P = 2^9 \bmod 11 = 512 \bmod 11 = 6,$$
$$b = Y^K M \bmod P = = 3^9 \cdot 5 \bmod 11 = 19683 \cdot 5 \bmod 11 = 9.$$

Получим шифротекст (a, b) = (6, 9).

Выполним расшифровывание этого шифротекста. Вычисляем сообщение M, используя секретный ключ X: $M = b/a^X \bmod P = 9/6^8 \bmod 11$. Выражение $M \equiv 9/6^8 \bmod 11$ можно представить в виде $6^8 \cdot M \equiv 9 \bmod 11$ или $1679616 \cdot M \equiv 9 \bmod 11$. Решая данное сравнение, находим $M = 5$.

В реальных схемах шифрования необходимо использовать в качестве модуля P большое целое простое число, имеющее в двоичном представлении длину 512…1024 бит. При программной реализации схемы Эль Гамаля скорость ее работы (на SPARC-II) в режимах зашифровывания и расшифровывания при 160-битовом показателе степени для различных длин модуля P определяется значениями, приведенными в табл. 4.10.

Таблица 4.10

Скорости работы схемы Эль Гамаля

Режим работы	Длина модуля, бит		
	512	768	1024
Зашифровывание	0,33 с	0,80 с	1,09 с
Расшифровывание	0,24 с	0,58 с	0,77 с

Комбинированный метод шифрования

Главным достоинством криптосистем с открытым ключом является их потенциально высокая безопасность: нет необходимости ни передавать, ни сообщать кому бы то ни было значения секретных ключей, ни убеждаться в их подлинности. В симметричных криптосистемах существует опасность раскрытия секретного ключа во время передачи. Однако алгоритмы, лежащие в основе криптосистем с открытым ключом, имеют следующие недостатки:

- генерация новых секретных и открытых ключей основана на генерации новых больших простых чисел, а проверка простоты чисел занимает много процессорного времени;
- процедуры зашифровывания и расшифровывания, связанные с возведением в степень многозначного числа, достаточно громоздки.

Поэтому быстродействие криптосистем с открытым ключом обычно в сотни и более раз меньше быстродействия симметричных криптосистем с секретным ключом.

Комбинированный (гибридный) метод шифрования позволяет сочетать преимущества высокой секретности, присущие асимметричным криптосистемам с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом. При таком подходе криптосистема с открытым ключом применяется для зашифровывания, передачи и последующего расшифровывания только секретного ключа симметричной криптосистемы. А симметричная криптосистема применяется для зашифровывания и передачи исходного открытого текста. В результате криптосистема с открытым ключом не заменяет симметричную криптосистему с секретным ключом, а лишь дополняет ее, позволяя повысить в целом защищенность передаваемой информации. Такой подход иногда называют схемой электронного цифрового конверта.

Если пользователь *A* хочет передать зашифрованное комбинированным методом сообщение *M* пользователю *B*, то порядок его действий будет таков:

1. Создать (например, сгенерировать случайным образом) симметричный ключ, называемый в этом методе сеансовым ключом K_S.

2. Зашифровать сообщение *M* на сеансовом ключе K_S.

3. Зашифровать сеансовый ключ K_S на открытом ключе K_O пользователя B.

4. Передать по открытому каналу связи в адрес пользователя *B* зашифрованное сообщение вместе с зашифрованным сеансовым ключом.

Действия пользователя *B* при получении зашифрованного сообщения и зашифрованного сеансового ключа должны быть обратными:

5. Расшифровать на своем секретном ключе K_C сеансовый ключ K_S.

6. С помощью полученного сеансового ключа K_S расшифровать и прочитать сообщение *M*.

При использовании комбинированного метода шифрования можно быть уверенным в том, что только пользователь *B* сможет правильно расшифровать ключ K_S и прочитать сообщение *M*.

Таким образом, при комбинированном методе шифрования применяются криптографические ключи как симметричных, так и асимметричных криптосистем. Очевидно, выбор длин ключей для каждого типа криптосистемы следует осуществлять таким образом, чтобы злоумышленнику было одинаково трудно атаковать любой механизм защиты комбинированной криптосистемы.

В табл. 4.11 приведены распространенные длины ключей симметричных и асимметричных криптосистем, для которых трудность атаки

полного перебора примерно равна трудности факторизации соответствующих модулей асимметричных криптосистем.

Таблица 4.11

Длины ключей для симметричных и асимметричных криптосистем при одинаковой их криптостойкости

Длина ключа симметричной криптосистемы, бит	Длина ключа асимметричной криптосистемы, бит
56	384
64	512
80	768
112	1792
128	2304

Комбинированный метод шифрования является наиболее рациональным, объединяя в себе высокое быстродействие симметричного шифрования и высокую криптостойкость, гарантируемую системами с открытым ключом.

4.4. Вопросы и задания для самопроверки

4.1. Какие существуют модели каналов с криптографическим кодированием информации?

4.2. Что является ключом в шифрующих таблицах?

4.3. Что является ключом в шифрующих таблицах при двойной перестановке?

4.4. В чем различия между аддитивной и афинной системой Цезаря?

4.5. Какой основной недостаток систем шифрования Цезаря?

4.6. К какому типу шифров относится система шифрования Вижинера?

4.7. Какие достоинства имеет система Вижинера по отношению к системам Цезаря?

4.8. Почему для большинства современных задач не рационально использование одноразовых шифров?

4.9. Что такое гамма шифра?

4.10. В чем принципиальная разница между шифрованием методом гаммирования и одноразовой системой шифрования?

4.11. К какому типу криптосистем относится алгоритм DES?

4.12. Сколько различных ключей существует в алгоритме DES?

4.13. Поясните принцип работы блока замены алгоритма DES.

4.14. Какие существуют режимы работы алгоритма DES?

4.15. Какой размер блока открытого текста в алгоритме DES?

4.16. Каким методом можно повысить криптостойкость алгоритма DES?

4.17. К какому типу криптосистем относится алгоритм ГОСТ 28147-89?

4.18. Сколько различных ключей существует в алгоритме ГОСТ 28147-89?

4.19. Сколько циклов используется в алгоритме ГОСТ 28147-89 для зашифровывания и расшифровывания информации?

4.20. Какой номер подключа, используемого на 9-м цикле расшифровывания информации в алгоритме ГОСТ 28147-89?

4.21. Какие основные достоинства и недостатки симметричных криптосистем?

4.22. Как находится модуль n в криптосистеме RSA?

4.23. Что такое функция Эйлера?

4.24. На чем основана криптостойкость RSA?

4.25. Как связаны открытый и секретный ключи в алгоритме RSA?

4.26. Что такое дискретный логарифм?

4.27. Как соотносится размер шифротекста и открытого текста в алгоритме Эль Гамаля?

4.28. Что такое цифровой конверт?

4.29. Что Вы понимаете под комбинированным методом шифрования?

4.30. Каковы преимущества комбинированных систем шифрования?

4.5. Практическое занятие №3

Классические шифры, ассиметричные криптосистемы

Теория для практического занятия представлена в разделе 4.1, 4.3

Тестовые задания

1. Результат зашифровывания сообщения КОМПЬЮТЕР методом простой перестановки (размер таблицы 3×3) выглядит как

a) КПТОЬЕМЮР;

b) КТПОЕЬМРЮ;

c) ПКТЬОЕЮМР;

d) МЮРКПТОЬЕ.

2. Результат зашифровывания сообщения МЕСТО ВСТРЕЧИ МОСТ методом двойной перестановки (ключ для столбцов 2143, ключ для строк 4132) выглядит как

a) ОВСТМОСТРЕЧИМЕСТ;

b) ОМРМВОЕЕССЧСТТИТ;

c) ЕМТСВОТСЕРИЧОМТС;
d) ВОТСОМТСЕРИЧЕМТС.

3. Результат зашифровывания сообщения ПРИЕДУ ДЕСЯТОГО методом Цезаря (K=5) выглядит как
a) ФХНКЙШЙКЦДЧИУ;
b) УЧДЦКЙШЙКНЧФ;
c) ЙКЦДЧИУФХНКЙШ;
d) ШЙКНЧФУЧДЦКЙ.

4. Результат зашифровывания сообщения ОШИБКА НАЙДЕНА афинной системой подстановок Цезаря (m = 32, a = 7, b = 4) выглядит как
a) ЖМЪЛКДЯДГАЗЯД;
b) РГНУФДЖМАЙФОГ;
c) КУСШТКЫЭЪОЫК;
d) МТРПКЫДЦГМЫЙЕ.

5. Результат зашифровывания сообщения ДОЖДЯ НЕ БУДЕТ методом Вижинера (Ключ: КАМЕНЬ) выглядит как
a) ГВВКЦТТРПФВШ;
b) ВУЛРЫФЩКЙЦРФ;
c) ООТЙМЗПБЯЙТМ;
d) ЛГНУЫВЖХЧМТА.

6. Асимметричной криптосистема называется потому, что
a) в процессе зашифровывания и расшифровывания используются разные ключи;
b) процесс зашифровывания не похож на процесс расшифровывания;
c) в процессе зашифровывания и расшифровывания используется один и тот же ключ;
d) шифротекст не обладает внутренней симметрией.

7. Функция Эйлера – $\varphi(n)$ находит
a) количество чисел больше n, которые являются взаимно простыми с n;
b) количество простых чисел меньше n;
c) количество чисел меньше n, которые не являются взаимно простыми с n;
d) количество простых чисел больше n;
e) количество чисел меньше n, которые являются взаимно простыми с n.

8. Алгоритмы RSA и Эль Гамаля соответственно относятся к криптосистемам
a) асимметричным и симметричным;
b) с открытом ключом;
c) симметричным и асимметричным;

d) с закрытым ключом.

9. Как связаны открытый и секретный ключи в алгоритме RSA?

a) являются обратными числами по модулю n;

b) никак не связаны;

c) их произведение равно n;

d) являются обратными числами по модулю $\varphi(n)$.

10. Как связаны открытый и секретный ключи в алгоритме Эль Гамаля?

a) являются обратными числами;

b) функцией дискретного логарифма;

c) никак не связаны;

d) функцией sin.

Задачи

1. Зашифруйте методом простой перестановки (размер таблицы 4×4) следующие сообщения:

a) КОМПЬЮТЕРНЫЙ ДИСК;

b) ПОГОДА БУДЕТ ЯСНАЯ;

c) МОБИЛЬНЫЙ ТЕЛЕФОН;

d) ЭЛЕКТРИЧЕСТВА НЕТ.

2.Зашифруйте методом двойной перестановки (ключ для столбцов 4321, ключ для строк 2143) следующие сообщения:

a) ЗВУКОВОЕ ДАВЛЕНИЕ;

b) КАТУШКА ЗАЖИГАНИЯ;

c) КОСМИЧЕСКАЯ СВЯЗЬ;

d) МИКРОЭЛЕКТРОНИКА.

3. Зашифруйте методом Цезаря (K=7) следующие сообщения:

a) ОРГАНИЗАЦИЯ ДОРОЖНОГО ДВИЖЕНИЯ;

b) КАСКАД УСИЛЕНИЯ;

c) ДЫРОЧНАЯ ПРОВОДИМОСТЬ;

d) ГАЛЬВАНИЧЕСКИЙ ЭЛЕМЕНТ.

4. Зашифруйте афинной системой подстановок Цезаря (m = 32, a = 13, b = 7) следующие сообщения:

a) ПРЕХОДНАЯ ФУНКЦИЯ;

b) ТЕЛЕВИЗИОННЫЙ СТАНДАРТ;

c) РАМОЧНАЯ АНТЕННА;

d) ГИЛЬБЕРТОВО ПРОСТРАНСТВО.

5.Зашифруйте методом Вижинера (Ключ: ГИЛЬЗА) следующие сообщения:

a) ТЕОРИЯ ВЕРОЯТНОСТЕЙ;

b) БАКТЕРИЦИДНАЯ ЛАМПА;
c) ДАЛЬНЯЯ СВЯЗЬ;
d) КВАРЦЕВЫЙ РЕЗОНАТОР.

6. Найдите значение функции Эйлера – $\varphi(n)$ следующих чисел:

a) 15; b) 72; c) 311; d) 128.

7. Зашифруйте методом RSA ($K_O = 91$, $n = 323$) следующие числа:

a) 35; b) 94; c) 248; d) 236.

8. Расшифруйте методом RSA ($K_C = 589$, $n = 667$) следующие числа:

a) 294; b) 531; c) 7; d) 111.

9. Зашифруйте методом Эль Гамаля ($P = 43$, $G = 33$, $Y = 26$, $K = 11$) следующие числа:

a) 16; b) 7; c) 31; d) 23.

10. Расшифруйте методом Эль Гамаля ($P = 47$, $X = 21$) следующие варианты шифротекста (a,b):

a) $(45,20)$; b) $(45,11)$; c) $(45,14)$; d) $(45,29)$.

4.6. Практическое занятие №4

Современные симметричные криптосистемы

Теория для практического занятия представлена в разделе 4.2.

Тестовые задания

1. Симметричной, криптосистема называется потому, что

a) ключ, используемый в процессе зашифровывания, симметричен ключу, используемому в процессе расшифровывания;
b) в процессе зашифровывания и расшифровывания используется один и тот же ключ;
c) шифротекст обладает внутренней симметрией;
d) в процессе зашифровывания и расшифровывания используются разные ключи.

2. Размер блока информации обрабатываемого алгоритмом DES равен

a) 32 бита; b) 56 бит; c) 64 бита; d) 128 бит.

3. Количество циклов в алгоритме DES равно

a)15; b) 4; c) 16; d) 1.

4. Разрядность подключей K_i, используемых на каждом цикле алгоритма DES, равна

a) 32 бита; b) 48 бит; c) 64 бита; d) 56 бит.

5. Количество различных ключей в алгоритме DES равно

a) 56; b) 2^{64}; c) 2^{56}; d) 2^{48}.

6. Размер блока информации обрабатываемого алгоритмом ГОСТ 28147-89 равен

a) 48 бит; b) 64 бита; c) 128 бит; d) 256 бит.

7. Количество различных ключей в алгоритме ГОСТ 28147-89 равно

a) 2^{127}; b) 2^{128}; c) 2^{64}; d) 11529215046068469 76.

8. Разрядность подключей K_i, используемых на каждом цикле алгоритма ГОСТ 28147-89 равна

a) 32 бита; b) 48 бит; c) 64 бита; d) 56 бит.

9. Количество циклов в алгоритме ГОСТ 28147-89 равно

a) 15; b) 32; c) 16; d) 64.

10. Номер подключа K_i, используемого на десятом цикле процедуры зашифровывания в алгоритме ГОСТ 28147-89, равен

a) 1; b) 2; c) 6; d) 0.

Задачи

1. Переведите число 3^{43} в двоичную систему счисления.

2. Пусть каждая из 16 первых букв русского алфавита (абвгдежзийклмноп) имеет четырехразрядный двоичный код, соответствующий ее номеру от 0 до 15, т.е. *а* – 0000_2, *б* - 0001_2, …, *п* - 1111_2. Составьте из этих букв произвольное сообщение состоящее из 32 букв, затем разбейте полученное сообщение на блоки длиной 64 бита. Значения полученных блоков запишите в десятичной системе счисления.

3. Найдите сумму по модулю 2 следующих пар чисел:

a) 224489301 и 28973675;
b) 3479913811 и 2301120149;
c) 3040958609 и 2781188359;
d) 3075166647 и 3785852425.

4. Найдите сумму по модулю 3 следующих пар чисел

a) 3496 и 3718; b) 3668 и 1419; c) 5563 и 6482; d) 6379 и 1215.

5. Найдите сумму по модулю 2^{32} следующих пар чисел:

a) 3037741847 и 1257225448;
b) 2706981523 и 1587985773;
c) 2597745569 и 1697221728.

6. Найдите состояние 28-разрядного двоичного регистра сдвига после циклического сдвига влево на 5 числа, предварительно записанного в регистр:

a) 179317333;b) 183838335; c) 250862994; d) 124140475.

7. Найдите состояние 64-разрядного двоичного регистра сдвига после сдвига (не циклического) влево на 3 числа, предварительно записанного в регистр:

a) 16858206854648677463;
b) 14106664103838071685;
c) 13974973798771614621;
d) 13862897145741766693.

8. Найдите 4-х битовое число на выходе блока замены S_2 (функции шифрования алгоритма DES), если на вход подать число

a) 15; b) 62; c) 23; d) 45.

9. Найдите 32-х битовое число на выходе блока замены *S* (алгоритма ГОСТ 28147-89), если на вход подать число

a) 2425314973; b) 2513012245; c) 3424196711; d) 3619058277.

10. Найдите 64-х битовое число на выходе блока перестановки *IP* алгоритма DES, если на вход подано число

a) 16175076002153763172;
b) 11870809655824700300;
c) 14436987034089088762;
d) 12727938706001950544.

Модуль 5

ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

Цель модуля – изучение моделей каналов передачи информации, наблюдаемых в них помех, параметров и границ кодов, контролирующих ошибки, вероятностей декодирования.

В результате изучения модуля студенты должны:

- знать основные модели каналов передачи информации;
- иметь представление об ошибках, возникающих в каналах передачи информации;
- знать основные параметры помехоустойчивых кодов;
- иметь представление о границах возможностей помехоустойчивых кодов контролировать ошибки;
- уметь рассчитывать вероятностные параметры при использовании помехоустойчивых кодов.

Содержание модуля

5.1. Ошибки в каналах передачи информации и их модели

В каналах передачи, хранения, обработки и распределения информации наблюдаются разнообразные помехи различной физической природы. Это могут быть случайные и преднамеренные электромагнитные помехи, шумы, загрязнения и дефекты в оптических, магнитных и полупроводниковых материалах, взаимные помехи функциональных узлов оборудования и т.п. Помехи приводят к ошибкам в принимаемой информации, которые делятся на случайные и зависимые.

Случайные ошибки возникают независимо друг от друга. При этом говорят о кратности ошибок t на длине сообщения. Зависимые ошибки делятся на пакетные и модульные (частный случай пакета ошибок). Пакет ошибок, который может находиться в произвольном месте на длине передаваемого сообщения, описывается длиной p и задается вектором ошибок $e=(e_0,e_1,..,e_{p-1})$, $e\in(0;1)$. Например, для $p = 4$ вектор ошибок может выглядеть следующим образом: $e_1=(1001)$ или $e_2=(1101)$, или $e_3=(1100)$, или на длине сообщения – E=(00... 0100100...0). Пакеты ошибок могут вызываться, например, замираниями в линиях связи, помехами от источников периодического шума, пылинками на магнитной ленте и т.д.

Модуль ошибок – это фазированный пакет ошибок, границы которого известны. Длина модуля ошибок обозначается через b. Наиболее характерным примером модульных ошибок являются ошибки, наблюдаемые в системах памяти, построенных на многоразрядных БИС ЗУ. Например, в 16-разрядной системе памяти, реализованной на микросхемах с четырехразрядными выходами, наиболее характерны модульные ошибки длиной $b = 4$ из-за отказов отдельных БИС ЗУ. Пакеты и модули бывают как однократными, так и многократными. Например, для $b = 4$ двукратный модуль ошибок может иметь вид: E =(⋮ … ⋮1011 ⋮..... ⋮1111 ⋮..... ⋮) , где ⋮ – границы модулей.

Ошибки по вероятности их возникновения разделяют на симметричные, асимметричные и однонаправленные. Симметричные ошибки характерны для двоично-симметричного канала (ДСК). В этом канале вероятность перехода «0»→ «1», «1»→ «0» одинакова и равна p (рис. 5.1).

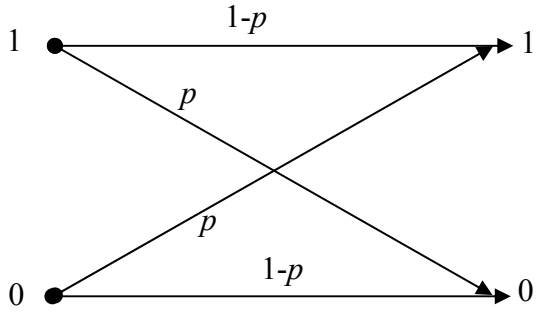


Рис. 5.1. Модель ДСК

В асимметричном канале вероятности перехода «0» → «1» и «1» → «0» различны, а в полностью асимметричном канале одна из вероятностей перехода, например «0» → «1», равняется нулю. Характерным примером полностью асимметричного канала является программируемые ПЗУ

с плавкими вставками, когда до программирования все неисправные ячейки памяти находятся, например, в единичном состоянии, а из-за дефектов производства отдельные ячейки находятся в нулевом состоянии, т.е. наблюдается только ошибочный переход из «1» → «0».

Однонаправленные ошибки – это ошибки, когда во всех разрядах сообщения присутствуют только нули или единицы, т.е. принимаемое из канала слово нулевое или единичное. Эти ошибки возникают, например, при отключении питания у плат микросхем по каким-либо причинам.

Успешное применение помехоустойчивых кодов в реальных каналах зависит от согласования корректирующей способности кодов с наблюдаемыми ошибками в сообщениях из-за помех. Поэтому исследуется статистика ошибок в различных каналах. Результаты экспериментов показывают, что, например, вероятности появления ошибок p на входе канального приемника (декодера) находятся в пределах:

- телефонного и радиорелейного $10^{-3} – 10^{-5}$;
- магнитной ленты $10^{-5} – 10^{-6}$;
- –телеметрического $10^{-6} – 10^{-10}$;
- волоконно-оптического $10^{-7} – 10^{-8}$;
- оптического диска $10^{-4} – 10^{-6}$.

Для цифровых устройств вероятность появления ошибок делится на вероятность ошибок из-за отказов и сбоев элементов. Анализ статистических данных показывает, что вероятность отказов на порядок ниже вероятности сбоев. Под отказом элемента понимается состояние элемента, которое не изменяется при входных воздействиях на элемент (в дальнейшем под отказом будем понимать понятие «дефект», как это принято в теории информации). *Сбой* – это состояние элемента, которое переходит из ложного в верное под воздействием входных сигналов.

Реальные каналы передачи информации описываются моделями, с помощью которых можно выбрать соответствующие коды и методы их обработки для эффективной борьбы с ошибками.

Наиболее широко используется модель канала с неизвестным местоположением ошибок как при кодировании, так и при декодировании информации (рис. 5.2).

Рис. 5.2. Модель канала с ошибками

На практике используется и канал со стираниями. *Стирание* – ошибка, местоположение которой известно при декодировании, но неизвестно истинное состояние ошибочного разряда (рис. 5.3).

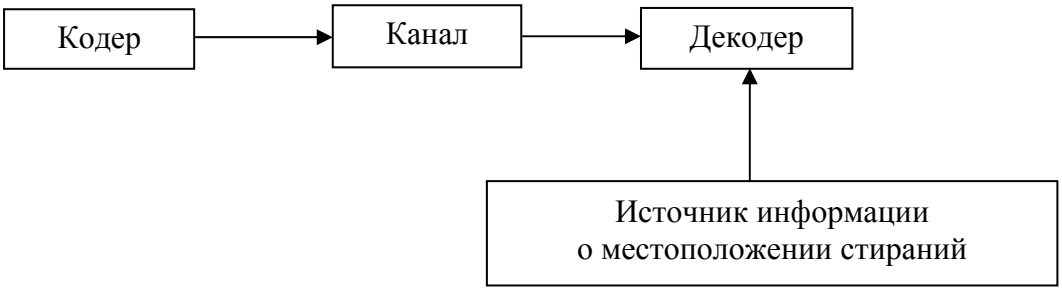


Рис. 5.3. Модель канала со стираниями

Информацию о местоположении стираний можно получить, например, для ЗУ с помощью диагностического оборудования, при применении кодов, обнаруживающих ошибки, или введя порог при обработке сигналов. В этом случае образуется троичный сигнал $\{0,1,S\}$, где, например, принятое сообщение может иметь вид $A^*=(0,1,S,0,S)$. Наличие дополнительной информации о стираниях (третьем символе) S позволяет в два раза повысить эффективность при использовании одного и того же кода по сравнению с каналом с ошибками.

Канал с дефектами симметричен каналу со стираниями, т.е. в этом канале известно местоположение ошибок из-за дефектов при кодировании информации. Одновременно известно, в каком состоянии находятся дефекты (рис. 5.4).

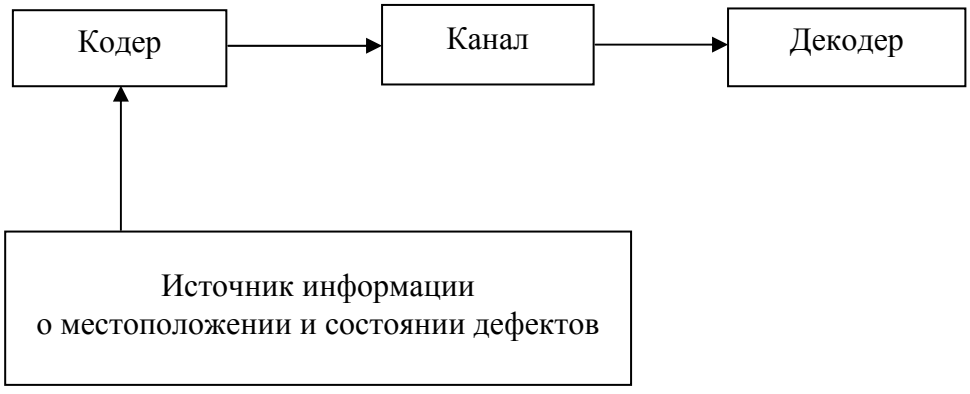


Рис. 5.4. Модель канала с дефектами

Т.е. в канале с дефектами имеется информация о согласовании или несогласовании передаваемой информации с состоянием дефектов. По аналогии, как и для канала со стираниями, наличие дополнительной информа-

ции позволяет более эффективно использовать корректирующие возможности существующих кодов.

Применение диагностического оборудования позволяет определять местоположение отказов элементов памяти супер-ЭВМ и эту информацию хранить в дополнительной памяти для обхода отказавших ячеек. В этом случае известно местоположение ошибок как при кодировании, так и при декодировании информации (рис. 5.5).

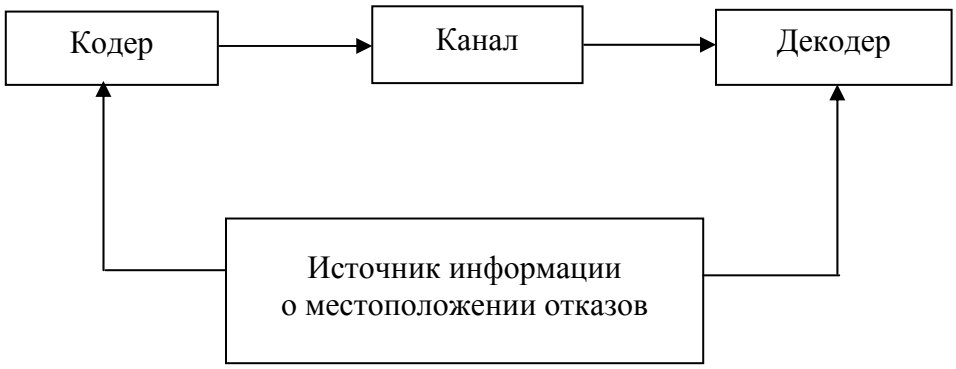
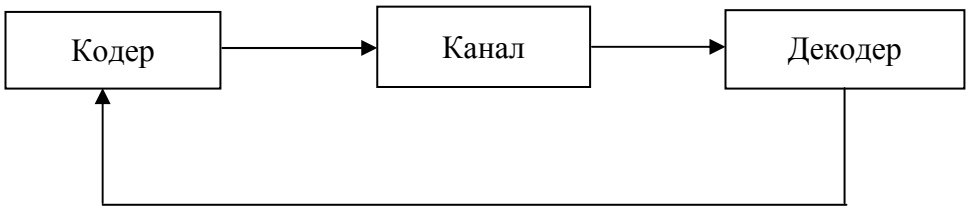


Рис. 5.5. Модель канала с отказами

Необходимо построить устройство переадресации (кодек) к исправным ячейкам по информации, хранящейся в дополнительной памяти.

В каналах передачи данных, где ошибки встречаются редко (волоконно-оптические линии связи, отдельные космические системы связи и т.д.) эффективным является режим обнаружения ошибок с последующей передачей сигнала о неправильно принятом сообщении и повторной передачи сообщения. Данная ситуация хорошо описывается моделью канала с обратной связью (рис. 5.6).



Сигнал о получении ошибочного сообщения

Рис. 5.6. Модель канала с обратной связью

В теории информации интенсивно изучается так называемый широковещательный канал передачи информации, к которому, например, относится спутниковое и кабельное телевидение, скрытая связь, когда

необходимо передавать информацию от многих источников через один передатчик (кодер) многим приемникам (декодерам) согласно предписанному алгоритму и заданной вероятности ошибки на выходе декодера (рис. 5.7).

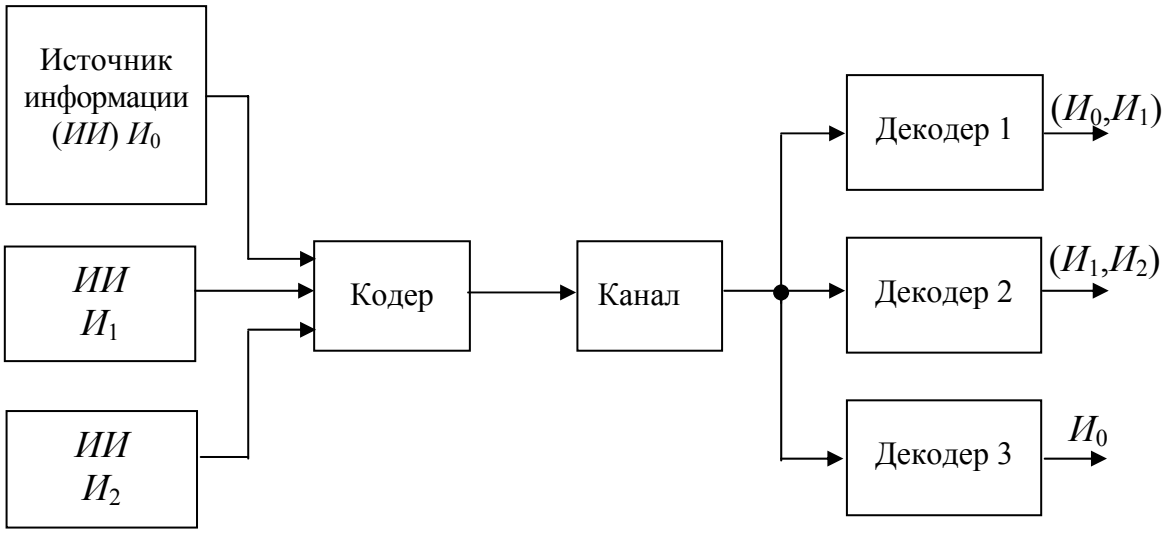


Рис. 5.7. Модель широковещательного канала передачи информации

Целенаправленное изучение причин и статистики ошибок, особенностей протоколов работы систем позволяет создать модель канала, на основании которой возможна разработка эффективных методов борьбы с ошибками.

5.2. Основные параметры кодов. Границы кодов, исправляющих ошибки

5.2.1. Кодовое расстояние. Контроль ошибок кодами

Любую кодовую комбинацию можно представить n-разрядным вектором (словом). Например, при $n = 3$ имеется восемь двоичных комбинаций, которые можно представить трехкоординатным вектором $X=(x, x_2, x_3)$ (рис. 5.8). При передаче сообщений под воздействием помех происходит искажение слов (изменение на обратное состояний отдельных разрядов слов).

Если для передачи использовались все слова кода, называемого в этом случае полным или безызбыточным, то, очевидно, что при искажениях одни слова переходят в другие и, следовательно, не имеется возможности контролировать ошибки. Если же все множество слов полного кода разделить на две части – разрешенных для передачи слов (кодовых) и запрещенных, –

то имеется возможность контролировать ошибки благодаря соответствующему выбору кодовых слов из всего множества слов полного кода. Разрешенные слова следует выбирать таким образом, чтобы они располагались в пространстве как можно дальше друг от друга. В этом случае принятые искаженные слова будут принадлежать к множеству запрещенных слов. Переход разрешенных слов в запрещенные под воздействием помех и используется для контроля за ошибками.

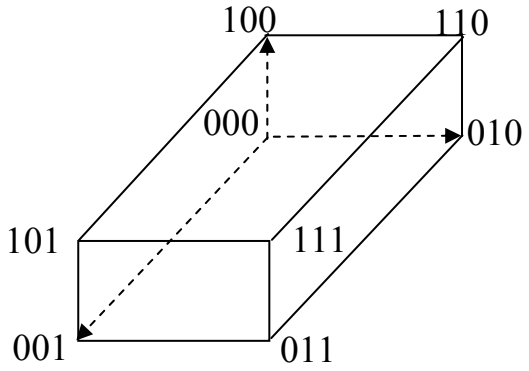


Рис.5.8. Векторное представление слов трехразрядного кода

Коды, в которых для передачи информации используются не все слова полного кода, получили название *избыточных кодов*. Очевидно, что для того чтобы передаваемые векторы (слова) можно было отличать друг от друга, необходимо располагать эти векторы в n-мерном пространстве как можно дальше друг от друга. Степень удаленности векторов измеряется расстоянием между разрешенными векторами, определяемыми числом ребер, которые необходимо пройти, чтобы попасть из вершины одного вектора в вершину другого. Для измерения расстояния используют расстояние Хэмминга d_x, которое определяется как число несовпадающих позиций при попарном сравнении векторов (слов) d_x=dist(A_i, A_j), или иначе, d_x показывает, сколько позиций в слове нужно исказить, чтобы перевести одно слово в другое. Например, при

A_i=(0 0 0 1 1 1 1)
$\underline{A_j}$=(0 0 1 0 1 0 0)
– – 1 1 – 1 1 $\rightarrow$ d_x=dist(A_i, A_j)=4.

Т.е. чтобы перевести слово A_i, в A_j, необходимо изменить четыре позиции. Геометрически расстояние Хэмминга d_x – число ребер в n-мерном кубе, которое нужно пройти, чтобы перевести i-слово в j-е. Если попарно сравнить все кодовые (разрешенные) слова между собой, то можно постро-

ить так называемый спектр весов кода (зависимость числа L слов, отстоящих на одно и то же d_x) (рис. 5.9).

Кодовое расстояние кода d – минимальное расстояние Хэмминга, $d = \min d_x$. Отметим, что в общем случае вес слова w – число ненулевых позиций в слове (например, w_{Ai}=4, w_{Aj}=2).

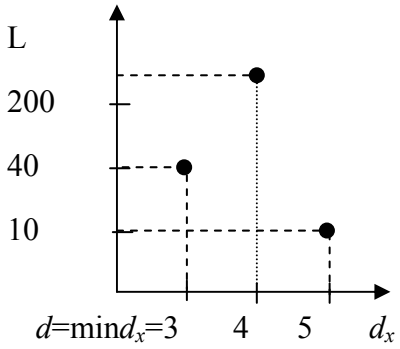


Рис. 5.9. Спектр весов кода

Известна теорема, согласно которой минимальное расстояние d линейного кода равно минимальному весу его ненулевых кодовых слов.

Возникает вопрос: каким должно быть кодовое расстояние d, чтобы код мог обнаруживать $t_{об}$ и исправлять $t_и$ ошибок, или как зависит кратность контролируемых кодом ошибок от кодового расстояния?

Обнаружение ошибок

Пусть слова кода отстоят друг от друга на кодовое расстояние $d = 2$, т.е. чтобы перевести слово A_i, в A_j, необходимо пройти два ребра, разделенных запрещенным словом. Если вокруг каждой кодовой точки (слова) A_i, провести окружности единичного радиуса, то искажение информации в одном разряде приведет к переходу разрешенного слова A_i в запрещенные слова, расположенные на окружности. Очевидно, что при этом любая однократная ошибка будет обнаруживаться, т.е. для того чтобы код обнаруживал ошибки кратностью $t_{об}$=1, достаточно кодового расстояния $d = 2$ (рис. 5.10).

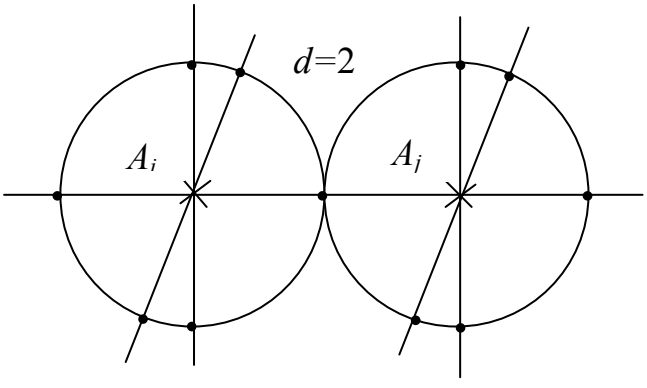


Рис. 5.10. Разрешенные (кодовые) (*) и запрещенные (•) слова кода при d=2

Если же кодовые слова A_i, и A_j разделить тремя ребрами (d = 3) и провести окружности радиусами два, то при этом можно обнаружить любую двукратную ошибку $t_{об}$ = 2 в сообщениях, поскольку при этой ошибке разрешенные слова всегда переходят в запрещенные, лежащие на данной окружности или внутри ее.

В общем случае $d \geq t_{об}+1$ или $t_{об} \leq d-1$.

Например, для того чтобы обнаружить ошибку кратностью $t_{об}$= 4, необходимо выбрать код с кодовым расстоянием d = 5.

Исправление ошибок

Пусть слова кода отстоят друг от друга на расстоянии d = 3, и вокруг каждого кодового слова проведем окружности единичного радиуса. При однократной ошибке эти слова переходят в запрещенные, расположенные на окружности вокруг каждого кодового слова, и эти слова при t_u=1 по принципу близости можно идентифицировать с центром окружности A_i (рис. 5.11).

По аналогии можно показать, что в общем случае для исправления t_u -кратных ошибок достаточно кодового расстояния $d \geq 2t_u+1$ или $t_u \leq (d-1)/2$.

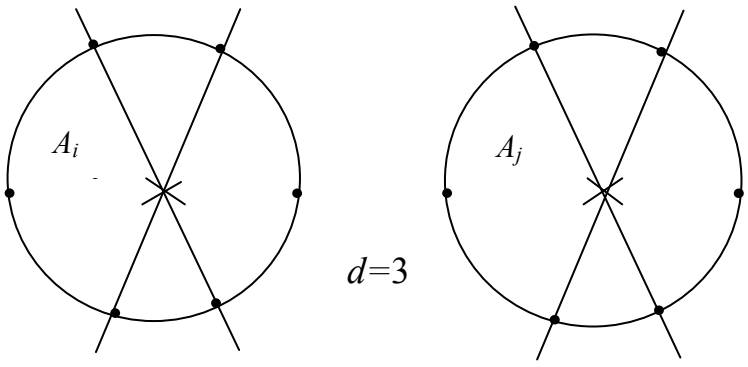


Рис. 5.11. Разрешенные (кодовые – *) и запрещенные (•) слова при d=3

Из рис. 5.9 для спектра весов кода следует, что код с кодовым расстоянием d гарантированно исправляет (обнаруживает) все ошибки кратностью t и меньше, однако может контролировать часть ошибок большей кратности, поскольку некоторые слова отстоят на расстоянии больше чем d.

Для коррекции t -кратных стираний или дефектов достаточно кодового расстояния $d \geq t+1$, т.е. с помощью одного и того же кода можно корректировать в два раза больше стираний и дефектов, чем ошибок.

5.2.2. Основные параметры кодов

Помехоустойчивый код характеризуют следующими параметрами:

1. Основание кода q – число элементарных символов, выбранных для передачи сообщений. Например, для двоичного и троичного кода q_2={0;1}, q_3={-1; 0; 1}.

2. Длина кода n – число символов, выбранных для передачи сообщений. Коды бывают блоковые и непрерывные. Блоковые коды в свою очередь делятся на равномерные, в которых длина кода постоянна n = const, и неравномерные – n = var.

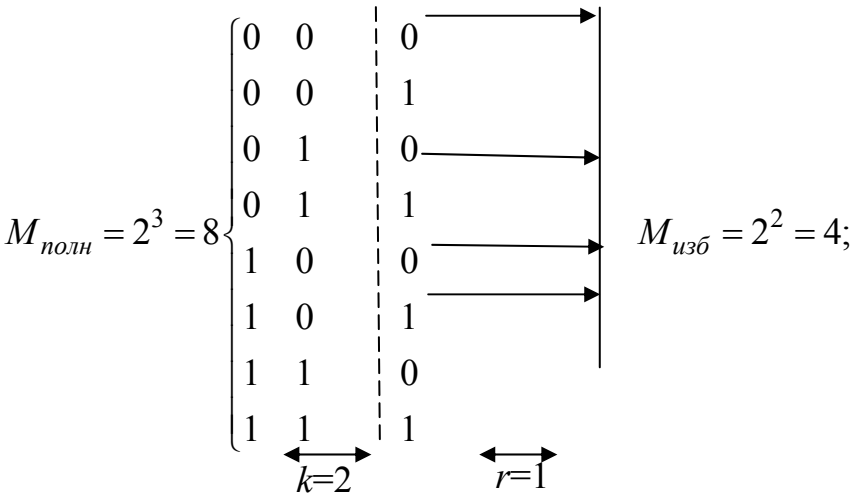
В непрерывных кодах, к которым относятся широко используемые сверточные коды, сообщения передаются без деления на блоки (у этих кодов роль длины блока n, на которой контролируется определенное число ошибок, играет кодовое ограничение).

3. Число информационных позиций в коде, выбранных для передачи данных, – k.

4. Число проверочных (контрольных) позиций в коде – $r=n-k$.

5. Мощность кода M – число кодовых комбинаций, выбранных для передачи сообщений. В полном безызбыточном коде $M_{полн}=q^n$, в помехоустойчивом (избыточном) коде $M_{изб}=q^k$. Очевидно, что всегда $M_{полн}<M_{изб}$.

Например, в коде с $q=2$, $n=3$ число кодовых комбинаций $M_{полн}$, $M_{изб}$ кодов равно



6. Скорость передачи кода $R=k/n$ характеризует качество кода.

7. Кратность контролируемой ошибки t ($t_{обн}$ или $t_и$).

8. Кодовое расстояние кода d характеризует возможности кода по контролю ошибок.

9. Вероятность (частота) ошибки на бит – p.

На практике коды, как правило, обозначают (n; k) или (n; k; d).

