ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М.А. БОНЧ-БРУЕВИЧА»

О.С. Когновицкий, В.М.Охорзин

ТЕОРИЯ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

ЧАСТЬ 1
ЦИКЛИЧЕСКИЕ КОДЫ

Учебное пособие

СПб ГУТ)))

САНКТ-ПЕТЕРБУРГ
2013

УДК
ББК

Рецензенты:
доктор технических наук *В. И. Комашинский* (СПб ГУТ) ,
кандидат технических наук *А. А. Березкин* (ЗАО НПП ИСТА-Системс).
Утверждено редакционно-издательским советом СПбГУТ в качестве учебного пособия

Когновицкий ,О.С.
Теория помехоустойчивого кодирования. Ч 1. Циклические коды: /О.С. Когновицкий, В.М. Охорзин; – СПб.: СПбГУТ, 2013. –84 с.

Излагаются алгебраические основы и принципы построения и реализации классов циклических кодов, используемых в современных системах телекоммуникаций и изучаемых в учебных дисциплинах СПб ГУТ: «Математическая теория помехоустойчивого кодирования», «Основы теории передачи информации», «Основы теории передачи дискретных сообщений »(для бакалавров), «Современные проблемы помехоустойчивого кодирования» (для магистров) по теме «Помехоустойчивые коды». Предназначено студентам специальности : 210700 «Инфокоммуникационные технологии и системы связи», профили: «Инфокоммуникационные технологии в сервисах и услугах связи», «Сети связи и системы коммутации».

Материалы пособия могут быть использованы для подготовки студентов, обучающихся по всем техническим специальностям.

УДК
ББК

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Повышение достоверности при передаче сообщений по каналам связи является одной из важнейших задач, решаемых специалистами в области телекоммуникаций. Этим вопросам уделяется большое внимание в программах подготовки инженерных кадров, бакалавров и магистров во многих высших учебных заведениях инфокоммуникационного профиля.

Как известно, одним из наиболее эффективных средств борьбы с ошибками при передаче дискретных сообщений по каналам связи с помехами является применение избыточных помехоустойчивых кодов, посредством которых большинство таких ошибок могут быть обнаружены и исправлены. Из всех известных помехоустойчивых кодов наиболее широкое применение нашли различные классы циклических кодов. Однако в последнее время, особенно в спутниковых и беспроводных системах связи, широко применяются сверточные и турбокоды с исправлением ошибок, имеющие высокую эффективность. Они обеспечивают достаточно высокую достоверность передачи сообщений в сравнительно плохих каналах связи.

В пособии «Теория помехоустойчивого кодирования» дается подробное изложение основ теории циклических кодов, сверточных и турбокодов на уровне, доступном для студентов разных факультетов, изучающих вопросы передачи и обработки информации в дискретном виде. Рассматриваются общие принципы обнаружения и исправления ошибок, а также вопросы построения кодирующих и декодирующих устройств.

Пособие издается в трех частях. Часть 1 посвящена теории и принципам построения циклических кодов. В части 2 будут рассмотрены вопросы теории и построения сверточных и турбо кодов. Часть 3 будет посвящена кодам с низкой плотностью проверок на четность и методам кодирования и декодирования рекуррентных последовательностей на основе двойственного базиса, разработанными на кафедре обработки и передачи дискретных сообщений.

В связи с широким применением помехоустойчивых кодов в различных системах передачи и обработки сообщений настоящее пособие может быть полезным для инженерно-технических работников, разрабатывающих аппаратуру связи, телемеханики и автоматики, вычислительной техники и средств хранения данных.

1. ВВЕДЕНИЕ В ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ

1.1.Основные понятия и определения помехоустойчивого кодирования

В теории передачи информации различают два понятия кодирования – кодирование источника и кодирование канала. Задачей кодирования источника является устранение избыточности дискретных сообщений с целью повышения скорости передачи информации. Второе кодирование – кодирование канала, напротив, вводит в передаваемое сообщение избыточность с целью повышения достоверности передачи информации. Введенная избыточность используется на приемной стороне для борьбы с ошибками в канале связи, вследствие этого кодирование канала называется помехоустойчивым.

Известные, широко применяемые коды источника, такие как коды Шеннона и коды Хаффмена, оптимизируют среднюю длину комбинации дискретных сообщений, которая по первой теореме Шеннона сколь угодно близко стремится, оставаясь больше к энтропии дискретного источника. Недостатками таких оптимальных кодов источника являются неравномерная длина кодовых комбинаций и размножение ошибок при декодировании принятой с ошибкой комбинации. Для уменьшения вероятности размножения ошибок в декодере источника принятую кодовую последовательность сначала декодируют в декодере помехоустойчивого кода, а затем в декодере источника. Очевидно, что большинство ошибок, которые могли бы вызвать размножение в декодере источника, будут исправлены помехоустойчивым кодом. Следует отметить, что размножения ошибок не происходит в случае использования равномерных первичных кодов (кодов источника), имеющих одинаковую длину комбинаций.

Приведем основные понятия и определения помехоустойчивых кодов.

Основание кода – это количество возможных значений отдельного символа кодовой последовательности. Наиболее широкое применение получили двоичные коды, т.е. коды с основанием 2, каждый символ которых принимает одно из двух значений, условно обозначаемых как 0 и 1. Значения символов p-ичного кода (основание кода p) принимают условные значения $0,1,2,\ldots,(p-1)$. В этом случае элементы p-ичного кода являются вычетами по $\bmod p$.

Различают *блоковые* и *непрерывные* помехоустойчивые коды.

Блоковые коды (называемые также блочными) состоят из множества B кодовых комбинаций длиной n символов каждая. Любой такой комбинации будет соответствовать определенная информационная комбинация из k символов, принадлежащая множеству A. Следовательно, количество кодовых комбинаций множества B равно количеству комбинаций множества A. Пусть это количество будет равно N. Тогда количество возможных комбинаций длины n при основании кода p будет равно p^n. Из них $N \leq p^k$

будут *разрешенными*, а $M = (p^n - N)$ – *запрещенными*. Количество запрещенных комбинаций M можно считать общей избыточностью блокового кода. Как будет показано ниже, от величины M зависят корректирующие свойства помехоустойчивого кода.

Так как любой разрешенной комбинации длины n соответствует определенная информационная комбинация длины k, будем обозначать кодовую комбинацию блокового кода в обобщенном виде (n,k).

Очевидно, что символьная *избыточность* кодовой комбинации такого равномерного (n,k)-кода будет равна $(n-k)$. А отношение числа избыточных символов к общей длине комбинации (n,k)-кода называют относительной избыточностью блокового кода, равной

$$\eta = \frac{n-k}{n} = 1 - \frac{k}{n}. \qquad (1.1)$$

Отношение числа информационных символов к общей длине кодовой комбинации (n,k)-кода называется скоростью кода и обозначается так:

$$R = \frac{k}{n}. \qquad (1.2)$$

Сравнивая (1.1) и (1.2), видим, что доля избыточных символов η выражается через скорость как $\eta = 1 - R$, и наоборот, скорость – через относительную избыточность блокового кода $R = 1 - \eta$. Отсюда следует очевидный вывод, что наиболее простой мерой эффективности блокового кода является величина $0 < R \leq 1$, т. е. чем больше R, тем меньше относительная избыточность и тем эффективнее будет код по скорости передачи информации. Однако, чем меньше относительная избыточность, тем ниже эффективность помехоустойчивого кода по корректирующим свойствам, т. е. по способности обнаруживать и исправлять ошибки. Поэтому эффективность блокового (n,k)-кода следует оценивать по меньшей мере по двум показателям – скорости и достоверности, ибо, улучшая один из них, мы ухудшаем другой показатель.