Например, обозначение (7; 4) или (7; 4; 3) характеризует блоковый код Хэмминга длиной n=7, числом информационных позиций k=4, кодовым расстоянием d=3.

5.2.3. Границы кодов, исправляющих ошибки

В теории кодирования доказывается ряд теорем о верхних и нижних границах кодов, которые указывают на возможности кодов контролировать ошибки. Эти возможности определяются структурой кодового пространства. Очевидно, что при увеличении кодового расстояния d кратность контролируемых ошибок t возрастает за счет увеличения числа избыточных символов r, а это приводит к уменьшению числа информационных символов k и, следовательно, множества кодовых слов $M=q^k$.

Между параметрами (n, d, M) существует взаимосвязь, определяющая оптимальный выбор кодов, решаемый, как правило, тремя способами.

1. При заданных M и n максимизируется d. Подобным образом построенные коды получили название *максиминных*.

2. При заданных M и d минимизируется длина сообщения n, т.е. отыскиваются коды с минимальной избыточностью.

3. При заданных n и d находят коды с максимальным числом кодовых слов M. Такие коды называют *максимальными.*

Данные три процедуры по нахождению кодов не приводят к одному и тому же решению. К настоящему времени ни одна из задач полностью не решена, однако существует большое число оценок (границ) качества кодов. На практике наиболее широко используют верхние границы Хэмминга и Плоткина, нижнюю границу Варшамова – Гильберта. На практике считается, что код является достаточно хорошим, если его параметры лежат выше границ Варшамова – Гильберта и ниже верхней границы Хэмминга (в заштрихованной области, рис. 5.12).

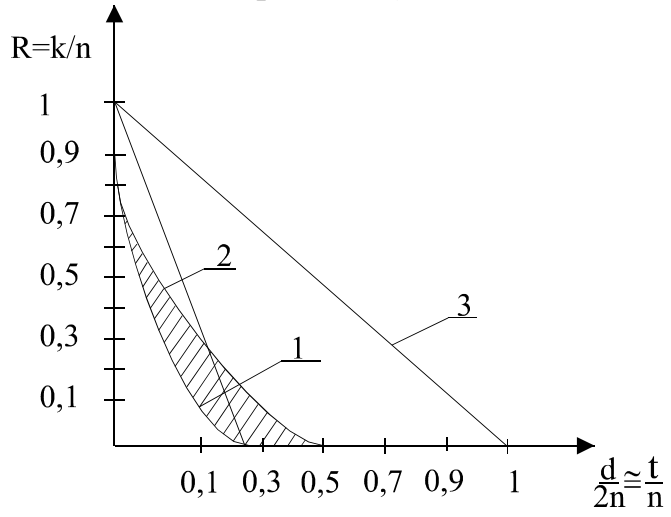


Рис. 5.12. Границы кодов, контролирующих ошибки:
1 – нижняя граница Варшамова – Гильберта; 2 – верхняя граница Хэмминга;
3 – верхняя граница для кодов, исправляющих дефекты

5.2.4. Вероятность правильного и ошибочного декодирования кодового слова

Для ДСК с вероятностью ошибки на символ p и вероятностью правильного приема символа $(1\text{-}p)$ вероятность возникновения ошибки P кратностью t в слове длины n в общем случае равна

$$P_t = C_n^t p^t \left(1-p\right)^{n-t},$$

где $C_n^t = \dfrac{n\left(n-1\right)\ldots\left(n-t+1\right)}{t!}$ указывает на все возможные конфигурации t-кратных ошибок на длине кода n.

Например, для $t = 0; 1; 2$ вероятности P равны

$$P_0=(1\text{-}p)^n,\ P_1=np(1\text{-}p)^{n\text{-}1},\ P_2 = \frac{n\left(n-1\right)}{2} p^2 \left(1-p\right)^{n-2}.$$

При использовании кода с кодовым расстоянием $d=2t+1$, корректирующего t-кратные ошибки, вероятность правильного декодирования принятого слова $P_{пр}$ равна

$$P_{пр} = \sum_{i=0}^{t} C_n^i p^i \left(1-p\right)^{n-i}.$$

Например, при $t = 0; 1;2$

$$P_{пр.0}=(1\text{-}p)^n=P_0,$$

$$P_{пр.1}=(1\text{-}p)^n+np(1\text{-}p)^{n\text{-}1}=P_0+P_1,$$

$$P_{пр.2} = \left(1-p\right)^n + np\left(1-p\right)^{n-1} + \frac{n\left(n-1\right)}{2} p^2 \left(1-p\right)^{n-2} = P_0 + P_1 + P_2.$$

Т.е. $P_{пр}$ определяется как сумма вероятностей всевозможных корректируемых конфигураций t-кратных ошибок на длине кода n. Вероятность ошибочного декодирования слова равна

$$P_{ош} = 1 - P_{пр} = 1 - \sum_{i=0}^{t} C_n^i p^i \left(1-p\right)^{n-i}.$$

5.3. Вопросы и задания для самопроверки

5.1. Что является причиной возникновения ошибок при передаче информации по каналам связи?

5.2. Какие ошибки называются случайными?

5.3. Что такое пакет ошибок?

5.4. Что такое модуль ошибок?

5.5. Поясните принципиальное отличие модульных от пакетных ошибок.

5.6. На какие классы подразделяют ошибки по вероятности их возникновения?

5.7. Что такое модель канала передачи информации?

5.8. Какой канал передачи информации называется двоично-симметричным?

5.9. Что понимается под термином «стирание»?

5.10. В чем отличия канала с дефектами от канала со случайными ошибками?

5.11. Что такое кодовое расстояние кода?

5.12. Чем отличается расстояние Хэмминга от кодового расстояния?

5.13. Что такое вес кодового слова?

5.14. Что такое спектр весов кодовых слов?

5.15. Можно ли по спектру весов кодовых слов определить кодовое расстояние кода?

5.16. Поясните физическую сущность помехоустойчивого кода, корректирующего ошибки кратностью t.

5.17. По исходным данным p, t и n, задаваемым преподавателем, определите P_t, $P_{пр}$, и $P_{ош}$ рассматриваемого кода.

5.18. Поясните физическую сущность заштрихованной области рис. 5.12 с позиций теории кодирования.

5.19. Какое число ошибок и стираний корректирует код с кодовым расстоянием $d = 7$?

Модуль 6

ЛИНЕЙНЫЕ КОДЫ

Цель модуля – изучение распространенных линейных кодов, способов их задания, методов их кодирования и декодирования, способов построения кодирующих и декодирующих устройств.

В результате изучения модуля студенты должны:

- уметь задавать коды Хемминга, Рида – Маллера, циклические, БЧХ-коды с помощью порождающей и проверочной матриц;
- уметь кодировать информацию линейными кодами;
- иметь представление о кодах, исправляющих зависимые ошибки;
- знать методы декодирования кодов: по максимуму правдоподобия, по синдрому, мажоритарный метод;
- уметь строить схемы умножения и деления многочленов по модулю два;
- уметь строить кодеры и декодеры для линейных кодов.

Содержание модуля

6.6.3. Синдромное декодирование циклических кодов

6.7. Кодирование для исправления зависимых ошибок

6.8. Вопросы и задания для самопроверки

6.9. Практическое занятие № 5

6.10. Практическое занятие № 6

6.11. Лабораторная работа № 2 «Кодирование и декодирование информации кодами Хэмминга» (Методические указания к выполнению лабораторных работ по курсу «Прикладная теория кодирования» для студентов специальности 1-39 01 01 «Радиотехника»// Сост. Богуш Р.П. – УО «ПГУ»: Новополоцк, 2005 – 48 с.)

6.12. Лабораторная работа № 3 «Кодирование и декодирование информации кодами Рида – Маллера» (Методические указания к выполнению лабораторных работ по курсу «Прикладная теория кодирования» для студентов специальности 1-39 01 01 «Радиотехника»// Сост. Богуш Р.П. – УО «ПГУ»: Новополоцк, 2005 – 48 с.)

6.1. Способы задания линейных кодов и кодирование информации

6.1.1. Общие сведения о линейных кодах

Среди известных кодов наиболее полно изучены к настоящему времени линейные (групповые) коды.

Код называется *групповым*, если кодовые комбинации образуют некоторую подгруппу группы всех последовательностей длиной n. Пусть, например, $n = 7$, т.е. имеется группа семиразрядных двоичных чисел. Среди этих чисел можно выделить следующие шестнадцать, которые удовлетворяют всем аксиомам группы, т.е. образуют подгруппу

k	$n–k$	k	$n–k$	k	$n–k$
0001	011	0111	010	1101	001
0010	110	1000	101	1110	100
0011	101	1001	110	1111	111
0100	111	1010	011	0000	000.
0101	100	1011	000		
0110	001	1100	010		

Рассмотренный код позволяет передать шестнадцать различных сообщений. Из рассмотрения кодовых слов можно видеть, что минимальное расстояние Хэмминга d_{min}=3, т.е. код позволяет исправлять любую одиночную ошибку. Если передаваемые сообщения представляют собой k-разрядные

двоичные числа, то групповой код всегда может быть построен таким образом, что эти сообщения будут являться первыми k символами кода, как это сделано для рассмотренного выше примера. Код, построенный таким образом, обозначается (n; k); иногда код обозначается как (n; k; d). Первое число в скобках показывает общее число символов в коде (длину кода), второе – число информационных символов, третье – кодовое расстояние кода. Для вышеприведенного примера обозначение кода будет таким: (7; 4; 3). Оставшиеся $r=n–k$ символов называются *проверочными*. В групповых кодах проверочные символы образуются путем суммирования по модулю два символов, стоящих на определенных позициях кодового слова. Поскольку операция суммирования по модулю два является линейной, то коды, образованные таким образом, называются *линейными.* Для рассмотренного примера правила образования проверочных символов будут следующие

$$a_5=a_1+a_2+a_3,$$
$$a_6=a_2+a_3+a_4,$$
$$a_7=a_1+a_2+a_4.$$

Линейные коды бывают систематическими (разделенными) и несистематическими (неразделёнными). Если известно местоположение (безразлично, на каких позициях они расположены) проверочных и информационных разрядов, то такой код называют *систематическими* (*разделенным*). В *несистематических* (*неразделенных*) кодах все символы являются кодовыми символами. Как правило, коды, используемые в цифровой обработке сигналов, являются неразделёнными. Нелинейные коды отличаются от линейных кодов тем, что для них не выполняется одна или ряд аксиом группы.

6.1.2. Задание линейных кодов и кодирование информации

Линейные коды задаются с помощью порождающей G и проверочной H матриц. Эти матрицы связаны основным уравнением кодирования

$$G \cdot H^T = 0.$$

Матрица G содержит k строк и n столбцов, ее элементами являются нули и единицы. В качестве строк матрицы G выбираются любые ненулевые линейно независимые n-значные векторы, отстоящие друг от друга не менее чем на заданное кодовое расстояние d.

Векторы v_1, v_2, …, v_k называются *линейно независимыми*, если выполняется условие

$$c_1 v_1 + c_2 v_2 + \ldots + c_k v_k = 0,$$

где c_i={0, 1}, а сложение выполняется по модулю два. Т.е. каким бы образом мы не суммировали различные строки матрицы G, не получим суммы, равной нулю. Все множество кодовых слов состоит из строк порождающей матрицы и всевозможных комбинаций этих строк, т.е. суммы по модулю два всех строк матрицы G сначала попарно, затем по три и так далее до суммы всех k строк.

С точки зрения алгебры все кодовые слова образуют некоторое векторное пространство, базисом которого являются строки матрицы G.

Поскольку линейно независимые векторы могут быть выбраны произвольным образом, то, очевидно, можно построить множество матриц G с одним и тем же кодовым расстоянием d. Коды, обладающие одними и теми же параметрами $\{n, k, d\}$, но задаваемые различными порождающими матрицами, получили название *эквивалентных кодов*. В этих кодах различная зависимость проверочных символов от информационных.

Известны эквидистантные коды – коды, в которых кодовые слова отстоят друг от друга на одно и то же кодовое расстояние.

Свойство линейной независимости векторов инвариантно относительно двух следующих операций (выполняя эти две операции, кодовое расстояние не меняется):

1. возможна произвольная перестановка строк и столбцов в матрице G;
2. замена i-й строки на сумму i-й и j-й строк и т.д. (эту операцию нельзя осуществлять со столбцами матрицы G).

Производя вышеуказанные операции, любую произвольную матрицу G можно привести к так называемому приведенно-ступенчатому виду

$$G = \underbrace{\left[I_k ; G^* \right]}_{n} = \left[I_k ; H^{*T} \right] \Big\updownarrow k ,$$

где I_k – единичная подматрица размерностью k.

Исходя из основного уравнения кодирования, проверочная матрица имеет вид

$$H = \underbrace{\left[H^* ; I_r \right]}_{n} = \left[G^{*T} ; I_r \right] \Big\updownarrow r \quad .$$

Операция кодирования информации линейным кодом заключается в умножении информационного вектора A_k размерностью k на порождающую матрицу G; в результате получаем вектор

$$Y = A_k \cdot G.$$

При умножении A_k на G^* в приведенно-ступенчатом виде на первых k позициях будет расположен без изменения информационный вектор A_k, т.к. при умножении какого-либо вектора на единичную матрицу этот вектор не изменяется. i-я проверочная позиция находится из информационных по i-му столбцу подматрицы G^* или в вычислении i-го проверочного символа участвуют те информационные символы, которым соответствуют единицы в i-й строке матрицы H.

Например, для H_i строки матрицы H кода (7; 4)

$$H = [1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0],$$

$$a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7$$

$$\text{k} \qquad\qquad \text{r}$$

$a_6=a_1+a_3$, т.е. для вычисления шестого символа a_6 (проверочного) необходимо просуммировать по модулю два информационные символы, расположенные на первой и третьей позициях (для $a_1 = 0$, $a_3 = 1$, $a_6 = 1$).

Пример.

Рассмотрим код Рида – Маллера (8;4), задаваемый следующей порождающей матрицей

$$G_{(8;\,4)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Пусть A_k=(1001), тогда

$$Y=A_k \cdot G=(1\ 1\ 1\ 1\ 0\ 0\ 0\ 0),$$

т.е. кодовое слово несистематического кода находится путем суммирования соответствующих строк матрицы G, номера которых соответствуют позициям «1» в векторе A_k (для данного примера необходимо просуммировать по модулю два первую и четвертую строки матрицы G).

Образуем систематический код, для этого просуммируем все строки и запишем сумму вместо первой строки

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

$$* \quad * \quad * \qquad\qquad * \qquad\qquad$$

В матрице G знаком * обозначены столбцы единичного веса; переставим их:

$$G = \begin{matrix} a_1 & a_2 & a_3 & a_4 & & a_5 & a_6 & a_7 & a_8 \end{matrix}$$
$$\begin{bmatrix} 1 & 0 & 0 & 0 & & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & & 0 & 1 & 1 & 1 \end{bmatrix} = \left[I_4 \vdots G^* \right],$$

$$H = \left[G^{*T} \vdots I_4 \right] = \begin{matrix} a_1 & a_2 & a_3 & a_4 & & a_5 & a_6 & a_7 & a_8 \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 1 & 0 & & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & & 0 & 0 & 0 & 1 \end{bmatrix} = \left[I_4 \vdots G^* \right].$$

В этом случае для A_k=(1001), Y=(1001 ⋮ 1001).

Из порождающей и проверочной матриц видно, что проверочные символы (a_5–a_8) равны соответственно

$$a_5 = a_1 + a_2 + a_3, \qquad a_7 = a_1 + a_3 + a_4,$$
$$a_6 = a_1 + a_2 + a_4, \qquad a_8 = a_2 + a_3 + a_4.$$

Кодирующее устройство (кодер) для данного кода представлено на рис. 6.1.

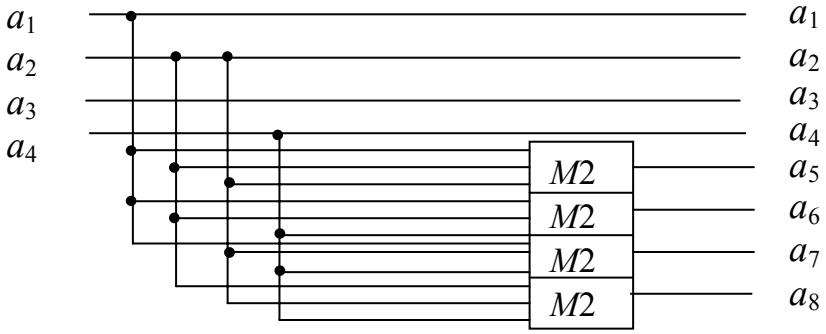


Рис. 6.1. Кодер кода (8;4)

6.2. Основные методы декодирования линейных кодов

6.2.1. Декодирование по максимуму правдоподобия

Пусть есть код A, содержащий $M=2^k$ слов, причем каждое слово имеет разрядность n,

$$A=(a_1, a_2, \ldots, a_M),\ a_i=(0, 1, \ldots, n-1).$$

На длине кода возможны следующие ошибки:

- одиночные, $t = 1$;
- двукратные, $t = 2$;

……

- n-кратные, $t = n$.

Суммарное число всевозможных ошибок равно $E=2^n-1$.

Задача декодирования состоит в том, чтобы по принятому вектору b_j однозначно определить переданное сообщение a_i, т.е. в качестве a_i следует принять такой вектор, для которого условная вероятность $p(a_i/b_j)$ максимальна. Декодирование, когда максимизируется $p(a_i/b_j)$, получило название декодирования по максимуму апостериорной вероятности. В теории информации доказывается, что $p(a_i/b_j)=C\,p(b_j/a_i)$, где C – константа. Следовательно, для максимизации $p(a_i/b_j)$ необходимо максимизировать условную вероятность $p(b_j/a_i)$. Для каждого отдельного канала эта вероятность имеет свое конкретное значение. Для наиболее широко применяемого двоично-симметричного канала зависимость этой вероятности от числа ошибок показана на рис. 6.2.

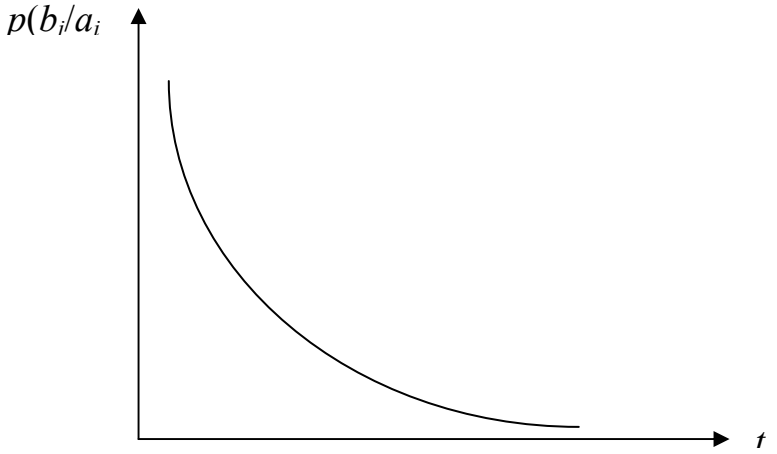


Рис. 6.2. Зависимость вероятности $p(b_j/a_i)$ от кратности ошибок

Из рис. 6.2 следует, что для максимизации $p(b_j/a_i)$ необходимо выбирать кратность ошибки как можно меньше. Отсюда следует правило декодирования: в качестве переданного слова следует выбирать такое слово, которое отличается от принятого слова в наименьшем числе позиций. Это можно определить так же, как декодирование по минимуму расстояния: принятое слово отождествляется с тем кодовым словом, от которого оно удалено на минимальное расстояние. В теории информации показывается,

что такое декодирование является наилучшим декодированием, т.к. обеспечивает минимальную вероятность ошибки.

На рис. 6.3 представлена структурная схема декодирующего устройства (декодера), реализующего декодирование по минимуму расстояния, где УСр – устройство сравнения, РУ – решающее устройство.

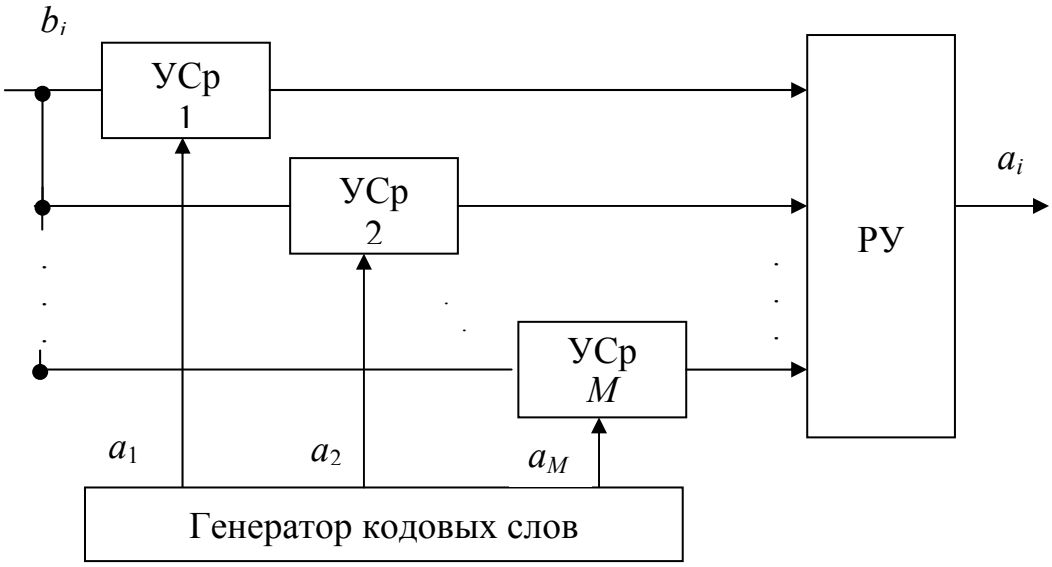


Рис. 6.3. Структурная схема декодера по минимуму расстояния

Часто декодирование по минимуму расстояния определяется как декодирование в целом, т.к. при декодировании принимается решение обо всем слове, а не об отдельном символе, когда принятое слово коррелируется со всеми кодовыми словами, и в качестве переданного принимается то, корреляция с которым будет максимальна. Такое декодирование называется декодированием по максимуму правдоподобия и реализуется многоканальным корреляционным приёмником, опорными словами которого являются все кодовые слова. Оценим сложность декодера. Пусть код имеет длину n и содержит 2^k кодовых слов. Для вычисления расстояния нужно умножить каждое из этих слов на принятое, вычислить скалярное произведение. Одно умножение требует выполнения n арифметических операций сложения двух чисел. Общее количество арифметических операций пропорционально величине $n \cdot 2^k$. Это число растет очень быстро с ростом k, поэтому декодирование по минимуму расстояния целесообразно использовать лишь в кодах с малым числом информационных символов (в низкоскоростных кодах, когда $r >> k$).

С формальной точки зрения операция корреляции сводится к умножению принятого вектора на кодовую матрицу. Для кодов с основанием $q = 2$

для подобного умножения требуется $n(n-1)$ операций сложения (вычитания). При больших n это приводит к большим вычислительным затратам.

6.2.2. Декодирование по синдрому

Декодирование по синдрому основано на таблице смежных классов. Пусть A – $(n; k)$-код и a_1 – его нулевой элемент, a_2, a_2, …,a_2^k – ненулевые кодовые слова. Кодовые слова являются элементами первой строки таблицы. В качестве элементов второй строки таблицы являются элемент e_1 и все его суммы с кодовыми словами. Элемент e_1 не принадлежит кодовым словам кода и является элементом, который наиболее вероятно получить при передаче нулевого кодового слова. Как правило, в качестве этого элемента выбирается элемент с минимальным весом. Аналогичным образом строится третья строка таблицы, однако в качестве первого элемента третьей строки выбирается элемент e_2, не принадлежащий ни первой, ни второй строке, и т.д.

Полученная таким образом таблица есть таблица смежных классов, первый столбец – образующий смежных классов. При декодировании в таблицу вводится дополнительный столбец S, называемый *синдромом*, причем для каждого смежного класса имеется свой синдром. При декодировании можно воспользоваться однозначным соответствием между смежными классами и синдромами.

Таблица 6.1.

Разложение на смежные классы

$\leftarrow n \rightarrow$					$\leftarrow r \rightarrow$
a_1=(000…0)	a_2	a_3	…	a_2^k	S_0=(000…0)
e_1	a_2+e_1	a_3+e_1	…	$a_2^k+e_1$	S_1
e_2	a_2+e_2	a_3+e_2	…	$a_2^k+e_2$	S_2
…	…	…	…	…	…

Справедливость этого утверждения доказывается следующими равенствами, исходя из основного уравнения кодирования:

$$(a_i + e_z)\, H^T = a_i\, H^T + e_z\, H^T = e_z\, H^T = S_z,\ (a_i\, H^T=0),$$
$$(a_j + \mathrm{e_z})\, H^T = a_j\, H^T + e_z\, H^T = e_z\, H^T = S_z,\ (a_j\, H^T=0).$$

Отсюда следует, что два вектора принадлежат одному и тому же смежному классу, если равны их синдромы, т.е. номер смежного класса однозначно определяется по синдрому. Следовательно, для однозначного декодирования синдромы должны быть различны.

Из этого вытекает следующее правило декодирования:

1. Вычисляется синдром S_j по принятому слову

$$S_j = Y^* H^T.$$

2. По синдрому S_j находится образующий смежного класса e_j, который и есть вектор ошибок E.

3. Из принятого вектора Y^* вычитается вектор ошибок; в результате получаем переданное слово, Y.

Если в качестве образующих выбраны векторы, имеющие минимальный вес в своем классе, то декодирование по синдрому совпадает с декодированием по минимуму расстояния, т.е. обеспечивает минимальную вероятность ошибки в двоично-симметричном канале.

Оценим сложность декодирования по синдрому. Очевидно, что увеличение кратности корректируемых ошибок приводит к увеличению числа проверочных символов r, т.е. к увеличению избыточности и, следовательно, к росту числа синдромов, равного 2^r. Каждый синдром имеет r разрядов, поэтому объем вычислений пропорционален величине $r \cdot 2^r$. Отсюда следует, что декодирование по синдрому проще декодирования по минимуму расстояния, если $r2^r < n2^k$. Заметим, что полученные оценки не являются точными. Они показывают лишь порядок роста сложности с ростом k и r. В обоих случаях сложность растет экспоненциально. Только в первом случае эта экспонента – от числа информационных символов, а во втором случае – от числа проверочных. Такая скорость роста сложности декодирования является серьезным препятствием для практического использования рассмотренных оптимальных методов.

Декодирование по синдрому применяется для коррекции ошибок малой кратности. Основная трудность при декодировании по синдрому связана с «проблемой селектора», который по синдрому должен однозначно установить разряды, в которых произошли ошибки, те. определить вектор ошибок E.

6.2.3. Мажоритарное декодирование

При посимвольном декодировании кодов, кроме синдромного, используется метод, который получил название мажоритарного декодирования.

Сущность мажоритарного декодирования базируется на системе проверочных равенств, вытекающих из основного уравнения кодирования. Поскольку код избыточен, то система неравенств может быть разрешена относительно каждой из переменных не единственным образом. *Мажорирование* есть голосование по большинству правильных проверочных уравнений.

Существует три способа построения систем проверочных уравнений при мажоритарном декодировании:

- с разделенными проверками;

- с λ-связанными проверками;
- с квазиразделенными проверками.

В системах с разделенными проверками некоторый символ, относительно которого разделяется система уравнений, входит во все уравнения.

Любой другой символ входит не более чем в одну проверку. Отсюда следует, что для коррекции t ошибок система должна состоять из $(2t + 1)$ уравнений и иметь на столько же входов мажоритарные элементы.

Пример. Пусть имеется код (8;4), задаваемый следующей проверочной матрицей

$$H = \left[\begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right].$$

Декодирование сводится к умножению принятого вектора $A^* = (a_1^* \; a_2^* a_3^* \; a_4^* \; a_5^* \; a_6^* \; a_7^* \; a_8^*)$ на матрицу H^T. В результате приравнивания данного произведения нулю получим систему уравнений

$$a_1^* + a_2^* + a_5^* = 0, \quad (6.1)$$
$$a_2^* + a_3^* + a_6^* = 0, \quad (6.2)$$
$$a_3^* + a_4^* + a_7^* = 0, \quad (6.3)$$
$$a_1^* + a_4^* + a_8^* = 0. \quad (6.4)$$

Из этой системы можно записать систему проверочных равенств, где равенство $a_i = a_i^*$ называется уравнением истинности:

для a_1 из (6.1) и (6.4)

$$\begin{cases} a_1 = a_2^* + a_5^*, \\ a_1 = a_4^* + a_8^*, \\ a_1 = a_1^*, \end{cases}$$

для a_4 из (6.3) и (6.4)

$$\begin{cases} a_4 = a_3^* + a_7^*, \\ a_4 = a_1^* + a_8^*, \\ a_4 = a_4^*. \end{cases}$$

Видно, что каждый из принятых символов a_i входит в данные системы один раз, и, следовательно, если он ошибочен, то ошибочным будет одно из трех уравнений системы; два остальных – правильны. По большинству правильных проверок мажоритарный элемент принимает правильное решение об оценке состояния разряда.

Для данного примера мажоритарный элемент реализует функцию

$$M = a_1a_2a_3v\overline{a}_1a_2a_3va_1\overline{a}_2ava_1a_2\overline{a}_3 .$$

Минимизировав эту логическую функцию каким-либо из известных методов, получим

$$M = a_1a_2va_1a_3va_2a_3 ,$$

что легко реализуется на элементах И – ИЛИ.

С ростом числа корректируемых ошибок растет число уравнений в системе и, следовательно, увеличивается сложность мажоритарного элемента. Например, для пятивходового мажоритарного элемента требуется (без минимизации) уже 16 элементов И для реализации функции

$$M = a_1a_2a_3a_5v\overline{a}_1a_2a_3a_4a_5v...va_1a_2a_3a_4\overline{a}_5v\overline{a}_1\overline{a}_2a_3a_4a_5v...va_1a_2a_3\overline{a}_4\overline{a}_5 .$$

Минимальную сложность мажоритарного элемента реализует схема счетчика и арифметического сумматора.

Рассмотрим мажоритарное декодирование кодов с квазиразделенными проверками.

Система проверок называется квазиразделенной, если она разделима относительно некоторой суммы символов.

Пусть, например, имеем код, задаваемый матрицей H,

$$H = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array},$$

Для суммы (a_1+ a_2) можно записать

$$\begin{cases} a_1 + a_2 = a_1 + a_2, \\ a_1 + a_2 = a_3 + a_6, \\ a_1 + a_2 = a_5 + a_7. \end{cases}$$

Положим,

$$a_1 + a_2 = C_0.$$

Увеличим индексы при a_i на единицу (это имеет место при мажоритарном декодировании циклических кодов); в результате получим

$$a_2 + a_3 = C_1.$$

Тогда для a_2 будем иметь следующую систему уравнений

$$\begin{cases} a_2 = a_1 + C_0, \\ a_2 = a_3 + C_1, \\ a_2 = a_2 \ . \end{cases}$$

Видно, что при мажоритарном декодировании с квазиразделенными проверками истинное значение символов осуществляется двухэтапно (может использоваться и большее число этапов): на первом этапе система разрешается относительно суммы элементов, а на втором – относительно конкретного символа a_i.

При мажоритарном декодировании с помощью λ-связанных проверок один символ, относительно которого разрешается система уравнений, входит в каждое уравнение системы. Любой другой символ входит не более чем в λ уравнений. Для того чтобы можно было бы корректировать t ошибок, необходимо $(2\lambda t + 1)$ уравнений в системе проверок и соответственно мажоритарный элемент с таким числом входов.

Пример. Пусть имеется проверочная матрица кода (7;4)

$$H = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}.$$

Из матрицы следует, что

$$\begin{cases} a_1 + a_2 + a_3 + a_5 = 0, & (6.5) \\ a_2 + a_3 + a_4 + a_6 = 0, & (6.6) \\ a_1 + a_3 + a_4 + a_7 = 0. & (6.7) \end{cases}$$

Разрешим данную систему относительно a_1

$$\begin{cases} a_1 = a_1, \\ a_1 = a_2 + a_3 + a_5, \\ a_1 = a_3 + a_4 + a_7 \ . \end{cases}$$

Анализ данной системы проверок показывает, что символ a_3 входит в два уравнения системы. Следовательно, система разделенных проверок не реализуется. Необходимо найти еще как минимум два уравнения. Для этого просуммируем уравнения (6.5) и (6.6), а также (6.6) и (6.7). В результате получим

$$\begin{cases} a_1 = a_4 + a_5 + a_6, & (6.8) \\ a_1 = a_2 + a_6 + a_7. & (6.9) \end{cases}$$

Т.е. система проверок, разрешенная относительно a_1 и состоящая из пяти уравнений, имеет $\lambda=2$, и, следовательно, с ее помощью можно исправить любую одиночную ошибку. Аналогичным образом могут быть реализованы системы проверок и для других информационных символов (a_2, a_3, a_4).

6.3. Коды с проверкой на четность и коды Хэмминга

Широчайшее применение в различных устройствах и системах информатики и радиоэлектроники получил код с простой проверкой четности, кодовое расстояние которого равно двум ($d = 2$), задаваемый следующими порождающей G и проверочной H матрицами,

$$G_{КПЧ} = \left[\begin{array}{c|c} & 1 \\ & 1 \\ I_k & . \\ & . \\ & 1 \end{array}\right], \qquad H_{КПЧ} = \left[1 \quad 1 \quad . \quad . \quad 1 \quad | \quad 1\right].$$

Например, для $k = 4$ матрицы и уравнение кодирования имеют вид

$$G_{(5;\,4)} = \left[\begin{array}{cccc|c} 1 & & & & 1 \\ & 1 & & & 1 \\ & & 1 & & 1 \\ & & & 1 & 1 \end{array}\right],$$

$$H_{(5;4)} = \begin{array}{ccccc} a_1 & a_2 & a_3 & a_4 & a_5 \\ \left[1 \right. & 1 & 1 & 1 & \left. 1\right] \end{array}, \; a_5 = a_1 + a_2 + a_3 + a_4 .$$

В данном коде в единственный проверочный (избыточный) разряд записывается сумма по модулю два состояний информационных разрядов; она равняется нулю, если число единиц при вычислении четно, и единице – в противном случае, т.е. в передаваемом сообщении число единиц всегда будет четно. На приемной стороне при декодировании вновь определяется, четно или нечетно число единиц в принятом сообщении, и если оно нечетно, то выставляется флаг обнаруженной ошибки.

Коды Хэмминга имеют параметры $(n; k) = (2^m–1; 2^m–1–m)$ и обычно задаются проверочной матрицей. Столбцами проверочной матрицы являются все ненулевые двоичные числа длиной m. Например, для кода (7; 4) проверочная матрица имеет следующий вид

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} = \left[\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

При $m = r$ полные коды Хэмминга имеют, например, параметры $(n; k)=(7; 4), (15; 11), (63; 57)$.

Определим корректирующую способность кодов Хэмминга. Предварительно напомним, что весом $w(a)$ слова a называется число ненулевых элементов, и в линейном коде расстояние $d(a, b)$ между двумя словами a и b равно весу некоторого третьего слова $a+b$, т.е. $d(a, b)=w(a+b)$.

Для определения минимального расстояния воспользуемся равенством

$$aH^T = 0.$$

Предположим, что $d(a, b)=1$. Тогда в коде существует слово весом 1, что невозможно, т.к. для слов весом один $aH^T\neq0$. Теперь предположим, что $d(a, b)=2$. Тогда в коде существует слово весом 2, что также невозможно, поскольку никакие два столбца проверочной матрицы не дадут в сумме нулевой вектор. Следовательно, $d(a, b)\geq3$. Простым подбором нетрудно найти слово весом 3, для которого $aH^T=0$ (т.е. в матрице H можно подобрать три столбца, которые в сумме дадут нулевой вектор). Окончательно получим d_{min} =3, т.е. коды Хэмминга исправляют и обнаруживают соответственно любую одиночную и двойную ошибки. Поскольку все $2^m–1$ векторов, задающих одиночные ошибки, принадлежат различным смежным классам, то этими смежными классами и кодовым пространством исчерпываются все 2^m смежных классов. Их образующими являются нулевой вектор и векторы единичного веса. Следовательно, коды Хэмминга являются совершенными.

Если передан вектор u, а принят вектор $e+u$, то синдром равен

$$(u+e)H^T = eH^T,$$

и для одиночных ошибок равен строке h^T, номер которой соответствует номеру искаженной позиции. Если векторы матрицы H^T упорядочены как двоичные числа (а не в приведенно-ступенчатой форме), то синдром говорит о номере искаженной позиции.

Пример. Построим кодирующее и декодирующее устройства кода Хэмминга (7;4). Матрицы G и H имеют вид

$$H = \left[H^* \;\middle|\; I_r \right] = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix},$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \left[I_k \;\middle|\; G^* \right].$$

Устройство кодирования (кодер) реализует нижеследующие уравнения и имеет вид (рис. 6.4)

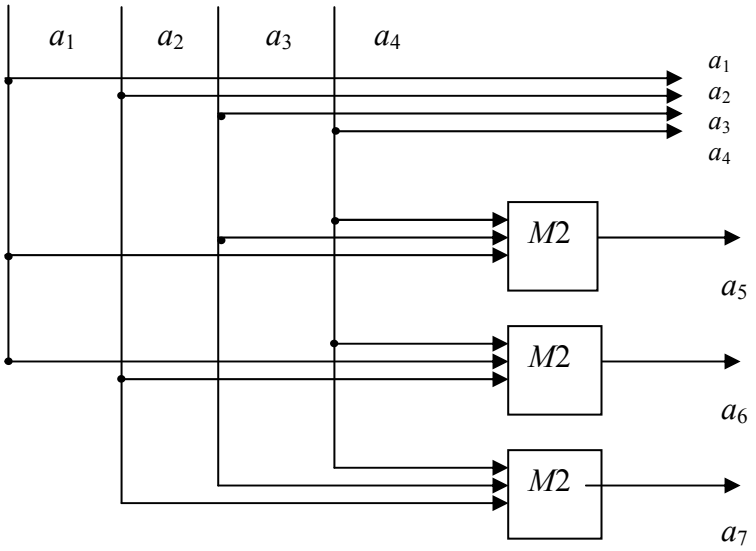


Рис. 6.4. Кодирующее устройство кода Хэмминга (7;4)

$$\begin{cases} a_5 = a_1 + a_3 + a_4, \\ a_6 = a_1 + a_2 + a_4, \\ a_7 = a_2 + a_3 + a_4. \end{cases}$$

Устройство декодирования (декодер) вычисляет синдром

$$A^* \cdot H^T = S = (S_1, S_2, S_3),$$

$$S_1 = a_1^* + a_3^* + a_4^* + a_5^*, \; S_2 = a_1^* + a_2^* + a_4^* + a_6^*, \; S_3 = a_2^* + a_3^* + a_4^* + a_7^*,$$

который реализуется блоком вычисления синдрома (БВС) (рис. 6.5).

Вычисленный синдром поступает на селектор (дешифратор) синдрома, который по виду синдрома находит вектор ошибок E. Если значение синдрома совпадает со значением i-го столбца матрицы H, то это значит, что ошибка произошла в i-м разряде.

Если произошло две ошибки в *i*-м и *j*-м разрядах, то синдром *S* равен сумме *i*-го и *j*-го столбцов матрицы *H*. Поскольку это различные двоичные числа, то результат всегда не равен нулю и, следовательно, любая двойная ошибка будет обнаружена кодом Хэмминга.

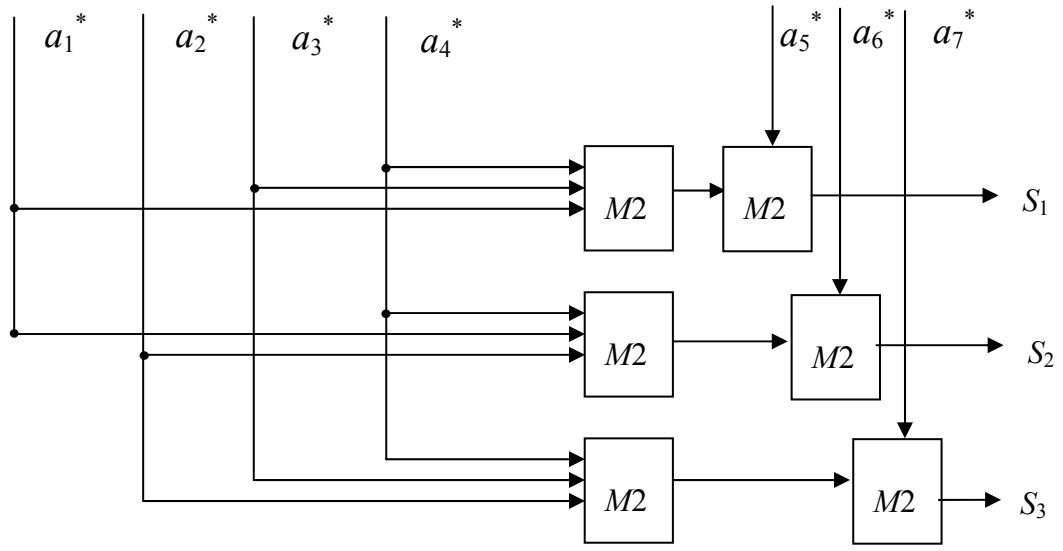


Рис. 6.5. Блок вычисления синдрома кода Хэмминга (7;4)

На рис. 6.6 представлен селектор, с помощью которого ошибки исправляются только в информационных разрядах. Как видно, селектор представляет собой неполный дешифратор.

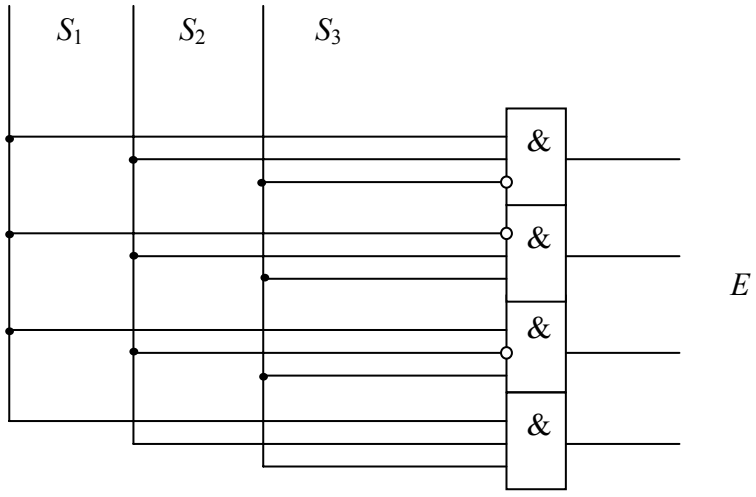


Рис. 6.6. Селектор кода Хэмминга (7; 4)

Коррекция ошибок в принятом сообщении A^* по вектору ошибок осуществляется в корректоре на сумматорах по модулю два. Обобщенная структурная схема синдромного декодера кода Хэмминга имеет вид (рис.6.7)

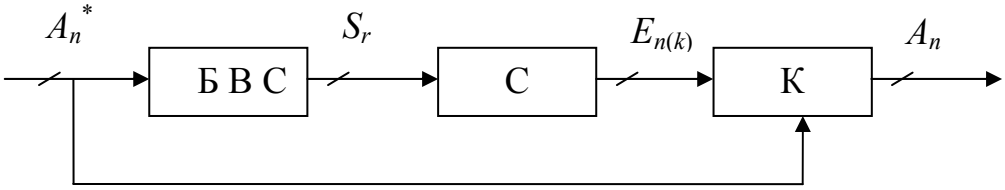


Рис. 6.7. Структурная схема декодера кода Хэмминга:
БВС – блок вычисления синдромов, С – селектор, К – корректор

Коды Хэмминга могут задаваться и с помощью элементов поля Галуа. Действительно, как показано выше, элементы поля Галуа могут быть представлены трояким образом с помощью ненулевых элементов поля, которые могут быть описаны как некоторая степень элемента α, где α – примитивный элемент поля, вычетами или ненулевыми двоичными числами (коэффициентами полиномов).

Элементы поля находятся путем деления многочлена степени меньшей, чем n–1, на полином, являющийся неприводимым. Поскольку в качестве столбцов матрицы H кода Хэмминга выбираются все ненулевые двоичные числа, то матрица H – матрица степеней элемента поля α, т.е.

$$H = \left[\alpha^0, \alpha^1, \alpha^2, \ldots \alpha^{n-1} \right].$$

Пример. Вычислим элементы поля, таблицу элементов поля и построим проверочную матрицу кода Хэмминга (7;4) для неприводимого полинома

$$q(x) = x^3 + x + 1.$$

$x^0 \mid x^3+x+1$; $R(x)=x^0 \mid 0$

$x^1 \mid x^3+x+1$; $R(x)=x^1 \mid 0$

$x^2 \mid x^3+x+1$; $R(x)=x^2 \mid 0$

$x^3 \mid x^3+x+1$; $R(x)=x+1 \mid 1$

x^4 ; x^4+x^2+x ; x^3+x+1 ; x ; $R(x)=x^2+x$

x^5 ; $x^5+x^3+x^2$; x^3+x+1 ; x^2+1 ; x^3+x^2 ; x^3+x+1 ; $R(x)=x^2+x+1$

x^6 ; $x^6+x^4+x^3$; x^3+x+1 ; x^3+x+1 ; x^4+x^3 ; x^4+x^2+x ; x^3+x^2+x ; x^3+x+1 ; $R(x)=x^2+1$

Таблица 6.2

Результаты расчета

α^j	$R(x)$	Коэффициенты $R(x)$
α^0	1	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = H^T$
α^1	x	
α^2	x^2	
α^3	$1 + x$	
α^4	$x + x^2$	
α^5	$1 + x + x^2$	
α^6	$1 + x^2$	

Отсюда

$$H_{(7;4)} = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Модификации кодов Хэмминга

Ниже на примере кода Хэмминга рассматриваются его модификации, осуществляемые с целью получения дополнительных полезных свойств, не присущих основному коду. Аналогичным образом можно модифицировать другие коды.

Удлиненные коды Хэмминга. Коды Хэмминга можно удлинить на один символ, добавив общую проверку на четность. Данные коды имеют длину на единицу больше обычных кодов, т.е. $(n; k)=(2^m; 2^m–m–1)$, здесь $r=m+1$. Добавление проверочного символа увеличивает вес слов на единицу, а следовательно, кодовое расстояние кода $d = 4$. Отметим, что параметры удлиненных кодов Хэмминга совпадают с параметрами кодов Рида – Маллера с $d = 4$.

В общем случае данный код исправляет одну ошибку, а обнаруживает три, но, как правило, в этом режиме код используется редко, а применяется в режиме идентификации двойных ошибок и коррекции одиночных. Идентификация – это когда по виду синдрома можно определить, произошла одиночная или двойная ошибка. В первом случае можно или исправлять, или обнаруживать ошибки, а во втором – исправлять и обнаруживать.

Для удлиненного кода Хэмминга

$$H_Y = \left[\begin{array}{ccccc|c} & & & & & 0 \\ & & H & & & 0 \\ & & & & & 0 \\ \hline - & - & - & - & - & - \\ 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right].$$

Например, матрица H для удлиненного кода Хэмминга (8; 4) имеет вид

$$H_{(8;4)} = \left[\begin{array}{cccc:ccc:c} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right],$$

т.е. последний разряд (a_8) – разряд общего контроля четности по всем разрядам.

Структурная схема кодера удлиненного кода Хэмминга представлена на рис. 6.8.

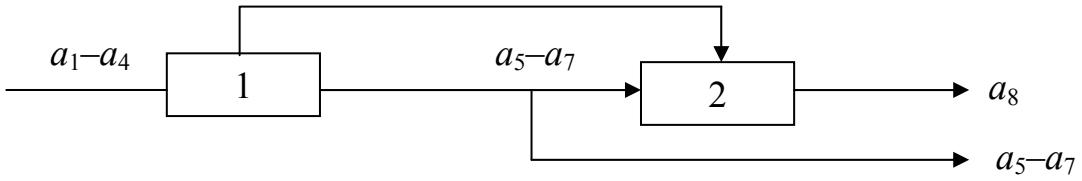


Рис. 6.8. Структурная схема кодера удлиненного кода Хэмминга

Структурная схема декодера (БАЧ – блок анализа четности) представлена на рис. 6.9.

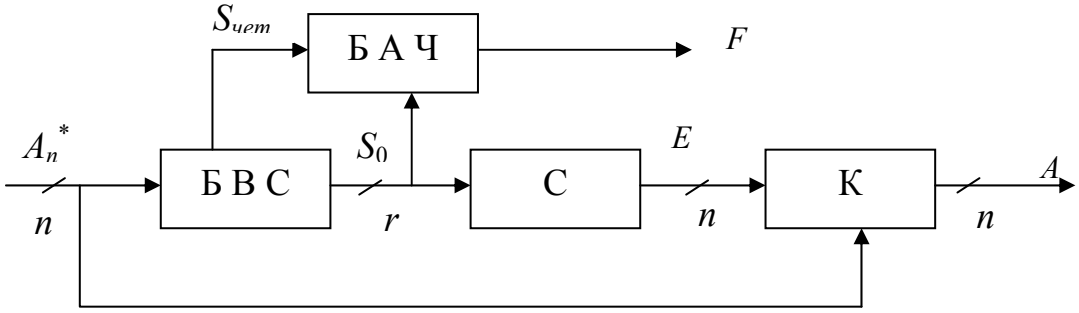


Рис. 6.9. Структурная схема декодера удлиненного кода Хэмминга

Декодирование удлиненных кодов Хэмминга осуществляется согласно следующему правилу:

1. Прежде вычисляется синдром S, который состоит из двух частей

$$S = (S_0, S_{чет}).$$

2. Если $S_0 \neq 0$, а $S_{чет}=1$, то произошла одна ошибка (при этом разрешается исправление ошибки).

3. Если $S_0 \neq 0$, а $S_{чет}=0$, то произошло две ошибки (при этом выставляется флаг F двукратной ошибки и коррекция ошибок не осуществляется).

4. Если $S_0=S_{чет}=0$, то предполагается, что ошибок не произошло.

Укороченные коды Хэмминга

Чтобы получить матрицу H кода Хэмминга с нужным числом информационных разрядов, необходимо в проверочной матрице полного кода исключить любые T-столбцы, относящиеся к информационным разрядам (не проверочным), иначе говоря, исключить нужное число столбцов весом $w \geq 2$. Например, если необходимо закодировать $k_1=16$ разрядов, то нужно укоротить полный код (31; 26) на $T=k-k_1=10$ разрядов. Например, матрица H укороченного кода (6; 3) может иметь вид

$$H_{(6;3)} = \left[\begin{array}{ccc:ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

При правильном укорочении можно получить модифицированные коды, которые могут обеспечить:

1. упрощение кодирующих и декодирующих схем;
2. регулярность и однородность схем коррекции;
3. контроль зависимых ошибок (модульных и пакетных);
4. уменьшение вероятности неправильного (ложного) декодирования;
5. коррекцию двойных ошибок из-за отказов элементов и другие свойства.

Упрощение кодирующих и декодирующих схем. Поскольку каждая единица в проверочной матрице есть сумматор по модулю два в блоках кодирования и вычисления синдрома, то вычеркивание столбцов с максимальным весом приводит к уменьшению сложности вышеназванных блоков при укорочении кодов. Кроме того, если производить операции, а именно суммирования, со строками матрицы H, то также можно уменьшить число единиц. Например, при суммировании всех строк в матрице кода (8; 4) и записи суммы в последнюю строку матрицы H можно получить матрицу вида

$$H_{(8;4)} = \left[\begin{array}{ccccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] = \left[\begin{array}{ccccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

В этом случае у кодов Хэмминга с $d = 4$ в матрице H вес столбцов нечетен. Следовательно, правило декодирования такого кода может быть записано в следующем виде:

1. если синдром нечетен и $S \neq 0$, то произошла ошибка кратностью один;
2. если синдром четен и $S \neq 0$, то произошла двукратная ошибка.

На рис. 6.10 представлена структурная схема декодера для данного правила декодирования (БАС – блок анализа синдрома).

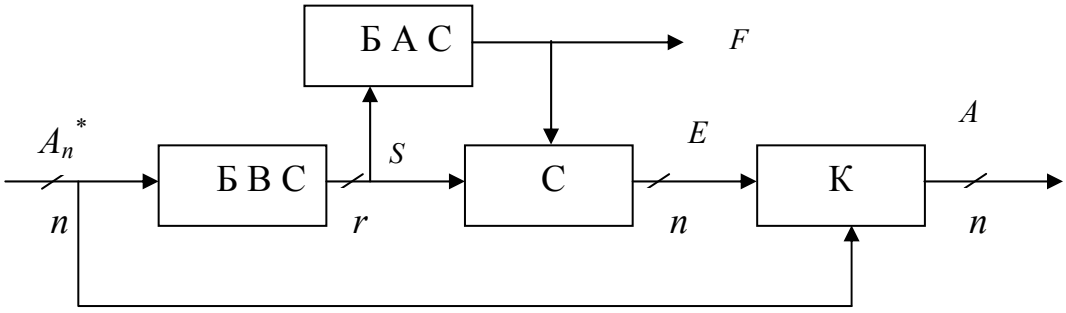


Рис. 6.10. Структурная схема декодера кода Хэмминга с $d = 4$

Блок анализа синдрома – схема контроля четности разрядов синдрома S, сигнал на выходе которой равен «1» при $t = 1$ и «0» при $t = 2$.

Обеспечение регулярности и однородности схем коррекции. Очевидно, что если матрица H будет состоять из однородных блоков и одинакового числа единиц в строках матрицы, то это приведет к однородной структуре при реализации кодирующих и декодирующих устройств, повышению быстродействия за счет уменьшения длины соединений между элементами, реализации алгоритмов коррекции ошибок при наращивании матриц H однотипными подматрицами.

Уменьшение вероятности неправильного декодирования. Выбрасывая определенные слова при укорочении кодов, можно увеличить кодовое расстояние и, следовательно, кратность обнаруживаемых и исправляемых ошибок. Для кодов Хэмминга с $d = 4$ установлено, что для увеличения вероятности правильного декодирования необходимо уменьшать число слов веса $w = 4$. Обозначим эти слова через B_4. Тогда известно, что вероятности ложного исправления трехкратной ошибки и не обнаружения четырехкратной ошибки соответственно равны $p_3 = 4B_4/C_n^3$, $p_4 = B_4/C_n^4$.

Например, для кода (22; 16) с $d = 4$ минимальное $B_4 = 250$; это позволяет получить $p_3 = 0{,}65$, $p_4 = 0{,}03$, т.е. практически все четырехкратные

ошибки могут быть обнаружены. Для кода Хэмминга с $d = 3$ при укорочении необходимо уменьшать число слов веса $w = 3$.

Обнаружение ошибок специального вида. Рассмотрим обнаружение ошибок специального вида (0000…0) и (1111…1), получивших название однонаправленных. Для обнаружения подобных ошибок необходимо исключить нулевое и единичные слова из числа кодовых слов, т.е. в канал эти слова не передавать. В этом случае, если на приемной стороне появилось нулевое или единичное слово, происходит обнаружение однонаправленных ошибок.

На рис. 6.11 представлена структурная схема канала с обнаружением данных ошибок.

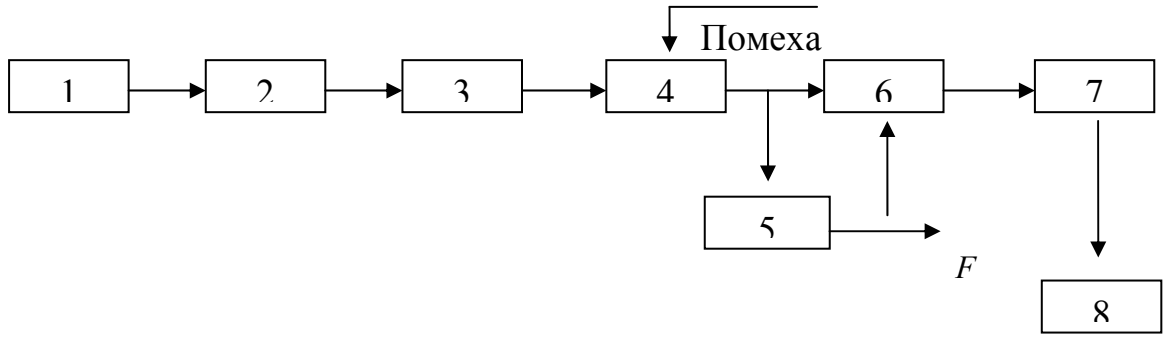


Рис. 6.11. Структурная схема канала с кодированием
для обнаружения однонаправленных ошибок:
1 – источник; 2 – кодер; 3 – блок инвертирования проверочных символов;
4 – канал передачи; 5 – блок анализа (элементы И, ИЛИ);
6 – блок инвертирования; 7 – декодер; 8 – приёмник

Поскольку в кодах Хэмминга нулевое и единичные слова являются кодовыми, то их можно исключить путем инвертирования 2 – 3 проверочных символов.

Нахождение кодов, кодовыми словами которых являются как прямые, так и инверсные слова. Коды, кодовыми словами которых являются как прямые, так и инверсные слова, получили название *транспарантных кодов.* Для того чтобы в коде содержались как прямые, так и инверсные слова необходимо, чтобы в коде было слово, состоящее из одних единиц (для выполнения аксиомы замкнутости). Отсюда из основного уравнения кодирования $GH^T = 0$ следует, что число единиц в строках проверочной матрицы должно быть четно.

Полные коды Хэмминга с $d = 3$ и $d = 4$ содержат в каждой строке матрицы H четное число единиц, следовательно, эти коды Хэмминга отно-

сятся к транспарантным кодам. Укороченные транспарантные коды с $d = 3$ существуют для параметра укорочения $T \geq 3$; с $d = 4$ – для $T \geq 4$ и длине кода n – четной.

Например, если нужно получить транспарантный укороченный код с $k = 21$ из полного кода Хэмминга (32; 26) с $d = 4$, то T=5. В этом случае длина n=32–5=27 и, следовательно, транспарантный код на такой длине не существует. Для получения подобного кода имеется два решения:

- увеличить k на единицу и передавать, например, постоянно в данном разряде «0» – символ;
- увеличить r на единицу, оставив k постоянным.

В обоих случаях можно найти укороченный транспарантный код Хэмминга с $d = 4$.

Коррекция двойных ошибок из-за отказов элементов методом двойного инвертирования. Рассмотрим два способа коррекции при использовании кода с $d = 4$.

1. Пусть имеется восьмиразрядное слово памяти, в котором, например, первые два разряда постоянно находятся в единичном состоянии «1» из-за отказов элементов памяти – (11– – – – – –), где чертой обозначены исправные разряды. При записи в память нулевого слова A=(0000 0000) при считывании получим слово A^*=(1100 0000). Поскольку с помощью кода с $d = 4$ можно по синдрому S идентифицировать кратность произошедших ошибок, то установится флаг некорректируемых двукратных ошибок, которые не может исправить данный код. Далее инвертируем A^* и записываем в память $\overline{A}^* = (0011\ 1111)$, считываем его из памяти: $\overline{A}^{**} = (1111\ 1111)$. Повторно слово $\overline{A}^{**}$ инвертируем и получаем правильное слово A. Очевиден недостаток способа – низкое быстродействие схем коррекции.

2. Повысить быстродействие можно, если исключить повторные обращения к памяти. Это можно осуществить благодаря введению информации в кодовое слово о том, каким хранится слово, прямым или инверсным. Пусть имеем код с порождающей матрицей

$$G_{(8;\ 4)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} = \left[\begin{array}{ccc|c|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ - & - & - & - & - & - & - & - \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right].$$

k_0 k_1 r_0

Один из информационных разрядов кода k_1 отведем под метку, которая указывает, каким хранится слово – прямым или инверсным. В исходном состоянии в этот разряд записывается «0» – символ, при инвертировании – «1». Благодаря этому не требуется инвертирования при повторном обращении к памяти.

6.4. Коды Рида – Маллера и БЧХ-коды

Коды Рида – Маллера

Коды Рида – Маллера (РМ-коды) нашли широкое применение в различных радиоэлектронных системах. Как правило, информация РМ-кодами кодируется таким образом, что в результате получается неразделимый код. При этом используется однородная и регулярная структура порождающей матрицы G, позволяющая упростить декодирование кодов.

Параметры РМ-кодов:

длина кода $n=2^m$, $d=2^{m-l}$, $k=1+\sum_{i=1}^{l} C_m^i$, $m\geq 3$, $l<m$, l – порядок кода.

Например, при $m = 3$, $l = 1$ (используется РМ-код первого порядка – РМ-1) с $n=2^3=8$, $k=4$, т.е. кодовое расстояние РМ-1 $d=2^2=4$ равно половине длины кода ($d=n/2$).

Коды Рида – Маллера первого порядка задаются порождающей матрицей G, первая строка которой состоит из единиц. В качестве столбцов остальных m строк используются все двоичные числа длиной m

$$G_{PM\text{-}1}=\begin{bmatrix} 1 & 1 & 1 & . & . & . & . & . & . & 1 \\ 0 & 1 & 0 & 1 & . & . & . & . & . & 1 \\ 0 & 0 & 1 & 1 & . & . & . & . & . & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & . & . & . & 1 \\ . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 & . & . & . & 1 \end{bmatrix} \begin{matrix} \\ q_0 \\ q_1 \\ q_2 \\ q_3 \\ \\ q_m \end{matrix}$$

Порождающая матрица РМ-кода второго порядка (РМ-2) состоит из порождающей матрицы $G_{PM\text{-}1}$ и попарных произведений векторов (q_1, q_2, …, q_m). Порождающая матрица кода РМ-3 состоит из матрицы $G_{PM\text{-}2}$ плюс все произведения по три вектора (q_1, q_2, …, q_m) и т. д.

Пример. Построим порождающие матрицы РМ-кодов на длине $n = 8$.

$$G_{PM\text{-}1}=\begin{bmatrix}1&1&1&1&1&1&1&1\\0&1&0&1&0&1&0&1\\0&0&1&1&0&0&1&1\\0&0&0&0&1&1&1&1\end{bmatrix}\begin{matrix}q_0\\q_1\\q_2\\q_3\end{matrix},\ d=4,$$

$$G_{PM\text{-}2}=\begin{bmatrix}&&&G_{PM\text{-}1}&&&&\\0&0&0&1&0&0&0&1\\0&0&0&0&0&1&0&1\\0&0&0&0&0&0&1&1\end{bmatrix}\begin{matrix}\\q_1q_2\\q_1q_3\\q_2q_3\end{matrix},\ d=2,$$

$$G_{PM\text{-}3}=\begin{bmatrix}&&&G_{PM\text{-}2}&&&&\\0&0&0&0&0&0&0&1\end{bmatrix}\begin{matrix}\\q_1q_2q_3\end{matrix},\ d=1.$$

Кодирование РМ-кодов. Кодирование РМ-кодов осуществляется стандартным образом – путем умножения исходного вектора на порождающую матрицу

$$AG=B.$$

Эта операция может реализоваться на сумматорах по модулю два. Поскольку кодовое слово является суммой базисных векторов ($q_0, q_1, \ldots, q_m$) с коэффициентами сообщения, то, учитывая, что базисные векторы могут быть получены как сигналы двоичного счетчика и их произведения, известна и реализация кодирующего устройства на счетчиках. Очевидно, что при подобном кодировании образуется неразделимый код.

Пример**.** Используем РМ-1 для кодирования четырехразрядного слова:

$$AG=\begin{matrix}a_0&a_1&a_2&a_3\end{matrix}\begin{bmatrix}1&1&1&1&1&1&1&1\\0&1&0&1&0&1&0&1\\0&0&1&1&0&0&1&1\\0&0&0&0&1&1&1&1\end{bmatrix}=\begin{matrix}b_0&b_1&b_2&b_3&b_4&b_5&b_6&b_7\end{matrix}$$
$$\begin{bmatrix}1&1&0&0\end{bmatrix}\qquad\qquad\begin{bmatrix}1&0&1&0&1&0&1&0\end{bmatrix}=B.$$

Здесь

$b_0=a_0$,
$b_1=a_0+a_1$,
$b_2=a_0+a_2$,
$b_3=a_0+a_1+a_2$,
$b_4=a_0+a_3$,
$b_5=a_0+a_1+a_3$,
$b_6=a_0+a_2+a_3$,
$b_7=a_0+a_1+a_2+a_3$.

Путем перестановки столбцов и суммирования строк можно привести порождающую матрицу G к приведенно-ступенчатому виду и, следовательно, получить разделимый код; однако при этом нарушается однородная структура кода.

Декодирование РМ-кодов. Как правило, РМ-коды декодируются мажоритарным способом (в последние годы для декодирования РМ-кодов используются процессоры быстрых преобразований).

В данном случае проверки находятся по порождающей матрице. Из примера следует, что если просуммировать рядом стоящие уравнения кодирования через одно, три уравнения, то получим следующую систему мажоритарных проверок:

$a_1=b_0+b_1$, $a_2=b_0+b_2$, $a_3=b_0+b_4$,
$a_1=b_2+b_3$, $a_2=b_1+b_3$, $a_3=b_1+b_5$,
$a_1=b_4+b_5$, $a_2=b_4+b_6$, $a_3=b_2+b_6$,
$a_1=b_6+b_7$, $a_2=b_5+b_7$, $a_3=b_3+b_7$.

Для вычисления a_0 необходимо использовать значения a_i, вычисленные на предыдущем этапе, а также уравнение $a_0=b_0$. Следует заметить, что поскольку код неразделимый, то в системах проверок не используются уравнения истинности $a_i=b_i$.

БЧХ-коды. Коды Боуза – Чоудхури – Хоквингема (БЧХ-коды) позволяют корректировать t-кратные ошибки, и, как правило, что будет показано ниже, задаются с помощью корней порождающего полинома. Эти коды можно задавать и традиционно, с помощью проверочной матрицы, элементами которой являются степени примитивного элемента α поля Галуа,

$$H=\begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & L & \alpha^{(n-1)} \\ \alpha^0 & \alpha^3 & \alpha^6 & L & \alpha^{3(n-1)} \\ & M & & & M \\ \alpha^0 & \alpha^{2t-1} & \alpha^{2(2t-1)} & L & \alpha^{(2t-1)(n-1)} \end{bmatrix}.$$

Для коррекции одной ошибки требуется матрица H, состоящая из одной строки, двух ошибок – двух строк и т.д.

Пример. Пусть $n = 7$, $t = 2$. Проверочная матрица БЧХ-кода, соответствующая этим параметрам, имеет вид

$$H=\begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ - & - & - & - & - & - & - \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} \end{bmatrix}=\begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ - & - & - & - & - & - & - \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \end{bmatrix}.$$

Степени элемента α образуют циклическую группу порядка 7, поэтому показатели степеней элементов α^9, α^{12}, α^{15}, α^{18} приводятся по модулю 7, т.е. $\alpha^{9(\mathrm{mod}\ 7)}=\alpha^2$, $\alpha^{12(\mathrm{mod}\ 7)}=\alpha^5$, $\alpha^{15(\mathrm{mod}\ 7)}=\alpha^1$, $\alpha^{18(\mathrm{mod}\ 7)}=\alpha^4$.

БЧХ-коды могут декодироваться одним из вышерассмотренных методов. При синдромном декодировании сложность схем коррекции при $t \geq 3$ определяется селектором, который по синдрому должен найти вектор ошибок. При $t \geq 3$ для этого необходимо решение кубических и более высокого порядка уравнений для нахождения локаторов ошибок, указывающих на местоположение ошибок. Это связано с большими вычислительными затратами, поэтому на практике коррекция многократных ошибок используется достаточно редко.

6.5. Задание циклических кодов. Линейные переключательные схемы для умножения и деления многочленов

6.5.1. Задание циклических кодов с помощью корней генераторного полинома

Основным препятствием к широкому использованию линейных кодов, контролирующих многократные случайные и зависимые ошибки, являются большие временные и аппаратурные затраты на декодирование кодовых слов. Это связано с параллельным декодированием всего кодового слова, когда за один такт необходимо исправить ошибки во всем слове. Очевидно, что параллельное декодирование кодовых слов не требуется при последовательном приеме сообщений, характерном для большого числа каналов передачи информации. Применение последовательной обработки информации и циклических кодов приводит к процедурам кодирования и декодирования, эффективным как с алгоритмической, так и с вычислительной точки зрения.

Циклические коды являются подклассом в классе линейных кодов, удовлетворяющим дополнительному свойству цикличности. Линейные коды, удовлетворяющие этому свойству, – *циклические*, т.е. такие коды, которые вместе с каждым кодовым словом $(a_0, a_1, ..., a_{n-1}, a_n)$ содержат также и его циклическую перестановку $(a_1, a_2, ..., a_n, a_0)$. При таком определении для построения кода достаточно задать одно кодовое слово. Отдельные кодовые слова образуются из исходного путем циклического сдвига и всех линейных комбинаций циклических сдвигов.

Выше при описании линейных кодов было показано, что, например, проверочная матрица кода Хэмминга (7; 4) есть матрица, состоящая из элементов α поля $GF(2^3)$,

$$H = \left[\alpha^0 \alpha^1 \alpha^2 \alpha^3 \alpha^4 \alpha^5 \alpha^6 \right].$$

Используя эту матрицу, кодовое слово можно определить как вектор над *GF*(2), такой, что в расширении поля $GF(2^3)$ он дает (исходя из основного уравнения кодирования $GH^T = 0$) нулевое произведение с H^T

$$AH^T = 0 \text{ или } \sum_{i=0}^{6} a_i \alpha^i .$$

Отсюда кодовое слово *A* можно представить в виде многочлена

$$A(x) = \sum_{i=0}^{n\text{-}1} a_i x^i ,$$

и операция умножения AH^T превращается в операцию вычисления многочлена $A(x)$ в точке $x = \alpha$. Условие того, что $A(x)$ – кодовое слово, превращается в равенство $A(\alpha) = 0$, т.е. двоичный многочлен $A(x)$ является кодовым словом тогда и только тогда, когда α – корень многочлена $A(x)$. Представление многочленами кода Хэмминга (7; 4) образует множество всех многочленов над полем *GF*(2) степени не выше шесть таких, что α из $GF(2^3)$ является корнем каждого из них. Таким образом, если $A(x)$ – переданное кодовое слово, то символами переданного слова будут коэффициенты многочлена $A(x)$ (для двоичного кода коэффициенты $\{0,1\}$). Например, при $A(x) = 1 + x^3 + x^5 + x^6$ переданное слово $A = (1001011)$.

Циклический код задается порождающим (генераторным) полиномом $g(x)$ степени *r*. Например, число проверочных разрядов $r = 3$ в коде, задаваемом полиномом $g(x) = 1 + x + x^3$. Для того чтобы полином был генераторным полиномом циклического кода, необходимо, чтобы он был делителем полинома вида $x^n + 1$, где *n* – длина кода. При делении полиномов действия производятся по правилам арифметики по модулю два, в которой вычитание равносильно сложению.

Для построения циклического кода необходимо знать разложение полинома $x^n + 1$ на множители.

Если $n = 2^m - 1$, то многочлен $x^n + 1$ представляется как произведение неприводимых многочленов степени не выше *m*. Каждый из этих многочленов, а также их произведения могут быть использованы как порождающие полиномы циклического кода.

Пример. Пусть $n = 2^3 - 1 = 7$ двоичных символов (бит). Полином $x^7 + 1$ разлагается на следующие неприводимые

$$x^7+1=(x+1)\left(x^3+x+1\right)\left(x^3+x^2+1\right),$$

которые можно использовать как порождающие полиномы:

$g_1(x)=x+1$ порождает код (7;6);

$g_2(x)=x^3+x+1$ порождает код (7;4);

$g_3(x)=x^3+x^2+1$ порождает код (7;4);

$g_1(x)\times g_2(x)$ и $g_1(x)\times g_3(x)$ порождают код (7;3).

Порождающая матрица циклического кода имеет в качестве строк векторы $g(x)$, $xg(x)$, ..., $x^{k-1}g(x)$

$$G=\begin{bmatrix} g(x) \\ xg(x) \\ \dots\dots\dots \\ x^{k-1}g(x) \end{bmatrix}=\begin{bmatrix} g_0 & g_1 & \dots & g_r & 0 & \dots & 0 \\ 0 & g_0 & q_1 & \dots & g_r & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & g_0 & \dots & g_r \end{bmatrix}\Big\updownarrow k,$$

$$\longleftrightarrow$$

$$n$$

где $g_{01}\dots g_r$ – коэффициенты генераторного полинома.

Проверочная матрица строится на основе полинома $h(x)=(x^n+1)/g(x)$ степени k и имеет вид

$$H=\begin{bmatrix} \overleftarrow{h(x)} \\ \overleftarrow{xh(x)} \\ \dots\ \dots\ \dots \\ \overleftarrow{x^{r-1}h(x)} \end{bmatrix}=\begin{bmatrix} 0 & \dots & 0 & h_k & \dots & h_2 & h_1 & h_0 \\ 0 & 0 & h_k & \dots & h_2 & h_1 & h_0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ h_k & \dots & h_2 & h_1 & h_0 & 0 & \dots & 0 \end{bmatrix}\Big\updownarrow r,$$

$$\longleftrightarrow$$

$$n$$

где $h_0, h_1, \dots$ – коэффициенты полинома $h(x)$, называемого *проверочным полиномом.*

Пример.

Пусть $g(x)=1+x^2+x^3+x^4$. Тогда

$$h(x)=x^7+1/g(x)=1+x^2+x^3,\ k=3,\ r=4,$$

т.е. g(x) порождает циклический код (7; 3) (код максимальной длины) с матрицами

$$G_{7;3}=\begin{bmatrix}1&0&1&1&1&0&0\\0&1&0&1&1&1&0\\0&0&1&0&1&1&1\end{bmatrix},\quad H_{7;3}=\begin{bmatrix}0&0&0&1&1&0&1\\0&0&1&1&0&1&0\\0&1&1&0&1&0&0\\1&1&0&1&0&0&0\end{bmatrix}.$$

Для построения циклического кода с матрицей G в приведенно-ступенчатом виде ($G=\left[I_k : G^*\right]$) необходимо вычислить остаток от деления $R_i(x)=x^i / g(x)$, $i=n-1,\ n-2,\ ...,\ n-k$.

Например, для кода (15; 7) задаваемого $g(x)=1+x^4+x^6+x^7+x^8$,

$R_{14}(x)=x^{14}/g(x)=x^3+x^5+x^6+x^7$, $R_8(x)=x^8/g(x)=1+x^4+x^6+x^7$, а

$$G=\begin{array}{c}\begin{matrix}14&13&12&11&10&9&8&7&6&5&4&3&2&1&0\end{matrix}\\\begin{bmatrix}1&&&&&&&1&1&1&0&1&0&0&0\\&1&&&&&&0&1&1&1&0&1&0&0\\&&1&&&&&0&0&1&1&1&0&1&0\\&&&1&&&&0&0&0&1&1&1&0&1\\&&&&1&&&1&1&1&0&0&1&1&0\\&&&&&1&&0&1&1&1&0&0&1&1\\&&&&&&1&1&1&0&1&0&0&0&1\end{bmatrix}\end{array}.$$

Пусть $A(x)=a_0+a_1x+...+a_{n-1}x^{n-1}$ – произвольное слово циклического кода A с генераторным полиномом $g(x)$ и пусть $\alpha_1,\alpha_2,\ ...,\alpha_r$ – корни полинома, среди которых нет повторяющихся.

Т.к. $A(x)$ делится на $g(x)$ без остатка, то все корни $g(x)$ должны быть корнями любого $a(x)\in A$, т.е.

$$a_0+a_1\alpha^i+a_2\left(\alpha^i\right)^2+...+a_{n-1}\left(\alpha^i\right)^{n-1}=0,\ i=1,2,...,r.$$

Эти равенства можно записать в матричной форме

$$(a_0,a_1,...,a_{n-1})\times\begin{bmatrix}1&\alpha^1&(\alpha^1)^2&...&(\alpha^1)^{n-1}\\1&\alpha^2&(\alpha^2)^2&...&(\alpha^2)^{n-1}\\...&...&...&...&...\\1&\alpha^r&(\alpha^r)^2&...&(\alpha^r)^{n-1}\end{bmatrix}^r=0,$$

или более компактно

$$a\left[\alpha\right]^T = 0\,.$$

Матрица $\left[\alpha\right]$ играет здесь роль проверочной матрицы.

Справедливо и обратное: если заданы r неповторяющихся величин $\alpha_1, \alpha_2, \ldots, \alpha_r$ в некотором расширении поля $GF(q)$, то множество последовательностей

$$\alpha = (a_0, a_1, \ldots, a_{n-1})\,,$$

обладающих свойством $a\left[\alpha\right]^T$, образует линейный циклический код, если $n-$ наименьшее общее кратное порядков величин $\alpha_1, \alpha_2, \ldots, \alpha_r$. Порождающим полиномом $g(x)$ является произведение всех различных неприводимых над $GF(q)$ полиномов, корнями которых служат величины $\alpha_1, \alpha_2, \ldots, \alpha_r$, т.е.

$$g(x) = HOK\left[g_1(x), g_2(x), \ldots, g_r(x)\right],$$

где $g_i(x) -$ неприводимый над $GF(q)$ полином, корнем которого является α_m.

Пример. Рассмотрим поле $GF(2^3)$ и неприводимый полином $g(x) = x^3 + x + 1$. Пусть примитивный элемент поля α. Образуем код, генераторный полином которого имеет своими корнями элементы

$$\eta_1 = \alpha\,, \eta_2 = \alpha^2,\ \eta_3 = \alpha^4\,.$$

Элементы поля равны

$$\alpha^0 = (100)\,,\ \alpha^1 = (010)\,,\ \alpha^2 = (001)\,,\ \alpha^3 = (110)\,,$$
$$\alpha^4 = (011)\,,\ \alpha^5 = (111)\,, \alpha^6 = (101)\,.$$

Из этих элементов составим проверочную матрицу

$$H = \begin{bmatrix} & & & & \\ 1 & \alpha^1 & (\alpha^1)^2 & \ldots & (\alpha^1)^6 \\ & & & & \\ 1 & \alpha^2 & (\alpha^2)^2 & \ldots & (\alpha^2)^6 \\ & & & & \\ 1 & \alpha^4 & (\alpha^4)^2 & \ldots & (\alpha^4)^6 \\ & & & & \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ - & - & - & - & - & - & - \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ - & - & - & - & - & - & - \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ \\ 4 \\ 5 \\ 6 \\ \\ 7 \\ 8 \\ 9 \end{matrix}.$$

После вычеркивания линейно зависимых строк получим (совпадают строки 1, 4, 7, 2 и 9; 3 и 5; 6 и 8, кроме того, являются суммой второй и третьей строк)

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix},$$

что с точностью до порядка следования столбцов совпадает с матрицей циклического кода Хемминга.

Кодирование информации циклическими кодами

Умножение сообщения на порождаемую матрицу эквивалентно умножению сообщения, представленного в виде полинома, на генераторный полином. Действительно, пусть сообщение имеет вид

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 +, \ ..., \ a_{k-1} x^{k-1}, \ a_i \in 0,1.$$

Тогда

$$(a_0 a_1 ... a_{k-1}) G = \left[a_0 g_0, (a_0 g_1 + a_1 g_0), (a_0 g_2 + a_1 g_1 + a_2 g_0) \right],$$

что эквивалентно умножению полиномов

$$a(x) g(x) = a_0 g_0 + (a_0 g_1 + a_1 g_0) x + ... \ .$$

Из способа кодирования следует, что любое кодовое слово должно делиться на $g(x)$. Пусть, например, генераторный полином равен $g(x) = 1 + x + x^3$. Этот полином является делителем полинома $x^7 + 1$, т. е. $n = 7$, а $h(x) = 1 + x + x^2 + x^4$.

Пусть передается сообщение (1010), т.е. $A(x) = 1 + x^2$. После кодирования получим следующее кодовое слово

$$A(x) g(x) = (1 + x^2)(1 + x + x^3) = 1 + x + x^2 + x^5,$$

т.е. (1110010). Аналогично находятся другие кодовые слова. Например, $(0001) \to (0001101), (1111) \to (1001011)$.

Рассмотренный способ кодирования не позволяет разделить информационные и проверочные символы, а код, который при этом получается, неразделимый код. Использование неразделимых кодов по многим причинам неудобно.

Циклический неразделимый код можно сделать разделимым следующим образом. Допишем к информационной последовательности $n - k$

нулей, что эквивалентно умножению ее полинома на x^{n-k}, после чего разделим на генераторный полином

$$\left[A(x)x^{n-k}\right]/g(x) = c + R(x)/g(x),$$

где $R(x)$ – некоторый остаток.

Умножим обе части на $P(x)$ и перенесем остаток в левую часть

$$A(x)x^{n-k} + R(x) = cg(x).$$

Т.к. c – целое, то $A(x)x^r + R(x)$ делится на $g(x)$, и, следовательно, является кодовым словом. Таким образом, для того чтобы закодировать сообщение, необходимо приписать к информационным символам остаток от деления $\left[A(x)x^{n-k}\right]/g(x)$. Можно показать, что при таком способе кодирования множество кодовых слов остается тем же самым, а изменяется только таблица соответствия сообщений и кодовых слов.

Пример. Пусть $g(x) = 1 + x + x^3$. Тогда для $A = (0001)$, $A(x) = x^3$, $A(x)x^3 = x^6 \to (0000001)$.

$$\begin{array}{l|l}
x^6 & \underline{x^3 + x\ \ +1} \\
\underline{x^6 + x^4 + x^3} & x^3 + x\ \ +1 \\
\quad x^4 + x^3 & \\
\quad \underline{x^4 + \quad\ x^2 + x} & \\
\quad\quad x^3 + x^2 + x & \\
\quad\quad \underline{x^3 + \quad\ x\ \ +1} & \\
R(x) = \quad\ x^2 + \quad\ 1\ . &
\end{array}$$

Следовательно, $A_k = (101;0001), A_k(x) = 1 + x^2 + x^6$.

Для $A = (0010), R = (111), A_k = (111;0010), A_k(x) = 1 + x + x^2 + x^5$.

Видно, что для получения неразделимого циклического кода требуется устройство, реализующее операцию умножения полиномов, один из которых постоянен, а другой – произвольный, а для образования разделимого циклического кода необходимо устройство деления полиномов. Дан-

ные операции реализуются с помощью линейных автоматов – линейных переключательных схем, реализованных на регистрах сдвига с обратными связями и без них.

6.5.2. Линейные переключательные схемы для умножения и деления многочленов

Рассмотрим умножение и деление некоторого многочлена $A(x) = a_0 x^k + a_{k-1} x^{k-1} + ... + a^0, a_i \in (0,1)$ на фиксированный многочлен $S(x) = s_n x^n + s_{n-1} x^{n-1} + ... + s_0, a_i \in (0,1)$ в конечных полях характеристики два. Так, умножение и деление производится по обычным правилам, однако суммирование осуществляется по модулю два.

В дальнейшем будем считать, что при всех операциях такие многочлены передаются, начиная с коэффициентов высших порядков. Это делается по той причине, что при делении у делителя сначала должны быть обработаны коэффициенты высших порядков. Так, например, коэффициенты многочлена $f(x) = f_n x^n + f_{n-1} x^{n-1} + ... + f_0$ будут подаваться на вход или появляться на выходе в следующем порядке: $f_n, f_{n-1}, ..., f_0$. Любому многочлену $A(x)$ можно поставить в соответствие входную последовательность линейного фильтра

$$X(D) = x_0 + x_1 D + ... + x_k D^k,$$

а фиксированному многочлену $S(x)$ – передаточную функцию фильтра

$$H(D) = h_0 + h_1 D + ... + h_k D^k.$$

Заметим, что здесь индексы возрастают в зависимости от времени, в то время как в многочленах уменьшаются.

Теперь умножение многочлена на X можно представить как сдвиг последовательности на один такт влево, а умножение на X^{-1} – как сдвиг последовательности на один такт вправо. При таком представлении отклик $z(D)$ линейной схемы $H(D)$ на входное воздействие $x(D)$ представляет собой произведение многочленов $A(x)$ и $S(x)$. Общий вид структурных схем для умножения многочленов, с вынесенными и встроенными сумматорами, представлен соответственно на рис. 6.12 и 6.13.

Рассмотрим работу этих схем. Предполагается, что в исходном состоянии элементы задержки $\boxed{i}$ содержат нули.