Обобщённой характеристикой кода, увязывающей скорость и корректирующие способности кода, является расстояние Хемминга d, которое равняется числу отличающихся друг от друга позиций в сравниваемых комбинациях. Параметром блокового помехоустойчивого кода является наименьшее значение хеммингового расстояния d для всех сравниваемых пар кодовых комбинаций. Этот параметр называется минимальным кодовым расстоянием и обозначается $d_{\min}$. Поэтому часто равномерный блоковый (n,k)-код, имеющий параметр $d_{\min}$, записывают как $(n,k,d_{\min})$ или просто (n,k,d).

Рассмотрим, как $d_{\min}$ влияет на корректирующие способности помехоустойчивого кода. Если мы хотим построить помехоустойчивый код, исправляющий все сочетания из t или менее ошибочных символов в любой принятой комбинации, то необходимо обеспечить значение минимального кодового расстояния, удовлетворяющее равенству

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor, \qquad (1.3)$$

где $\lfloor \cdot \rfloor$ обозначает целую часть дроби.

Таким образом, для исправления всех сочетаний ошибок из t или менее ошибочных символов необходимо и достаточно, чтобы каждая разрешенная комбинация отличалась от любой другой разрешенной комбинации кода не

менее, чем в (2t+1) позициях. Формулу (1.3) можно довольно наглядно проиллюстрировать условными графическими представлениями, показанными на рис.1.1 для минимальных кодовых расстояний 4 и 5.

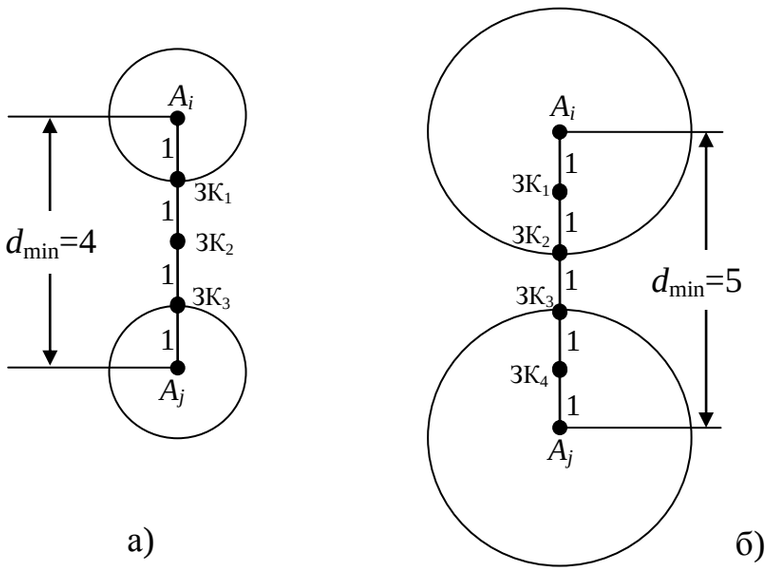


Рис.1.1. Связь минимального кодового расстояния d_{min} и корректирующих свойств помехоустойчивого кода для d_{min}=4 (а) и d_{min}=5 (б) («1» на рисунке – ошибка)

Для большинства блоковых помехоустойчивых кодов декодер принимает решение по принципу максимального правдоподобия, суть которого состоит в том, что декодер декодирует принятую n-символьную комбинацию в ближайшую к ней разрешенную кодовую комбинацию по расстоянию Хемминга.

Как видно из демонстрационного рис.1.1,а, при d_{min}=4 переданная разрешенная кодовая комбинация A_i вследствие однократной ошибки (t =1) перейдет в некоторую запрещенную комбинацию ЗК$_1$, которая по расстоянию Хемминга находится ближе всего к разрешенной комбинации A_i и поэтому декодер примет правильное решение, т. е. будет декодирована переданная комбинация. Если в принятой комбинации возникнут 3 ошибки, которые переведут комбинацию A_i в запрещенную комбинацию ЗК$_3$, то эта комбинация окажется ближе всего к другой разрешенной комбинации A_j и декодер примет решение, что была передана именно эта комбинация, т. е. произойдет неправильное декодирование. Наконец, если возникнет двукратная ошибка, которая переведет разрешенную комбинацию A_i в запрещенную комбинацию ЗК$_2$, то декодер сформирует отказ от декодирования принятой комбинации в силу её равноудаленности и от A_i, и от A_j.

Таким образом, при d_{min}=4 декодер гарантированно исправляет однократные ошибки и может обнаружить, но не исправить, двукратные ошибки. Тем самым мы показали, что в общем виде при d_{min} чётном декодер сможет гарантированно исправить все ошибки до кратности t включительно, где значение t определяется выражением (1.3).

Анализируя рис.1.1, б, придем к заключению, что при d_{min}=5 декодер, используя критерий наиболее близкого расстояния к какой-либо разрешенной комбинации, сможет гарантированно исправить все однократные и

двукратные ошибки, тогда как ошибки кратности 3, 4 и 5 приведут к неправильному декодированию. Таким образом, для любого нечетного $d_{\min}$ декодер гарантированно исправляет все ошибки до кратности $t = \frac{d_{\min} - 1}{2}$ включительно.

Из рассмотренного выше следует вывод, что по критерию наименьшего расстояния Хемминга блоковый код гарантийно исправляет все ошибки до половины минимального расстояния Хемминга (так называемого конструктивного расстояния).

Если помехоустойчивый код работает только в режиме обнаружения ошибок, то, анализируя рис. 1.1, можно заключить, что в этом режиме помехоустойчивый код гарантийно обнаруживает все ошибки до кратности σ включительно, где $\sigma \le (d_{\min} - 1)$. Кроме того, код будет обнаруживать и определенную долю ошибок большей кратности, которые переводят разрешенную кодовую комбинацию в запрещенную.

Теперь покажем на примере двоичных кодов (основание кода 2), как связана избыточность блокового (n,k)-кода с кратностью исправляемых ошибок. В случае исправления всех ошибок до кратности t включительно подмножество комбинаций длины n, соответствующее некоторой разрешенной комбинации A_i, будет содержать саму разрешенную комбинцию A_i (безошибочный прием) и $(C_n^1 + C_n^2 + ... + C_n^t)$ запрещенных комбинаций длины n, где C_n^i – число сочетаний из n по i .

Отсюда следует, что число N разрешенных комбинаций блокового (n,k)-кода будет ограничено значением

$$N \le \frac{2^n}{\sum_{i=0}^{t} C_n^i}, \tag{1.4}$$

где C_n^i – количество запрещенных комбинаций длины n, отличающихся от A_i в i позициях.

Наиболее часто встречаются блоковые (n,k)-коды, у которых число разрешенных комбинаций равно $N = 2^k$, в этом случае выражение (1.4) принимает вид

$$2^k \le \frac{2^n}{\sum_{i=0}^{t} C_n^i} \tag{1.5}$$

или

$$2^{n-k} \ge \sum_{i=0}^{t} C_n^i. \tag{1.6}$$

Логарифмируя по основанию 2 выражение (1.6), получим, что относительная избыточность блокового (n,k)-кода, исправляющего до t ошибок включительно, должна удовлетворять неравенству

$$n - k \ge \log_2 \sum_{i=0}^{t} C_n^i. \tag{1.7}$$

Выражение (1.7) представляет собой границу Хемминга для избыточного блокового (n,k)-кода при заданных значениях k и t .

Обобщение этой границы для недвоичных кодов (основание кода $p \neq 2$) имеет следующий вид:

$$n-k \geq \log_p \left[\sum_{i=0}^{t} C_n^i (p-1)^i \right]. \qquad (1.8)$$

Из этого следует, что относительная избыточность двоичного помехоустойчивого кода, исправляющего однократные ошибки (t=1), определяется неравенством $n-k \geq \log_2(1+n)$.