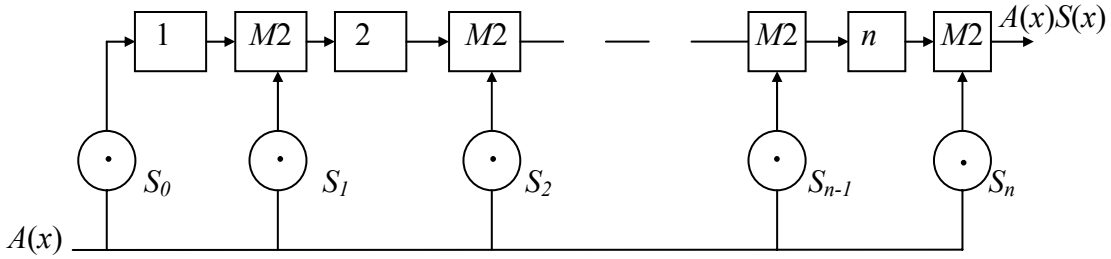


Рис. 6.12. Структурная схема для умножения полиномов
с вынесенными сумматорами по модулю два

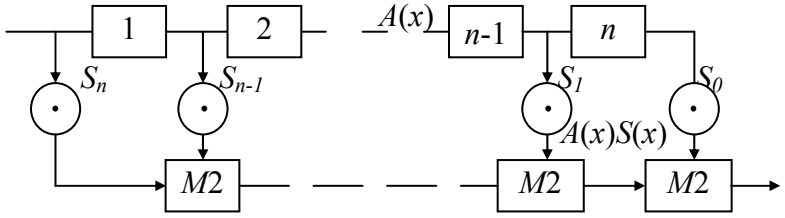


Рис. 6.13. Структурная схема для умножения полиномов
с встроенными сумматорами по модулю два

Пусть на вход схемы рис. 6.12 поступает многочлен $A(x)$, начиная с коэффициентов высших порядков. Когда на входе появляется первый коэффициент многочлена a_k, на выходе появляется первый коэффициент произведения $a_k s_n$. Во втором такте коэффициент a_k, прошедший элемент задержки, умножается на S_{n-1}, а второй коэффициент a_{k-1} умножается на S_n. На выходе формируется коэффициент произведения, равный $(a_{k-1}S_n)+(a_kS_{n-1})$ и т.д. Через $n+k$ сдвигов выход равен последнему коэффициенту произведения a_0S_0. Схема рис. 6.13 работает следующим образом. В первом такте в элементы задержки записываются произведения a_kS_0, ..., a_kS_n, а на выходе появляется первый коэффициент произведения a_kS_n. Во втором такте в элементы задержки записываются величины $a_{k-1}S_0(a_kS_0+a_{k-1}S_1)...(a_{k-1}S_{n-1}+a_kS_{n-1})$, а на выходе появляется второй коэффициент произведения $(a_kS_{n-1}+a_{k-1}S_n)$ и т. д.

Пример. Пусть $S(x)=1+x+x^4$. Тогда соответствующие схемы умножения представлены на рис. 6.14.

Многотактные переключательные схемы могут иметь более одного входа. Например, схема, показанная на рис. 6.15, имеет два входа. Верхняя

половина схемы производит умножение многочлена $A(x)$ на многочлен $g(x)$. Нижняя половина производит умножение многочлена $B(x)$ на многочлен $S(x)$. В результате перемножения и суммирования на выходе получим $f(x) = A(x)g(x) + B(x)S(x)$.

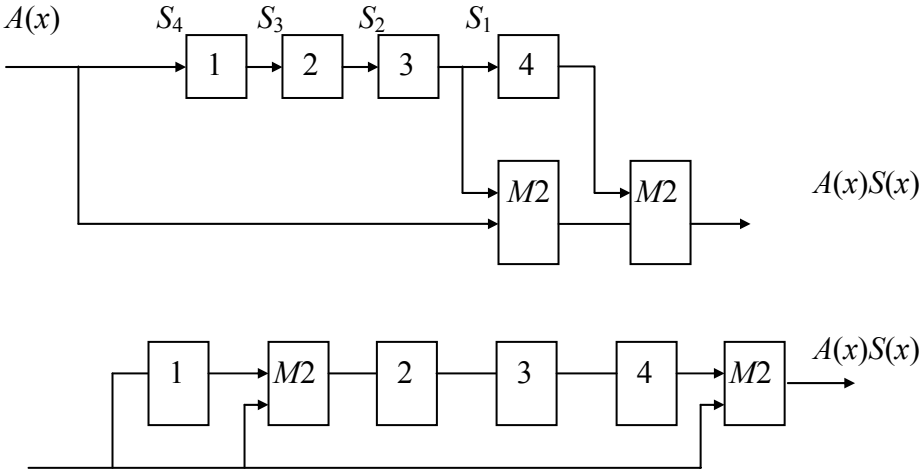


Рис. 6.14. Структурные схемы для умножения полинома $A(x)$ на полином $S(x) = 1 + x + x^4$

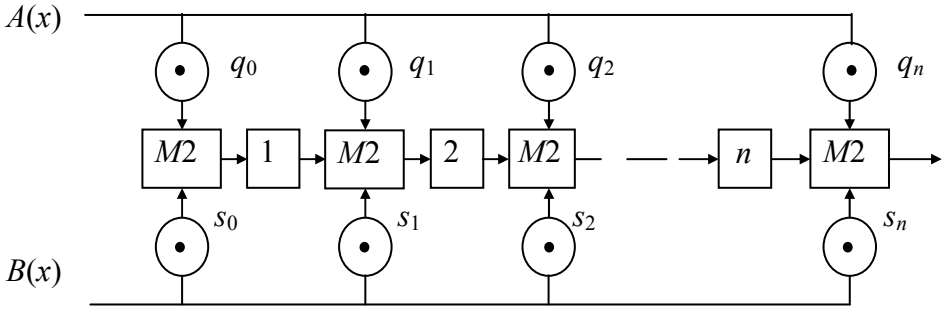


Рис. 6.15. Структурная схема для умножения двух многочленов

Аналогично можно выполнить схему на большее число входов. В том случае, если степени входных полиномов различны, часть старших коэффициентов в полиноме большей степени полагается равной нулю.

Инверсные схемы имеют обратную передаточную функцию и могут быть использованы для деления многочленов. Пусть, например, нужно разделить многочлен $x^8 + x^6 + x^3 + x + 1$ на многочлен $x^4 + x + 1$. Производя деление столбиком, получаем

$$
\begin{array}{llllllll|l}
x^8+ & & x^6+ & & & x^3+ & x+ & 1 & x^4+x+1 \\
\cline{9-9}
x^8+ & & & x^5+ & x^4 & & & & x^4+x^2+x+1 \\
\cline{1-5}
 & & x^6+ & x^5+ & x^4+ & x^3 & & & \\
 & & x^6+ & & & x^3+ & x^2 & & \\
\cline{3-6}
 & & & x^5+ & x^4+ & & x^2+ & x & \\
 & & & x^5+ & & & x^2+ & x & \\
\cline{3-6}
 & & & & x^4+ & & & 1 & \\
 & & & & x^4+ & & x+ & 1 &
\end{array}
$$

$$R(x) = x - \textit{остаток}\,.$$

То же самое можно проделать в двоичной записи, пользуясь только коэффициентами многочлена

$$
\begin{array}{l|l}
101001011 & 10011 \\
\cline{2-2}
10011 & 10111 \\
\cline{1-1}
\ 11110 & \\
\ 10011 & \\
\cline{1-1}
\ 11011 & \\
\ 10011 & \\
\cline{1-1}
\ 10001 & \\
\ 10011 & \\
\cline{1-1}
\ 10 &
\end{array}
$$

Из последней записи видно, что деление заключается в последовательном сложении по модулю два коэффициентов делителя со старшими коэффициентами делимого, получения остатка и повторения указанной операции с коэффициентами остатка и т. д.

Схема для выполнения деления может быть, как и для умножения, реализована в двух вариантах: на регистрах с вынесенными сумматорами (рис. 6.16, а) и встроенными сумматорами (рис. 6.16, б)

С помощью вычислений нетрудно убедиться, что схемы производят деление многочленов. Рассмотрим, например, работу схемы на рис. 6.16, б. Как только первый разряд частного появляется на выходе, происходит суммирование по модулю два делителя и старших разрядов делимого, а остаток через сумматоры записывается в регистр. При появлении на выходе второй единицы частного в регистр записывается второй остаток и т.д. Объединение прямой и инверсной схем позволяет построить комбинированную схему для умножения и деления многочленов.

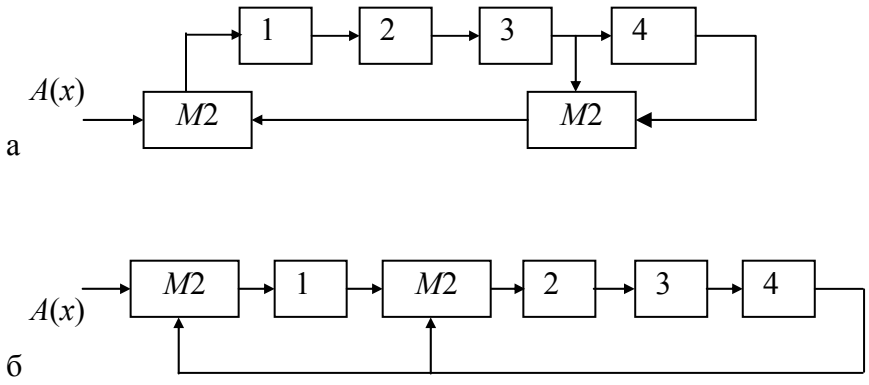


Рис. 6.16. Структурные схемы для деления на многочлен $S(x)=1+x+x^4$

6.6. Построение кодирующих и декодирующих устройств циклических кодов

6.6.1. Кодирующие устройства циклических кодов

Кодирование неразделимым циклическим кодом заключается в умножении полинома сообщения на генераторный полином и производится с помощью схем, описанных в предыдущем разделе.

При кодировании разделимым кодом существует два варианта построения кодирующего устройства в зависимости от соотношения между k и r:

1. если $k > r$, то кодер реализуется по порождающему полиному $g(x)$;
2. если $k < r$, то кодер реализуется по проверочному полиному $h(x)$.

При этом минимизируется число ячеек памяти в регистре сдвига.

Рассмотрим основное проверочное соотношение:

$$(a_0, a_1, ..., a_{n-1}) \cdot \begin{bmatrix} ... & ... & h_k & ... & h_2 & h_1 & h_0 \\ ... & h_k & ... & h_2 & h_1 & h_0 & ... \\ ... & ... & ... & ... & ... & ... & ... \\ h_k & ... & h_2 & h_1 & h_0 & ... & ... \end{bmatrix}^T = 0.$$

Поскольку $h_k = 1$, то из произведения вектора сообщения на первую строку матрицы получим выражение для первого проверочного символа

$$a_{n-k-1} = \sum_{i=0}^{k-1} a_{n-1-i} h_i .$$

Из произведения на вторую строку получим

$$a_{n-k-2} = \sum_{i=0}^{k-1} a_{n-2-i} h_i$$

и т. д.

$$a_{n-k-j} = \sum_{i=0}^{k-1} a_{n-1-j-i} h_i \ ,$$

т.е. можно найти $n-k$ проверочных символов по k информационным символам. Схема кодера, выполняющего эту операцию, приведена на рис. 6.17. Он работает следующим образом: сначала ключ K находится в положении 1, и на вход подаются информационные символы. После k тактов информационные символы занимают все k ячеек регистра. Затем ключ переводится в положение 2, и регистр совершает еще n тактов, при каждом из которых на выходе появляется очередной символ кодового слова. Уже при первом из этих n тактов в первой ячейке формируется проверочный символ. За $(n-k)$ тактов весь кодовый вектор сформирован, $(n-k)$ символов выданы на выход, а остальные k символов находятся в регистре. Ключ возвращается в положение 1, и в регистр вводится k информационных символов следующего вектора, а оставшиеся в регистре k символы предыдущего вектора выводятся на выход кодера.

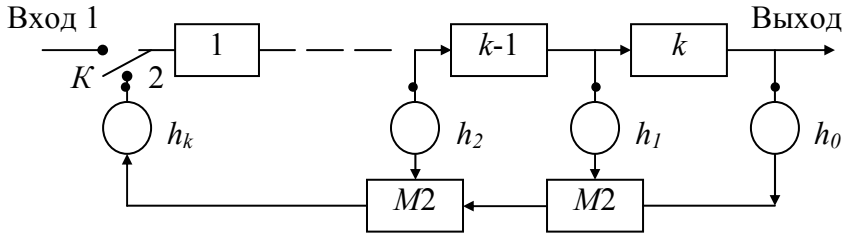


Рис. 6.17. Кодер с k-разрядным регистром, реализованный по полиному $h(x)$

Второй вариант кодирующего устройства содержит $(n-k)$-разрядный регистр сдвига и определяет остаток от деления $A(x)x^r$ на генераторный полином $q(x)$ в соответствии с правилами образования кода, приведенными выше.

Соответствующая схема такого кодера приведена на рис. 6.19 и работает следующим образом: вначале ключ $K1$ находится в положении 1, а ключ $K2$ замкнут. Информационные символы, подаваемые на вход через ключ $K1$, поступают на выход, а через ключ $K2$ – в кодирующее устрой-

ство, где через k тактов образуется $(n-k)$ проверочных символов. После этого ключ $K1$ переводится в положение 2, а ключ $K2-$ размыкается. Затем регистр делает еще $(n-k)$ тактов, выдавая контрольные символы из ячеек регистра на выход кодера.

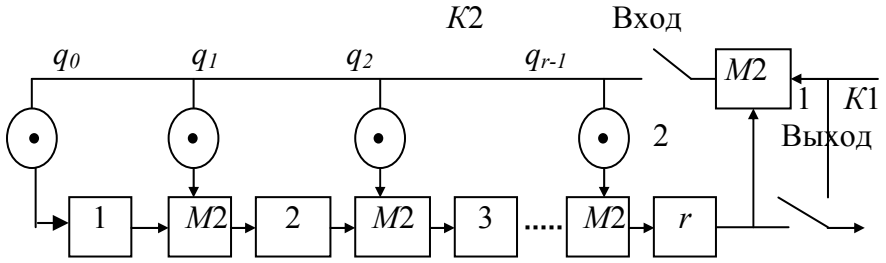


Рис.6.18. Кодер с *r*- разрядным регистром, реализованный по полиному $g(x)$

Пример. Пусть $g(x)=1+x+x^3$, а $h(x)=1+x^2+x^3+x^4$. Тогда соответствующие кодеры циклических кодов имеет вид (рис. 6.19)

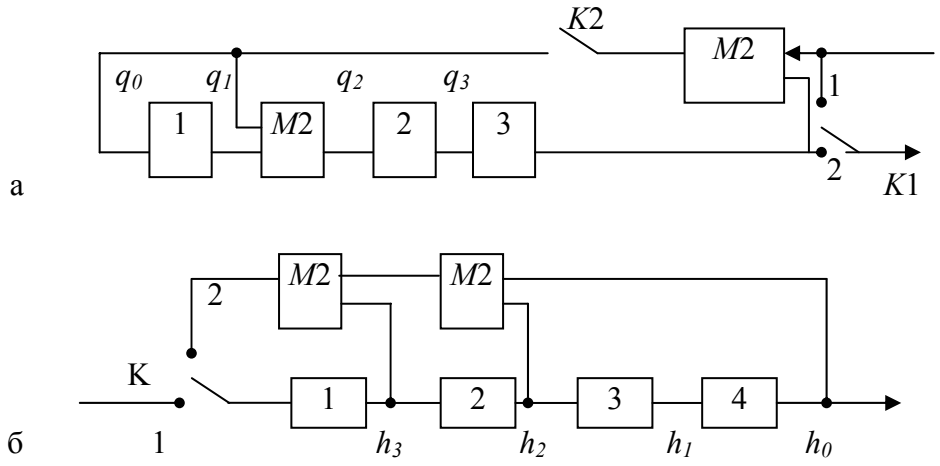


Рис. 6.19. Кодеры циклических кодов, задаваемых полиномами $g(x)=1+x+x^3$ (а) и $h(x)=1+x^2+x^3+x^4$ (б)

6.6.2. Мажоритарное декодирование циклических кодов

Из основного проверочного соотношения

$$(a_0, a_1, \ ..., a_{n-1})H^T = 0,$$

где

$$H = \begin{bmatrix} h_{00} & h_{01} & \dots & h_{0,n-1} \\ \dots & \dots & \dots & \dots \\ h_{r-1,0} & h_{r-1,1} & \dots & h_{r-1,n-1} \end{bmatrix},$$

следует система уравнений

$$\begin{cases} a_0 h_{00} + a_1 h_{01} + \dots + a_{n-1} h_{0,n-1} = 0, \\ a_0 h_{r-1,0} + a_1 h_{r-1,1} + \dots + a_{n-1} h_{r-1,n-1} = 0. \end{cases}$$

Выбрав из них те, для которых $h_{i0} \neq 0$, получим:

$$\begin{cases} a_0 = a_1 h_{i_1} 1 + \dots + a_{n-1} h_{i_1} n - 1, \\ a_0 = a_1 h_{i_2} 1 + \dots + a_{n-1} h_{i_2} n - 1. \end{cases}$$

Отсюда следует, что символ a_0 можно определить по принципу большинства, полагая $a_0 = 0$, если правая часть большинства уравнений последней системы равна нулю, и $a_0 = 1$ в противном случае. Такую же процедуру можно проделать для символов $a_1, a_2, \dots, a_{n-1}$. В случае, если код циклический, система проверочных уравнений для символов $a_1, a_2, \dots, a_{n-1}$ получается из системы для символа a_0 циклическим сдвигом. Для декодирования символа a_i достаточно произвести циклическую перестановку кодового слова на i позиций и использовать ту же самую систему проверок. (Напомним, что для линейных кодов (нециклических) для декодирования каждого символа должна быть найдена своя система проверок.

Система проверок, как и при декодировании линейных кодов, может быть разделенной, λ-связанной и квазиразделенной. В качестве примера рассмотрим построение системы λ-связанных проверок и соответствующего мажоритарного декодирования.

Пример. Пусть имеется код (7; 3), задаваемый порождающим полиномом $g(x)=1 + x + x^2 + x^4$. Найдем проверочный полином

$$h(x) = \left(x^n + 1\right) / g(x) = \left(x^7 + 1\right) / \left(1 + x + x^2 + x^4\right) = x^3 + x + 1.$$

Тогда проверочная матрица H имеет вид

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Просуммируем вторую и третью строки и запишем вместо третьей строки, далее переставим столбцы для получения H в приведенно-ступенчатом виде

$$H = \begin{matrix} \\ \\ \\ \end{matrix}\begin{matrix} a_0 \\ \end{matrix}\begin{matrix} a_1 \\ \end{matrix}\begin{matrix} a_2 \\ \end{matrix}\begin{matrix} a_3 \\ \end{matrix}\begin{matrix} a_4 \\ \end{matrix}\begin{matrix} a_5 \\ \end{matrix}\begin{matrix} a_6 \\ \end{matrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} = \left[\begin{array}{ccc:cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right].$$

Отсюда

$$\begin{cases} a_0 + a_1 + a_3 = 0 & (6.10) \\ a_1 + a_2 + a_4 = 0 & (6.11) \\ a_0 + a_1 + a_3 + a_5 = 0 & (6.12) \\ a_0 + a_2 + a_6 = 0 & (6.13) \end{cases}$$

Разрешим эту систему относительно a_0

$$\begin{cases} a_0 = a_0 \\ a_0 = a_1 + a_3 \\ a_0 = a_1 + a_2 + a_5 \\ a_0 = a_2 + a_6 \end{cases}$$

Видно, что в данной системе символы a_1 и a_3 входят дважды, следовательно, для реализации проверок с $\lambda = 2$ необходимо дополнить первую систему одним уравнением. Для этого просуммируем (6.11) и (6.12). В результате получим $a_0 = a_4 + a_5$, а вторая система преобразуется в следующую систему проверок, реализованную на рис. 6.20.

$$\begin{cases} a_0 = a_0, \\ a_0 = a_1 + a_3, \\ a_0 = a_1 + a_2 + a_5, \\ a_0 = a_2 + a_6, \\ a_0 = a_4 + a_5. \end{cases}$$

При приеме сообщения ключ находится в положении 1, и принятый вектор за n тактов записывается в буферный регистр. Далее ключ переводится в положение 2; в результате образуется обратная связь с выхода на вход декодера для реализации систем проверок для a_i. На первом этапе декодируется символ a_0, а затем a_1 из системы

$$\begin{cases} a_1 = a_1, \\ a_1 = a_2 + a_4, \\ a_1 = a_2 + a_3 + a_6, \\ a_1 = a_3 + a_0, \\ a_1 = a_5 + a_6. \end{cases}$$

и т.д.

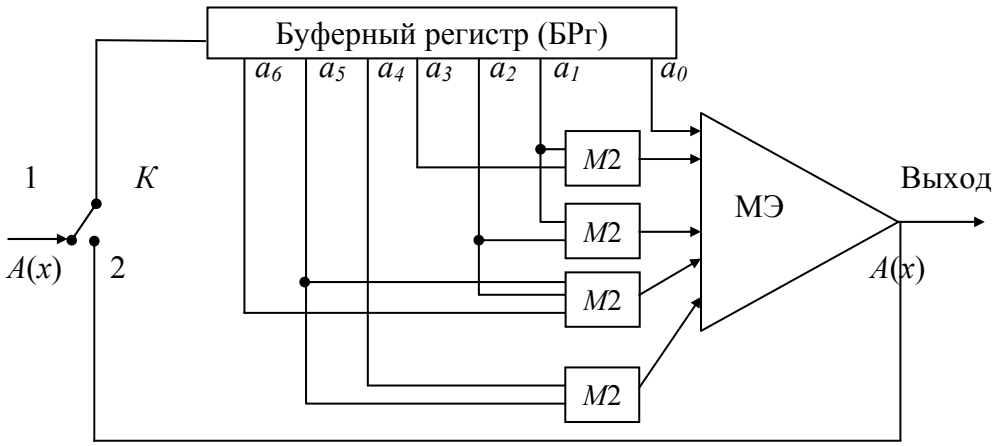


Рис. 6.20. Структурная схема мажоритарного декодера циклического кода (7; 3)

6.6.3. Синдромное декодирование циклических кодов

Обобщенная структурная схема синдромного декодера циклического кода представлена на рис. 6.21.

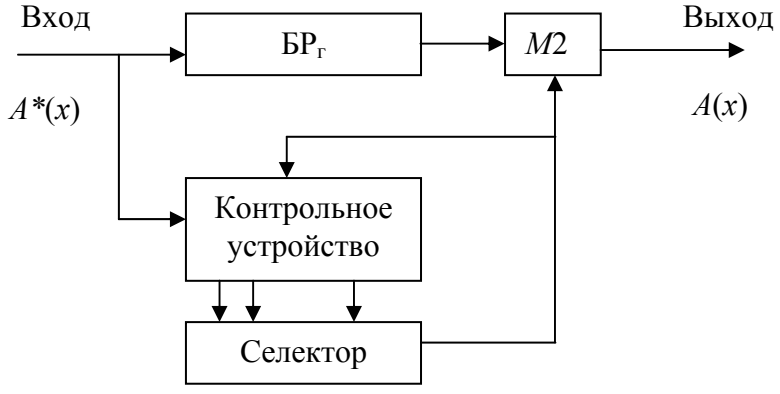


Рис. 6.21. Обобщенная структурная схема синдромного декодера циклического кода

Последовательность $A^*(x)$ записывается в *n*-разрядный буферный регистр так, что через n тактов все слово оказывается записанным в регистр. Одновременно последовательность поступает в контрольное устройство (делитель на $g(x)$), которое производит вычисление синдрома $A \cdot H^T$. Ранее было показано, что все слова циклического кода должны делиться на порождающий полином $g(x)$. Если принятое слово принадлежит коду, то остаток от его деления на полином $g(x)$ равен нулю, ненулевой остаток свидетельствует об ошибке. По виду остатка можно определить ошибку. Контрольное устройство совместно с селектором и производит эти операции. В контрольном устройстве производится вычисление остатка от деления $A^*(x)$ на $g(x)$. Селектор анализирует полученный остаток и выдает корректирующий сигнал в тот момент, когда ошибочный символ покидает буферный регистр, одновременно в контрольное устройство выдается сигнал, обозначающий, что осталась некоторая более простая комбинация ошибок. Если после $2n$ сдвигов, т.е. когда последний символ покидает буферный регистр, состояние контрольного устройства будет ненулевым, это означает, что произошла некорректируемая ошибка.

Рассмотрим подробнее работу делителя при одиночной ошибке в принятом слове. После *n* тактов в регистре делителя будет находиться ненулевой остаток от деления $R(x) \neq 0$, т.е. $R(x) = x^i S(x)$. Последующие i сдвигов в регистре делителя соответствуют изменению степени этого полинома

$$R^*(x) = x^{i-1} S(x),$$
$$R^{**}(x) = x^{i-2} S(x),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$R^i(x) = S(x).$$

Видно, что остаток от деления будет воспроизведен через i тактов. Поскольку в случае одиночных ошибок $E(x) = x^i$, то $S(x) = 1$ и $R(x) = 1$, что соответствует содержимому ячеек памяти r-разрядного регистра (00...01). Таким образом, для коррекции одиночной ошибки селектор в декодере циклических кодов должен быть настроен на комбинацию (00...01), причем единица расположена в старшем разряде делителя.

Это следует и из того факта, что после первых *n* тактов входная информация не поступает в делитель, и он превращается в генератор поля Галуа, генерируя такт за тактом элементы поля a^i вплоть до появления элемента поля a^i с единичным коэффициентом в старшем разряде.

Работу синдромного декодера циклического кода Хэмминга (7; 4), исправляющего одиночные ошибки, задаваемого порождающим полиномом $g(x) = 1 + x + x^3$, проследим на примере декодера, представленного на рис. 6.22.

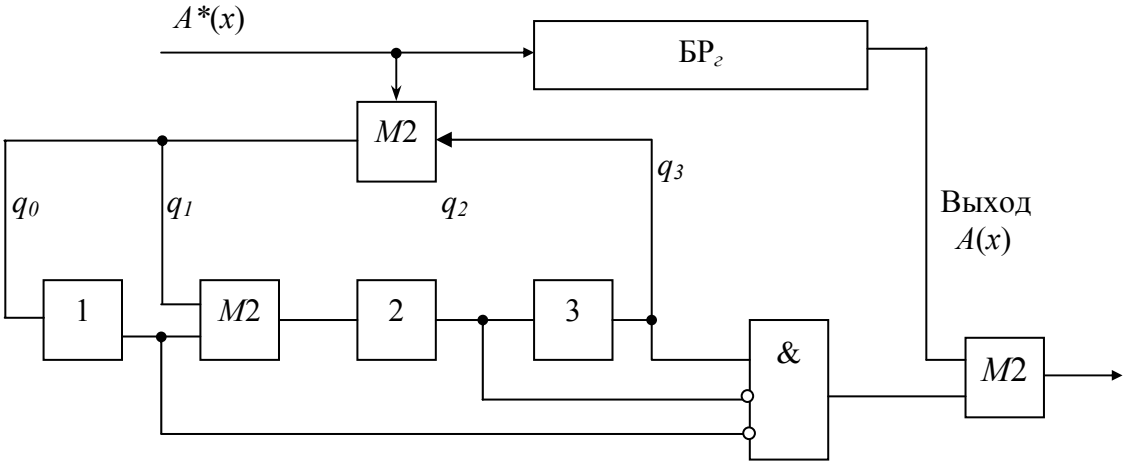


Рис. 6.22. Схема синдромного декодера циклического кода Хэмминга (7; 4)

Схема декодирующего устройства состоит из семиразрядного буферного регистра, трехразрядного делителя и селектора, настроенного на комбинацию (001). Единичный сигнал на выходе селектора появляется только при этой комбинации в ячейках контрольного устройства. Пусть передается кодовая комбинация, состоящая из одних нулей, т.е. (0000000), но в результате ошибки принимается комбинация (0000001). Состояния ячеек делителя в последовательные моменты времени, соответствующие этой комбинации, представлены в табл. 6.3. Через семь тактов весь принятый вектор будет записан в буферном регистре, причем искаженный первый символ будет записан в седьмую ячейку буферного регистра. При этом в контрольном устройстве находится комбинация (001). На восьмом такте искаженный символ покидает буферный регистр. Одновременно с этим с селектора выдается единичный сигнал, и происходит исправление ошибки, а в контрольном устройстве остается комбинация (000).

Математически процесс исправления ошибки может быть представлен следующим образом: поскольку принятое слово $A(x) = x^6$, то

1. $x^6 x^r = x^6 x^3 = x^9$,
2. $x^9 / g(x) = x^9 /(1 + x + x^3) = x^2 = R(x) \to (001)$.

В общем случае декодирующее устройство получается значительно более сложным, чем в рассмотренном примере, т.к. с увеличением длины

кода и кратности исправляемых ошибок число селектируемых комбинаций возрастает.

Таблица 6.3

Состояние элементов памяти регистра сдвига делителя

Такты	Состояние элементов памяти		
	1 (q_1)	2 (q_2)	3 (q_3)
1	1	1	0
2	0	1	1
3	1	1	1
4	1	0	1
5	1	0	0
6	0	1	0
7	0	0	1

В табл. 6.4 приведено число селектируемых комбинаций для кода длиной $n = 63$. Из табл. 6.4 видно, что уже при исправлении двойных ошибок сложность селектора превышает сложность контрольного устройства. Поэтому рассмотренное декодирующее устройство находит в основном применение для исправления ошибок малой кратностью $t \leq 2$, а также для обнаружения ошибок.

Таблица 6.4

Число селектируемых комбинаций циклических кодов

Кратность исправляемых ошибок	1	2	3
Число селектируемых комбинаций	1	63	1955
Количество элементов памяти в регистре делителя	6	12	18

6.7. Кодирование для исправления зависимых ошибок

Коды Рида – Соломона

Коды Рида – Соломона (PC, RS) являются линейными блочными исправляющими ошибки кодами с широким диапазоном применений в цифровых системах связи и хранения информации. Примерами областей использования являются:

- устройства хранения цифровой информации (цифровые кассеты, компакт-диски, DVD-диски, штриховые коды);
- беспроводные и мобильные системы связи (например, в сотовых телефонах);
- цифровое телевидение;
- высокоскоростные модемы, такие как ADSL, xDSL и т.д.;

Коды Рида – Соломона применялись NASA для связи со спутниками Voyager, Gallileo, Magellan и других при их запусках к планетам Солнечной системы.

Коды Рида – Соломона являются важным подклассом кодов БЧХ. Примитивным кодом БЧХ является блоковый код длиной $n = q^m - 1$ над полем $GF(q)$, для которого элементы $\alpha^{m_0}, \alpha^{m_0+1}, ..., \alpha^{m_0+2\cdot t-1}$ (для произвольного m_0) являются корнями порождающего многочлена $g(x)$, где α – примитивный элемент $GF(q^m)$, t – кратность исправления ошибки. Порождающий многочлен кода БЧХ можно записать как

$$g(x) = HOK\left[m_{m_0}(x), m_{m_0+1}(x), ..., m_{m_0+2\cdot t-1}(x)\right],$$

где минимальные полиномы – $m_i(x) = \prod_{j \in C_i}\left(x - \alpha^j\right)$;

C_i – множество $C_i = \left\{i, q\cdot i, ..., q^{d-1}\cdot i\right\}$;

d - наименьшее целое, удовлетворяющее условию $iq^d = (i \bmod n)$;

НОК – наименьшее общее кратное.

Коды Рида – Соломона – коды БЧХ с $m = m_0 = 1$. Таким образом, можно дать определение: q-ичным кодом Рида – Соломона или кодом RS(n,k) называется q-ичный код БЧХ – BCH(q-1,k) с длиной $n = q$-1.

Таким образом, длина q-ичного кода Рида – Соломона равна числу ненулевых элементов поля $GF(q)$. Т.е. для $GF(8)$ число ненулевых элементов поля – 1, 2, 3, 4, 5, 6, 7 – равно семи, соответственно и длина кода Рида – Соломона равна семи.

Т.к. $n = q$-1, и корни полинома $x^{q\text{-}1}$-1 являются ненулевыми элементами поля $GF(q)$, то

$$x^n - 1 = x^{q-1} - 1 = \prod_{\alpha \in GF(q)}(x - \alpha)$$

$$m_i(x) = \left(x - \alpha^i\right)$$

Как результат, генераторный (порождающий) полином кода Рида – Соломона, исправляющего t ошибок, задается формулой

$$g(x) = \prod_{i=1..2t} m_i(x) = \prod_{i=1..2t}\left(x - \alpha^i\right) = \left(x - \alpha^1\right)\cdot\left(x - \alpha^2\right)\cdot\ldots\cdot\left(x - \alpha^{2\cdot t}\right).$$

Таким образом, кодер, построенный на основе данного полинома, получает $k = n$-2t символов данных, каждый из которых состоит из m бит, где m – степень простого числа, при возведении которого в эту степень получается

q. К этим k данным кодер добавляет проверочные символы в количестве n-k для получения кодового слова длиной n. Проверочные n-k символы также состоят из m бит. Декодер Рида – Соломона может исправлять до t ошибок включительно и определить, что произошло до $2t$ ошибок, но не сможет исправить их, где $t = \frac{n-k}{2}$. Эти ошибки подразумеваются относительно передаваемых символов, а не передаваемых битов. Т.е., при ошибке в передаче t символов может случиться от t ошибок в битах, каждая из которых по одной встречается в различных кодовых символах, до $t \cdot m$ ошибок в битах, при этом в ошибочных символах неправильны все биты.

Рассмотрим пример. Пусть $q=2^3=8$, таким образом: $n=q-1=8-1=7$; $m=3$. Примитивным полиномом над данным полем является, например, полином $f(x) = x^3 + x + 1$. Исходя из этого, корень α этого примитивного полинома дает нам генераторный полином для кода, исправляющего $t=2$ ошибки (при этом $k=n-2t=7-2\cdot 2=3$; $m_i(x)$ вычисляется для $i=1..2t$):

$$m_1(x) = (x - \alpha)$$
$$m_2(x) = \left(x - \alpha^2\right)$$
$$m_3(x) = \left(x - \alpha^3\right)$$
$$m_4(x) = \left(x - \alpha^4\right)$$

$$g(x) = m_1(x) \cdot m_2(x) \cdot m_3(x) \cdot m_4(x) =$$
$$= (x - \alpha) \cdot \left(x - \alpha^2\right) \cdot \left(x - \alpha^3\right) \cdot \left(x - \alpha^4\right) =$$
$$= x^4 + \alpha^3 \cdot x^3 + x^2 + \alpha \cdot x + \alpha^3$$

Кодер для данного кода RS(7, 3) принимает $k=3$ информационных символа (каждый из которых состоит из $m=3$ бит), добавляет к ним n-$k=4$ проверочных символа, получая кодовое слово длиной $n=7$ (все символы также имеют 3 бита), при этом исправляется $t=2$ ошибки в кодовых символах.

Например, если были закодированы данным кодом RS(7, 3) информационные символы 6, 2 и 4, то кодовое слово после кодирования выглядит как {6, 2, 4, 6, 0, 0, 2}. При двух ошибках декодер может получить, например, кодовое слово {6, 2, 6, 6, 0, 0, 5} или {6, 2, 3, 6, 0, 0, 6}.

В первом случае вместо 4 принято 6 и вместо 2 принято 5. Т.е. в бинарном выражении вместо 100_2 принято 110_2, а вместо 001_2 принято 101_2. Следовательно, при данной передаче произошло 2 ошибки в кодовом слове, выраженные двумя ошибками в битах (наилучший случай).

Во втором случае вместо 4 принято 3 и вместо 2 принято 6. В двоичной записи: вместо 100_2 принято 011_2, а вместо 001_2 принято 110_2. Следовательно, при данной передаче произошло 2 ошибки в кодовом слове, выраженные шестью ошибками в битах (наихудший случай).

Оба принятых сообщения декодер правильно декодирует. Однако может произойти три ошибки в битах, но каждый из этих бит находится в различных символах кодового слова, следовательно, произойдет три ошибки, и декодер не сможет достоверно исправить ошибки.

Кодирование и декодирование кода Рида – Соломона может быть реализовано программно или аппаратно. Этот процесс базируется на арифметике в конечных полях или поле Галуа.

Кодовое слово строится как: $u(x) = g(x) \cdot \inf(x)$, где $u(x)$ – кодовое слово на выходе кодера; $g(x)$ – генераторный (порождающий) полином, $\inf(x)$ – информационные символы.

Алгоритм процесса кодирования представляется следующим образом:

Операция первая. Получаемые информационные символы $\inf(x)$ умножаются на x^{n-k}. Эта операция также называется *сдвигом вверх*, при этом получается полином со степенными коэффициентами от (n-k+1) до n, то есть информационные символы, не изменяясь, первыми поступают на кодер, как того требует алгоритм. Получаемый полином назовем $\inf_c(x)$.

Операция вторая. 2*t проверочных символов получаются в результате вычисления полиномиального остатка от деления по правилам полей Галуа полученного полинома $\inf_c(x)=\inf(x)\cdot x^{n-k}$ на генераторный полином $g(x)$. Это можно записать как: $p(x)=\inf_c(x) \bmod g(x)$; $g(x)=(\inf(x)\cdot x^{n-k}) \bmod g(x)$, где $p(x)$ – остаток от вышеописанного деления, полином с степенными коэффициентами от (n-k) до 1.

Операция третья. Суммируем полином $\inf_c(x)$ с полиномом $p(x)$ – получаем кодовое слово $u(x)$ – полином степени n.

Перемежение в системах с кодированием

В случаях когда нужно учитывать пакеты ошибок, одно из возможных решений состоит в применении каких-либо кодера и декодера, предназначенных для исправления случайных ошибок, вместе с парой устройств, состоящей из устройства перемежения и устройства восстановления после перемежения. При таком подходе последовательность на выходе декодера подвергается перемежению до передачи по каналу и восстанавливается перед декодированием, так что на входе декодера ошибки распределяются более равномерно. Структурная схема системы показана на рис. 6.23. Если демодулятор квантует каждый кодовый символ на Q бит, то устройство восстановления требует в Q раз большую память, чем устройство перемежения.

Устройство перемежения переупорядочивает (переставляет) символы в последовательности некоторым детерминированным образом. С устройством перемежения связано устройство восстановления после перемежения, с помощью которого осуществляется обратная перестановка и восстанавливается исходный порядок символов. Существует много типов таких устройств. Два важных класса устройств перемежения – это периодические и псевдослучайные. Периодические устройства перемежения во многих случаях оказываются предпочтительнее псевдослучайных из-за простоты. Однако псевдослучайные устройства перемежения характеризуются большей устойчивостью, чем периодические. Поэтому псевдослучайные устройства перемежения могут оказаться предпочтительнее в некоторых практических ситуациях, когда характеристики пакетов в канале могут меняться со временем.

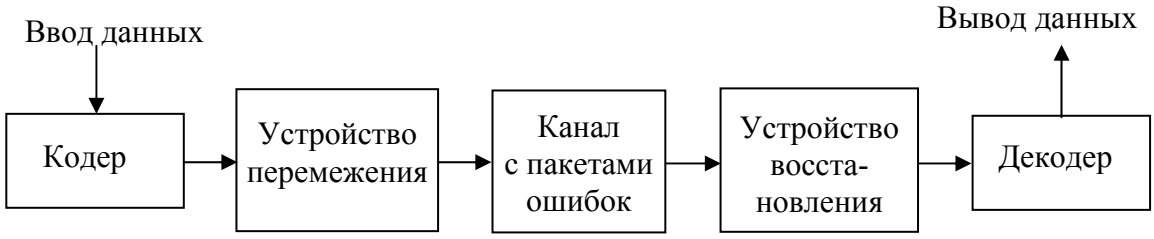


Рис. 6.23. Структурная схема применения внешних устройств перемежения и восстановления после перемежения

Периодические устройства перемежения

Периодическим называется такое устройство перемежения, в котором перестановка является периодической функцией времени. Обычно используется устройство перемежения одного из двух типов. На вход блоковых устройств перемежения символы поступают блоками, и устройство производит одну и ту же перестановку каждого блока символов. Сверточные устройства перемежения не имеют фиксированной блоковой структуры; они осуществляют периодическую перестановку полубесконечной последовательности кодовых символов. Различие между устройствами перемежения двух типов очень похоже на различие между блоковыми и сверточными кодами.

Блоковые устройства перемежения

Типичное блоковое устройство перемежения работает следующим образом. Кодовые символы записываются в столбцы матрицы, состоящей из N строк и B столбцов. Перестановка состоит в том, что для передачи по каналу символы считываются из матрицы по строкам. Такое устройство

называется *блоковым* (B, N) –устройством перемежения. Устройство восстановления после перемежения осуществляет обратную операцию: записывает символы по строкам, а считывает их по столбцам. Ясно, что такие устройства перемежения и восстановления легко реализуются с помощью современной цифровой техники.

Наиболее важные свойства при таком способе перемежения состоят в следующем:

1. любой пакет ошибок длиной $b \le B$ переходит на выходе устройства восстановления в одиночные ошибки, каждые две из которых разделены не менее чем N символами;

2. любой пакет ошибок длиной $b = rB \quad (r > 1)$ переходит в пакеты ошибок длиной, не большей $[r]$ символов, каждые два из которых разделены не менее чем $N - [r]$ символами;

3. периодическая последовательность одиночных ошибок, разделенных B символами, переходит в один пакет ошибок длиной N на выходе устройства восстановления;

4. задержка устройства составляет $2NB$ символов (в дополнение к задержке в канале), и каждое из устройств требует наличия памяти емкостью NB символов.

В типичных случаях параметры устройства перемежения выбирают такими, чтобы длина b всех ожидаемых пакетов ошибок не превышала B. Если, однако, характеристики пакетов ошибок существенно нестационарны, то устройства перемежения такого типа согласно свойству (3) могут быть неустойчивыми.

Выбор параметра N зависит от используемой схемы кодирования. В обычных приложениях влияния памяти в канале не проявляется на любом отрезке из N символов на выходе устройства восстановления. Поэтому параметр N должен быть выбран большим, чем длина отрезка, на котором производится декодирование. Для блоковых кодов значение N должно быть больше длины блока, а для сверточных кодов – больше длины кодового ограничения. При этом каждый пакет ошибок длиной $b \le B$ приведет не более чем к одной ошибке в каждом кодовом слове блокового кода. Аналогично для сверточных кодов существует не более одной ошибки на каждом отрезке, длина которого равна длине кодового ограничения.

Использование блоковых устройств перемежения приводит практически к тем же задачам синхронизации, что и использование блоковых кодов. Устройство восстановления не может правильно работать до тех пор, пока не будет точно известно начало каждого блока устройства перемежения. Конеч-

но, в этом случае не может надежно происходить процесс исправления ошибок. Устройства перемежения и восстановления могут быть синхронизированы с помощью стандартных методов кадровой синхронизации, когда специальное синхронизирующее слово с хорошими корреляционными свойствами периодически вставляется в кодовую последовательность в устройстве перемежения, а затем восстанавливается кадровым синхронизатором в устройстве восстановления. При таком методе в последовательность вставляется обычно 1... 2% дополнительных символов. Другой подход (не требующий введения дополнительных символов) состоит в том, что некоторые кодовые символы заменяются синхрословом. Затем эти символы стираются на входе декодера. Во избежание существенного ухудшения характеристик требуется позаботиться, чтобы эти стертые символы находились далеко друг от друга на входе декодера. Построение кадрового синхронизатора для систем с кодированием обычно несколько труднее, чем для систем без кодирования, поскольку значение E_s / N_0 для первых обычно более низкое.

Псевдослучайные устройства перемежения

Псевдослучайное устройство перемежения представляет собой блоковое устройство, которое берет блоки из L символов канала после декодирования и переставляет их псевдослучайным образом. Это можно сделать, записав L символов последовательно в память с произвольной выборкой (ЗУПВ) и затем считав их псевдослучайным образом. Требуемую перестановку можно записать в постоянную память (ПЗУ), а затем использовать эту перестановку для адресации памяти устройства перемежения.

Такой метод обеспечивает высокую степень устойчивости при изменении параметров пакетов, однако его сложность превышает, конечно, сложность блокового или сверточного устройства перемежения того же объема. Наиболее интересными являются применения таких устройств для защиты от организационных помех (ЗП), при этом используется та или иная система широкополосной модуляции. Именно на такие случаи применения будет сделан упор в последующем изложении.

Если в каждом блоке осуществляется одна и та же перестановка, то будут существовать некоторые комбинации аддитивных помех, которые могут существенно ухудшить характеристики. В тех системах, где возникновение таких комбинаций весьма вероятно (как, например, в системах борьбы с организационной помехой при условии, что организатор помехи знает перестановку), эту перестановку нужно часто менять (например, после каждого блока). Один из возможных здесь методов состоит в записи в ПЗУ некоторого числа M различных перестановок и в случайном выборе одной из

них для перемежения каждого блока. Параметр M должен быть достаточно большим, для того чтобы исключить уязвимость по отношению к помехам, организованным с учетом знания этого множества перестановок. Величина M зависит от критериев, по которым определяется требуемое качество системы передачи, и от оценок способности схемы организованных помех. Типичными являются значения, лежащие между 10 и 100.

Типичная структурная схема устройства перемежения (управляемого генератором псевдослучайного шума) показана на рис. 6.24. Символы канала последовательно записываются в память устройства перемежения. После записи всего блока эти символы переставляются путем считывания, осуществляемого с использованием псевдослучайной перестановки, записанной в адресном ПЗУ. Для правильной работы устройства необходимы два ЗУПВ, работающие в противофазе: во время записи в одно из ЗУПВ происходит считывание из другого. После завершения этого процесса роли двух ЗУПВ меняются и выбирается новая псевдослучайная перестановка.

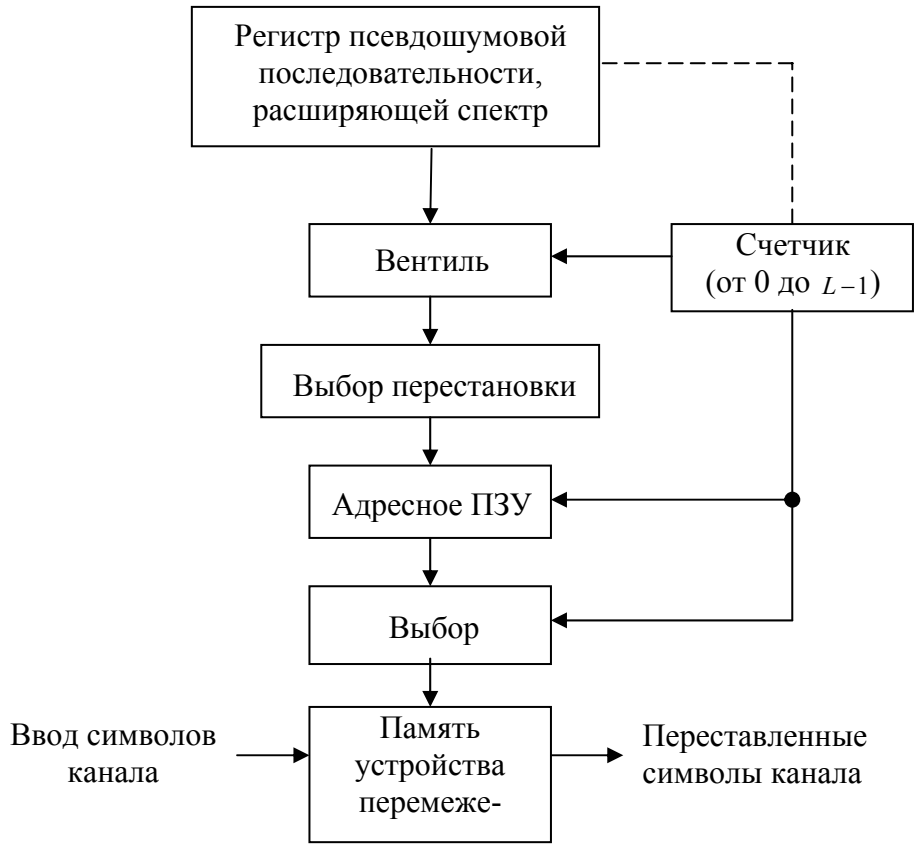


Рис. 6.24. Структурная схема
псевдослучайного устройства перемежения

Работа устройства перемежения может синхронизироваться фиксированием связи между состоянием регистра псевдослучайного шума, используемого для порождения последовательности, расширяющей спектр, и счетчиком принятых символов. При захвате псевдошумовой последовательности, устанавливаются границы блока перемежения. На рисунке показано также, что состояние регистра псевдослучайного шума на границе блока используется для выбора новой перестановки (одной из M возможных) для следующего перемежения. В других случаях применения (отличных от систем с организованными помехами) устройство восстановления после перемежения может быть синхронизировано с помощью стандартных методов кадровой синхронизации, учитывающих пакетный характер ошибок в канале.

Для случайных пакетов с ограниченной максимальной длиной обычно выбирается периодическое перемежение. Однако периодические пакеты стираний обрабатываются гораздо легче, и известно, что для пакетов стираний умеренной длины (не больше половины длины кодового ограничения в переданных символах) при низкой интенсивности ухудшение незначительно даже без перемежения. Для более длинных пакетов, когда перемежение необходимо, периодическое устройство перемежения, параметры которого согласованы с номинальным параметром пакетов, оказывается устойчивым к сравнительно большим изменениям длины пакетов. Имеются, однако, случаи, когда периодическое перемежение не обладает достаточной устойчивостью (например, при организованных помехах). Случайное перемежение обладает большей степенью устойчивости; его рекомендуется применять в случаях, когда условия интерференции неизвестны.

Таблица 6.5.

Применение перемежения для различных процессов возникновения стираний

Тип стираний	Без перемежения	Периодическое перемежение	Случайное перемежение
Случайные	Да	Нет	Нет
Известные периодические пакеты	Да, для умеренных значений длины пакета	Используется при больших длинах пакета	Нет
Случайные пакеты	Хорошие Характеристики для малых значений длины пакета	Наилучшие характеристики	Иногда требуется память меньшей емкости, чем при периодическом
Организованные помехи	Нет	Неустойчиво для некоторых периодических пакетов стираний	Максимальная устойчивость

6.8. Вопросы и задания для самопроверки

6.1. Какие коды называются групповыми?

6.2. Чем отличаются систематические коды от несистематических?

6.3. Как можно задать линейный код?

6.4. Какие коды называются эквидистантными?

6.5. Какие операции над матрицами можно использовать для приведения порождающей матрицы к приведено-ступенчатому виду?

6.6. Как связаны порождающая и проверочная матрицы?

6.7. В чем заключается процедура кодирования линейным кодом?

6.8. Назовите и поясните сущность параметров линейных кодов.

6.9. При каких условиях помехоустойчивого кодирования возможно только обнаружение ошибок, коррекция ошибок, идентификация и коррекция ошибок?

6.10. Назовите основные методы декодирования линейных кодов.

6.11. Поясните сущность метода декодирования по максимуму правдоподобия.

6.12. Что такое мажоритарный элемент?

6.13. Что такое синдром принятого вектора?

6.14. Поясните последовательность действий декодера при синдромном декодировании.

6.15. Что такое смежный класс?

6.16. Назовите параметры кодов Хэмминга.

6.17. Какое кодовое расстояние имеют коды с проверкой на четность.

6.18. Какое кодовое расстояние имеют удлиненные коды Хэмминга?

6.19. Поясните алгоритм декодирования удлиненных кодов Хэмминга.

6.20. Как задать код Рида-Маллера первого порядка?

6.21. Какое кодовое расстояние имеют коды Рида-Маллера первого порядка?

6.22. Каким методом декодируются коды Рида-Маллера?

6.23. Какие коды называются циклическими?

6.24. Как связаны генераторный и проверочный полиномы?

6.25. Как производиться кодирование циклических кодов?

6.26. Поясните процедуру кодирования разделимым циклическим кодом.

6.27. Чем отличается кодирование по генераторному полиному от кодирования по проверочному полиному?

6.28. Как вычисляется синдром при декодировании циклических кодов?

6.29. Какие существуют методы декодирования циклических кодов?

6.30. Почему кодовые слова циклических кодов отождествляют с полиномами?

6.9. Практическое занятие №5

Линейные переключательные схемы для умножения и деления многочленов по модулю два

Теория и методы решений для практического занятия представлены в разделе 6.5.2.

Тестовые задания

1. Количество триггеров в схеме умножения произвольного полинома (множимое) на фиксированный полином (множитель) по модулю два определяется

a) количеством ненулевых коэффициентов перед степенями x в множимом;

b) количеством ненулевых коэффициентов перед степенями x в множителе;

c) степенью множимого;

d) степенью множителя.

2. Количество триггеров в схеме деления произвольного полинома (делимое) на фиксированный полином (делитель) по модулю два определяется

a) количеством ненулевых коэффициентов перед степенями x в делимом;

b) количеством ненулевых коэффициентов перед степенями x в делителе;

c) степенью делимого;

d) степенью делителя.

3. Количество двухвходовых сумматоров по модулю два в схеме умножения произвольного полинома (множимого) на фиксированный полином (множитель) по модулю два определяется

a) количеством ненулевых коэффициентов перед степенями x в множимом минус один;

b) количеством ненулевых коэффициентов перед степенями x в множителе минус один;

c) степенью множимого;

d) степенью множителя.

4. Количество двухвходовых сумматоров по модулю два в схеме деления произвольного полинома (делимое) на фиксированный полином (делитель) по модулю два определяется

a) количеством ненулевых коэффициентов перед степенями x в делимом минус один;
b) количеством ненулевых коэффициентов перед степенями x в делителе минус один;
c) степенью делимого;
d) степенью делителя.

5. Максимальная степень полинома, являющегося целой частью при делении произвольного полинома (делимое) на фиксированный полином (делитель), определяется
a) как степень делимого минус степень делителя;
b) как степень делимого минус степень делителя минус один;
c) степенью делимого;
d) степенью делителя.

6. Максимальная степень полинома, являющегося остатком при делении произвольного полинома (делимое) на фиксированный полином (делитель), определяется как
a) степень делителя минус степень делимого;
b) степень делимого минус степень делителя плюс один;
c) степень делимого минус один;
d) степень делителя минус один.

7. Если на схему умножения произвольного полинома (множимое) на фиксированный полином (множитель) по модулю два, в которой множимое подается начиная со старших разрядов, подать множимое начиная с младших разрядов, то результат будет равен
a) произведению множителя на множимое;
b) произведению множителя на полином реверсивный множимому;
c) сумме множителя и множимого;
d) разности множимого и множителя.

8. Количество тактов в схеме умножения произвольного полинома (множимое) на фиксированный полином (множитель) по модулю два, необходимое для нахождения произведения, определяется как
a) степень множимого плюс один;
b) степень множимого плюс степень множителя;
c) степень множимого плюс степень множителя плюс один;
d) степень множимого плюс степень множителя минус один.

9. Количество тактов в схеме деления произвольного полинома (делимое) на фиксированный полином (делитель) по модулю два, необходимое для нахождения целой части и остатка, определяется как

a) степень делимого плюс один;
b) степенью делимого;
c) степень делимого минус один;
d) степень делителя плюс один.

10. Где находятся целая часть и остаток в схеме деления произвольного полинома (делимое) на фиксированный полином (делитель) по модулю два?
a) целая часть выдвигается из схемы старшими разрядами вперед, остаток находится в триггерах после количества тактов равного степени делимого плюс один;
b) остаток выдвигается из схемы старшими разрядами вперед, целая часть находится в триггерах после количества тактов равного степени делимого плюс один;
c) целая часть выдвигается из схемы младшими разрядами вперед, остаток находится в триггерах после количества тактов равного степени делимого;
d) остаток выдвигается из схемы младшими разрядами вперед, целая часть находится в триггерах после количества тактов равного степени делимого.

Задачи

1. Постройте схему, реализующую задержку входного вектора на следующее количество тактов:

a) 2; b) 3; c) 5; d) 7.

2. Найдите результат умножения двух полиномов:
a) $1+x+x^3$ и $1+x^2+x^3$;
b) $1+x^2+x^3$ и $1+x+x^3+x^4$;
c) $1+x+x^5$ и $1+x$;
d) $1+x^2+x^3+x^7+x^8$ и $1+x^2+x^5+x^7+x^8$.

3. Найдите частное и остаток от деления полиномов:
a) $1+x+x^7$ и $1+x^2+x^3$;
b) $1+x^6+x^8$ и $1+x+x^3+x^4$;
c) $1+x^3+x^9$ и $1+x$;
d) $1+x^2+x^4+x^5+x^8$ и $1+x^2+x^5$.

4. Постройте схему (со встроенными сумматорами), реализующую умножение по модулю два любого полинома (подаваемого старшими разрядами вперед) на заданный полином:
a) $1+x+x^3$; b) $1+x^2+x^3$; c) $1+x+x^5$; d) $1+x^2+x^3+x^7+x^8$.

4. Постройте схему (со встроенными сумматорами), реализующую умножение по модулю два любого полинома (подаваемого младшими разрядами вперед) на заданный полином:

a) $1+x+x^3+x^4$;

b) $1+x^2+x^5$;

c) $1+x^5+x^7$;

d) $1+x\ +x^2+x^7+x^9$.

5. Постройте схему (с вынесенными сумматорами), реализующую умножение по модулю два любого полинома (подаваемого старшими разрядами вперед) на заданный полином:

a) $1+x^2+x^3+x^4$;

b) $1+x+x^2+x^5$;

c) $1+x^5+x^6+x^7$;

d) $1+x\ +x^2+x^6+x^7+x^9$.

6.Постройте схему (с вынесенными сумматорами), реализующую умножение по модулю два любого полинома (подаваемого младшими разрядами вперед) на заданный полином:

a) $1+x+x^3+x^4$; b) $1+x^2+x^5$; c)$1+x^5+x^7$; d) $1+x\ +x^2+x^7+x^9$.

7. Постройте схему (с вынесенными сумматорами), реализующую деление по модулю два любого полинома (подаваемого старшими разрядами вперед) на заданный полином:

a) $1+x+x^3+x^4$; b) $1+x^2+x^5$; c) $1+x^5+x^7$; d) $1+x\ +x^2+x^7+x^9$.

8. Постройте схему (со встроенными сумматорами), реализующую деление по модулю два любого полинома (подаваемого старшими разрядами вперед) на заданный полином:

a) $1+x+x^3+x^4+x^5$;

b) $1+x^2+x^3+x^4+x^5$;

c) $1+x^2+x^5+x^6+x^7$;

d) $1+x\ +x^2+x^7+x^8+x^9$.

9. Найдите количество тактов необходимое для умножения следующих пар полиномов:

a) $1+x+x^6$ и $1+x^2+x^4$;

b) $1+x^6+x^7$ и $1+x+x^3+x^4+x^5$;

c) $1+x^3+x^4$ и $1+x^2$;

d) $1+x^2+x^8$ и $1+x^2+x^3$.

10. Найдите количество тактов необходимое для деления следующих пар полиномов:

a) $1+x+x^7$ и $1+x^2+x^4$;

b) $1+x^6+x^7+x^{11}$ и $1+x+x^3+x^4+x^5$;

c) $1+x^3+x^{40}$ и $1+x^{21}$;

d) $1+x^2+x^{83}$ и $1+x^2+x^{31}$.

6.10. Практическое занятие № 6

Задание, кодирование и декодирование циклических кодов

Теория и методы решений для практического занятия представлены в разделах 6.5., 6.6.

Тестовые задания

1. Для того чтобы полином $g(x)$ был генераторным полиномом циклического кода, необходимо, чтобы он был

a) неприводимым;

b) примитивным;

c) делителем полинома вида x^n+1, где n – длина кода;

d) приводимым.

2. Степень генераторного полинома определяет

a) кодовое расстояние кода;

b) количество информационных разрядов;

c) количество проверочных разрядов;

d) длину кода.

3. Степень проверочного полинома определяет

a) длину кода;

b) количество информационных разрядов;

c) количество проверочных разрядов;

d) кодовое расстояние кода.

4. Генераторный полином $g(x)$ и проверочный полином $h(x)$ связаны друг с другом следующим образом:

a) никак не связаны;

b) $g(x)\cdot h(x)=x^n+1$;

c) $g(x)+h(x)=1$;

d) $g(x)\cdot h(x)=0$.

5. Умножение по модулю два информационного полинома на генераторный полином $g(x)$ эквивалентно

a) умножению информационного вектора на порождающую матрицу *G*;
b) умножению информационного вектора на проверочную матрицу *H*;
c) умножению кодового слова на проверочную матрицу *H*;
d) умножению кодового слова на порождающую матрицу *G*.

6. Генераторный полином $g(x)=1+x^2+x^3$ задает циклический код с параметрами:
a) $n=15$, $k=4$, $r=11$, $d=3$;
b) $n=7$, $k=4$, $r=3$, $d=3$;
c) $n=15$, $k=11$, $r=3$, $d=3$;
d) $n=7$, $k=3$, $r=4$, $d=3$.

7. Мощность кода определяется
a) количеством всех кодовых слов;
b) количество всех кодовых слов умножить на длину кода;
c) количеством всех исправляемых ошибок;
d) средним весом всех кодовых слов.

8. Скорость (n,k) кода равна

a) k/n; b) n/k; c) n/r; d) r/k.

9. Генераторный полином $g(x)=1+x+x^4$ задает циклический код с параметрами:
a) $n=15$, $k=4$, $r=11$, $d=5$;
b) $n=7$, $k=4$, $r=3$, $d=3$;
c) $n=15$, $k=11$, $r=3$, $d=3$;
d) $n=7$, $k=3$, $r=4$, $d=3$.

10. Сколько требуется хранить в ПЗУ синдромных многочленов, чтобы было возможно исправление всех ошибок циклическим кодом (15;11)?

a) 15; b) 11; c) 30; d) 26.

Задачи

1. Определите параметры циклического кода, задаваемого генераторным полиномом $g(x)$:

a) $1+x+x^4$; b) $1+x+x^3$; c) $1+x^2+x^3$; d) $1+x^3+x^4$.

2. Закодируйте неразделимым циклическим кодом $g(x)=1+x+x^3$ следующие информационные полиномы:

a) $1+x+x^2$; b) $1+x$; c) $1+x^2$; d) $1+x^3$.

3. Закодируйте разделимым циклическим кодом $g(x)=1+x^2+x^3$ следующие информационные полиномы:

a) $x+x^3$; b) 1; c) x^2; d) 1.

4. Постройте таблицу синдромов корректируемых ошибок циклического кода, задаваемого генераторным полиномом $g(x)$:

a) $1+x^3+x^4$; b) $1+x+x^3$; c) $1+x+x^4$; d) $1+x^2+x^3$.

5. Постройте кодер неразделимого циклического кода задаваемого генераторным полиномом $g(x)$:

a) $1+x^3+x^4$; b) $1+x+x^3$; c) $1+x+x^4$; d) $1+x^2+x^3$.

6. Постройте кодер разделимого циклического кода, задаваемого генераторным полиномом $g(x)$:

a) $1+x^3+x^4$; b) $1+x+x^3$;c) $1+x+x^4$; d) $1+x^2+x^3$.

7. Постройте структурную схему синдромного декодера циклического кода задаваемого генераторным полиномом $g(x)$:

a) $1+x^3+x^4$; b) $1+x+x^3$; c) $1+x+x^4$; d) $1+x^2+x^3$.

8. Постройте структурную схему мажоритарного декодера циклического кода, задаваемого генераторным полиномом $g(x)$:

a) $1+x^3+x^4$; b) $1+x+x^3$; c) $1+x+x^4$; d) $1+x+x^4$.

9. Постройте все кодовые слова разделимого и неразделимого циклического кода, задаваемого генераторным полиномом $g(x)=1+x+x^3$.

10. Декодируйте сообщения синдромным методом для кода, задаваемого генераторным полиномом $g(x)=1+x+x^3$:

a) $1+x^3+x^5+x^6$;

b) $1+x^3+x^4+x^5$;

c) $1+x+x^4+x^5$;

d) $1+x+x^6$.

Модуль 7

СВЕРТОЧНЫЕ КОДЫ

Цель модуля – изучение способов представления и описания сверточных кодов, основных алгоритмов кодирования и декодирования информации сверточными кодами.

В результате изучения модуля студенты должны:

- знать описание сверточных кодов с помощью многочленов и матриц;
- знать правила кодирования на основе порождающих матриц и кодовых деревьев; уметь кодировать информацию такими алгоритмами;
- знать алгоритм Витерби для декодирования сверточных кодов;
- иметь представление о кодах Вайнера – Эша.

Содержание модуля

В случае использования блоковых кодов поток данных делится на блоки по k информационных символов, и каждый блок кодируется n символами кодового слова. Кодовые слова для последовательных k-символьных блоков никоим образом не связываются кодером.

Однако возможна другая схема кодирования, при которой поток данных разбивается на гораздо меньшие блоки длины k_0, которые называют *кадрами информационных символов*, которые включают лишь несколько символов. Кадры информационных символов кодируются кадрами кодового слова длины n_0 каждый. Однако вместо того, чтобы независимо кодировать отдельные кадры информационных символов в отдельные кадры кодового слова, кодирование каждого кадра информационных символов в отдельный кадр кодового слова производится с учетом предыдущих m кадров информационных символов. Поэтому процедура кодирования связывает между собой последовательные кадры кодовых слов.

Коды, получаемые таким образом, называются древовидными кодами. Сверточными кодами являются древовидные коды, которые обладают дополнительными свойствами линейности, и постоянства во времени.

7.1. Структура и описание сверточных кодов

7.1.1. Основные понятия и параметры, классификация древовидных кодов

Сверточный кодер обычно представляется в виде регистра сдвига, и многие основные определения могут быть введены с использованием такой схемы. Рассмотрим кодер, представленный на рис. 7.1. Информационная последовательность вводится в кодер, начиная с нулевого момента времени и до бесконечности. Поток входящих информационных символов разбивается на сегменты, которые содержат no k_0 символов и называются *кадрами информационных символов*. Кадр информационных символов может, в частности, состоять из единственного символа, что нередко имеет место на практике. В кодере может храниться m кадров. В течение каждого временного кадра в регистр сдвига вводится новый кадр информационных символов, а кадр информационных символов, дольше остальных хранившийся в нем, выводится из него и сбрасывается. В конце каждого временного кадра в кодере хранятся последние m из поступивших в него кадров (всего mk_0 информационных символов). В начале каждого временного кадра кодер по введенному кадру информационных символов и m хранящимся в нем кадрам вычисляет один кадр кодового слова, имеющий длину n_0 символов. Этот кадр кодового слова выводится из кодера, как только поступает следующий кадр информационных символов. Следовательно, каждым k_0 информационным символам соответствует передача по каналу n_0 кодовых символов.

Бесконечное множество всех бесконечно длинных кодовых слов, получаемых при поступлении в этот кодер всех возможных входных последовательностей, называется древовидным (n_0, k_0)-кодом.

Формальное определение древовидного кода. *Древовидный* (n_0, k_0)-*код* – это такое отображение на себя множества полубесконечных последовательностей элементов из $GF(q)$, что если для любого M первые Mk_0 компонент двух полубесконечных последовательностей совпадают, то первые Mn_0 компонент отображений этих последовательностей тоже совпадают. Древовидный код лучше всего можно представить себе, обратившись к кодеру, изображенному на рис. 7.1.

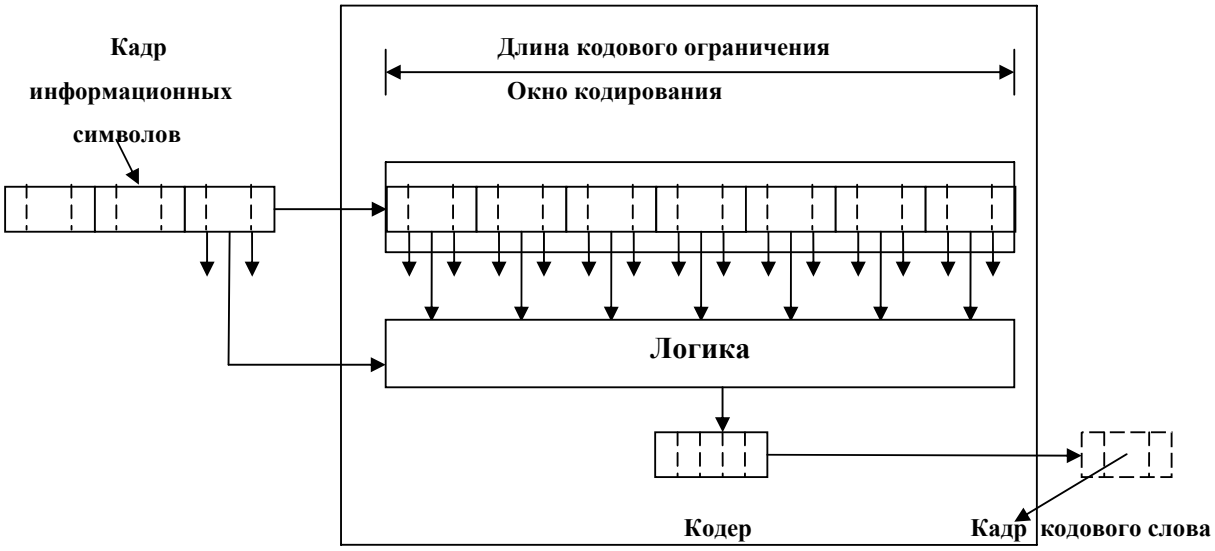


Рис. 7.1. Сверточный кодер

Частные случаи древовидных кодов получаются различными комбинациями следующих четырех свойств.

Конечность длины кодового ограничения. Длина кодового ограничения – $v=mk_0$ может быть конечной или бесконечной. Практически древовидные коды всегда имеют конечную длину кодового ограничения. Однако в теоретических исследованиях иногда полезны коды с бесконечной длиной кодового ограничения. Древовидный (n_0, k_0)-код с конечной длиной кодового ограничения v, длиной слова $k=v+k_0$ и кодовой длиной блока n называется также *решетчатым* (n, k)-*кодом.*

Постоянство во времени. Если две различные входные последовательности совпадают во всем, но с временным сдвигом на целое число кадров, то соответствующие им кодовые последовательности также совпадают во всем, но с временным сдвигом на то же самое целое число кадров.

*Линейность***.** Кодовая последовательность любой линейной комбинации двух информационных последовательностей совпадает с такой же линейной комбинацией кодовых последовательностей этих двух информационных последовательностей. Иначе говоря, если d_1, и d_2 являются двумя информационными последовательностями с кодовыми последовательностями $G(d_1)$ и $G(d_2)$, то $ad_1 + bd_2$ соответствует кодовая последовательность

$$G\,(ad_1 + bd_2) = aG\,(d_1) + bG\,(d_2).$$

Систематичность. *Систематическим древовидным кодом* называется код, в котором каждый кадр информационных символов составляет первые k_0

символов первого из тех кадров кодовой последовательности, на которые влияет данный кадр информационных символов.

Линейный постоянный во времени древовидный (n_0, k_0)-код, имеющий конечную длину слова k=(n + 1) k_0, называется *сверточным (n, k)-кодом*. Сверточный (n, k)-код, удовлетворяющий условию систематичности, называется *систематическим сверточным (n, k)-кодом*. Известно, что можно называть один и тот же код древовидным (n_0, k_0)-кодом или сверточным (n, k)-кодом. На практике k значительно больше k_0, и поэтому недоразумений не возникает. Постоянный во времени древовидный (n_0, k_0)-код, имеющий конечную информационную длину слова k, называется скользящим блоковым (n, k)-кодом. Следовательно, линейный скользящий блоковый код является сверточным кодом.

На рис. 7.2 графически показаны связи между различными классами древовидных кодов.

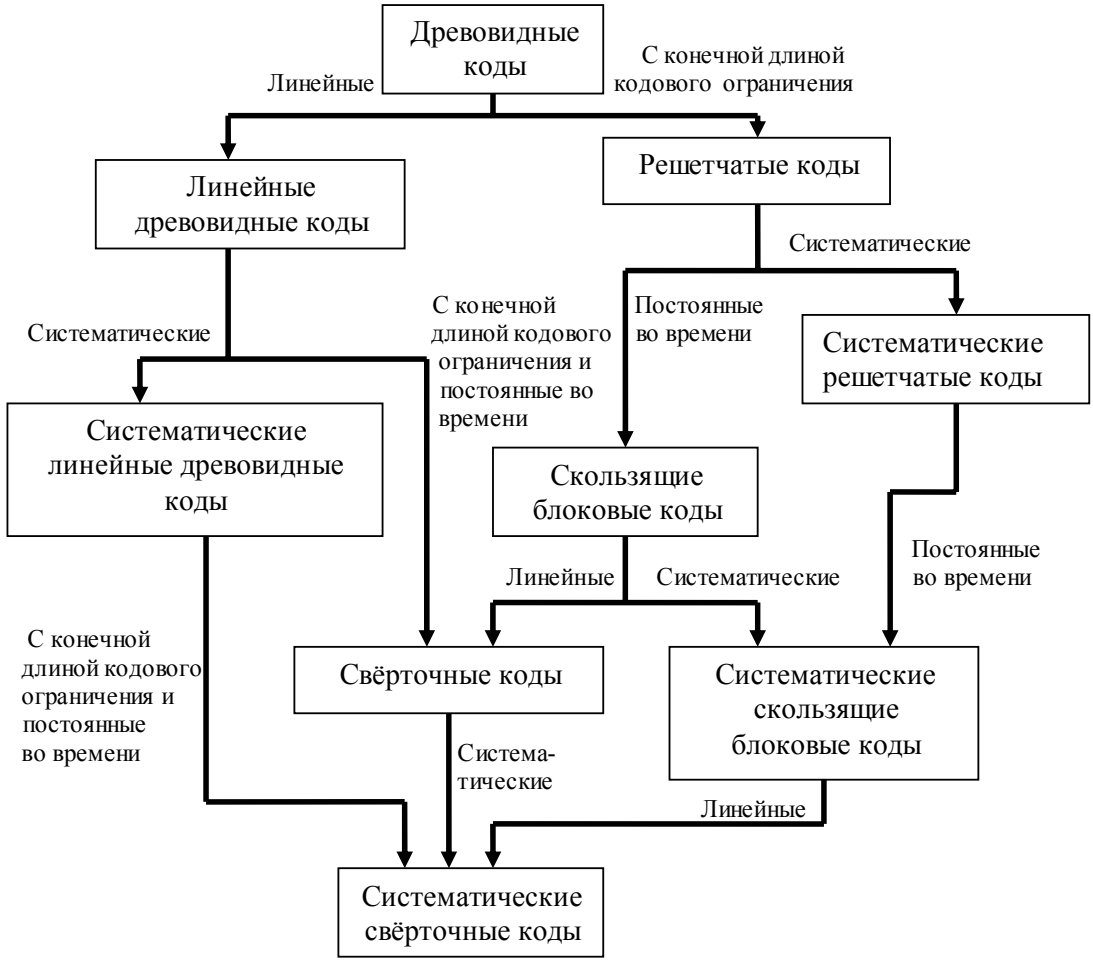


Рис. 7.2. Классификация древовидных кодов

Основными характеристиками сверточных кодов являются:

- k_0 – размер кадра информационных символов;
- n_0 – размер кадра кодовых символов;
- m – длина памяти кода;
- $k = (m+1) \cdot k_0$ – информационная длина слова;
- $n = (m+1) \cdot n_0$ – кодовая длина блока – это длина кодового слова, на которой сохраняется влияние одного кадра информационных символов;
- $R = k/n$ – скорость, которая характеризует степень избыточности кода, вводимой для обеспечения справляющих свойств кода.

На рис. 7.1. изображен кодер, у которого k_0=3, n_0=5, v=21, n=40. Кодовая длина блока – это длина кодового слова, на которой сохраняется влияние одного кадра информационных символов. Из соображений удобства реализации на практике значения n_0 и k_0 для древовидных кодов выбираются равными небольшим целым числам; в типичном случае k_0 равно единице. Это означает, что выбор скорости кода ограничен. Невозможно построить практический древовидный код со скоростью, очень близкой к единице.

7.1.2. Описание сверточных кодов с помощью многочленов и матриц

Сверточный $((m+1)n_0, (m+1)k_0)$-код над $GF(q)$ с длиной кодового ограничения $v=mk_0$ можно генерировать с помощью наборов фильтров с конечной импульсной характеристикой (КИХ-фильтров); каждый набор состоит из k_0 КИХ-фильтров над $GF(q)$. Поэтому последовательность на выходе кодера можно рассматривать как свертку импульсной характеристики кодера с входной последовательностью. Импульсная переходная характеристика фильтра (ИПХ) (а кодер сверточного кода, по сути дела, является фильтром) есть реакция на единичное воздействие вида $\overline{\delta}$= (10000.....). Рассмотрим примеры кодирования последовательностей с использованием импульсной характеристики эквивалентного фильтра.

Пусть m = (110 ... , тогда для кодера с импульсной переходной характеристикой H = (11.00.00.01.00.00....

U = 11.00.00.01.00.00…

11.00.00.01.00…

U = 11.11.00.01.01.00.00… ,

m = (1011000…)

$$
\begin{array}{r}
U = 11.00.00.01.00.00.00\ldots \\
11.00.00.01.00\ldots \\
11.00.00.01\ldots \\
\hline
U = 11.00.11.10.00.01.01.00\ldots\ .
\end{array}
$$

На рис. 7.3. показан кодер для двоичного сверточного кода с n_0=5 и k_0=1. В кодер поступает поток символов со скоростью k_0 символов в единицу времени, а из него выходит в канал поток символов со скоростью n_0 символов в единицу времени.

Такой кодер состоит из серии фильтров и выходного регулирующего буфера, который необходим для согласования выходной скорости со скоростью фильтров.

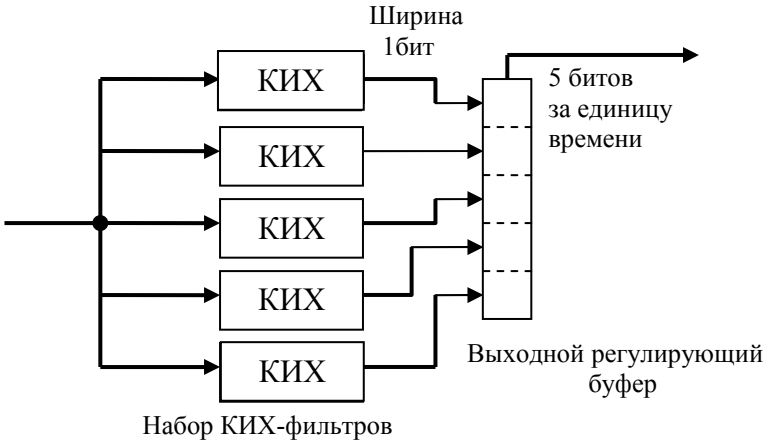


Рис. 7.3. Свёрточный кодер

Каждый КИХ-фильтр может быть представлен многочленом степени не выше m. Если входной поток записать как многочлен (быть может, бесконечной длины), то работа фильтра может быть описана как умножение многочленов. В этом случае кодер сверточного кода может быть представлен множеством многочленов; поэтому и сам код может быть представлен посредством того же множества многочленов. Иначе говоря, код является множеством кодовых слов, которое порождается данным множеством многочленов. Эти многочлены называются *порождающими многочленами кода***.** Наибольшая степень порождающих многочленов равна m.

В отличие от блоковых кодов, которые описываются единственным порождающим многочленом, сверточный код требует для своего описания нескольких порождающих многочленов – в общем случае $k_0 n_0$ многочленов. Сверточный код состоит из бесконечного числа кодовых слов бесконечной длины. Он линеен и может быть описан бесконечной порождающей кодо-

вой матрицей. Для описания каждого может быть использовано огромное количество порождающих матриц, но удобно оперировать только некоторыми из них. Даже в лучшем случае порождающая матрица сверточного кода более громоздка, чем порождающая матрица блокового кода. Пусть g_{ij} (x) – множество порождающих многочленов, i= 1, ... k_0, j= 1, ..., n_0. Они могут быть объединены в матрицу размера $k_0 \times n_0$, называемую *порождающей матрицей* из многочленов

$$G(x)=[g_{ij}(x)].$$

При k_0, большем единицы, некоторые порождающие многочлены могут равняться нулю.

Порождающая матрица сверточного кода, усеченного до блокового кода длины n, записывается в виде

$$G_n = \begin{bmatrix} G_0 & G_1 & G_2 & ... & G_m \\ 0 & G_0 & G_1 & ... & G_{m-1} \\ 0 & 0 & G_0 & ... & G_{m-2} \\ . & & . & & \\ . & & . & & \\ . & & . & & \\ 0 & 0 & 0 & & G_0 \end{bmatrix},$$

где символом 0 обозначена $(k_0 \times n_0)$-матрица, целиком состоящая их нулей. Порождающей матрицей сверточного кода является матрица

$$G = \begin{bmatrix} G_0 & G_1 & G_2 & ... & G_m & 0 & 0 & 0 & 0 & ... \\ 0 & G_0 & G_1 & ... & G_{m-1} & G_m & 0 & 0 & 0 & ... \\ 0 & 0 & G_0 & ... & G_{m-2} & G_{m-1} & G_m & 0 & 0 & ... \\ . & & & & & & & & & \\ . & & & & & & & & & \\ . & & & & & & & & & \end{bmatrix},$$

бесконечно продолжаясь вниз и вправо. За исключением диагональной полосы, состоящей из m нулевых подматриц, все ее подматрицы являются нулевыми.

Рассмотрим входной кадр как k_0 параллельно поступающих символов, а последовательность входных кадров как k_0 параллельных последовательностей символов. Они могут быть представлены k_0 информационными многочленами $d_i(x)$, i=1, …, k_0 или вектором-строкой таких многочленов

$$d(x) = [d_1(x), d_2(x), ..., d_{k0}(x)].$$

Аналогично выходное кодовое слово может быть представлено n_0 многочленами кодового слова $c_j(x)$, j=1,..., n_0 или вектором этих многочленов

$$c(x) = [c_j(x)] = [c_1(x), c_2(x), ..., c_{n0}(x)].$$

Коэффициенты многочленов кодового слова перемежаются в порядке их прохождения по каналу. Теперь операцию кодирования можно компактно описать с помощью векторно-матричного произведения

$$c(x) = d(x)G(x),$$

Проверочная матрица $H(x)$ из многочленов является $[(n_0–k_0)\times n_0]$-матрицей, элементами которой являются многочлены и которая удовлетворяет условию

$$G(x)H^T(x) = 0.$$

Проверочной матрицей является любая матрица, H удовлетворяющая условиям

$$G^{(\ln_0)}(H^{(\ln_0)})^T = 0,\ l = 0,\ 1,\ 2...$$

где $G^{(\ln_0)}$ и $H^{(\ln_0)}$ – стоящие в левых верхних углах матриц G и H подматрицы, соответствующие l кадрам. Бесконечная проверочная матрица может быть построена по порождающей. В качестве примера рассмотрим систематический двоичный сверточный (4,2) код с k_0=1, m=1, P_0=1 и P_1=1 размера 1×1. Следовательно:

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & & & & & \\ & & 1 & 1 & 0 & 1 & & & \\ & & & 1 & 1 & 0 & 1 & & \\ & & & & & \dots & \dots & \dots & \\ & & & & & & & \dots & \dots \end{bmatrix}. \quad H = \begin{bmatrix} 1 & 1 & & & & & \\ 1 & 0 & 1 & 1 & & & \\ & & 1 & 0 & 1 & 1 & \\ & & & & \dots & \dots & \\ & & & & & \dots & \dots & \dots \end{bmatrix}.$$

7.2. Кодирование и декодирование сверточными кодами

7.2.1. Понятие решетчатой диаграммы и кодового дерева

Сверточные и другие решетчатые коды удобно описывать специальным графом, называемым решёткой. *Решеткой* называется граф, узлы которого находятся в прямоугольной координатной сетке, полубесконечной

справа; число узлов в каждом столбце конечно. Конфигурация ребер, соединяющих узлы каждого столбца с узлами столбца справа, одинакова для всех столбцов. Узлы, которые не могут быть достигнуты при движении вправо из верхнего левого узла, обычно не указываются. Типичная решетка для двоичного кодового алфавита представлена на рис. 7.4. Ее маркировка соответствует кодеру изображенному на рис. 7.5. Каждый последующий столбец представляет собой набор состояний в следующий момент времени.