Другой характеристикой блоковых (n,k)-кодов является *вес кодовой комбинации*, определяемый числом ненулевых элементов в данной кодовой комбинации. При этом обобщённой характеристикой кода будет *весовой спектр* $A(w_i)$, представляющий собой распределение количества комбинаций с весом w_i. Знание весового спектра упрощает оценку корректирующих свойств кода, в частности, вероятности необнаруженных ошибок в режиме обнаружения или вероятности неправильного декодирования комбинации в режиме исправления ошибок.

Большинство блоковых кодов обладают свойством линейности. *Линейные коды* характеризуются следующими особенностями: поэлементная сумма двух кодовых комбинаций (например, по модулю 2 для двоичных кодов) образует третью комбинацию, также принадлежащую данному коду. Свойство линейности, как будет показано ниже, существенно упрощает процедуру кодирования и декодирования таких кодов, в частности, позволяя выразить каждую кодовую комбинацию в виде линейной суммы некоторых базисных комбинаций, составляющих образующую (порождающую) код матрицу.

Блоковый линейный код является *групповым*, так как множество его кодовых комбинаций образует конечную группу (см. раздел 2.1) по операции поэлементного сложения по модулю 2. Для такого группового кода характерно, что минимальный вес $w_{\min}$ ненулевой кодовой комбинации равен минимальному расстоянию Хемминга $d_{\min}$.

В отличие от блоковых равномерных помехоустойчивых кодов применяют *непрерывные* коды, к числу которых относятся так называемые цепные или рекуррентные коды [16], а в последнее время *сверточные* коды, выходные комбинации которых образуются как свертка непрерывной входной последовательности с импульсной характеристикой кодера. Наиболее часто встречается декодер сверточного кода, который, как и в блочных кодах, принимает решение по максимуму правдоподобия (алгоритм Витерби).

Наконец, по своим возможностям помехоустойчивые коды с исправлением ошибок делятся на коды, исправляющие независимые ошибки, и коды, исправляющие ошибки, группирующиеся в пакеты (пачки).

1.2. Эффективность помехоустойчивого кодирования

Эффективность помехоустойчивого кода, наряду с такими показателями как скорость R кода и достоверность передачи информации, часто оценивается энергетическим выигрышем от применения такого кодирования [2]. В качестве энергетического показателя системы связи выбирают отношение энергии, приходящейся на один информационный символ, к спектральной мощности шума E_b/N_0, которое требуется для достижения заданной вероятности ошибки. Энергетический выигрыш оценивается как разность значений E_b/N_0, которые обеспечивают заданную вероятность ошибки в системе без помехоустойчивого кодирования и системе с

применением помехоустойчивого кода. При этом энергетический выигрыш измеряется, как правило, в дБ.

Ряд авторов [16, 17] оценивают эффективность помехоустойчивого кодирования эквивалентной вероятностью ошибки $p_э$. Эта вероятность определяется как вероятность ошибочного приема кодового символа, при которой в некотором гипотетическом (эквивалентном) канале с безызбыточным кодом обеспечивается такая же вероятность правильного приема комбинации из k информационных символов, как и в канале с использованием избыточного помехоустойчивого кода.

Эквивалентная вероятность ошибки по Л. М. Финку [16] определяется следующим образом.

Пусть вероятность правильного приема кодовой комбинации (n,k)-кода в реальном канале с помехоустойчивым кодом будет равна $P_{пп.}$. Те же k информационных элементов в эквивалентном канале с применением кода без избыточности будут приняты без ошибок с вероятностью $(q_э)^k$, где $q_э$ – вероятность правильного приема одного кодового символа. Приравняв обе вероятности, получим $(q_э)^k = P_{пп} = 1 - P_{оп}$, где $P_{оп}$ – вероятность ошибочного декодирования комбинации избыточного (n,k)-кода. Полученное равенство можно записать как $q_э=1–p_э=(1– P_{оп})^{1/k}$.

Тогда, при $P_{оп}<<1$ можно считать, что эквивалентная вероятность ошибочного приёма одного символа будет равна

$$p_э \approx P_{оп} / k . \qquad (1.9)$$

2. ЦИКЛИЧЕСКИЕ КОДЫ

Широкое применение в различных системах находят циклические коды, используемые как для обнаружения, так и для исправления ошибок. В протоколах канального уровня, как правило, используются циклические коды для обнаружения ошибок, вследствие чего их реализация довольно проста. Требуемая эффективность, при этом, достигается за счет применения обратной связи. Однако во многих системах применение обратной связи неприемлемо, так как она вносит существенные задержки. Особенно это характерно для трафика реального времени. В таких системах находят широкое применение более мощные помехоустойчивые коды – циклические эквидистантные коды (коды максимальной длины), Рида–Маллера, Боуза–Чоудхури–Хоквингема (БЧХ), Рида–Соломона, обеспечивающие исправление ошибок в принятой комбинации.

Коды максимальной длины $(2^k–1,k)$ характеризуются высокими корректирующими свойствами, простотой реализации и одинаковым весом ненулевых комбинаций, равным $2^{k–1}$. Поэтому такой код называют эквидистантным. Кодовые последовательности эквидистантного кода находят широкое применение в таких системах, где необходим псевдослучайный характер последовательности, например, в широкополосных системах.

Коды Рида–Маллера (РМ) являются двоичными групповыми кодами, эквивалентными циклическим кодам с дополнительной проверкой на четность. Код РМ первого порядка есть последовательность максимальной

длины с дополнительной проверкой на четность. Положительное свойство кодов Рида–Маллера состоит в том, что эти коды могут быть декодированы с использованием алгоритмов порогового декодирования.

Коды БЧХ составляют большой класс циклических кодов, исправляющих независимые ошибки кратностью t и менее.

Среди циклических кодов, исправляющих ошибки, наибольшее применение в настоящее время находят коды Рида–Соломона (РС), которые представляют собой недвоичные циклические коды БЧХ.

К примеру, коды РС применяют в системах с широкополосным беспроводным доступом (стандарт IEEE 802.16), в системах беспроводной связи с кодовым уплотнением (стандарт IS-95), для цифровой видеозаписи и т. п.

Для декодирования кодов БЧХ и РС, в частности, во временной области, используются алгебраические процедуры, в результате которых находятся синдромы и определяются номера ошибочных позиций путем решения многочлена локаторов ошибок по алгоритму Ченя, а для кодов РС определяются еще и значения ошибок.

Ценным свойством циклических кодов РС является то, что они имеют максимально возможное минимальное кодовое расстояние и, тем самым, обладают наибольшей корректирующей способностью. Это свойство, прежде всего, и послужило основным фактором того, что коды РС находят сегодня широкое применение.

Для борьбы с пачками ошибок в каналах с памятью применяют циклические коды Файра.

В [18] описана эффективная процедура алгебраического мажоритарного декодирования эквивалентных циклических кодов, основанная на использовании двойственного базиса поля Галуа GF(2^k). При этом кодовые комбинации рассматриваются как рекуррентные последовательности.

2.1. Алгебраические основы теории циклических кодов

Теория циклических кодов основана на математическом аппарате высшей алгебры и полях Галуа. Поэтому, прежде чем приступить к изучению самих циклических кодов и их построению, рассмотрим некоторые необходимые элементы высшей алгебры и теории полей Галуа.

Циклический код образуется из равномерного k-элементного кода, множество комбинаций которого представляет собой конечную группу порядка p^k, где p – основание кода.

Группой называется множество G объектов или элементов (числа, матрицы, подстановки и т. д.), для которых определена некоторая операция, позволяющая для каждых двух элементов a и b группы найти третий элемент c той же группы по однозначной функциональной зависимости $f(a, b) = c$.

Операцию называют сложением, если между элементами группы выполняется зависимость $a + b = c$ или умножением при $a \cdot b = c$. Как правило, эти операции не являются арифметическим сложением или арифметическим умножением.

Для элементов группы должны выполняться следующие аксиомы.