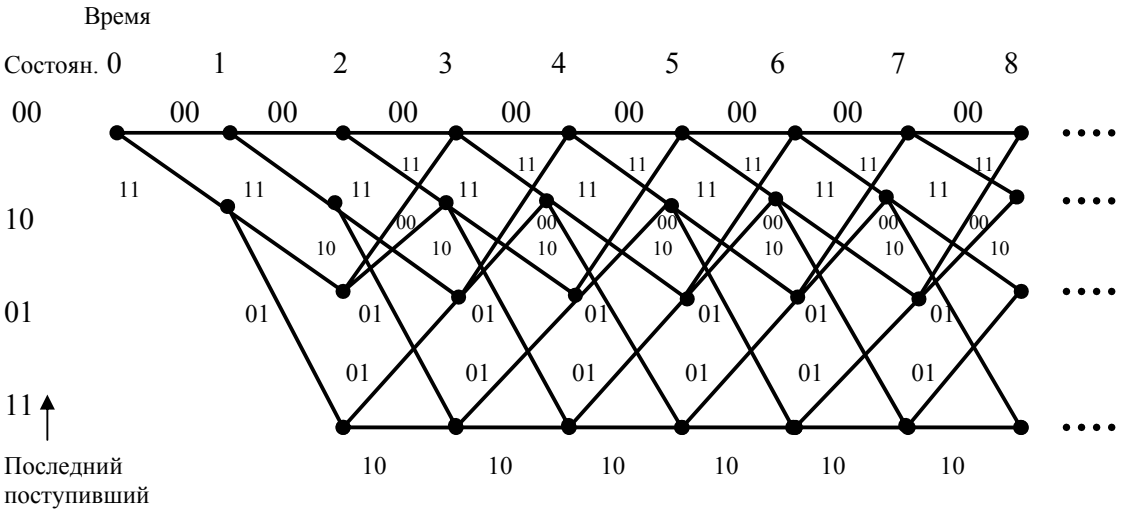


Рис. 7.4. Решетчатая диаграмма свёрточного (6, 3)-кода

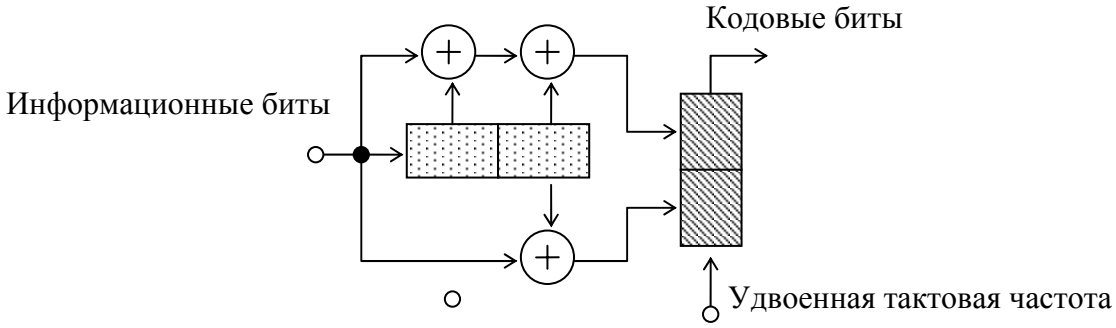


Рис. 7.5. Кодер для двоичного свёрточного (6, 3)-кода

Поступление на вход нового кадра приводит к изменению состояния регистра сдвига, соответствующему ребру, которое ведёт к следующему узлу. В примере каждое ребро помечено двумя двоичными символами, передаваемыми в канал при переходе в следующее состояние регистра сдви-

га, а ведущая из произвольного узла верхняя прямая соответствует нулевому входному двоичному символу, а нижняя – единичному. В общем же случае каждое ребро помечается k_0 входными символами, которые ему соответствуют.

Маркированная решетка описывает сверточный код в том смысле, что все пути слева направо по решетке обозначают кодовые снова. Маркировка ребер одинакова для каждого сегмента и линейна в том смысле, что линейная комбинация маркировок любого множества ребер является маркировкой некоторого ребра.

Решетка может быть маркирована и при более слабых ограничениях. Если маркировка не обладает свойством линейности, то код называется *скользящим блоковым кодом*. Если маркировка меняется от кадра к кадру, то такой код известен под общим названием *решетчатого кода*. Наконец, если число состояний в следующих друг за другом временных кадрах продолжает неограниченно расти, то такой код называется *общим древовидным кодом.*

Решетчатый код может быть также представлен особым графом, называемым *деревом.* Каждый узел дерева – это состояние, соответствующее всем поступившим в него информационным символам, начиная с нулевого момента времени. Для кодов с бесконечной длиной кодового ограничения или даже умеренно большой длиной кодового ограничения дерево – это именно тот граф, который их описывает. Кодовые слова соответствуют путям по дереву. Для кодов с малой длиной кодового ограничения удобнее использовать решетку. Код полностью описывается последними v поступившими в кодер символами, и их достаточно для определения состояния.

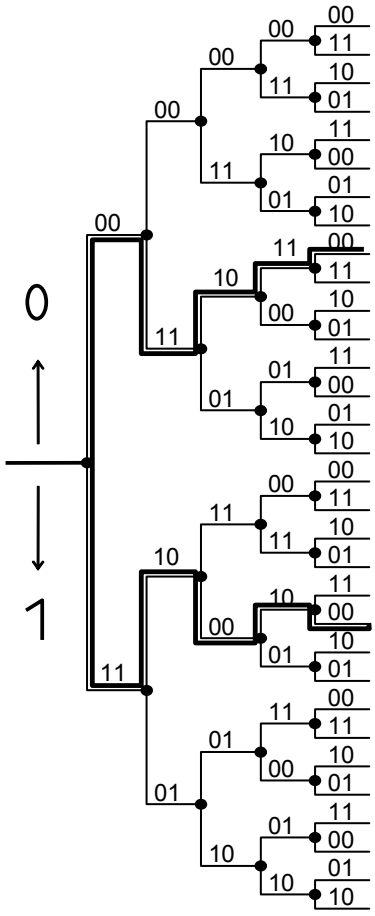


Рис.7.6. Кодовое дерево свёрточного (6, 3)-кода

На рис. 7.6. представлено кодовое дерево для кодера, показанного на рис. 7.5. Кодирование состоит в движении вправо по кодовому дереву, если входной символ 0-– движение вверх по дереву, 1 – вниз. Например, входная по-

следовательность m=(01000……) кодируется как U=(0011101100000…), последовательность m = (1010100000…) – как U=(1110001000…). Анализ строения приведенного на рис. 7.6 кодового дерева, показывает, что начиная с четвертого ребра его структура повторяется, т.е. каким бы ни был первый шаг, начиная с четвертого ребра значения выходных символов кодера повторяются. Одинаковые же узлы могут быть объединены, и тогда начиная с некоторого сечения размер кодового дерева перестанет увеличиваться. Другими словами, выходная кодовая последовательность в определенный момент перестает зависеть от значений входных символов, введенных в кодер ранее. Действительно, когда третий входной символ вводится в кодер, первый входной символ покидает сдвиговый регистр и не сможет в дальнейшем оказывать влияния на выходные символы кодера. С учетом этого неограниченное кодовое дерево (см. рис. 7.6) переходит в ограниченную решетчатую диаграмму (см. рис. 7.4).

Правило кодирования сверточными кодами с использованием решетчатой диаграммы: очередные символы входной последовательности определяют направление движения из узлов решетки: если 0, то идем по верхнему ребру, если 1 – по нижнему ребру. Диаграмма не разрастается по ширине с каждым шагом, а имеет, начиная с третьего сечения, постоянную ширину.

В качестве примера закодируем с помощью решетчатой диаграммы несколько информационных последовательностей. Пусть m=(0110000…). Соответствующая ей кодовая последовательность будет иметь вид: U = (001101011100…). Для m=(110100…) – U=(1101010010110000…).

7.2.2. Коды Вайнера – Эша

Класс двоичных сверточных кодов, исправляющих одну ошибку и называемых кодами Вайнера-Эша, аналогичен классу кодов Хэмминга. Для каждого положительного целого m существует $((m+1)2^m,((m+1)(2^m-1))$ – код Вайнера – Эша. Такой код определяется проверочной матрицей $H'(2^m-1,2^m-1-m)$ – кода Хэмминга. Это проверочная $\left[m\times(2^m-1)\right]$-матрица, в которой все (2^m-1) столбцов различны и ненулевые. Выберем такую матрицу, используем ее строки для определения множества $\left[1\times(2^m-1)\right]$-матриц $P_1^T,...,P_m^T$ и обозначим через P_0^T вектор-строку, все 2^m-1 элементов которой равны единице. Тогда проверочная матрица кода Вайнера-Эйша запишется в виде

$$H = \begin{bmatrix} P_0^T & 1 & 0 & 0 & 0 & 0 & 0 & . & . & . \\ P_1^T & 0 & P_0^T & 1 & 0 & 0 & 0 & & & \\ P_2^T & 0 & P_1^T & 0 & P_0^T & 1 & 0 & & & \\ . & & . & & P_1^T & & & & & \\ . & & . & & & & & & & \\ . & & . & & & & & & & \\ P_m^T & 0 & & & & & & & & \\ 0 & 0 & P_m^T & 0 & & & & & & \\ . & . & & & & & & & & \\ . & . & & & & & & & & \\ . & . & & & & & & & & \end{bmatrix}$$

$$H^{((m+1)2m)} = \begin{bmatrix} P_0^T & 1 & 0 & 0 & 0 & 0 & & & \\ P_1^T & 0 & P_0^T & 1 & 0 & 0 & & & \\ P_2^T & 0 & P_1^T & 0 & P_0^T & 1 & & & \\ . & & & & & & & & \\ . & & & & & & & & \\ . & & & & & & & & \\ P_m^T & P_{m-1}^T & 0 & P_{m-2}^T & 0 & P_{m-3}^T & \ldots & P_0^T & 1 \end{bmatrix},$$

Минимальное расстояние d^* кода Вайнера – Эша равно 3; таким образом, он является сверточным кодом, исправляющим одну ошибку. Например, $(12,9)$ – код Вайнера – Эша соответствует $m = 2$. Его проверочная матрица равна H и при таком усечении кода, чтобы его проверочная матрица соответствовала длине блока 12, получается проверочная матрица $H^{(12)}$:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & & & & & & & & & & & & \\ 1 & 1 & 0 & 0 & & 1 & 1 & 1 & 1 & & & & & & & \\ 1 & 0 & 1 & 0 & & 1 & 1 & 0 & 0 & & 1 & 1 & 1 & 1 & & \\ 0 & 0 & 0 & 0 & & 1 & 0 & 1 & 0 & & 1 & 1 & 0 & 0 & & 1 & 1 & 1 & 1 & \\ 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 & & 1 & 0 & 1 & 0 & & 1 & 1 & 0 & 0 & \ldots \\ & . & & & & & & . & & & & & & . & & & & & \\ & . & & & & & & . & & & & & & . & & & & & \\ & . & & & & & & . & & & & & & . & & & & & \end{bmatrix}.$$

$$H^{(12)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Порождающая матрица $(12,9)$-кода Вайнера – Эша равна

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & & & & & & & & & & \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & & & & & & & & & \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & & & & & \\ & & & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & & & & & & \\ & & & & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & & & & & \\ & & & & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & & & & & \\ & & & & & & & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & & \\ & & & & & & & & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \ldots \\ & & & & & & & & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & \\ & & & & & & & & \vdots & & & & & & & & & & & & & \end{bmatrix}$$

а проверочная матрица кода, усеченного до длины блока 12, представляется в виде

$$G^{(12)} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Непосредственным рассмотрением $G^{(12)}$ убеждаемся, что в пределах блока длины 12 каждое ненулевое слово имеет вес не менее трех. Следовательно, в блоке 12 код может исправлять одну ошибку.

Кодер для $(12,9)$ – кода Вайнера – Эша представлен на рис.7.7. Каждому порождающему многочлену на схеме соответствует отдельно КИХ-

фильтр. Большинство известных наиболее употребительных сверточных кодов было получено с помощью ЭВМ.

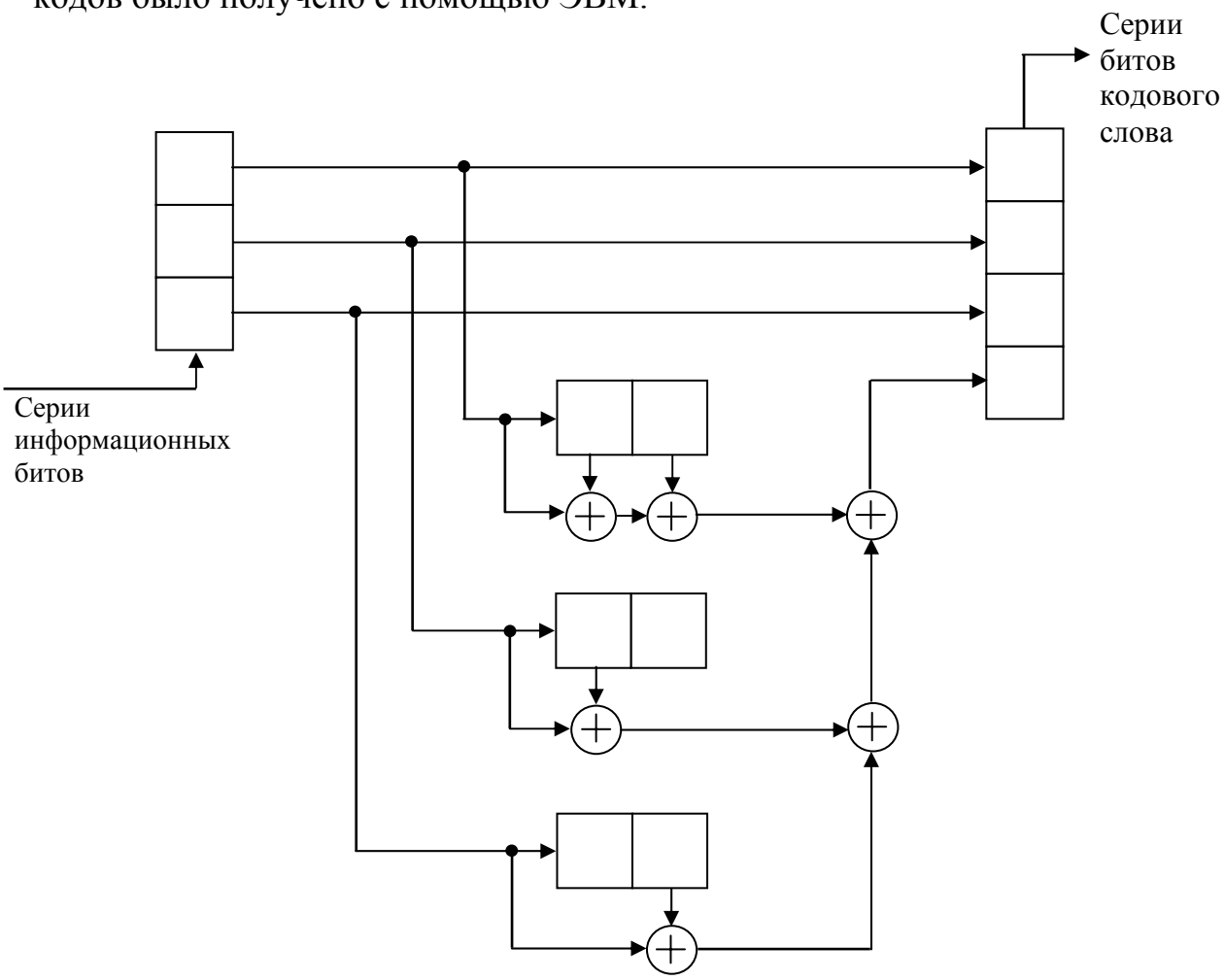


Рис. 7.7. Кодер (12, 9) – кода Вайнера – Эша

7.2.3. Декодирование сверточных кодов

Изучение процедур декодирования сверточных кодов в большинстве случаев ограничивается вопросом исправления ошибок в первом кадре. Если этот кадр может быть исправлен и декодирован, то первый информационный кадр известен. Влияние информационных символов этого кадра на последующие кадры кодовых слов может быть учтено и исключено из них. Следовательно, задача декодирования второго кадра кодового слова аналогична задаче декодирования первого кадра кодового слова.

При успешном исправлении первых j кадров проблема декодирования $(j + 1)$-го кадра аналогична проблеме декодирования первого кадра. Известно много подобных процедур декодирования. Процедура, исполь-

зующая информационные символы исправленного кадра для явного исключения их влияния на последующие кадры, называется *процедурой с обратной связью*. Другие декодеры оперируют таким образом, чтобы соответствующим образом декодированные предшествующие кадры не оказывали никакого влияния на текущий кадр.

В любом декодере может случиться так, что в связи со слишком большим количеством ошибок первый кадр кодового слова не будет исправлен должным образом. В некоторых декодерах это приводит к введению ошибок в последующие кадры, вызывая их неправильное кодирование. Если ошибка в декодировании одного кадра приводит к появлению в кодовом слове бесконечного числа дополнительных ошибок, то говорят, что в декодере происходит распространение ошибок. Если распространение ошибок может быть устранено выбором алгоритма декодирования, это явление называют *обычным распространением ошибок*; если же распространение ошибок вызывается выбором катастрофического порождающего многочлена сверточного кода, то говорят о *катастрофическом распространении ошибок*. Выбор надлежащей конструкции системы позволяет избежать обеих этих возможностей.

Число символов, которые декодер может хранить в памяти, называется *шириной окна декодирования*. Если ставить своей целью обнаружить как можно больше конфигураций ошибок, то в общем случае увеличение ширины окна декодирования всегда приводит к улучшению характеристик, однако, в конце концов происходит насыщение. Ширина окна декодирования должна быть не меньше длины блока n и зачастую в несколько раз превышает последнюю.

Минимальное расстояние Хэмминга для любых начальных сегментов длины l кадров всех пар кодовых слов, отличающихся начальным кадром, называется *l-м минимальным расстоянием сверточного кода* и обозначается через d^*_l. Если l равно m+1, то оно называется просто *минимальным расстоянием* и обозначается через d*. Последовательность d^*_1, d^*_2, d^*_3, ... называется *дистанционным профилем сверточного кода*.

Т.к. сверточный код линеен, одно из двух кодовых слов может целиком состоять из нулей. В этом случае l-е минимальное расстояние равно минимальному из всех весов сегментов длины l кадров кодовых слов с ненулевым первым кадром. Предположим, что сверточный код имеет l-е минимальное расстояние d_l. Если в первых l кадрах произошло не более t ошибок, причем t удовлетворяет неравенству

$$2t+1 \le d^*_1,$$

то ошибки, которые появились в первом кадре кодового слова, могут быть исправлены. В частности, выберем $l = m + 1$; минимальное расстояние кода d^* равно d^*_{m+1}. Тогда при t, удовлетворяющем неравенству

$$2t+1 \le d^*$$

код исправит первый кадр кодового слова, если на длине первого блока появилось не более t ошибок. Такой код называется сверточным кодом, исправляющим t ошибок.

Рассмотрим работу алгоритма декодирования Витерби. Предположим, на вход декодера поступил сегмент последовательности r длиной b, превышающей кодовую длину блока n. Назовем b окном декодирования. Сравним все кодовые слова данного кода (в пределах сегмента длиной b) с принятым словом и выберем кодовое слово, ближайшее к принятому. Первый информационный кадр выбранного кодового слова примем в качестве оценки информационного кадра декодированного слова. После этого в декодер вводится n_0 новых символов, а введенные ранее самые старые n_0 символов сбрасываются, и процесс повторяется для определения следующего информационного кадра.

Таким образом, декодер Витерби последовательно обрабатывает кадр за кадром, двигаясь по решетке, аналогичной используемой кодером. В каждый момент времени декодер не знает, в каком узле находится кодер, и не пытается его декодировать. Вместо этого декодер по принятой последовательности определяет наиболее правдоподобный путь к каждому узлу и определяет расстояние между каждым таким путем и принятой последовательностью. Это расстояние называется *мерой расходимости пути*. В качестве оценки принятой последовательности выбирается сегмент, имеющий наименьшую меру расходимости. Путь с наименьшей мерой расходимости называется *выжившим путем*.

Рассмотрим работу декодера Витерби на примере. Полагаем, что кодирование производится с использованием сверточного (6,3)-кода (схема – см. рис. 7.5, решетчатая диаграмма, соответствующая этому кодеру, – см. рис. 7.4). Пользуясь решетчатой диаграммой кодера, попытаемся, приняв некоторый сегмент r, проследить наиболее вероятный путь кодера. При этом для каждого сечения решетчатой диаграммы будем отмечать меру расходимости пути к каждому ее узлу.

Предположим, что передана кодовая последовательность U=(0000000…), а принятая последовательность имеет вид r=(10001000 00...), то есть в первом и в третьем кадрах кодового слова возникли ошибки. Процедура и результат декодирования не зависят от передаваемого кодового слова и определяются только ошибкой, содержащейся в принятой последовательности. Поэтому проще всего считать, что передана нулевая последовательность, то есть U = (0000000…).

Приняв первую пару символов (10), определим меру расходимости для первого сечения решетки (см. рис. 7.4), приняв следующую пару символов (00), – для второго сечения и т.д. При этом из входящих в каждый узел путей оставляем путь с меньшей расходимостью, поскольку путь с большей на данный момент расходимостью уже не сможет стать в дальнейшем короче. Для рассматриваемого примера начиная с четвертого уровня метрика (или мера расходимости) нулевого пути меньше любой другой метрики. Поскольку ошибок в канале больше не было, ясно, что в конце концов в качестве ответа будет выбран именно этот путь. Из этого примера также видно, что выжившие пути могут достаточно долго отличаться друг от друга. Однако на шестом – седьмом уровне первые семь ребер всех выживших путей совпадут друг с другом. В этот момент согласно алгоритму Витерби и принимается решение о переданных символах, т.к. все выжившие пути выходят из одной вершины, т.е. соответствуют одному информационному символу.

Процедура декодирования последовательности с двумя ошибками иллюстрируется последовательностью, представленной на рис. 7.8.

Глубина, на которой происходит слияние выживших путей, не может быть вычислена заранее; она является случайной величиной, зависящей от кратности и вероятности возникающих в канале ошибок. Поэтому на практике обычно не ждут слияния путей, а устанавливают фиксированную глубину декодирования. Из рис. 7.8 видно, что уже на уровне E степень различия метрик правильного и неправильного путей достаточно велика ($d_{пр}$=2, $d_{ош}$=4), т.е. в данном случае можно было бы ограничить глубину декодирования величиной b≤6. Но иногда более длинный к данному сечению путь может оказаться в конечном итоге самым коротким, поэтому особенно увлекаться уменьшением размера окна декодирования b с целью упрощения работы декодера не стоит. Из рис. 7.8 видно также, что, несмотря на наличие в принятом фрагменте двух ошибок, его декодирование произошло без ошибки и в качестве ответа будет принята переданная нулевая по-

следовательность. На практике глубину декодирования обычно выбирают в диапазоне $n<b\leq n+l$, где l – число исправляемых данным кодом ошибок.

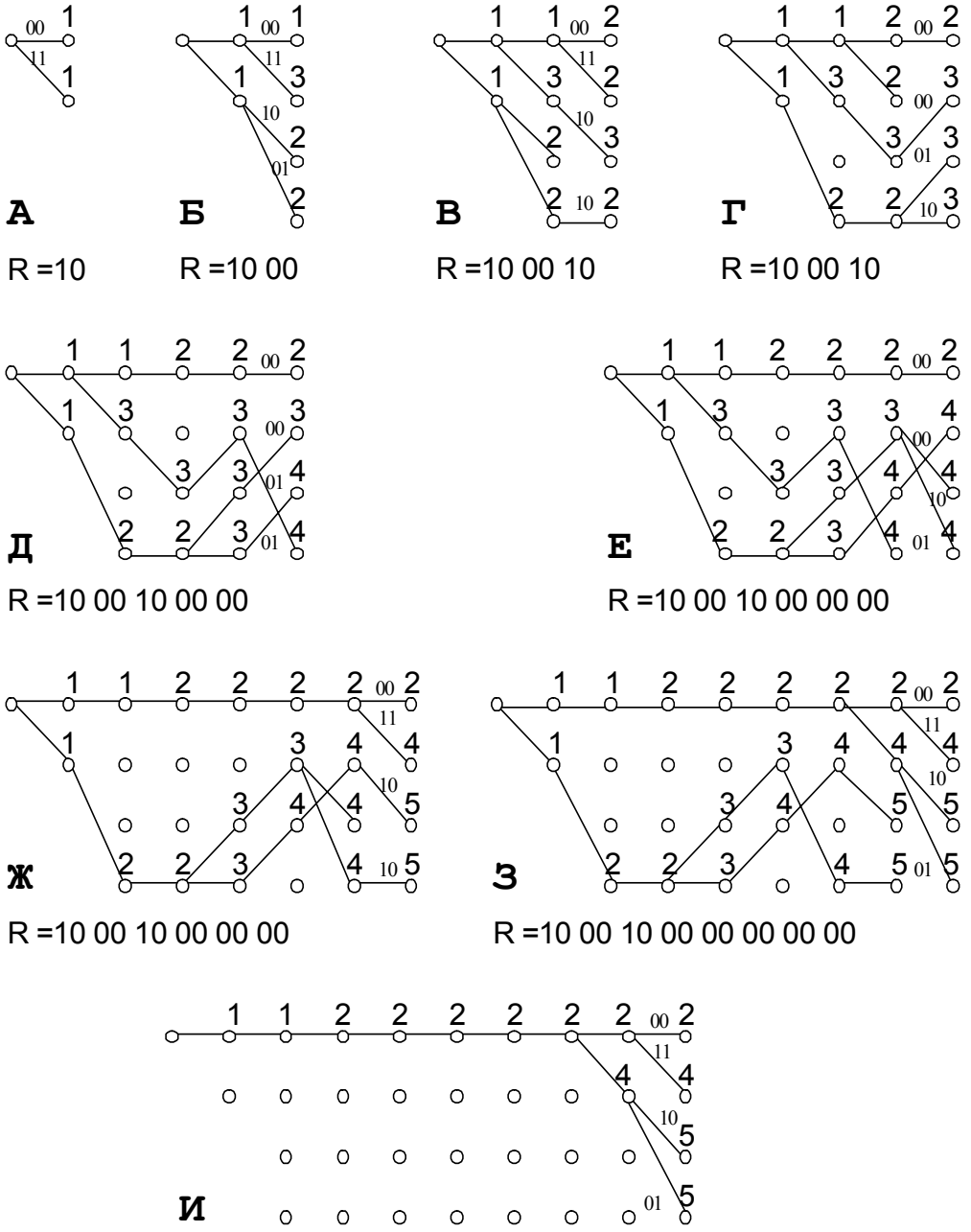


Рис. 7.8. Декодирование по алгоритму Витерби

7.3. Вопросы и задания для самопроверки

7.1. Дайте определение древовидным кодам.

7.2. Представьте схему сверточного кодера и поясните ее работу.

7.3. В каком случае древовидный код является систематическим?

7.4. Представьте схему классификации древовидных кодов.

7.5. В чем заключается основное отличие между сверточными и блочными кодами?

7.6. Назовите основные характеристики сверточных кодов.

7.7. Постройте схему сверточного кодера для (6,3)-кода и объясните ее работу?

7.8. Как получить импульсную переходную характеристику сверточного кода?

7.9. Каким образом с использованием импульсной переходной характеристики кода закодировать информационное сообщение? Приведите пример.

7.10. Представьте общий вид порождающей матрицы сверточного кода и порождающей матрицы сверточного кода, усеченного до блокового длины n.

7.11. Представьте проверочную матрицу сверточного кода и приведите примеры порождающей и проверочной матрицы.

7.12. Как называются графы, с помощью которых можно описать сверточный код?

7.13. Приведите пример маркированной решетки для сверточного кода и поясните ее строение.

7.14. Приведите пример кодового дерева и поясните принцип кодирования по нем. В чем разница между кодовым деревом и решетчатой диаграммой?

7.15. Сколько ошибок можно исправить с использованием кодов Вайнера – Эша?

7.16. Запишите проверочную и порождающие матрицы для (12,9)-кода Вайнера – Эша.

7.17. Представьте схему кодера (12, 9)-кода Вайнера-Эша.

7.18. Дайте определение минимальному расстоянию сверточного кода.

7.19. Сформулируйте алгоритм Витерби декодирования сверточных кодов.

7.20. Приведите пример декодирования сверточного кода по алгоритму Витерби.

7.4. Практическое занятие №7

Помехоустойчивое кодирование информации сверточными кодами

Теория и методы решений для практического занятия представлены в теоретической части модуля 7.

Тестовые задания

1. Древовидные коды формируют по принципу:

a) независимо кодируются отдельные кадры информационных символов в отдельные кадры кодового слова;

b) кодирование каждого кадра информационных символов в отдельный кадр кодового слова производится с учетом предыдущих кадров информационных символов;

c) независимо кодируются первые m кадры информационных символов в отдельные кадры кодового слова, а последующие – с учетом предыдущих кадров;

d) кодирование каждого кадра информационных символов для первых m кадров в отдельный кадр кодового слова производится с учетом предыдущих кадров информационных символов, а последующие – кодируются независимо.

2. Отметьте правильное формальное определение древовидного (n_0, k_0)-кода:

a) древовидный (n_0, k_0)-код – это такое отображение на себя множества полубесконечных последовательностей элементов из $GF(q)$, что если для любого M первые Mn_0 компонент двух полубесконечных последовательностей совпадают, то первые Mk_0, компонент отображений этих последовательностей тоже совпадают;

b) древовидный (n_0, k_0)-код – это такое отображение на себя множества полубесконечных последовательностей элементов из $GF(q)$, что если для любого M любые Mk_0 компонент двух полубесконечных последовательностей совпадают, то любые Mn_0, компонент отображений этих последовательностей тоже совпадают;

c) древовидный (n_0, k_0)-код – это такое, отображение на себя множества полубесконечных последовательностей элементов из $GF(q)$, что если для любого M любые Mn_0 компонент двух полубесконечных последовательностей совпадают, то любые Mk_0, компонент отображений этих последовательностей тоже совпадают;

d) древовидный (n_0, k_0)-код – это такое отображение на себя множества полубесконечных последовательностей элементов из $GF(q)$, что если для любого M первые Mk_0 компонент двух полубесконечных последовательно-

стей совпадают, то первые Mn_0, компонент отображений этих последовательностей тоже совпадают.

3. На основе какого выражения определяется информационная длина слова сверточного кода?

a) $n=(m+1)\cdot n_0$; b) $k=(m+1)\cdot k_0$; c) $k=(n+1)\cdot n_0$; d) $n=(m+1)\cdot n_0$.

4. На основе какого выражения определяется кодовая длина блока сверточного кода?

a) $n=(m+1)\cdot n_0$; b) $k=(m+1)\cdot k_0$; c) $m=(n+1)\cdot n_0$; d) $n=(m+\mathrm{k})\cdot n_0$.

5. Какое число многочленов, в общем случае, требуется для описания сверточного кода?

a) k_0+n_0; b) k_0/n_0; c) k_0n_0; d) $n_0k_0+n_0$.

6. Выберите ответ, в котором дано правильное определение решетке сверточного кода:

a) решеткой называется граф, узлы которого находятся в прямоугольной координатной сетке, полубесконечной слева, а число узлов в каждом столбце конечно;

b) решеткой называется граф, узлы которого находятся в прямоугольной координатной сетке, полубесконечной справа, а число узлов в каждой строке конечно;

c) решеткой называется граф, узлы которого находятся в прямоугольной координатной сетке, полубесконечной справа; число узлов в каждом столбце конечно;

d) решеткой называется граф, узлы которого находятся в прямоугольной координатной сетке, полубесконечной слева, а число узлов в каждой строке конечно.

7. Что понимают под шириной окна декодирования?

a) число кадров, которые декодер может хранить в памяти;

b) число символов, которые декодер может хранить в памяти;

c) число символов, которые декодер может параллельно обрабатывать;

d) число кадров, которые декодер может параллельно обрабатывать.

8. Определите правило кодирования сверточными двоичными кодами с использованием решетчатой диаграммы:

a) очередные символы входной последовательности определяют направление движения из узлов решетки: если 1, то идем по верхнему ребру, если 0 – по нижнему ребру;

b) очередные символы входной последовательности определяют направление движения из узлов решетки: если 0, то идем по верхнему ребру, если 1 – по нижнему ребру;
c) первые *m* символы входной последовательности определяют направление движения из узлов решетки: если 1, то идем по верхнему ребру, если 0 – по нижнему ребру;
d) первые *m* символы входной последовательности определяют направление движения из узлов решетки: если 0, то идем по верхнему ребру, если 1 – по нижнему ребру.

9. В каком случае минимальное расстояние Хэмминга для любых начальных сегментов длины l кадров всех пар кодовых называется просто минимальным расстоянием слов?
a) если l равно *m*;
b) если l равно *n*+1;
c) если l равно *m*+1
d) если l равно *n*.

10. В каком ответе приведена правильная методика работы декодера Виттерби?
a) декодер Витерби параллельно обрабатывает кадры, двигаясь по решетке. В каждый момент декодер по принятой последовательности определяет наиболее правдоподобный путь к каждому узлу и определяет расстояние между каждым таким путем и принятой последовательностью. В качестве оценки принятой последовательности выбирается сегмент, имеющий наименьшую меру расходимости;
b) декодер Витерби последовательно обрабатывает кадр за кадром, двигаясь по решетке. В каждый момент декодер по принятой последовательности определяет наиболее правдоподобный путь к каждому узлу и определяет расстояние между каждым таким путем и принятой последовательностью. В качестве оценки принятой последовательности выбирается сегмент, имеющий наименьшую меру расходимости;
c) декодер Витерби параллельно обрабатывает кадры, двигаясь по решетке, и зная в каком узле в каждый момент находиться кодер, успешно декодирует последовательность;
d) декодер Витерби последовательно обрабатывает кадр за кадром, двигаясь по решетке. В каждый момент декодер знает в каком узле находиться кодер и поэтому декодирует последовательность.

Задачи

1. Для систематического сверточного кода с порождающими многочленами $g_1(x)=1$, $g_2(x)=1+x$ и параметрами $R=1/2, v=1$ построить:

1.1. порождающую матрицу;

1.2. проверочную матрицу;

1.3. кодовое дерево;

1.4. решетчатую диаграмму;

1.5. схему кодера.

2. Постройте кодер (12,9)-кода Вайнера – Эша, использующий один двоичный регистр сдвига длины 3.

3. Постройте кодер для (32,28)-кода Вайнера – Эша и определите его импульсную характеристику.

4. Закодируйте сообщение (011000….), используя решетчатую диаграмму и кодовое дерево из задания 1.

5. По алгоритму Витерби декодируйте принятую последовательность (0100100…..), если используется для передачи сверточный (6,3)-код, с порождающими полиномами $g_1(x)=1+x+x^2$, $g_2(x)=1+x^2$.

Модуль 8

НИЗКОСКОРОСТНЫЕ КОДЫ

Цель модуля – изучение методов формирования и декодирования низкоскоростных кодов, включая алгоритмы быстрого декодирования.

В результате изучения модуля студенты должны:

- знать алгоритмы формирования кодов максимальной длины, кодов Голда и квадратично-вычетных кодов и уметь их использовать для генерирования таких кодов;
- знать правила построения генераторов низкоскоростных кодов на сдвиговых регистрах и уметь их синтезировать для кодов заданных длин;
- иметь представление о матрицах Адамара и построении быстрых алгоритмов вычисления векторно-матричного произведения для данного типа матриц;
- иметь представление о корреляционном декодировании низкоскоростных кодах и его особенностях;
- знать алгоритмы быстрого декодирования низкоскоростных кодов и уметь их использовать для обработки кодов.

Содержание модуля

8.1. Построение низкоскоростных кодов

8.1.1. Общие сведения о низкоскоростных кодах

К *низкоскоростным* кодам относятся коды, у которых количество проверочных символов значительно превышает количество информационных символов и, соответственно, скорость передачи мала. Однако низкоскоростные коды характеризуются значительным кодовым расстоянием $d \approx n/2$. Поэтому такие коды корректируют примерно четверть ошибок на длине n и занимают особое положение в теории и практике помехоустойчивого кодирования. Следует отметить, что для большинства из них разработаны эффективные алгоритмы формирования и декодирования. С точки зрения теории кодирования они являются классическими кодами, с другой стороны, свойства кодов позволяют использовать их в качестве основы для формирования так называемых сложных сигналов для систем связи, синхронизации, локации, навигации, систем передачи и криптографической защиты информации. Под *сложными* обычно понимают такие сигналы, для которых произведение их длительности на занимаемую полосу частот значительно больше единицы. Помимо термина «сложный» сигнал часто используется понятие *широкополосного сигнала* (в зарубежной литературе их называют *сигналами* «*с растянутым спектром*» (spread spectrum)).

На практике находят применение как отдельные кодовые слова низкоскоростных кодов, так и их ансамбли. В радиотехнических системах со сложными кодированными сигналами на передающей стороне к каждому отсчету передаваемых данных ставится в соответствие своя дискретная псевдослучайная последовательность, спектр частот которой значительно превышает полосу данных. Отдельные кодовые слова рассматриваются как кодовые последовательности символов $u_l=(u_0, u_1, ..., u_{n-1})$. Поэтому большое значение имеют периодические и апериодические корреляционные свойства кодовых последовательностей. Периодическая автокорреляционная функция двоичной последовательности U_l определяется следующим образом

$$R\ (\tau) = \sum_{l=0}^{n-1} (-1)^{u_l + u_{l+\tau}}\ ,$$

где $\tau = 0,1,...,n-1$, а сумма $l+\tau$ берется по модулю n.

Ансамбли кодовых последовательностей используются для формирования систем сигналов, обладающих оптимальными корреляционными свойствами при кодовом разделении сигналов различных абонентов, использующих для передачи информации общий канал. Определяющим в

синтезе ансамбля является критерий минимума боковых выбросов автокорреляционных функций и минимума значений взаимокорреляционных функций, который находится для пары последовательностей $u=\{u_i\}$ и $v=\{v_i\}$ следующим выражением

$$R_{uv}(\tau)=\sum_{i=0}^{n-1}(-1)^{u_i+v_{i+\tau}}.$$

К низкоскоростным кодам принадлежат последовательности Холла, Якоби, квадратично-вычетные и характеристические последовательности, а также *М*-последовательности и коды на их основе (коды Голда, Кассами и др.), последовательности Гордона–Милса–Велча.

8.1.2. Формирование низкоскоростных кодов

Формирование кодов максимальной длины

Коды максимальной длины являются наиболее изученными низкоскоростными кодами. Отдельные кодовые слова данных кодов называют *М-последовательностями*. *Двоичные М-последовательности* – класс кодовых последовательностей, используемых для формирования сигналов с идеальными автокорреляционными свойствами (боковые выбросы всегда равны -1). Коды максимальной длины представляют собой примитивные БЧХ-коды, имеющие следующие параметры: $n=2^k-1$, $d=2^{k-1}$, $t=2^{k-2}-1$. М-последовательности, являющиеся линейными циклическими кодами, обладают свойством периодичности: $\{u_i\}=\{u_{i+n}\}$. Подстановка $V: i=l_i$, $(l,n)=1$ переводит *М*-последовательность саму в себя либо в последовательность другой формы. Общее число различных последовательностей определяется формулой

$$M=\varphi(n)/k,$$

где $\varphi(n)$ – функция Эйлера числа n.

Обычно *М*-последовательности формируются при помощи линейных автоматов, описываемых проверочными – $h(x)$ и генераторными – $g(x)$ полиномами. Если для построения генератора используется проверочный полином, то его основу составляет операция деления на $h(x)$. Из k символов, формирующих начальный блок символов, образуют полином, который после умножения на x^n является делимым. Полином частного образует формируемую *М*-последовательность. Например, для информационного блока (1001) информационный полином равен x^3+1, когда после его умножения на x^{15} образуется полином $x^{18}+x^{15}$, который при последующем

делении на $h(x)=x^4+x+1$ дает полином $x^{14}+x^{10}+x^7+x^6+x^4+x^2+x^1+1$. Это соответствует КП (100010011010111). Изменяя начальный блок, можно получить все возможные сдвиги М-последовательности. На рис. 8.1 представлена схема формирователя *M*-последовательности для $h(x)=x^4+x+1$.

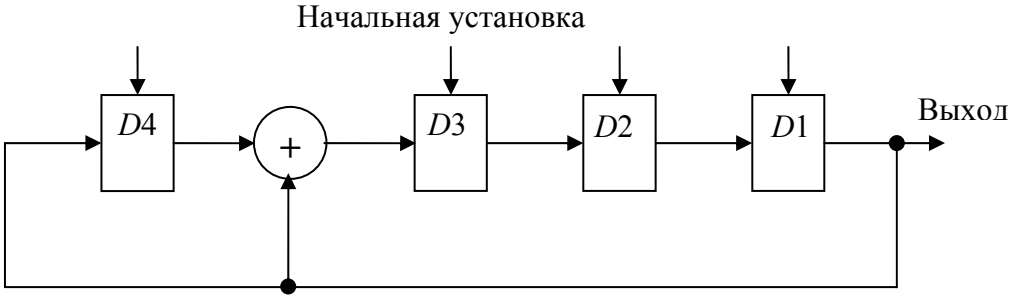


Рис. 8.1. Формирователь М-последовательности,
задаваемой полиномом $h(x)=x^4+x+1$

Использование кодера, реализуемого по порождающему полиному, связано с аппаратной сложностью делителя, т.к. число ячеек памяти регистра сдвига определяется максимальной степенью порождающего полинома. Например, при $n=1023$ потребуется регистр сдвига длиной 1013 разрядов. При программной же реализации это выливается в чрезмерно большое количество операций. Для *M*-последовательности степень проверочного полинома всегда меньше степени генераторного полинома, поэтому обычно применяется $h(x)$.

Процесс формирования *M*-последовательности можно представить как результат векторно-матричного умножения. При этом кодовое слово образуется как векторное произведение вектора A, полученного из последовательности информационных символов, на порождающую матрицу G

$$C=A\times G.$$

Первая строка матрицы G образуется из коэффициентов $g(x)$, а остальные являются их циклическими сдвигами. Для кода длиной $n=15$ и $g(x)=x^{11}+x^8+x^7+x^5+x^3+x^2+x+1$ матрица G имеет вид

$$G=\begin{bmatrix}0&0&0&1&0&0&1&1&0&1&0&1&1&1&1\\0&0&1&0&0&1&1&0&1&0&1&1&1&1&0\\0&1&0&0&1&1&0&1&0&1&1&1&1&0&0\\1&0&0&1&1&0&1&0&1&1&1&1&0&0&0\end{bmatrix}.$$

Путем линейных операций над строками проверочная матрица приводится к виду:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Столбцы матрицы G представляют собой элементы поля Галуа $GF(2^k)$, записанные в порядке возрастания степени примитивного элемента поля a, являющегося корнем полинома $h(x)$,

$$G = [a^0 \quad a^1 \quad a^2 \quad a^3 \quad a^4 \quad a^5 \quad a^6 \quad a^7 \quad a^8 \quad a^9 \quad a^{10} \quad a^{11} \quad a^{12} \quad a^{13} \quad a^{14}].$$

Предложены две практические схемы формирователей ортогональных последовательностей, основанных на матричных произведениях, имеющих примерно одинаковую аппаратную сложность. В первом случае используются формирователь поля Галуа и схема, обеспечивающая суммирование строк порождающей матрицы, выбранных при помощи логической схемы (рис. 8.2).

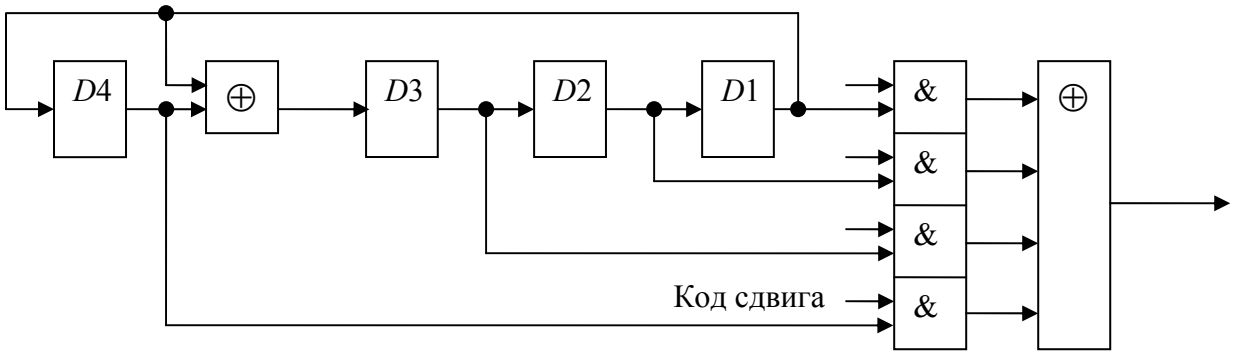


Рис. 8.2. Формирователь ансамбля *M*-последовательностей на основе генератора поля Галуа

Второй формирователь использует рекуррентные зависимости между отдельными символами *M*-последовательности. Они выводятся из известных соотношений, получающихся из записи проверочного полинома в следующем виде

$$h_k x^k = h_{k-1} x^{k-1} + \ldots + h_1 x + h_0 .$$

Тогда с учетом того, что h_k и h_0 равны единице, получаем

$$c_k = c_{k-1}h_{k-1} + c_{k-2}h_{k-2} + ... + c_1h_1 + c_0 .$$

В общем случае

$$c_i = c_{i-1}h_{k-1} + c_{i-2}h_{k-2} + ... + c_{i-k+1}h_1 + c_{i-k} .$$

При этом $c_i = a_i$ для $0 \le i \le k-1$.

На рис. 8.3 приведен генератор *M*-последовательностей, задаваемых полиномом $h(x) = x^4 + x + 1$ с рекуррентным правилом $c_i = c_{i-3} + c_{i-4}$.

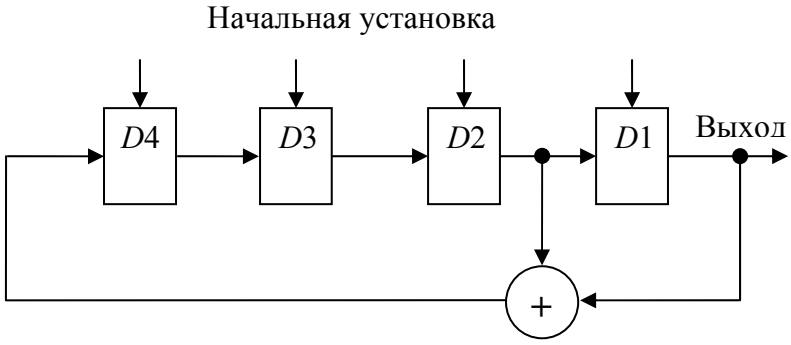


Рис. 8.3. Рекуррентный генератор *M*-последовательности, задаваемой полиномом $h(x) = x^4 + x + 1$

Все формирователи должны быть дополнены счетчиками для получения длины последовательности, равной $2^k - 1$. При формировании периодической последовательности необходимо предусмотреть возможность устранения запрещенного состояния регистра сдвига, когда все разряды заполнены нулям.

Формирование кодов Голда

Последовательности Голда формируются путем сложения по модулю два двух последовательностей максимального периода длиной $n = 2^k - 1$ каждая. Выполняется подбор пар *M*-последовательностей $\{a_i\}$ и $\{b_{i+j}\}$, имеющих периодические взаимокорреляционные функции требуемой формы, и суммирование одной *M*-последовательности со всеми $2^k - 1$ циклическими сдвигами второй *M*-последовательности. Ансамбль будет содержать обе исходные *M*-последовательности $\{a_i\}$ и $\{b_i\}$, а также сумму вида $\{a_i\} \oplus \{b_{i+j}\}$, где j указывает на циклический сдвиг второй последовательности, $j = \overline{0, n-1}$. В качестве первой последовательности берется любая *M*-последовательность с проверочным полиномом $h_1(x)$ степени k.

Этот полином является минимальным многочленом примитивного элемента α поля $GF(2^k)$, сформированного при помощи примитивного полинома $h_1(x)$. В качестве второй M-последовательности берется последовательность с проверочным полиномом $h_2(x)$, корнями которого являются элементы поля $GF(2^k)$ вида α^{2^l+1}, где $(l,n)=1$. Данное правило справедливо для любых k, кроме $k \equiv 0 \bmod 4$. Полученные описанным методом последовательности имеют ПВКФ, выбросы которых могут принимать только следующие значения

$$R(\tau)=\begin{cases} -1, \\ 2^{\frac{k+1}{2}}-1 \quad \text{и} \quad -(2^{\frac{k+1}{2}}+1), \text{ для нечетных } k, \\ 2^{\frac{k+2}{2}}-1 \quad \text{и} \quad -(2^{\frac{k+2}{2}}+1), \text{ для четных } k. \end{cases}$$

Пример. Пусть имеется проверочный полином $h_1(x)$ с $k=5$, в качестве второго можно взять полиномы, корнями которых являются элементы поля $\alpha^3, \alpha^5, \alpha^9, \alpha^{17}$. При этом элементы α^3 и α^{17} являются корнями полинома $h_2(x)$, а α^5 и α^9 корнями полинома $h_3(x)$. Таким образом, образуются две пары корней проверочных полиномов последовательностей Голда – (α, α^3) и (α, α^5). Кроме этого, можно получить еще две пары, если воспользоваться следующей процедурой: возведем элементы α и α^3 в степень m, которое выбирается из условия $3m \equiv 1 \bmod(2^5-1)$. В результате получим новую пару α и α^m. Аналогичным образом поступаем со второй парой (α, α^5). Таким образом, дополнительно имеем (α, α^7) и (α, α^{11}). Если в качестве первого полинома выбран полином $h_1(x)=x^5+x^2+1$, то другие равны

$$h_2(x)=x^5+x^4+x^3+x^2+1 \text{ для } \alpha^3;$$
$$h_3(x)=x^5+x^4+x^2+x+1 \text{ для } \alpha^5;$$
$$h_4(x)=x^5+x^3+x^2+x+1 \text{ для } \alpha^7;$$
$$h_5(x)=x^5+x^4+x^3+x+1 \text{ для } \alpha^{11}.$$

ПАКФ имеет только три значения: –1, –9, +7.

Для формирования последовательностей Голда применяют два способа. В первом используется два генератора М-последовательностей. Один из них генерирует периодически повторяющуюся последовательность, а вто-

рой генератор имеет возможность изменять фазу (задержку) второй последовательности относительно первой. Изменять фазу можно также двумя способами: изменением начальной фазы второго генератора (на рис. 8.4 представлено устройство для формирования кодов Голда) или путем сложения по модулю сигналов, снимаемых с различных отводов регистра сдвига второго генератора (рис. 8.5), здесь используется свойство линейности М-последовательностей.

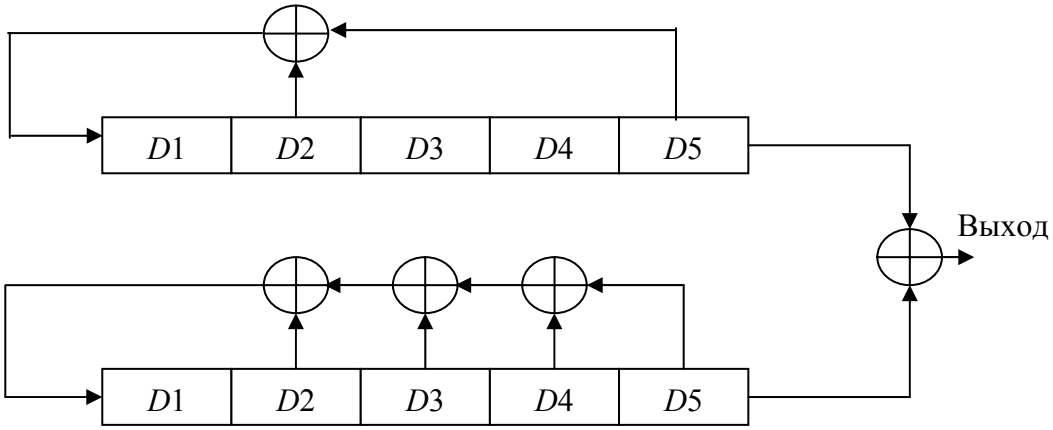


Рис. 8.4. Схема формирования кода Голда

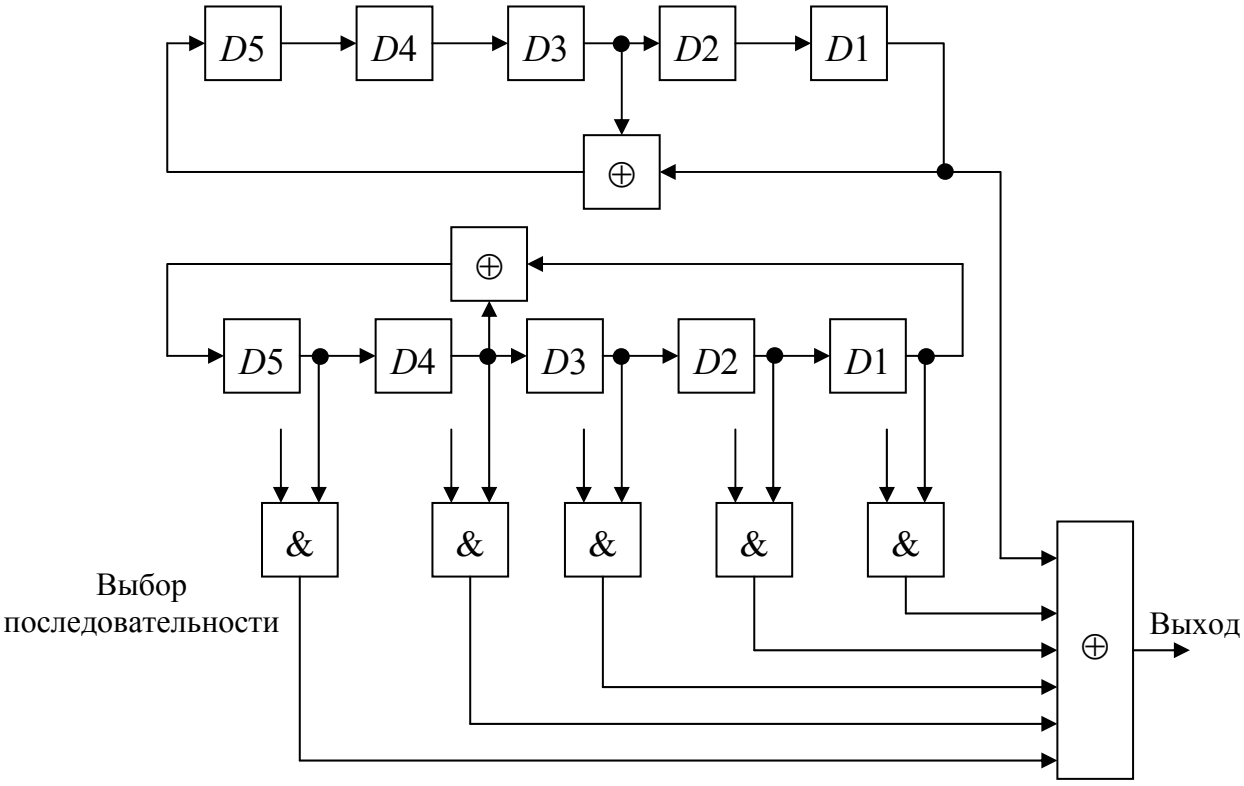


Рис. 8.5. Формирователь кода Голда

Второй способ формирования последовательностей Голда предполагает использование одного генератора, который строится по проверочному полиному $h(x) = h_1(x)h_2(x)$, где $h_1(x)$ и $h_2(x)$ – проверочные полиномы *M*-последовательностей. Для выбора конкретной формируемой последовательности в регистр сдвига вводится начальный блок длиной 2*k* символов. В схеме с двумя генераторами управление выбором формируемой последовательности проще, поскольку для этого требуется устанавливать начальную фазу только одного генератора, имеющего *k*-й разрядный регистр сдвига.

Построение кодов квадратичных вычетов

Квадратично-вычетные коды существуют для длин $n = 4k \pm 1$, где *k* – любое число, а *n* – простое число. Например, если *k*=3, то квадратично-вычетный код существует для длины 11 (*n*=4×3-1) и для длины 13 (*n*=4×3+1). Если *k*=4, то квадратично-вычетный код существует только для длины 17 (*n*=4×4+1).

Периодическая автокорреляционная функция последовательностей квадратичных вычетов обладает рядом свойств, позволяющих их использовать в широкополосных системах связи, и принимает значения

$$R_{n=4k-1} = 1, \quad R_{n=4k+1} = \begin{cases} -1, \\ 3. \end{cases}.$$

Пронумеруем в порядке возрастания элементы поля *GF*(*p*), тогда

$$GF(p) = \{a_i : i = 0,1,..., p-1\} = \{i : i = 0,1,..., p-1\}.$$

Правило кодирования сформулируем в терминах квадратичных вычетов

$$u = \{u_i : i = 0,1,..., p-1\},$$

$$u_0 = -1,$$

$$u_i = \begin{cases} 1, & \text{если } i \text{ - квадратичный вычет,} \\ -1, & \text{если } i\text{-квадратичный невычет,} \end{cases}$$

$$i \neq 0 \ (\text{mod p}).$$

Если сравнение

$$a \equiv x^2 (\text{mod } p), \quad a, x \in GF(p), \quad a \neq 0 \ (\text{mod } p),$$

имеет решение, то а называется *квадратичным вычетом поля GF*(*p*), в противном случае – *квадратичным невычетом*.

Пример. Построить код *U* для длины 7. Согласно правилу кодирования, необходимо определить является или нет индекс *i* квадратичным вычетом поля *GF*(7).

1^2=1 mod(7) 3^2=2 mod(7) 5^2=4 mod(7)
2^2=4 mod(7) 4^2=2 mod(7) 6^2=5 mod(7)
Таким образом, квадратичными вычетами являются числа 1, 2,4, т.к.
$1=1^2$ mod(7) $2=3^2$ mod(7) $4=2^2$ mod(7)
Кодовое слово имеет вид

$$u_0=-1,\ u_1=1,\ u_2=1,\ u_3=-1,\ u_4=1,\ u_5=-1,\ u_6=-1.$$

$$u=\{-1,1,1,-1,1,-1,-1\}.$$

Для перехода в алфавит представления (1,0) необходимо заменить (-1) на (1), а (1) на (0). Для рассмотренного примера получим $u=\{1001011\}$.

Квадратичные вычеты являются циклическими кодами, поэтому остальные слова кода формируются путем сдвига опорного кодового слова. Для квадратичных вычетов длиной n=7, ансамбль кодовых слов имеет вид

$$U'=\begin{bmatrix} -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & -1 \end{bmatrix}$$

8.2. Декодирование низкоскоростных кодов

8.2.1. Корреляционное декодирование

Кодовые последовательности, сформированные на основе низкоскоростных кодов, используются в телекоммуникационных, радиотехнических, вычислительных системах для синхронизации, передачи информации, локации, диагностики и других применений. Для их декодирования могут использоваться процедуры декодирования, основанные на посимвольном приеме. Однако низкоскоростные коды в основном используются в каналах с большим уровнем помех, поэтому целесообразно реализовать декодирование по методу максимального правдоподобия. Это приводит к повышению достоверности процессов декодирования, помехоустойчивости систем при передаче и обработке информации.

Рассмотрим задачу беспоисковой синхронизации в векторно-матричной форме. В этом случае принятый сигнал X длиной n может быть представлен в виде вектора $X=(x_0,x_1,\ldots,x_{N-1})^T$, а сигнальная матрица C – в

виде матрицы-циркулянта, строками которой являются все циклические сдвиги синхропоследовательности $\{c_i\}$, i=0, 1,…, N-1:

$$C = \begin{bmatrix} c_0 & c_1 & \dots & c_{N-1} \\ c_{N-1} & c_0 & \dots & c_{N-2} \\ \dots\dots & \dots & \dots & \dots\dots \\ c_1 & c_2 & \dots & c_0 \end{bmatrix}$$

Фаза принятого сигнала первоначально неизвестна, поэтому модулирующая кодовая последовательность может быть одной из строк вышеуказанной матрицы. Для определения фазы требуется установить, какой строке соответствует принятый сигнал. Поэтому формальной сущностью устройства синхронизации является вычисление вектора

$$Y=C\times X=(y_0,y_1,\dots,y_{n-1})^T$$

и определение в полученном векторе номера максимальной компоненты.

Аналогичная задача возникает и при передаче информации. При кодировании набор информационных символов однозначно определяет строку матрицы, которая используется для формирования сигнала. При декодировании необходимо однозначно установить номер использованной строки.

Последовательность отсчетов принятого сигнала образует вектор $X=(x_0,x_1,\dots,x_{n-1})^T$. Его форма зависит от передаваемой последовательности. При распространении в канале энергия сигнала уменьшается, к нему добавляются помехи. Поскольку в точке приема форма сигнала неизвестна, то принятый сигнал может содержать одну из строк матрицы и необходимо решить задачу распознавания сигналов. Реально надо определить, какая их строк матрицы C была использована для формирования сигнала. Для этого вектор принятого сигнала X умножается на матрицу C и по максимальной компоненте полученного корреляционного вектора принимается решение о принятом сигнале.

При практической реализации корреляционных алгоритмов важнейшим параметром становится сложность вычисления. Обычно под сложностью понимается количество операций сложения и количество операций умножения, необходимых для реализации конкретного алгоритма. Каждая арифметическая операция для своего выполнения требует определенного времени. Оно включает в себя время на извлечение исходных данных из памяти, время, расходуемое на непосредственное выполнение конкретных операций, и время на возврат результатов вычислений в память. Т.к. быстродействие устройств и систем передачи, обработки, хранения и распределения информации, как правило, ограничено, то по этому параметру мож-

но судить о реализуемости алгоритма, а также сравнивать различные алгоритмы между собой. При этом следует принимать во внимание и то, что время вычисления сильно зависит от используемой элементной базы и языков программирования. В режиме обработки сигналов в реальном масштабе времени декодирование должно быть осуществлено за время, равное периоду сигнала. Поэтому для сигналов большой длины невысокое быстродействие аппаратуры обработки существенно ограничивает длину кода, если использовать прямое умножение вектора принятого сигнала на опорную (кодовую) матрицу.

8.2.2. Декодирование *M*-последовательностей методом максимального правдоподобия

Рассмотрим задачи декодирования кода максимальной длинны по методу максимального правдоподобия (эту задачу можно также интерпретировать как задачу синхронизации) при приеме периода последовательности. Пусть A – циркулянт M -последовательности. Строками являются все циклические перестановки M -последовательности, в которой произведена замена символов $0 \to 1,\ 1 \to \text{-}1$. Матрица A путем перестановки строк и столбцов может быть преобразована к виду, допускающему применение алгоритма быстрого преобразования Уолша – Адамара. Быстрое преобразование Уолша – Адамара применяется для матриц Адамара. Матрица Адамара размерностью 2^k, k – целое, получается из матрицы размерностью 2^{k-1}

$$H_{2^k} = H_{2^{k-1}} \otimes H_2 = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}.$$

Форма операции умножения, которая здесь использована, называется *кронекеровским умножением*, а H_2 – матрица минимально возможного размера

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Например, матрица Уолша – Адамара размером 4×4 имеет вид

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix};$$

Если X вектор, элементами которого являются 2^k чисел, то прямое и обратное преобразования Уолша – Адамара вычисляются как векторно-матричные произведения

$$Y = H_{2^k} X^T,\ X^T = H_{2^k}^{-1} Y .$$

Матрица обратного преобразования Уолша – Адамара с точностью до постоянного множителя совпадает с матрицей прямого преобразования

$$H_{2^k}^{-1} = 2^{-k} H_{2^k} .$$

Свойством кронекеровского произведения матриц является то, что оно легко факторизуется,

$$H_{2^k} = C_{2^k}^1 C_{2^k}^2 ... C_{2^k}^k .$$

Умножение вектора на факторизованную матрицу Уолша – Адамара выполняется за $k2^k$ действительных сложений. Таким образом значительно сокращаются вычислительные затраты при декодировании методом максимального правдоподобия.

Упорядочим кодовые слова в порядке возрастания чисел в первых k символах последовательности. Это эквивалентно умножению слева исходной кодовой матрицы A на перестановочную матрицу P. Матрицу с переставленными строками обозначим S . Тогда S=PA. В подматрице матрицы S, образованной первыми k столбцами, записаны в порядке возрастания все двоичные числа от 1 до N-1. Совокупность этих чисел образует по столбцам функции Радемахера R_1, R_2, …, R_k без первого символа. Из правил формирования последовательности максимальной длины следует, что остальные столбцы будут произведениями k первых, т.е. функциями Уолша. Переставим столбцы так, чтобы они дали матрицу Адамара без первой строки. Это эквивалентно умножению матрицы S справа на перестановочную матрицу Q . Тогда будет справедливым следующее выражение

$$H_N = \begin{bmatrix} 1 & | & 1 & . & . & . & 1 \\ - & + & - & - & - & - & - \\ 1 & | & & & & & \\ . & | & & & & & \\ . & | & & PAQ & & & \\ . & | & & & & & \\ 1 & | & & & & & \end{bmatrix}$$

Задача декодирования сведется к задаче умножения вектора на матрицу Адамара без первой строки и первого столбца.

Поскольку размер матрицы Адамара на единицу больше размера входного вектора, то для согласования размеров длина входного вектора увеличится на единицу путем введения нулевого символа на первой позиции. При анализе результатов вычислений блок выбора максимального значения первая позиция вектора не принимается во внимание. Если слова кода (сдвиги M-последовательности) пронумерованы в порядке возрастания информационной части, то необходимость в перестановке выходных данных отпадает. В этом случае для реализации вычислителя необходимо определить лишь явный вид перестановки выходных данных.

Перестановка столбцов (умножение на матрицу Q) эквивалентна перестановке отсчетов входного вектора. Перестановка строк (умножение на матрицу P) эквивалентна перестановке отсчетов выходного вектора. Поэтому техническая реализация декодера содержит блоки перестановок входных и выходных данных, блок быстрого преобразования Адамара и блок выбора максимального значения (БВМ) (рис. 8.6). Поскольку размерность матрицы Адамара на единицу больше размерности входного вектора, то для согласования размерностей длина входного вектора увеличивается на единицу путем введения нулевого символа на первой позиции. При анализе результатов вычислений блоком выбора максимального значения первая позиция выходного вектора не принимается во внимание. Число необходимых вычислительных операций равно $n\log_2 n$.

Если слова кода пронумеровать в порядке возрастания первых k символов, то необходимость в перестановке выходных данных отпадает.

Для реализации декодера необходимо определить явный вид перестановки, задаваемой матрицей Q . Вид перестановки определяется следующей теоремой.

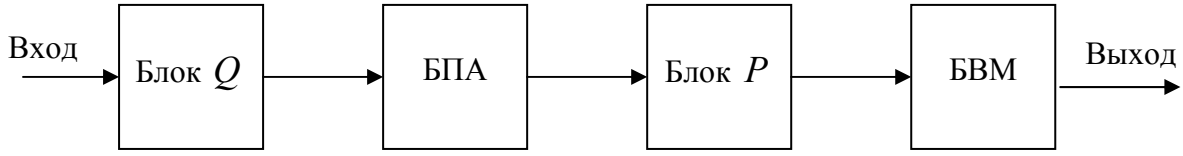


Рис. 8.6. Многоканальный коррелятор с использованием БПА

При упорядочении кодовых слов в матрице A по возрастанию их информационной части (первых k символов) первые k столбцов матрицы A являются функциями Уолша с одной ненулевой позицией в двоичном коде индекса (функциями Радемахера). Двоичные коды индексов этих функций

представляют собой первые k степеней примитивного элемента α поля $GF(2^k)$: $\alpha^0 \to 10...00$, $\alpha^1 \to 0,1...00$, …, $\alpha^{k-1} \to 00...01$. Остальные столбцы матрицы S являются произведениями k первых, выбранных по закону проверочного полинома кода.

Столбцы матрицы A представляют собой функции Уолша с исключенными первыми символами, упорядоченные по степеням примитивного элемента поля $GF(2^k)$, т.е. индекс функции Уолша, расположенной в l-ом столбце, равен α^{i-1}.

Подстановка

$$L: i \to \alpha^{i-1}, \tag{8.1}$$

примененная к столбцам матрицы A, переводит A в матрицу Адамара без первой строки и первого столбца.

Действительно, подстановка (8.1) располагает столбцы матрицы A в порядке возрастания двоичного кода индекса, т.е. вводит упорядочение функций Уолша по Адамару. В силу симметрии функций Уолша относительно индекса и аргумента строки A будут также упорядочены по Адамару, что и дает матрицу Адамара без первой строки и первого столбца.

Перестройка столбцов матрицы A равносильна перестановке позиций вектора X, поэтому процесс декодирования сводится к следующему:

1. Позиции вектора X переставляются в соответствии с выражением (8.1), после чего X дополняется слева одним нулевым символом. Обозначим результат этих операций X'.

2. Выполняется умножение вектора X' на матрицу Адамара H.

3. Определяется максимальная компонента произведения HX'.

Примером. Пусть код длинны $n = 2^k - 1$ задается проверочным полиномом $X^3 + X + 1$. Все слова этого кода записаны в алфавите 1, -1 и образуют матрицу

$$A = \begin{bmatrix} 1 & 1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & -1 & 1 \end{bmatrix}.$$

Очевидно, что первые три столбца этой матрицы образуют функции Радемахера R_1 , R_2 , R_3 без первого символа. Степени примитивного элемента поля $GF(2^3)$ равны: $\alpha^0 = 100$; $\alpha^1 = 010$; $\alpha^2 = 001$; $\alpha^3 = 110$; $\alpha^4 = 011$; $\alpha^5 = 111$; $\alpha^6 = 101$. Подстановка (8.1) переставляет столбцы матрицы A следующим образом: $001 \to \alpha^0 = 100$; $010 \to \alpha^1 = 010$; $011 \to \alpha^2 = 001$; $100 \to \alpha^4 = 011$; $110 \to \alpha^5 = 111$; $111 \to \alpha^6 = 101$, или в табличном виде

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 2 & 1 & 6 & 3 & 7 & 5 \end{pmatrix}.$$

Применяя эту подстановку к столбцам матрицы A, получаем матрицу

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix},$$

отличающуюся от матрицы Адамара отсутствием первого столбца и первой строки. Перестановка входных данных может быть аппаратурно выполнена устройством, показанным на рис.8.7

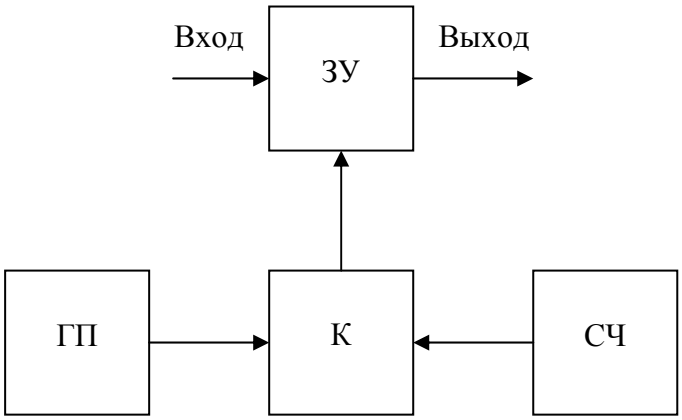


Рис. 8.7. Устройство перестановки

Устройство содержит запоминающее устройство с произвольной выборкой ЗУ, генератор поля Галуа ГП и двоичный счетчик СЧ. Адресные

входы запоминающего устройства подключены к генератору поля и счетчику через коммутатор *К*.

Работа устройства происходит следующим образом. При записи информации управление производится генератором поля Галуа, поэтому первый символ *M*-последовательности записывается в ячейку с номером α^0, второй – в ячейку с номером α^1 и т.д. После того как все символы будут приняты, производится считывание. Процессом считывания управляет двоичный счетчик. Первым будет считан символ по адресу один $(0...01)$, вторым – по адресу два $(0...10)$ и т.д. При таком алгоритме работы выходные данные представляются в соответствии с выражением (8.1).

При больших N рассмотренный декодер существенно проще многоканального коррелятора и легко реализуется средствами временной микроэлектроники. Он может быть использован также для декодирования при поэлементном приеме. В этом случае перестановка (8.1) приводит код максимальной длины в код Рида – Маллера первого порядка, который легко декодируется мажоритарным методом.

При программной реализации используются стандартные алгоритмы БПА, а записью реализации в память выполняет программа вычисления элементов поля Галуа.

8.2.3. Быстрое декодирование кодов Голда методом максимального правдоподобия

Пусть $\overline{A}$ – матрица размера $2^{2\cdot k} \times n$, строками которой являются нулевая последовательность и все последовательности Голда; A – матрица, полученная из $\overline{A}$ преобразованием $0 \to 1 \quad 1 \to -1$; M_1- и M_2-последовательности максимальной длины с проверочными полиномами $h_1(X)$ и $h_2(X)$, выбранные для образования кода Голда. Обозначим соответствующие коды максимальной длины как K_1 и K_2 . Каждый код состоит из нулевой последовательности и всех циклических перестановок последовательности максимальной длины. Ансамбль последовательности Голда является прямым произведением множеств K_1 и K_2.

Упорядочим эти последовательности в матрице $\overline{A}$ следующим образом. В строках с номерами 1, $2^k \cdot j + 1$, j=1,2, …, 2^K-1 разместим слова кода K_1 в порядке возрастания информационной части (первые k символов), т.е. в первой строке запишется слово из нулей A_0, в строке с номером 2^K+1 – слово A_1, первые k символов которого образуют число 1, а в строке с но-

мером $2^K \cdot 2+1$ – слово A_2, первые k символов которого образуют число 2, и т.д. Эти строки называют *образующими*. В остальных строках матрицы $\overline{A}$ запишем суммы по модулю два элементов кода K_2 с соответствующей образующей строкой A_j. Построенная таким образом матрица $\overline{A}$ имеет следующий вид

$$\overline{A} = \begin{bmatrix} 0 & 0 & .... & 0 & | 0 & .... & 0 \\ & & A_0 & & | & +K_2 & \\ 0 & 0 & .... & 1 & | & ..... & \\ & & A_1 & & | & +K_2 & \\ 1 & 1 & .... & 1 & | & .... & \\ & & A_{2^{k-1}} & & | & +K_2 & \end{bmatrix}$$

Матрица представляет собой разложение кода Голда на смежные классы по подкоду K_2.

Для реализации декодирования кодов Голда по методу максимального правдоподобия необходимо вычислить векторно-матричное произведение (ВМП) кодовой матрицы и входного вектор-сигнала. Для декодирования кодов Голда применяют два алгоритма вычисления ВМП, которые различаются по способу факторизации кодовой матрицы.

Переставим столбцы матрицы $\overline{A}$ при помощи подстановки (8.1) и получим матрицу Адамара. Известно, что для матриц Адамара справедливо рекуррентное соотношение

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_{2^{k+1}} = \begin{bmatrix} H_{2^k} & H_{2^k} \\ H_{2^k} & -H_{2^k} \end{bmatrix}.$$

Из чего следует, что матрицу A можно представить в виде

$$A = \begin{bmatrix} B_1 & B_2 \\ B_1 & -B_2 \end{bmatrix},$$

где B_1 – матрица размера $2^{2k-1} \times (2^{k-1} - 1)$, а B_2 – матрица размером $2^{2k-1} \times 2^{k-1}$. Эта процедура деления может быть продолжена рекурсивно с матрицами B_1 и B_2.

Разобьем вектор входного сигнала на две части $X = (X_1, X_2)^T$, где X_1 имеет длину $2^{k-1} - 1$, а X_2 – длину 2^{k-1}. Тогда:

$$Y = A \cdot X = (B_1 X_1 + B_2 X_2, B_1 X_1 - B_2 X_2)^{\mathsf{T}}.$$

Если известны произведения $B_1 X_1$, $B_2 X_2$, то для вычисления AX требуется выполнить 2^{2k} операций. Эти вычисления сводятся к умножению вектора $Y_{k-1} = (B_1 X_1, B_2 X_2)^T$ на матрицу

$$C_k = H_2 \otimes I_2 \otimes .... \otimes I_2,$$

где I_2 – единичная матрица порядка два, а знак $\otimes$ обозначает прямое произведение матриц. Для вычисления произведения $B_1 X_1$ и $B_2 X_2$ можно снова воспользоваться структурными свойствами матриц B_1 и B_2, а именно

$$B_1 = \begin{bmatrix} l_1 & l_2 \\ l_1 & -l_2 \end{bmatrix} \quad B_2 = \begin{bmatrix} q_1 & q_2 \\ q_1 & -q_2 \end{bmatrix}.$$

Блок l_1 имеет размер $2^{2k-2} \times (2^{k-2} - 1)$, а блоки l_2, q_1, q_2 – размер $2^{2k-2} \times 2^{k-2}$. Аналогично предыдущему выполнение этих операций эквивалентно умножению на матрицу

$$C_{k-1} = I_2 \otimes H_2 \otimes I_2 \otimes .... \otimes I_2.$$

Эта итеративная процедура закончится на *i*-м шаге, где *i* равно минимальному решению неравенства

$$2k - i \geq 2^{k-i}.$$

Неравенство определяет размер блочных подматриц, на которые делится матрица *A*. Блоки первого столбца имеют размер $2^{2k-i} \times (2^{k-i} - 1)$, а остальные блоки – размер $2^{2k-i} \times 2^{k-i}$. Все они образуются первыми 2^{2k-i} строками матрицы *A*. Число различных блоков равно 2^i. Обозначим их $\gamma_0, \gamma_1, ..., \gamma_{2^i - 1}$. Характерной чертой матриц γ_j, *j*=1,…, $2^i - 1$ является то, что в своих строках они содержат либо полный код, либо часть слов полного кода с повторениями. Исходная матрица теперь может быть записана в следующей факторизованной форме

$$A = C_k C_{k-1}, ..., C_{k-i-2} \operatorname{diag}(\gamma_0, \gamma_1, ..., \gamma_{2^i - 1}).$$

Умножение вектора на любую их матриц γ_j не сложнее, чем умножение на матрицу полного кода. Поэтому количество операций для вычислений не превышает $2^{2k} s(2^{k-i})$, где $s(2^{k-i}) \approx 0.5$.

8.3. Вопросы и задания для самопроверки

8.1. Какие коды называются низкоскоростными? Приведите примеры.

8.2. Запишите выражения для определения АКФ и ВКФ последовательностей.

8.3. Каким образом формируются коды максимальной длины? Приведите пример формирования *M*-последовательности для длины 15.

8.4. Каковы основные параметры кодов максимальной длины?

8.5. Постройте схему формирователя *M*-последовательностей на основе генератора поля Галуа.

8.6. Приведите схему рекуррентного генератора *M*-последовательности, задаваемого полиномом $h(x) = 1 + x^2 + x^3$.

8.7. Сформулируйте алгоритм формирования кодов Голда.

8.9. Постройте схему формирования кодов Голда для длины 15.

8.10. Определите все возможные длины квадратично-вычетных последовательностей в диапазоне чисел от 10 до 100.

8.11. Сформулируйте алгоритм формирования квадратично-вычетных кодов.

8.12. Постройте квадратично-вычетный код длиной n=11 и практически определите его корреляционные свойства.

8.13. Какая математическая процедура лежит в основе корреляционного декодирования кодов?

8.14. Назовите достоинства и недостатки корреляционного декодирования низкоскоростных кодов.

8.15. Какой тип преобразования используется при декодировании кодов максимальной длины?

8.16. Сформулируйте алгоритм перестановки строк и столбцов матрицы кодовых слов *M*-последовательностей, который приводит ее к матрице типа Адамара.

8.17. Назовите основные шаги алгоритма декодирования кодов максимальной длины и приведите пример.

8.18. Определите вычислительную сложность умножения вектора на матрицу Адамара размером 64×64.

8.19. Постройте структурную схему устройства декодирования кодов максимальной длины.

8.20. Сформулируйте алгоритм быстрого декодирования кодов методом максимального правдоподобия.

8.4. Практическое занятие №8

Формирование и декодирование низкоскоростных кодов

Теория и методы решений для практического занятия представлены в разделах 8.1 и 8.2.

Тестовые задания

1. Для низкоскоростных кодов выполняется условие:

a) $k >> r$; b) $k << r$; c) $k \approx r$; d) $k = n/2$.

2. На основе какого выражения можно определить автокорреляционную функцию двоичной последовательности?

a) $R\ (\tau) = \sum_{l=0}^{n-1} (1)^{u_l + u_{l+\tau}}$;

b) $R\ (\tau) = \sum_{l=0}^{n-1} (-1)^{u_l + u_{l+\tau}}$;

c) $R_{uv}(\tau) = \sum_{i=0}^{n-1} (1)^{u_i + v_{i+\tau}}$;

d) $R_{uv}(\tau) = \sum_{i=0}^{n-1} (-1)^{u_i + v_{i+\tau}}$.

3. Коды максимальной длины имеют параметры:

a) $n = 2^{k-1} - 1,\ d = 2^{k-1}$;

b) $n = 2^k - 1,\ d = 2^{k-1}$;

c) $n = 2^k - 1,\ d = 2^{k-2}$;

d) $n = 2^k,\ d = 2^{k-1}$.

4. Сколько начальных символов необходимо при формировании *M*-последовательности длиной 15?

a) 3; b) 4; c) 5; d) 15.

5. Какой длины потребуется регистр при формировании *M*-последовательности длиной 255?

a) 255; b) 8; c) 510; d) 126.

6. Для каких длин можно построить квадратично-вычетный код:

a) 11; b) 13; c) 15; d) 17.

7. Матрица Адамара размером 4×4 имеет вид:

a) $H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$;

b) $H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$;

c) $H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$; d) $H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$.

8. Чтобы привести матрицу кодовых слов максимальной длины к матрице типа Адамара, необходимо выполнить:

a) перестановку строк;

b) перестановку столбцов;

c) увеличить размерность кодовой матрицы в 2 раза;

d) «правый» циклический сдвиг опорной *M*-последовательности.

9. Матрица кодов Голда имеет размерность:

a) $2^{2 \cdot n} \times n$; b) $2^{2 \cdot k-1} \times n$; c) $2^{2 \cdot k} \times n$ d) $2^{2 \cdot k} \times 2n$.

10. Каков основной недостаток корреляционного декодирования низкоскоростных кодов:

a) низкая помехоустойчивость;

b) высокая вычислительная сложность;

c) возможность применения только для биортогональных кодов;

d) невозможность использования для нелинейных кодов.

Задачи

1. Рассчитайте автокорреляционную функцию последовательности (1, 1, -1, -1, 1, -1 ,1).

2. Для последовательности максимальной длины

(1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1)

2.1. постройте код максимальной длины;

2.2. постройте генератор данной *M*-последовательности и проверьте его работу;

2.3. преобразуйте матрицу кода в матрицу Адамара;

2.4. запишите получившиеся перестановки строк и столбцов;

2.5. определите вычислительную сложность декодирования данного кода.

3. Используя проверочные полиномы

$$h_1(x) = 1 + x + x^3,\ h_2(x) = 1 + x^2 + x^3$$

3.1. сформируйте коды максимальной длины;

3.2. используя строки первой матрицы в качестве образующих, постройте код Голда;

3.3. постройте генератор кодов Голда и проверьте его работу;

3.4. на основе полученного кода приведите пример построения быстрого алгоритма декодирования кодов Голда.

Модуль 9

КОДОВЫЕ МЕТОДЫ ПОВЫШЕНИЯ НАДЕЖНОСТИ ЦИФРОВЫХ УСТРОЙСТВ

Цель модуля – изучение понятий дефекта, модели канала с дефектами, кодов, позволяющих контролировать дефекты, методов кодирования и декодирования.

В результате изучения модуля студенты должны:

- знать различия в понятиях дефекта и ошибки;
- иметь представление о различиях моделей каналов с ошибками и дефектами;
- знать коды, контролирующие дефекты;
- знать алгоритмы кодирования и декодирования кодов исправляющих дефекты.

Содержание модуля

9.1. Коды, исправляющие дефекты

9.1.1. Понятие канала с дефектами

Увеличение кратности корректируемых ошибок приводит к резкому возрастанию аппаратурных и временных затрат на исправление ошибок. Ниже на примере исправления ошибок в канале хранения показывается, что применение понятия «дефект» позволяет существенно снизить эти затраты.

При использовании избыточности совокупность элементов памяти (ЭП) накопителя запоминающего устройства (ЗУ) рассматривается как канал передачи информации, в котором последняя передается не в пространстве, а во времени. Для нейтрализации ошибок подлежащая хранению информация кодируется корректирующим кодом. При считывании происходит декодирование, во время которого ошибочная информация исправляется. Анализ особенностей канала хранения показывает следующее:

- изменяя конструкцию ЭП и ЗУ, можно варьировать характер наиболее вероятных ошибок в канале хранения, т.е. управлять состоянием канала;
- местоположение и состояние отказавших ЭП может быть известно до занесения информации в ЗУ.

Указанные особенности приводят к рассмотрению устройств хранения как специфического канала передачи информации. Возникающие при этом ситуации достаточно хорошо описываются с помощью обобщенной модели канала хранения (рис. 9.1).

На рис. 9.1 обратная связь с выхода декодера на вход кодера характеризует необходимость предварительного исправления ошибок в кодовом слове при записи информации только в часть разрядов слова, например в один ЭП, что наблюдается в одноразрядных БИС ЗУ. Блок управления состоянием канала характеризует возможность получения наиболее вероятных ошибок в канале путем изменения конструкции ЭП БИС и блоков памяти для более полного согласования ошибок хранения с имеющейся избыточностью.