1. *Условие замкнутости.* Для любых двух элементов a и b группы существует вполне определенный, принадлежащий этой же группе элемент c, который может быть представлен как $c = a + b$ для операции сложения или $c = a \cdot b$ для операции умножения.

2. *Условие сочетательности или ассоциативности.* Для любых трех элементов a, b и c группы $(a + b) + c = a + (b + c)$, если операция записана как сложение, или $(a \cdot b)\, c = a\, (b \cdot c)$, если операция записана как умножение.

3. *Условие существования единичного элемента*. Если операция названа сложением, то единичный элемент называется нулем, обозначается 0 и определяется из уравнения $0 + a = a + 0 = a$, которое должно выполняться для любого элемента группы. Если операция названа умножением, то единичный элемент называется единицей, обозначается e и определяется из уравнения $e \cdot a = a \cdot e = a$.

4. *Условие существования обратного элемента*. Если операция называется сложением, то обратный элемент, соответствующий элементу a, обозначается $(-a)$ и определяется из уравнения $a + (-a) = (-a) + a = 0$. Если операция называется умножением, то обратный элемент обозначается a^{-1} и определяется из уравнения $a \cdot a^{-1} = a^{-1} \cdot a = e$.

Кроме перечисленных аксиом, элементы группы могут удовлетворять условию коммутативности или переместительности, т. е. равенству $a + b = b + a$, если операция называется сложением, или равенству $a \cdot b = b \cdot a$, если операция названа умножением. В этом случае группа называется абелевой или коммутативной.

Группа называется конечной, если она состоит из конечного числа элементов; в противном случае она называется бесконечной.

Число элементов конечной группы называется ее порядком.

Группу, элементы которой можно представить как степени одного принадлежащего группе элемента, называют циклической.

Кольцо. Пусть R − некоторое множество элементов $a, b, c,,..$. Эти элементы могут быть самой разнообразной природы: числа, матрицы, многочлены и др. Множество R называется кольцом, если:

а) выполняется замкнутость множества R по отношению к операциям сложения и умножения, т. е. сумма $a + b$ и произведение $a \cdot b$ любых двух элементов a, b являются также элементами множества R;

б) выполняются сочетательные (ассоциативные) законы $(a + b) + c = a + (b + c)$ и $(ab)\, c = a\, (bc)$ для любых элементов a, b и c из множества R;

в) операция сложения перестановочна (коммутативна) $a + b = b + a$ для любых элементов a и b из множества R;

г) выполняется обратимость сложения, т. е. для любых элементов a и b из множества R уравнение $a + x = b$ разрешимо, где x принадлежит множеству R;

д) выполняется распределительный (дистрибутивный) закон: $a(b + c) = ab + ac$ и $(b + c)a = ba + ca$ для любых элементов a, b и c из множества R.

Если коммутативный закон также справедлив для операции умножения для любых элементов a и b из множества R, т. е. $ab = ba$, то кольцо называется *коммутативным*.

Полем называется такое коммутативное кольцо, в котором уравнение $ax = b$ при $a \neq 0$ всегда разрешимо (т. е. удовлетворяется выполнимость деления). Поле, кроме нуля 0 ($a + 0 = a$) и противоположных элементов a и ($-a$) [$a + (-a) = 0$], содержит также единичный элемент e и обратные элементы a^{-1}, для которых справедливо: $ae = ea = a$; $a \cdot a^{-1} = e$. Элемент поля 0 называют аддитивной единицей, а элемент e – мультипликативной единицей.

Итак, *группа* – это система, в которой заданы одна основная операция и операция, ей обратная, например сложение и обратная операция – вычитание; или умножение и обратная операция – деление. В *кольце* определены две основные операции – сложение и умножение, и операция, обратная первой из этих операций – вычитание. В *поле* определены две основные операции, а именно: сложение и умножение, и операции, обратные к ним обеим, т. е. вычитание и деление.

Идеал. Пусть R – коммутативное кольцо, для которого e является единичным элементом. Для такого кольца идеалом ((p)) называют подмножество элементов кольца R , состоящее из всех кратных rp, где $r \in R$. Например, идеал ((2)) в кольце целых чисел состоит из всех четных чисел. Следовательно, если e – единичный элемент кольца R , то единичный идеал, порожденный элементом e, будет содержать все элементы кольца. Нулевой идеал – это идеал, состоящий из одного элемента 0.

Классы вычетов. Фактор-кольцо. Идеал ((p)) кольца R определяет разбиение этого кольца на смежные классы или классы вычетов по идеалу. Например, разбиение кольца целых чисел на смежные классы по идеалу ((p)) будет следующим:

$$\begin{array}{ccccccc} 0 & \pm p & \pm 2p & \pm 3p & \pm 4p & \pm 5p & \ldots \\ 1 & 1 \pm p & 1 \pm 2p & 1 \pm 3p & 1 \pm 4p & 1 \pm 5p & \ldots \\ 2 & 2 \pm p & 2 \pm 2p & 2 \pm 3p & 2 \pm 4p & 2 \pm 5p & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ (p-1) & (p-1) \pm p & (p-1) \pm 2p & (p-1) \pm 3p & (p-1) \pm 4p & (p-1) \pm 5p & \ldots \end{array} \qquad (2.1)$$

Первый ряд представляет собой идеал, а остальные − смежные классы, число которых равно ($p - 1$). Нуль принадлежит идеалу, а числа 1, 2,..., ($p - 1$) являются образующими смежных классов или классов вычетов по идеалу ((p)).

Два элемента a и b называют сравнимыми по модулю p и записывают $a \equiv b \pmod{p}$, если они принадлежат одному и тому же смежному классу, т. е. если $(a - b) \in (p)$. Действительно, если взять два элемента из первого смежного класса $1 + 4p$ и $1 + p$, то их разность $(1 + 4p) - (1 + p) = 3p$ принадлежит идеалу, т.е. первому ряду из (2.1). Следовательно, $(1 + 4p) \equiv (1 + p) \pmod{p}$.

Простые поля. Характеристика поля. Пусть имеется некоторое поле F. Известно, что пересечение произвольного множества подполей поля R также является подполем. На рис. 2.1 k_1, k_2, k_3 – подполя поля F ; p – пересечение этих подполей.

Пересечение p всех подполей поля F есть наименьшее подполе, которое не содержит других подполей, отличных от p.

Поле (подполе) называется *простым*, если оно не содержит никаких подполей, отличных от него самого. Таким образом, поле (подполе) p будет простым, а поле F содержит единственное простое поле p , которое, как

всякое подполе, содержит в себе аддитивную единицу 0 и мультипликативную единицу e, входящие также в состав поля F и его подполей k_1, k_2 и k_3.

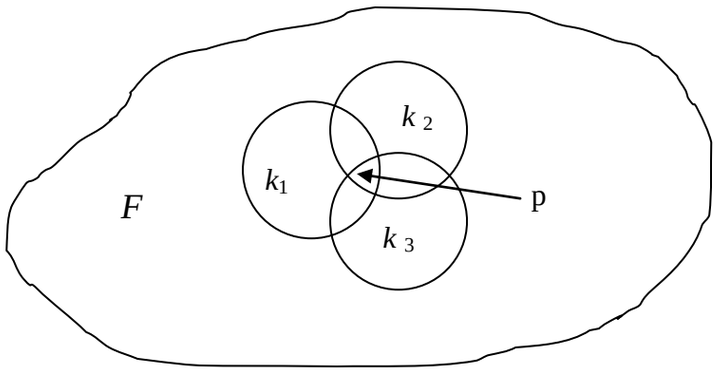


Рис. 2.1. Графическое изображение поля

Рассмотренный на рис 2.1 пример полей называют полями с характеристикой p ($p > 0$), где число p должно быть простым.