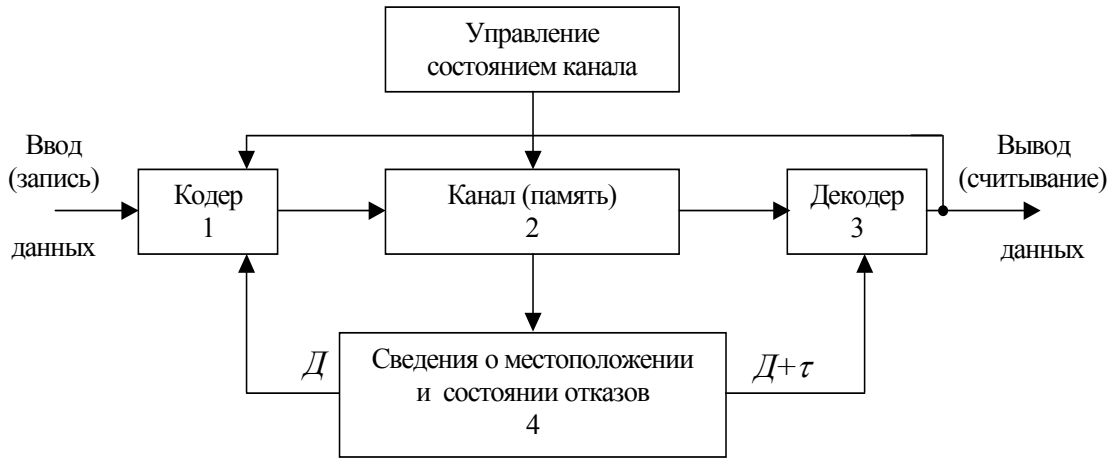


Рис. 9.1. Обобщенная модель канала хранения
с корректирующими кодами

Если сведения о местоположении отказов отсутствуют или не используются, то в качестве модели канала хранения берется обычный двоичный канал с ошибками, в котором блоки записи и считывания совмещены с кодером и декодером, а также совмещены отдельные элементы этих блоков (см. рис. 9.1). Если имеются сведения о местоположении $Д$ отказов при отсутствии ошибок (t=0), то благодаря этим сведениям при записи и считывании можно использовать исправные резервные элементы, исключив обра-

щение к отказавшим; в результате канал становится эквивалентным каналу без ошибок (см. рис. 9.1). Однако может возникнуть ситуация, когда до записи данных имеется $Д$ отказавших разрядов, а после записи и хранения данных отказывают еще τ разрядов. В этом случае канал неэквивалентен каналу без ошибок, так как кодеру известно местоположение $Д$ отказов, а декодеру – $Д+\tau$. При этом возможные методы кодирования и декодирования зависят от наличия у декодера сведений о местоположении $Д$ и τ отказов. Если декодеру известно местоположение $Д$ разрядов, отказавших до записи данных, и информация, подлежащая хранению в этих разрядах, то при появлении в процессе хранения τ отказов, истинные состояния которых неизвестны, канал становится подобен каналу со стираниями (τ-стирания). При $Д$=0 канал полностью отражает достаточно хорошо изученную в теории кодирования задачу об исправлении стираний (см. рис. 9.1).

Если же местоположение отказавших разрядов неизвестно декодеру, а известно кодеру и τ=0, что может иметь место как для постоянной, так и для оперативной памяти благодаря средствам диагностики, то для данной ситуации канал получил название *канала с дефектами* (см. рис. 9.1). Путем сочетания приведенных моделей каналов хранения можно получить и другие реально наблюдаемые на практике ситуации и более эффективно использовать избыточное кодирование. Так комбинация моделей каналов с ошибками и канала с дефектами приводит к каналу хранения с дефектами и ошибками, когда местоположение ошибок, произошедших в процессе хранения из-за t сбоев или отказов ЭП, неизвестно, а из-за $Д$ дефектов (отказов) до записи информации известно благодаря, например, сведениям, полученным от специальных блоков контроля или двукратного выполнения цикла записи – считывания и сравнения исходной информации с инвертированной (рис. 9.2).

Применение предложенных моделей каналов хранения информации позволяет более полно согласовать состояние канала с вводимой избыточностью, уменьшить ее, сформулировать требования к корректирующему коду, предложить эффективные методы защиты памяти от многократных ошибок. Так, например, в зависимости от рассматриваемой модели каналов хранения отказы в памяти могут исправляться как ошибки, стирания, дефекты, их сочетания и использоваться решения, требующие различного уровня избыточности. Окончательное решение принимается, как правило, лишь после сравнения различных вариантов с учетом особенностей организации, специфики применения защищаемых устройств памяти, требуемых технических параметров.

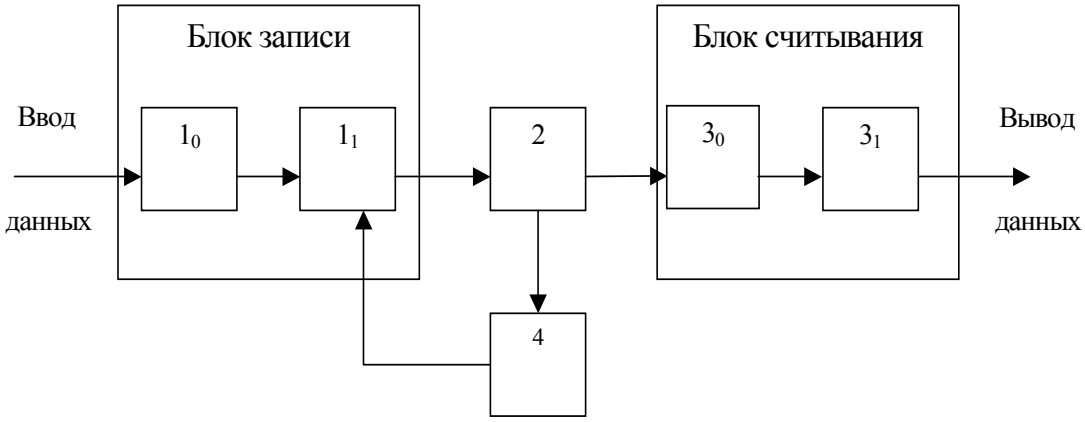


Рис. 9.2. Модель канала хранения с кодами,
исправляющими дефекты и ошибки

9.1.2. Задание кодов, исправляющих дефекты

При описании кодов под дефектными понимаются ЭП, которые при считывании всегда выдают либо логический «0», либо логическую «1» и не изменяют своего состояния под действием управляющих сигналов. Считается, что местоположение таких ЭП известно или может быть определено при помощи специальных измерений при кодировании информации. Проблема заключается в том, чтобы найти способ сообщить эти сведения потребителю в достаточно компактной и универсальной форме, отличной от тривиального перечисления адресов дефектных ЭП.

Основная идея надежного хранения информации в памяти с дефектами заключается в следующем. Пусть требуется хранить двоичное слово (вектор) u длины k. Символы этого слова могут не совпадать с состоянием дефектных ЭП ЗУ. Тогда можно попытаться найти некоторый покрывающий вектор C, который при суммировании с вектором u давал бы вектор $\varphi=u+C$, такой, чтобы его символы, предназначенные для хранения в дефектных ЭП, совпадали с состояниями этих ЭП. Следовательно, если занести в память вектор φ, то он будет храниться без искажений. При считывании следует выполнить обратную операцию, т.е. вычислить $u=\varphi+C$. Для того чтобы сообщить потребителю сведения об используемом векторе C, следует ввести r дополнительных разрядов, которые называются *проверочными*. Построенные таким образом коды получили название *аддитивных*.

Из приведенных рассуждений следует, что аддитивный код задается покрывающей (ξ,n)-матрицей C, строками которой являются все покрывающие векторы, и кодирующей функцией $C(u, Д)$, сопоставляющей с сообщени-

ем *и* и дефектом *Д* одну из строк матрицы *С*. Символ, записываемый в дефектный ЭП, называется *согласованным* с дефектом, если состояние ЭП совпадает со значением символа, и *несогласованным* в противном случае.

Пара (*и, Д*) может быть охарактеризована *n*-мерным вектором ω=(--0-11-) с нулевыми символами на согласованных позициях и единичными на несогласованных. Говорят, что строка матрицы *С* покрывает дефект *Д*, если она совпадает с ω в дефектных позициях. Для фиксированной пары (*и, Д*) значением С(*и, Д*) является любая из покрывающих строк матрицы *С*. Из конструкции аддитивного кода вытекает, что матрица *С* должна удовлетворять следующим условиям:

- строки *С* должны покрывать все дефекты кратности *t*, т.е. подматрица, составленная из любых *t* столбцов *С*, должна содержать в своих строках все 2^t двоичных чисел длиной *t*;
- в матрице должна существовать подматрица из *r* столбцов, строки которой содержат без повторений ξ двоичных чисел длиной *r*. Эта подматрица называется *контрольной.* Таким образом, построение кода сводится к построению матрицы *С* с указанными свойствами.

Групповые аддитивные коды

Эти коды строятся на базе проверочных матриц линейных кодов. Введем некоторые определения. Пусть $\tilde{u} = (0, ..., 0, u_{r+1}, ..., u_n)$ – слово длиной *n*, первые *r* символов которого нули, а последние *k*=(*n*-*r*) – символы, подлежащие хранению. Пусть *С* – матрица, строками которой являются все слова линейного корректирующего кода с порождающей матрицей *G*. Напомним, что строками *С* являются линейные комбинации строк матрицы *G* и строка, состоящая из нулей.

Теорема 9.1. Матрица *С* задает код, исправляющий все дефекты кратностью *t*, если в качестве порождающей матрицы *G* этого кода использована проверочная матрица *H* систематического линейного кода с минимальным расстоянием, не меньшим чем *t*+1 ($d \le t+1$), а функция *С*(*и, Д*) определяется следующим образом

$$C(u,\ Д) = v^* \cdot G\ , \tag{9.1}$$

где v^* – матрица-строка с *r* элементами, представляющая собой одно из решений системы *t* ($r \ge t$) линейных уравнений

$$v \cdot C_d = \omega. \tag{9.2}$$

В подматрице C_d (с размерами $r \cdot t$) *i*-й столбец является столбцом матрицы *G* с номером, равным номеру *i*-й слева дефектной позиции, а ω – матрица-строка с *t* элементами. *i*-й элемент соответствует *i*-й слева де-

фектной позиции и равен нулю, если эта позиция у $\tilde{u}$ и $Д$ согласована, и равен единице в противном случае.

Таким образом, строки матрицы C фактически являются словами линейного кода, двойственного коду с минимальным расстоянием $d \leq t+1$. Теорема также показывает, что при помощи линейного кода с исправлением t ошибок можно построить групповой аддитивный код с исправлением $2t$ дефектов. Кроме того, линейный код, обнаруживающий t модулей ошибок, исправляет t модулей дефектов, а код с исправлением t модулей ошибок исправляет $2t$ модулей дефектов.

Решение v^* системы $v \cdot C_d = \omega$ можно найти путем подбора, приняв на первом шаге v=(0... 0 1), и увеличением его на единицу до тех пор, пока не выполнится равенство. Весь алгоритм работы вычислителя для занесения кода в ЗУ показан на рис. 9.3.

При декодировании по префиксной части (проверочным символам) восстанавливается соответствующая строка покрывающей матрицы, которая далее суммируется по модулю два со считанным из ЗУ вектором.

Пример. Код для исправления одиночного дефекта в слове имеет один избыточный разряд. Матрица C содержит две строки длиной n, одна из которых состоит целиком из нулей, а другая – из единиц. Таким образом, матрица C имеет вид

$$C = \begin{bmatrix} 0 & 0 & 0 & . & . & . & 0 & 0 \\ 1 & 1 & 1 & . & . & . & 1 & 1 \end{bmatrix}. \qquad (9.3)$$

Функция $C(u, Д)$ определяется следующим образом: $C(u, Д)$=(0, ... 0), если u согласуется с дефектом $Д$; $C(u, Д)$=(1, ... 1), если u не согласуется с дефектом $Д$.

Следовательно, если код согласован с дефектом, то его информационные разряды сохраняются без изменения, а в единственный проверочный разряд записывается нулевой символ. Если код не согласован с дефектом, то его информационные разряды инвертируются, а в проверочный разряд записывается единичный символ.

Декодирование происходит следующим образом. Если на проверочной позиции расположен нулевой символ, то никаких действий над выбранным словом не производится. Если на проверочной позиции стоит единичный символ, то это говорит о несогласовании записанной информации с состоянием дефекта и, следовательно, все считываемое слово инвертируется. Таким образом, в ошибочных словах инвертирование производится дважды – при записи и при считывании. Для исправных элементов памяти эти инвертирования не изменяют выходного символа. Для дефект-

ных элементов памяти инвертирование записываемого слова при записи позволяет согласовать состояние дефекта с записываемой информацией, а инвертирование при считывании – восстановить исходную информацию.

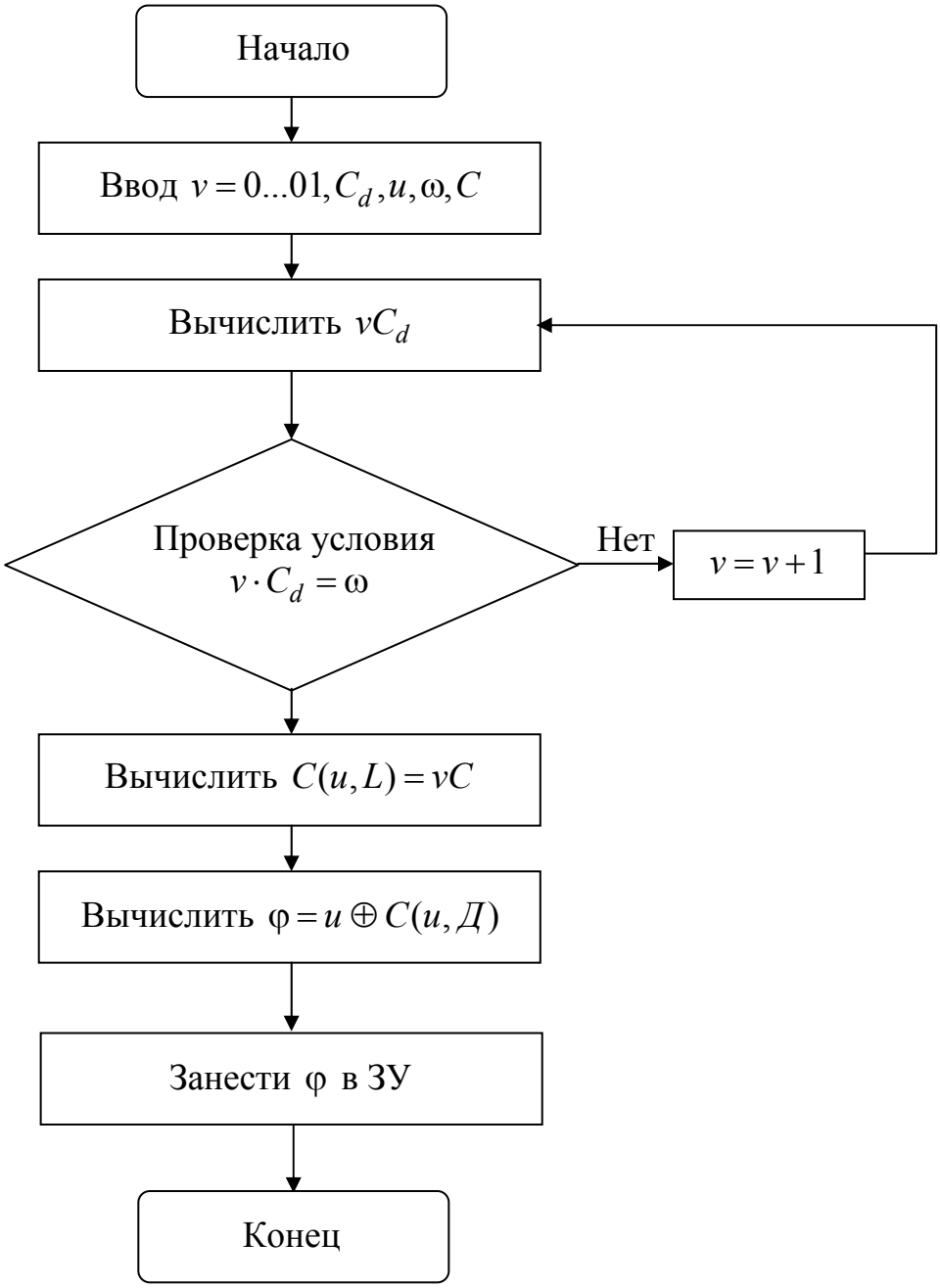


Рис. 9.3. Алгоритм работы вычислителя для занесения в ЗУ группового кода, исправляющего дефекты

Пример. Рассмотрим нахождение кодирующей функции *C(u, Д)* и построение декодирующего устройства, позволяющего исправить дефекты кратности два. Пусть, например, дефекты расположены на позициях 3, 5 и

число информационных символов равно k=4. Матрицей C^* в этом случае может быть проверочная матрица кода Хэмминга (7;4)

$$H=\begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}=\begin{bmatrix}1&0&0&1&1&0&1\\0&1&0&1&0&1&1\\0&0&1&0&1&1&1\end{bmatrix},$$

$$C^*=\begin{bmatrix}1&0&0&1&1&0&1\\0&1&0&1&0&1&1\\0&0&1&0&1&1&1\end{bmatrix},$$

$$C_d=\begin{bmatrix}0&1\\0&0\\1&1\end{bmatrix}.$$

Предположим, что в проверочной позиции 3 дефектный элемент памяти находится в состоянии 0, а в информационной позиции 5 записываемый символ не согласован с состоянием дефекта, тогда ω=(01). Далее решается система

$$(v_1v_2v_3)\cdot\begin{bmatrix}0&1\\0&0\\1&1\end{bmatrix}=01$$

$$\text{или}\ \begin{cases}v_3=0\\v_1+v_3=1\end{cases}.$$

Одно из решений этой системы:

v_1=1, v_2=0, v_3=0,

т.е. v=(100).

Поэтому

$$C(u,Д)=(100)\cdot\begin{bmatrix}1&0&0&1&1&0&1\\0&1&0&1&0&1&1\\0&0&1&0&1&1&1\end{bmatrix}=(1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1).$$

Отсюда при записи информации в проверочные разряды слова, содержащего дефектные элементы памяти, заносится код (100), а информационные разряды инвертируются с учетом кода (1101). Таким образом, состояние дефектной проверочной позиции 3 согласовано с записываемой проверочной информацией, а состояние дефектной информационной позиции 5 – с программируемым кодом.

Очевидно, что нахождение вектора $C(u, Д)$ по алгоритму (см. рис. 9.3) связано с большими временными затратами. Упростить и ускорить процедуру кодирования при исправлении дефектов кратности t позволяет метод комбинированного кодирования. Сущность метода состоит в кодировании на первой ступени слова кодом для исправления ошибок с кодовым расстоянием d. На второй ступени происходит кодирование кодом для исправления одиночных дефектов. Если число несогласованных ошибок больше половины t, то слово, подлежащее записи в память, инвертируется, в противном случае оно остается без изменения. Корректирующая способность метода совпадает с корректирующей способностью предыдущего при четных d и на единицу выше при нечетных d, т.е. для нечетных $d=t$. Действительно, например, при применении кодов с t=3,5 можно исправить данным методом d=3, 5 дефектов, т.к. соответственно при 2, 3 и 3, 4, 5 несогласованных дефектах и инвертировании эти дефекты становятся согласованными, а возможное появление 1 и 1, 2 несогласованных дефектов (ранее согласованных) будет исправлено как ошибки кодами с d=3, 5.

Необходимым условием такой коррекции дефектов служат коды, словами которых являются как прямые, так и инверсные слова, например, транспарантных кодов, в сочетании с отведением одного из информационных разрядов кода для исправления ошибок под информацию о метке: прямое или инверсное слово хранится в памяти. Как будет показано ниже, в качестве таких кодов могут быть выбраны смежные коды для исправления одного дефекта и t ошибок. Нетрудно убедиться, что требуемое при данном методе число проверочных разрядов при четных d на единицу больше, а при нечетных d совпадает с числом проверочных разрядов групповых аддитивных кодов для исправления дефектов. Если согласовать вектор u с дефектом $Д$ в максимальном числе разрядов при условии, что несогласованные разряды не принадлежат первым r разрядам, то комбинированное кодирование приводит к конструкциям с более высокой скоростью передачи при больших n.

Негрупповые аддитивные коды

Эти коды имеют меньшее, по сравнению с групповыми, количество проверочных разрядов, поскольку для построения кодирующей матрицы используются не все линейные комбинации строк матрицы G.

Коды для исправления двойных дефектов

Конструкции лучших кодов даются следующими теоремами.

Теорема 9.2. Пусть B – матрица, столбцами которой являются все двоичные числа длиной $l > 2$, B – матрица, полученная из $\overline{B}$ инвертированием всех элементов $(0 \to 1, 1 \to 0)$. Тогда код с $n=2^l$, задаваемый матрицей

$$C = \begin{bmatrix} B \\ \overline{B} \\ 1\ 1\ \ .\ \ .\ \ .\ \ .\ \ 1 \\ 0\ 0\ \ .\ \ .\ \ .\ \ .\ \ 0 \end{bmatrix}, \qquad (9.4)$$

исправляет все дефекты кратностью два и содержит $r = \lceil \log 2(l+1) \rceil$ проверочных символов.

Например, матрица C, определяемая теоремой 9.2 для $n=8$, имеет вид

$$C^{**} = \begin{bmatrix} 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0 \\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{bmatrix} = \begin{bmatrix} 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1 \\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1 \\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0 \\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0 \\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0 \\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{bmatrix}.$$

Некоторое улучшение, достигаемое за счет присоединения контрольных разрядов к $\left[B : \overline{B}\right]^T$, можно получить, если воспользоваться следующей конструкцией. Пусть C_l – матрица вида (9.5), где матрицы B и $\overline{B}$ определены в теореме 9.2; I – единичная матрица с числом строк, равным $r = \lceil \log 2(l+1) \rceil$, $l > 2$; $\overline{I}$ – матрица, полученная из I циклическим сдвигом строк на одну позицию вниз и инвертированием всех элементов; L – матрица с числом строк ℓ-1. Строки L должны быть различными, не быть инверсией друг друга и иметь вес, отличный от 0, 1, r-1, r. В остальном матрица L может быть произвольной. Матрица $\overline{L}$ образуется из L инвертированием всех элементов.

Теорема 9.3. Код, задаваемый матрицей (9.5), исправляет все дефекты кратностью два и содержит $r = \lceil \log 2(l+1) \rceil$ проверочных символов.

Доказательство теорем 9.2, 9.3 основывается на существовании покрытий (0 1) или (1 0) в матрицах B (инверсные покрытия образуются в $\overline{B}$) благодаря наличию в кодирующих матрицах в качестве столбцов двоич-

ных чисел. Покрытия (0 0), (1 1) даются двумя последними строками.

$$C_l = \left[\begin{array}{c:c} I & B \\ \hdashline L & \\ \hdashline \bar{I}^1 & \bar{B} \\ \hdashline \bar{L} & \\ \hdashline 1\ 1\ .\ .\ 1 & 1\ 1\ .\ .\ 1 \\ 0\ 0\ .\ .\ 0 & 0\ 0\ .\ .\ 0 \end{array}\right], \qquad (9.5)$$

Теорема 9.4. Пусть $C^{\omega}_{2\omega-1}$ – матрица размерностью $(2\omega\text{-}1)\cdot C^{\omega}_{2\omega-1}$, столбцами которой являются все слова двоичного равновесного кода длиной 2ω-1 и весом ω. Тогда код, задаваемый матрицей

$$C = \left[\begin{array}{ccccc} & & C^{\omega}_{2\omega-1} & & \\ 0 & 0 & . & . & . \ 0\ 0 \end{array}\right] \qquad (9.6)$$

размерностью $(2\omega \times C^{\omega}_{2\omega-1})$, исправляет все дефекты кратностью два и содержит $r = \lceil \log(2\omega\text{-}1) \rceil$ проверочных символов.

Доказательство основывается на использовании свойств равновесного кода. Параметры рассмотренных кодов приведены в табл. 9.1.

Таблица 9.1

Параметры негрупповых кодов для исправления двойных дефектов

Конструкция	Число проверочных символов r	Число информационных слов k
1	2	3
Теорема 9.2	3	2…5
	4	6…124
	5	125… (2^{15}=5)
Теорема 9.3	3	2…8
	4	9…128
	5	129…2^{15}
Теорема 9.4	3	2…32
	4	33…6431
	5	6432… (C^{15}_{33}-5)

Анализ данных табл. 9.1 показывает, что все три конструкции отличаются друг от друга не более чем на один проверочный символ. Поэтому определяющим фактором при выборе кода является удобство практической реализации, а окончательное решение принимается лишь с учетом назначения и специфики памяти.

9.1.3. Декодирование кодов, исправляющих дефекты

Кодирование информации наиболее просто осуществляется с помощью специализированного вычислителя, реализующего алгоритм рис. 9.3 и находящего согласующие векторы $C(u, Д)$.

При декодировании по информации, расположенной на проверочных позициях кода, исправляющего дефекты, однозначно находится согласующий вектор $C(u, Д)$. Этот вектор вычитается из принятого сообщения φ, в результате находится истинное значение переданного (хранимого) слова $\tilde{u} = \varphi + C(u, Д)$.

При использовании групповых аддитивных кодов вектор $C(u, Д)$ находится путем умножения проверочных символов на кодирующую матрицу C. Поэтому сложность и быстродействие декодирующего устройства группового кода для исправления дефектов зависят от числа единиц в кодирующей матрице C и равны затратам на блок вычисления синдрома кодов, исправляющих ошибки.

Схема декодера представлена на рис. 9.6.

Декодирование сводится к умножению хранимых символов кода проверочных разрядов (U_1, U_2, U_3) на матрицу C^* и сложению результата умножения со считанными информационными символами (U_4, U_5, U_6, U_7).

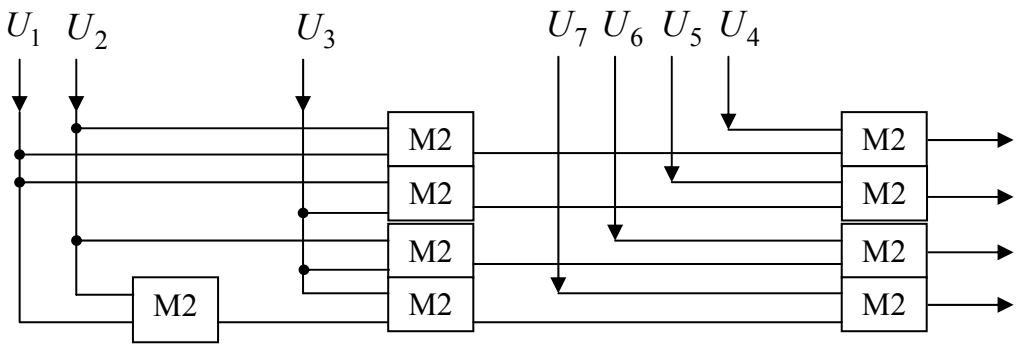


Рис. 9.6. Декодер группового кода, исправляющего дефекты

При применении негрупповых кодов для исправления дефектов декодирующая схема состоит из двух ступеней. Сначала при помощи дешифратора по считываемым проверочным символам определяется адресная часть вектора $C(u, Д)$, т.е. какая это по счету строка в матрице C.

Затем по этой части с помощью блока элементов ИЛИ синтезируется сам вектор $C(u, Д)$. Например, для кодирующей матрицы C^{**} негруппового кода с n=8, задаваемого теоремой 9.2, декодер представлен на рис. 9.7.

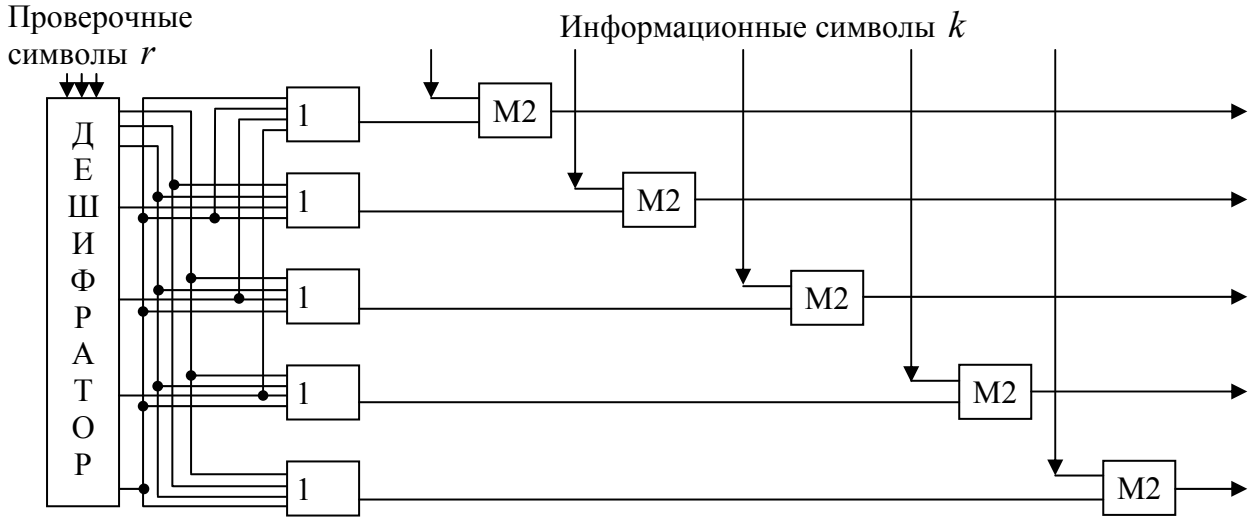


Рис. 9.7. Декодер негруппового кода, исправляющего дефекты

По сравнению с декодирующей схемой для группового кода данная реализация обеспечивает высокое быстродействие (задержку сигнала на трех элементах – дешифраторе, элементе ИЛИ, корректоре). Это весьма важно для ППЗУ, которые, как правило, являются высокоскоростными устройствами. Однако при достаточно больших r происходит усложнение элементов ИЛИ из-за увеличения числа входов и разводки к ним выходных шин дешифратора. Это приводит при микроэлектронной реализации к большим потерям площади кристалла (число входов у элементов ИЛИ примерно равно 2^{r-1}, а число шин – 2^r). Уменьшить затраты на эти схемы можно, используя в качестве матриц C слабозаполненные кодирующие матрицы с малым числом единиц в информационных разрядах. Это позволяет уменьшить число входов у элементов ИЛИ. При применении нижеследующих кодирующих матриц C_2, C_3 обеспечивается дальнейшее сокращение затрат на декодирование за счет исключения элементов ИЛИ и учета асимметрии дефектов в проверочных разрядах

$$C_2 = \left[\begin{array}{c|c} A_1 & I_k \\ \hline 1\ 1\ \ldots\ 1 & 1\ 1\ \ldots\ 1 \end{array}\right],\quad C_3 = \left[\begin{array}{cccccccc} A_2 & & & & & & & I_k \\ \hline \bar{A}_2 & & & & & & & \bar{I}_k \\ \hline 0 & 1 & 1 & 1 & 1 & 0 & \ldots & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & \ldots & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & \ldots & 0 \\ \hline 1 & 1 & \ldots & 1 & 1 & 1 & \ldots & 1 \end{array}\right], \quad (9.7)$$

где A_1 – матрица размерностью ($r \times k$), строки которой различны и не содержат слов веса $w=r$;

A_2 – матрица размерностью ($r \times k$), строки которой различны, не имеют инверсий и не содержат единичного слова и слов веса (r-1);

$\overline{A_2}$ – матрица, инверсная к A_2;

I_k – единичные матрицы;

$\overline{I_k}$ – матрица, инверсная к I_k.

Путем непосредственной проверки можно доказать, что эти матрицы задают соответственно негрупповые коды для исправления асимметричных дефектов кратностью два и три и содержат $r_2=\log(k+1)$; $r_3=\log(2k+1)$ проверочных разрядов. На рис. 9.8, 9.9 приведены декодеры, реализующие эти коды.

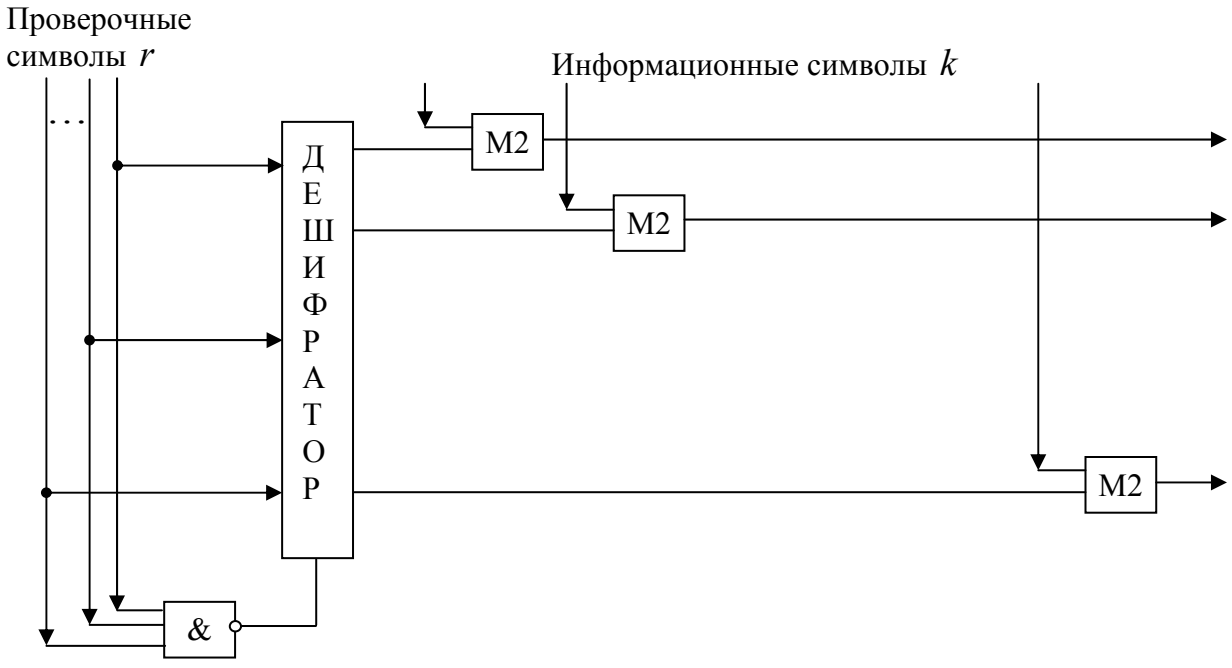


Рис. 9.8. Декодер негруппового кода,
исправляющего асимметричные дефекты кратностью два

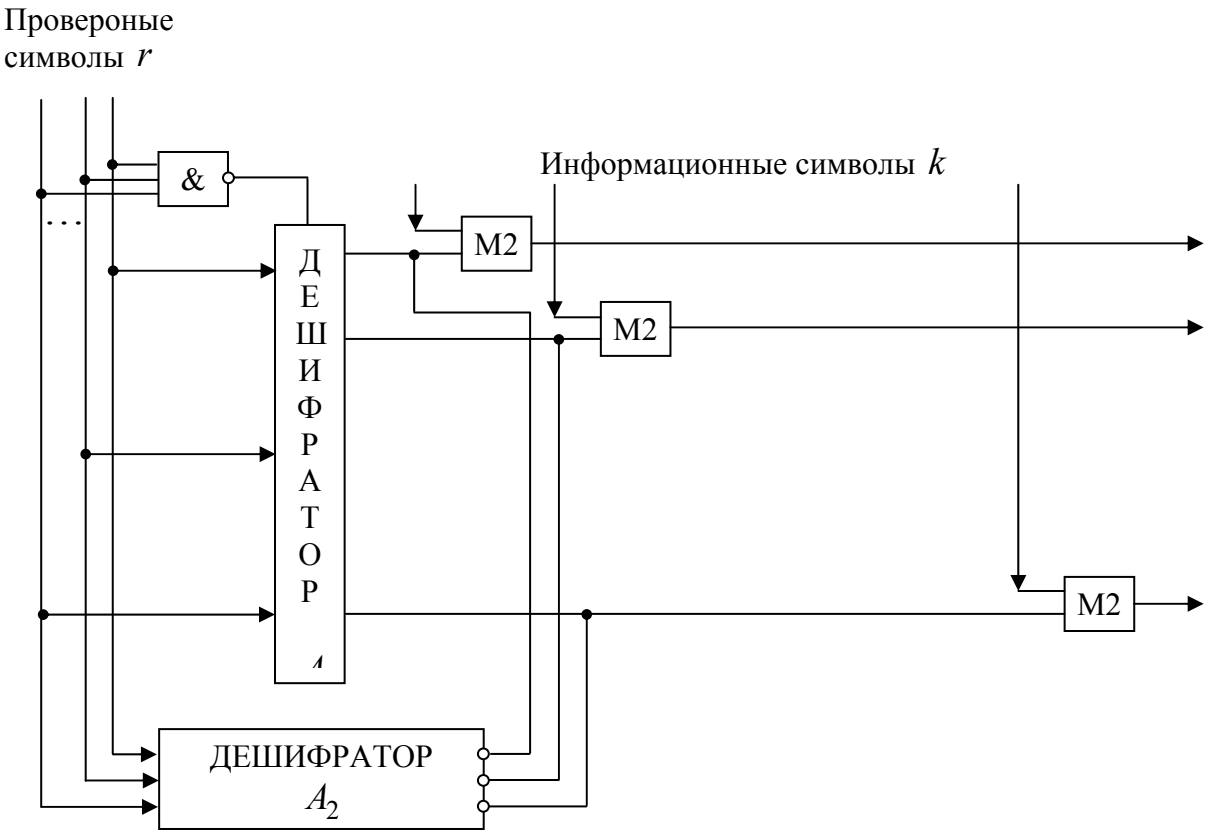


Рис. 9.9. Декодер негруппового кода,
исправляющего асимметричные дефекты кратностью три

Отличительной особенностью их является то, что при установлении сигнала лог. «0» на выходе элемента И-НЕ: дешифраторы стробируются и на их выходах устанавливаются сигналы лог. «1». Одноименные выходы дешифраторов A_2 и $\overline{A_2}$ объединены и реализуют функцию «проводное ИЛИ», что возможно, поскольку всегда один дешифратор открыт, а второй закрыт. Платой за более простую реализацию декодирующих схем является повышение числа проверочных разрядов по сравнению с параметрами лучших негрупповых кодов (на два – пять символов при k = 32–512).

9.2. Вопросы и задания для самопроверки

9.1. Что такое дефект?

9.2. Поясните отличие модели канала для записи и воспроизведения информации в системах памяти от моделей каналов передачи данных.

9.3. В чем отличие задания кодов для исправления дефектов от кодов, исправляющих ошибки?

9.4. Объясните способы задания групповых и негрупповых кодов, исправляющих дефекты, на примере коррекции трехкратных дефектов на длине кода n=16.

9.5. Объясните механизм исправления дефектов помехоустойчивыми кодами.

9.6. Для чего находятся и как определяются согласующие слова кода, исправляющего дефекты?

9.7. Для кода (15; 11) с кодовым расстоянием 3 записать порождающую матрицу, найти кодовое слово $C(u, Д)$ при условии, что имеется два дефекта, из которых первый является согласованным.

9.8. Какой кратности дефекты позволяет исправлять код исправляющий дефекты, построенный на основе группового кода с кодовым расстоянием $d = 7$.

9.9. Объясните алгоритм работы вычислителя для занесения в ЗУ группового кода, исправляющего дефекты.

9.10. Поясните принципиальные отличия декодера негруппового кода, исправляющего асимметричные дефекты кратностью два и три.

ЛИТЕРАТУРА

1. Аршинов М.Н., Садовский Л.Е. Коды и математика. – М.: Наука, 1983
2. Берлекэмп Э. Алгебраическая теория кодирования: Пер. с англ./ Под ред. С.Д. Бермана. – М.: Мир, 1971
3. Блейхут Р. Теория и практика кодов контролирующих ошибки. – М.: Мир, 1986
4. Кларк Дж.. мл., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи: Пер. с англ. – М.: Радио и связь, 1987
5. Мак-Вильямс Ф.Дж., Слоэн Н. Дж. А. Теория кодов, исправляющих ошибки: Пер. с англ./ Под ред. Л.А. Бассалыго. – М.: Связь, 1979
6. Молдовян А.А., Молдовян Н.А., Советов Б.А. Криптография. – СбП.: Лань, 2000
7. Основы теории передачи информации. Ч. I. Экономное кодирование /В.И. Шульгин. – Учеб. пособие. – Харьков: Нац. аэрокосм. ун-т «Харьк. авиац. ин-т», 2003
8. Основы теории передачи информации. Ч. 2. Помехоустойчивое кодирование /В.И. Шульгин. – Учеб. пособие. – Харьков: Нац. аэрокосм. ун-т «Харьк. авиац. ин-т», 2003
9. Теория прикладного кодирования. В 2 т. Т. 1: Учеб. Пособие /В.К. Конопелько, В.А. Липницкий, В.Д. Дворников и др.; Под ред. проф. В.К. Конопелько. – Мн.: БГУИР, 2004
10. Теория прикладного кодирования. В 2 т. Т. 2: Учеб. Пособие /В.К. Конопелько, В.А. Липницкий, В.Д. Дворников и др.; Под ред. проф. В.К. Конопелько. – Мн.: БГУИР, 2004
11. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки: Пер. с англ. /Под ред. Р.Д. Добрушина и С.И. Самойленко. – М.: Мир, 1976

Учебное издание

Составители
БОГУШ Рихард Петрович
КУРИЛОВИЧ Андрей Владимирович

ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ

Учебно-методический комплекс
для студентов специальности 1-39 01 01
«Радиотехника»

Редактор Т.А. Дарьянова

Подписано в печать 14.06.05. Формат 60x84 1/16. Гарнитура Таймс. Бумага офсетная.
Отпечатано на ризографе. Усл. печ. л. 14,85. Уч.-изд. л. 12,42. Тираж 60. Заказ №

Издатель и полиграфическое исполнение
Учреждение образования «Полоцкий государственный университет»

ЛИ № 02330/0133020 от 30. 04. 04 г. ЛП № 02330/0133128 от 27.05.04

211440, г. Новополоцк, ул. Блохина, 29