Переходим теперь к рассмотрению конечных полей более общего вида. Из сказанного ранее ясно, что в каждом конечном поле должно заключаться числовое поле, соответствующее некоторому простому числу p, поэтому в состав рассматриваемого конечного поля должны входить элементы, обозначаемые знаками

$$0, 1, 2, ..., p-1. \qquad (2.2)$$

Предположим, что кроме этих элементов, в поле существует еще элемент x, отличный от элементов (2.2), тогда по определению поля в нем должны также существовать элементы

$$x, 2x, 3x, ..., (p-1)x. \qquad (2.3)$$

Складывая элементы ряда (2.2) с элементами (2.3), получаем элементы вида

$$\alpha_1 + \alpha_2 x, \qquad (2.4)$$

где коэффициенты α_1 и α_2 принимают значения 0, 1, 2,..., $p-1$, т. е. пробегают полную систему вычетов по модулю p. Получаем p^2 элементов поля. Нетрудно убедиться, что все они различны между собой, так как решение равенства $\alpha_1 + \alpha_2 x = \alpha_1' + \alpha_2' x$ относительно x давало бы элемент числового ряда (2.2). Однако по предположению элемент x отличен от элементов ряда (2.2).

Возможно, что рассматриваемое поле исчерпывается элементами вида (2.4), тогда число элементов конечного поля будет равно p^2. Если же поле не исчерпывается элементами (2.4), тогда любой элемент поля будет иметь вид: $\alpha_1 + \alpha_2 x + \alpha_3 y$, где все коэффициенты α_1, α_2, α_3 пробегают полную систему вычетов по простому модулю p.

Продолжая рассуждения аналогичным образом, получим выражение для элемента поля в общем виде: $\alpha_1 + \alpha_2 x + \alpha_3 y + \ldots + \alpha_{n-1} t + \alpha_n u$, где коэффициенты α_i – вычеты по простому модулю p. Тогда число элементов конечного поля будет равно p^n, где n – число натуральное, и такое поле называют расширенным с характеристикой простого поля p.

Поля Галуа и их свойства. Конечное поле, называемое именем французского математика Эвариста Галуа и обозначаемое GF(q), представляет

собой конечное множество, состоящее из q элементов, обладающих свойствами поля. Число элементов поля GF(q) может быть простым числом или степенью простого числа. Если q – простое число, то поле GF(q) будет простым с характеристикой q, элементами которого будут числа 0, 1. 2, …, (q–1). При этом в соответствии со свойствами поля сложение и умножение элементов такого поля осуществляется с приведением по модулю p. Если q является степенью, например, простого числа p, т.е. $q= p^m$, где m – целое, то конечное поле GF(p^m) будет расширением простого поля, а элементами расширенного поля будут многочлены степени (m – 1) вида

$$a_0 + a_1x + a_2x^2 + ... + a_{m-1}x^{m-1}, \qquad (2.5)$$

где все коэффициенты a_i пробегают полную систему вычетов по модулю p, т. е. принадлежат простому полю GF(p).

Сложение двух элементов

$$A = a_0 + a_1x + a_2x^2 + ... + a_{m-1}x^{m-1};$$
$$B = b_0 + b_1x + b_2x^2 + ... + b_{m-1}x^{m-1}$$

производится с приведением коэффициентов по модулю p. Тогда элемент $C = c_0 + c_1x + c_2x^2 + ... + c_{m-1}x^{m-1}$ будет суммой двух элементов A и B, т. е. $C = A + B$, где $c_i \equiv (a_i+b_i) \mathrm{mod} p$.

Для умножения двух элементов поля A и B перемножим их алгебраически как многочлены независимой переменной x, затем найдем остаток D от деления произведения $A \cdot B$ на некоторый специальный многочлен $P(x)$ степени m с коэффициентами, принадлежащими простому полю GF(p). Этот многочлен должен обладать тем свойством, что его нельзя разложить на множители, используя только многочлены с коэффициентами из простого поля GF(p). Такой многочлен называют неприводимым. Тогда полученный остаток D с коэффициентами, принадлежащими простому полю GF(p), можно рассматривать как вычет по двойному модулю – по mod$p$ и mod$P(x)$, т.е. $A \cdot B \equiv D[\mathrm{mod}\ p,\ P(x)]$. При таких правилах сложения и умножения совокупность рассматриваемых p^m элементов вида (2.5) образует конечное поле, называемое полем Галуа.

Итак, поле Галуа может быть представлено как поле классов вычетов по двойному модулю.

Многочлены $P(x)$ и элементы поля GF(p^m) обладают целым рядом свойств, используемых при построении и описании циклических кодов. Приведем основные из этих свойств.

Свойство 1.1. Все отличные от нуля элементы поля GF(p^m) образуют мультипликативную циклическую группу порядка p^m –1. Тогда для любого ненулевого элемента поля ε имеет место равенство $\varepsilon^{p^m-1} = 1.$

Вытекающее из этого свойства равенство $\varepsilon^{p^m} = \varepsilon$ выполняется также и для нулевого элемента поля ε=0.

Отметим, что, учитывая свойства двучленных уравнений, ненулевые элементы поля являются корнями многочлена $x^{p^m-1} - 1$, а все элементы поля являются корнями многочлена $x^{p^m} - x$.

Свойство 1.2. В поле GF(p^m) всегда существует первообразный элемент ε, т. е. элемент, порядок которого равен p^m – 1. При этом каждый ненулевой элемент поля может быть представлен как некоторая степень одного и того же первообразного элемента ε. Иными словами: мультипликативная группа поля Галуа циклична.

В теории циклических кодов важную роль играет следующее свойство.

Свойство 1.3. Всякий неприводимый по модулю p многочлен $P(x)$ степени m, если он существует, есть делитель по этому модулю двучлена $x^{p^m-1}-1$.

Примитивным называется такой неприводимый по модулю p многочлен $P(x)$ степени m, корни которого являются первообразными элементами поля GF(p^m), т .е. имеющими порядок $p^m - 1$.

Порядок корней неприводимого по модулю p многочлена называют *показателем*, к которому этот многочлен принадлежит. Если неприводимый многочлен принадлежит показателю k, то он является делителем многочлена x^k–1, но не является делителем многочленов вида x^n–1, где $n < k$. Следовательно, неприводимый многочлен $P(x)$ степени m является примитивным тогда и только тогда, когда он принадлежит показателю $p^m - 1$.

Приведем теперь свойство, позволяющее определить элементы поля GF(p^m), являющиеся корнями функции $P(x)$, если известно, что один из этих корней ε.

Свойство 1.4. В простом поле характеристики p имеет место равенство $(a + b)^p=a^p+b^p$.

Действительно, $(a + b)^p=a^p+ C_p^1a^{p-1}b + C_p^2a^{p-2} b^2 +\ldots + C_p^{p-1}ab^{p-1} +b^p$. Но, так как для всех $0 <i < p$ имеем $C^i_P= 0 \pmod p$, то, следовательно, получаем $(a+b)^p=a^p+b^p$, что и требовалось доказать.

Аналогичным образом доказывается, что в поле характеристики p имеют место формулы:

$$(a+b)^{p^n} = a^{p^n} + b^{p^n}$$

$$(a_1 + a_2 + a_3 + \ldots + a_n)^p = a_1^p + a_2^p + a_3^p + \ldots + a_n^p$$

Важной является также *малая теорема Ферма* [11,12], которая гласит:

Для каждого класса вычетов a по модулю p, взаимно простого с модулем, выполняется сравнение $a^{\varphi(p)}\equiv 1 \pmod p$, *где* $\varphi(p)$ *– функция Эйлера. Если p простое число, то функция Эйлера* $\varphi(p) = p - 1$ *и тогда* $a^{p-1}\equiv 1 \pmod p$.

На основании этой теоремы и свойства 1.4 легко доказывается следующее важное для теории циклических кодов свойство.

Свойство 1.5. Для простого модуля p существует сравнение

$$[P(x)]^p\equiv P(x^p) \pmod p,$$

где $P(x)$ – произвольный многочлен, коэффициенты которого принадлежат простому полю GF(p).

Пусть теперь элемент ε поля GF(p^m) является корнем неприводимого многочлена $P(x)$ степени m. Тогда $P(\varepsilon) = 0$, а по свойству 1.5 $P(\varepsilon)]^p = P(\varepsilon^p) = 0$.

Следовательно, если ε корень неприводимого многочлена $P(x)$, то ε^p является также его корнем. Аналогично можно показать, что для неприводимого многочлена $P(x)$ степени m корнями будут также элементы поля $\varepsilon^{p^2},\varepsilon^{p^3},\ldots,\varepsilon^{p^{m-1}}$. Таким образом, доказано следующее свойство полей Галуа.

Свойство 1.6. Если элемент ε поля GF(p^m) является корнем неприводимого по модулю p многочлена $P(x)$ степени m, то остальными корнями многочлена $P(x)$ будут элементы $\varepsilon^p,\varepsilon^{p^2},\varepsilon^{p^3},\ldots,\varepsilon^{p^{m-1}}$.

Важную роль в теории циклических кодов играют следующие свойства.

Свойство 1.7. Многочлен $x^k - 1$ является делителем многочлена $x^n - 1$, если k – делитель n.

По этому свойству в поле GF(p^m) будут иметь место непервообразные элементы ε^i, которые относятся к показателю k, являющимся делителем порядка $x^{p^m-1}-1$. Такие элементы поля GF(p^m) будут корнями двучлена $x^k - 1$, т.е. $(\varepsilon^i)^k=1$, а их количество равно функции Эйлера $\varphi(k)$.

Свойство 1.8. Если неприводимый по модулю p многочлен $P_1(x)$ степени k является делителем двучлена $x^{p^m-1}-1$, то степень k должна быть делителем числа m. И наоборот: всякий неприводимый по модулю p многочлен $P_1(x)$, степень которого k есть делитель числа m, будет делителем по модулю p двучлена $x^{p^m-1}-1$.

Свойство 1.9. Всякий неприводимый по модулю p многочлен $P_1(x)$, который является делителем двучлена $x^{p^m-1}-1$, входит в состав этого двучлена однократно.

Таким образом, двучлен $x^{p^m-1}-1$ раскладывается на ряд различных неприводимых сомножителей-многочленов, степени которых d будут делителями числа m. Сомножители, имеющие степень, равную некоторому определенному делителю d, обозначим через $\Phi_d(x)$. Тогда в числовом поле по модулю p существует равенство

$$x^{p^m} - x = \Pi\Phi_d(x), \qquad (2.6)$$

где произведение П распространяется на все d делители числа m (причем, одним из множителей будет x). Многочлены $\Phi_d(x)$ называются *многочленами деления круга*.

Свойство 1.10. Для любого простого числа p и любого неприводимого по модулю p многочлена $P(x)$ степени m существует только одно поле Галуа GF(p^m), иными словами, поля Галуа GF(p^m), образованные различными неприводимыми примитивными многочленами степени m, изоморфны.

Свойство 1.11. Для каждого делителя $n > 0$ числа m в поле GF(p^m) существует подполе GF(p^n). Элемент ε^i поля GF(p^m) принадлежит подполю GF(p^n), если он удовлетворяет уравнению $(\varepsilon^i)^{p^n-1}=1$, т. е. если его порядок (в мультипликативной группе поля GF(p^m)) является делителем числа $p^n - 1$.

Тогда все ненулевые элементы поля GF(p^m), принадлежащие подполю GF(p^n), должны быть корнями двучлена $x^{p^n-1}-1$. Следовательно, двучлен $x^{p^n-1}-1$ должен быть делителем многочлена $x^{p^m-1}-1$, а по свойству 1.7 число ($p^n - 1$) должно быть делителем числа ($p^m - 1$), а n делителем m.

Все приведенные выше свойства полностью характеризуют поля Галуа и позволяют их построить. Кроме того, если в поле Галуа GF(p^m) существует какое-либо подполе Галуа, то все, относящееся к полю Галуа, должно быть справедливым и к подполю Галуа.

В заключение этого раздела рассмотрим некоторые примеры.

Пример 2.1. Пусть требуется построить поле Галуа GF(p^m), где p = 2, а m = 4, т. е. GF(2^4). По свойству 1.10 для построения поля Галуа GF(2^4) может быть выбран любой неприводимый примитивный многочлен $P(x)$ степени m=4.

Для нахождения неприводимых и примитивных многочленов можно воспользоваться таблицами неприводимых многочленов, приведенными в [7, 19].

Выберем в качестве порождающего поле GF(2^4) неприводимый по модулю 2 примитивный многочлен четвертой степени $P(x) = 1 + x + x^4$. Пусть ε является корнем данного многочлена, тогда он будет первообразным элементом поля GF(2^4). В соответствии со свойством 1.2 все 15 ненулевых элементов поля будут следующими (все степени ε приводятся по mod $P(\varepsilon)$):

Таблица 2.1

ε^0	=	1						=	(1000)
ε^1	=			ε				=	(0100)
ε^2	=					ε^2		=	(0010)
ε^3	=						ε^3	=	(0001)
ε^4	=	1	+	ε				=	(1100)
ε^5	=			ε	+	ε^2		=	(0110)
ε^6	=					ε^2	+ ε^3	=	(0011)
ε^7	=	1	+	ε			+ ε^3	=	(1101)
ε^8	=	1	+			ε^2		=	(1010)
ε^9	=			ε	+		ε^3	=	(0101)
ε^{10}	=	1	+	ε	+	ε^2		=	(1110)
ε^{11}	=			ε	+	ε^2	+ ε^3	=	(0111)
ε^{12}	=	1	+	ε	+	ε^2	+ ε^3	=	(1111)
ε^{13}	=	1	+			ε^2	+ ε^3	=	(1011)
ε^{14}	=	1+					ε^3	=	(1001)
ε^{15}	=	1						=	(1000)

В правой колонке табл.2.1 элементы поля $\varepsilon^i = a_0 + a_1\varepsilon^1 + a_2\varepsilon^2 + a_3\varepsilon^3$ представлены в векторной форме записи (a_0, a_1, a_2, a_3), где коэффициенты a_i принадлежат простому полю GF(2).

Пример 2.2. Определить делители двучлена $x^{p^m-1} - 1$, где $p = 2$, а $m = 4$.

Из свойства 1.8 следует, что делителями многочлена $x^{p^m-1} - 1$ будут все неприводимые многочлены, степень которых является делителем числа m. В примере делителями числа $m = 4$ будут 4, 2, 1. Находим по таблицам неприводимых многочленов все неприводимые многочлены этих степеней. Они и дают разложение двучлена ($x^{15} - 1$) на множители: ($x^{15} - 1$) = ($x - 1$) ($x^2 + x + 1$) ($x^4 + x + 1$) ($x^4 + x^3 + 1$) ($x^4 + x^3 + x^2 + x + 1$).

Пример 2.3. При построении поля Галуа GF(2^4) был выбран неприводимый примитивный многочлен $P(x) = 1 + x + x^4$, ε – корень этого многочлена. Определить, какие элементы поля GF(2^4) относятся к каждому из множителей в разложении двучлена $x^{15} - 1$, т. е. найти корни для каждого из множителей, приведенных в примере 2.2.

При решении этой задачи необходимо использовать свойства 1.6 и 1.7. Корнями многочлена $P(x)$ будут элементы: ε, ε^2, ε^4, ε^8. Берем следующий неиспользованный элемент низшей степени ε^3 и по тем же свойствам устанавливаем, что элементы ε^3, $(\varepsilon^3)^2 = \varepsilon^6$, $(\varepsilon^3)^4 = \varepsilon^{12}$ и $(\varepsilon^3)^8 = \varepsilon^9$ являются корнями следующего неприводимого многочлена $P_1(x)$ четвертой степени. Следующий элемент ε^5. Возводя его в квадрат, имеем $(\varepsilon^5)^2 = \varepsilon^{10}$, в то же время $(\varepsilon^{10})^2 = \varepsilon^5$. Следовательно, два элемента ε^5 и ε^{10} являются корнями неприводимого многочлена $P_2(x)$ второй степени. Аналогично находим, что ε^7, ε^{11}, ε^{13} и ε^{14} являются корнями неприводимого многочлена $P_3(x)$ четвертой степени. Последний элемент ε^{15} является корнем многочлена первой степени, а именно $P_4(x) = x - 1$. Следовательно, можно составить табл.2.2.

Таблица 2.2

Многочлен	Степень многочлена	Корни многочлена
$P(x)$	4	$\varepsilon, \varepsilon^2, \varepsilon^4, \varepsilon^8$
$P_1(x)$	4	$\varepsilon^3, \varepsilon^6, \varepsilon^9, \varepsilon^{12}$
$P_2(x)$	2	$\varepsilon^5, \varepsilon^{10}$
$P_3(x)$	4	$\varepsilon^7, \varepsilon^{11}, \varepsilon^{13}, \varepsilon^{14}$
$P_4(x)$	1	$\varepsilon^{15}=1$

Задание для самостоятельной работы. Найти значения многочленов $P_i(x)$, зная корни многочленов и используя формулы Виета и значения элементов поля из табл. 2.1.

2.2. Основные действия над многочленами в поле двоичных чисел и их реализация

Условимся, что векторному представлению $(a_0\ a_1\ a_2 \ldots a_{n-2}\ a_{n-1})$ будет соответствовать многочлен $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}\ x^{n-1}$, где коэффициенты a_i представляют собой наименьшие неотрицательные вычеты по модулю p, т. е. принимают значения 0, 1, 2,..., $(p - 1)$. Например, для двоичных полей с характеристикой $p = 2$ коэффициенты a_i принимают значения 0 и 1. Например, двоичной комбинации (101101) соответствует многочлен $1 + x^2 + x^3 + x^5$.

Сложение многочленов

Правило сложения многочленов сводится к суммированию коэффициентов при одинаковых степенях x и приведению суммы по модулю p. Пусть

$$f_1(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}\ x^{n-1}, \quad f_2(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1}, \quad (2.7)$$

где коэффициенты a_i и b_i принимают значения 0, 1,2, ..., $(p - 1)$.

Тогда сумма многочленов будет:

$$f_1(x) + f_2(x) = (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 + \ldots + (a_{n-1} + b_{n-1})x^{n-1} = \\ = c_0 + c_1x + c_2x^2 + \ldots + c_{n-1}\ x^{n-1} \pmod p.$$

Очевидно, что сумма $(a_i + b_i)$ сравнима с c_i по модулю p, т. е. $(a_i + b_i) \equiv c_i \pmod p$.

Иногда просто говорят, что коэффициенты a_i и b_i складываются по модулю p. Пусть $p = 3$, $a_i = 2$ и $b_i = 2$, тогда $(a_i + b_i) = (2 + 2) = 4 \equiv 1 \pmod 3$.

На основании этого сумма полиномов $f_1(x) = 1 + x^3 + x^5$ и $f_2(x) = x + x^3 + x^7$ с коэффициентами – вычетами по модулю 2, будет равна

$$f_1(x) + f_2(x) = 1 + x + x^5 + x^7.$$

В дальнейшем мы будем рассматривать действия над элементами двоичных полей, поэтому приведем правило сложения двоичных элементов по mod 2:

$$1 + 1 = 0, \qquad 0 + 1 = 1, \qquad 1 + 0 = 1, \qquad 0 + 0 = 0.$$

Умножение многочленов

Умножение многочленов осуществляется по обычным правилам перемножения. Пусть даны два многочлена (2.7), тогда произведение их будет равно:

$$f_1(x)\cdot f_2(x) = (a_0b_0) + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 + \ldots + a_{n-1}b_{n-1}x^{2n-2} \equiv$$
$$\equiv c_0 + c_1x + c_2x^2 + \ldots + c_{2n-2}\,x^{2n-2} \pmod p.$$

Таким образом, коэффициент при x^i будет равен

$$c_i \equiv (a_0b_i + a_1\,b_{i-1} + a_2b_{i-2} + \ldots + a_{i-1}b_1 + a_ib_0) \pmod p.$$

Пусть даны два многочлена с коэффициентами из двоичного поля

$$f_1(x) = 1 + x^3 + x^5 + x^6, \qquad f_2(x) = x + x^4.$$

Их произведение после приведения коэффициентов по модулю 2

$$f(x) = f_1(x)\cdot f_2(x) = x + x^6 + x^9 + x^{10}.$$

Деление многочленов

Операция деления многочленов осуществляется по обычным правилам деления с приведением коэффициентов по mod$p$. Например, деление двоичных многочленов ($p = 2$) осуществляется следующим образом:

$$\begin{array}{l|l}
x^8 + x^6 + x^4 + x^3 + 1 & x^3 + x^2 + x \\
\underline{x^8 + x^7 + x^6} & \underline{x^5 + x^4 + x^3 + 1} \\
x^7 + x^4 + x^3 + 1 & \\
\underline{x^7 + x^6 + x^5} & \\
x^6 + x^5 + x^4 + x^3 + 1 & \\
\underline{x^6 + x^5 + x^4} & \\
x^3 + 1 & \\
\underline{x^3 + x^2 + x} & \\
x^2 + x + 1 \text{ (остаток)} &
\end{array}$$

В общем виде эта операция может быть записана так:

$$\begin{array}{l|l}
a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x + a_0 & b_kx^k + b_{k-1}x^{k-1} + \ldots + b_1x + b_0 \\
\underline{\frac{a_n}{b_k}b_kx^n + \frac{a_n}{b_k}b_{k-1}x^{n-1} + \ldots + \frac{a_n}{b_k}b_0x^{n-k}} & \frac{a_n}{b_k}x^{n-k} + \ldots. \\
(a_{n-1} - \frac{a_n}{b_k}b_{k-1})x^{n-1} + \ldots + (a_{n-k} - \frac{a_n}{b_k}b_0)x^{n-k} + \ldots + a_1x + a_0 &
\end{array}$$

При этом коэффициенты $(a_{n-1}-\frac{a_n}{b_k}b_{k-1}),...(a_{n-k}-\frac{a_n}{b_k}b_0)$ приводятся по модулю p. Для двоичных полей (p = 2) операция вычитания равноценна операции сложения, так как – 1 = 1 (mod 2). Действительно, – 1 = –1 + 2=1 (mod 2).

Операции сложения, деления и умножения многочленов могут быть осуществлены над комбинациями коэффициентов этих многочленов.

Пусть $f_1(x) = 1 + x^3 + x^5 + x^6 + x^8$, $f_2(x) = x + x^2 + x^3$.
Многочлену $f_1(x)$ соответствует комбинация (100110101), а многочлену $f_2(x)$ – (0111).

Начало комбинации соответствует младшему разряду, т. е. нулевой степени x:

а) сложение $f_1(x) + f_2(x)$:

$f_1(x) \rightarrow$(1 0 0 1 1 0 1 0 1)
+
$f_2(x) \rightarrow$(0 1 1 1 0 0 0 0 0)
(1 1 1 0 1 0 1 0 1)
$f_1(x) + f_2(x) = 1 + x + x^2 + x^4 + x^6 + x^8$;

б) умножение $f_1(x)\cdot f_2(x)$:

$f_1(x) \rightarrow$(100110101)
$\times f_2(x)$ { $\times 0$ 000000000; $\times 1(x)$ 100110101; $\times 1(x^2)$ 100110101; $\times 1(x^3)$ 100110101 } $\oplus$
(011110001011);

$f_1(x)\cdot f_2(x) = x + x^2 + x^3 + x^4 + x^8 + x^{10} + x^{11}$.

Таким образом, если начало комбинации $f_1(x)$ соответствует младшему разряду, т. е. x^0, то умножению на x^i соответствует сдвиг комбинации $f_1(x)$ на число шагов, равное степени x. Полученный таким образом ряд комбинаций складывают по модулю 2. Как правило, нулевые комбинации (соответствующие умножению на 0) не записывают;

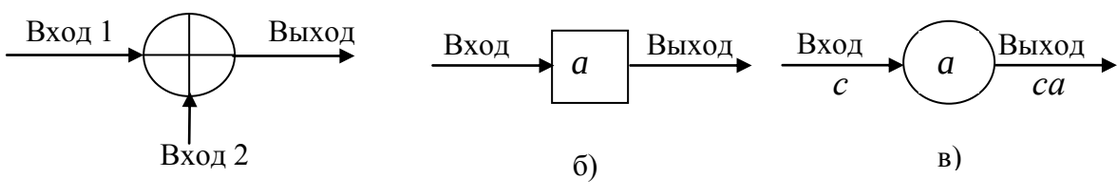
в) деление $f_1(x):f_2(x)$.
При делении комбинации $f_1(x)$ и $f_2(x)$ записывают в двоичной форме со старшего разряда и делят следующим образом:

$f_1(x)$	$f_2(x)$
101011001	1110
1110	111001 $\rightarrow x^5 + x^4 + x^3 + 1$ (частное)
1001	
1110	
1111	
1110	
0010	
1110	
0100	
1110	
1001	
1110	
111 $\rightarrow x^2 + x + 1$ (остаток)	

Реализация операций умножения и деления многочленов в поле двоичных чисел.

Операции умножения и деления многочленов реализуются с помощью регистров сдвига, состоящих из сумматоров, запоминающих устройств и устройств умножения. Сумматор (рис. 2.2,а) представляет собой устройство, осуществляющее сложение по модулю 2 величин на входах 1 и 2. Запоминающее устройство (рис. 2.2,б) служит для хранения в течение определенного времени постоянной величины *a*.

Устройство умножения (рис. 2.2,в). осуществляет умножение некоторой входной величины *c* на постоянный множитель *a*.

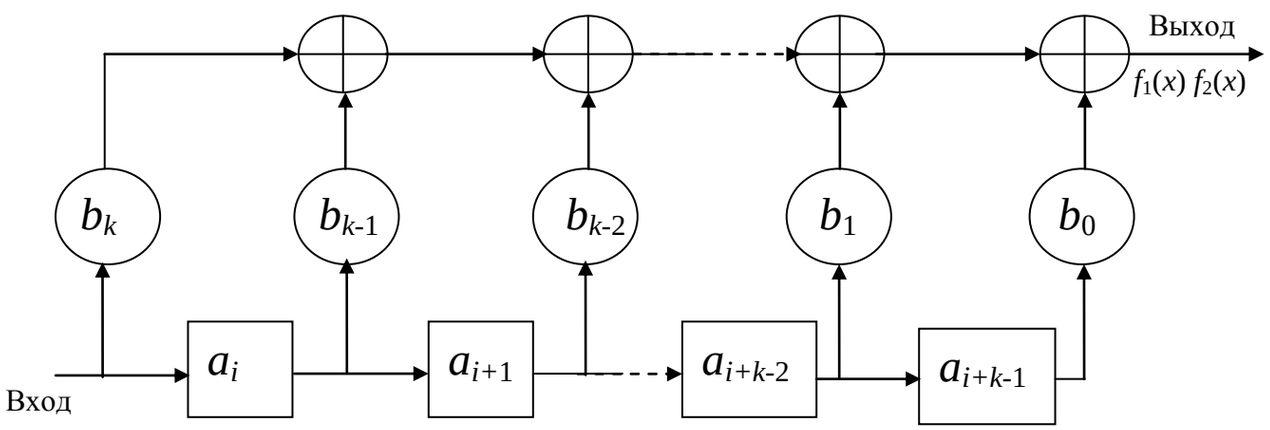


а) б) в)

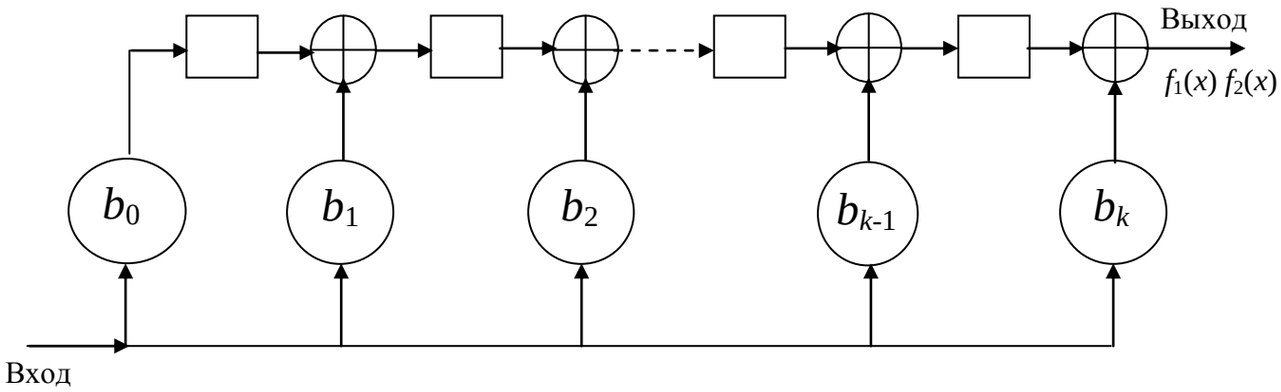
Рис. 2.2. Составные элементы устройств умножения и деления многочленов: сумматор (а), запоминающее устройство (б) и устройство умножения (в)

Реализация умножения многочленов

Умножение некоторого произвольного многочлена $f_1(x) = a_0 + a_1x + a_2x^2 + ... + a_{n-1}\,x^{n-1} + a_nx^n$ на фиксированный многочлен $f_2(x) = b_0 + b_1x + b_2x^2 + ... + b_{k-1}\,x^{k-1} + b_k\,x^k$ осуществляется с помощью регистров сдвига, изображенных на рис. 2.3. Эти регистры строятся по виду многочлена $f_2(x)$ и отличаются лишь расположением сумматоров. Предполагается, что регистр первоначально находится в нулевом состоянии. На вход регистра поступают коэффициенты многочлена $f_1(x)$, начиная со старшей степени, а затем следуют k нулей.



а)



б)

Рис. 2.3. Реализация операции умножения многочленов с помощью регистров сдвига с вынесенными (а) и встроенными (б) сумматорами

Схема, изображенная на рис.2.3,а и представляющая собой регистр с вынесенными сумматорами, работает следующим образом. Когда на входе имеет место первый коэффициент a_n, то на выходе появляется первый коэффициент произведения $f_1(x)\cdot f_2(x)$, равный $a_n b_k$. Кроме того, коэффициент a_n запоминается в первой ячейке памяти регистра. Со вторым тактом на входе появляется коэффициент $a_{n\text{-}1}$ и записывается в первую ячейку памяти, а коэффициент a_n этим же тактом переписывается из первой во вторую ячейку регистра. При этом на выходе схемы появляется значение второго коэффициента произведения $f_1(x)\cdot f_2(x)$, равное $(a_{n\text{-}1}b_k+a_n b_{k\text{-}1})$ и так далее.
Таким образом, схема работает в соответствии с табл. 2.3.

Процедура перемножения многочленов по тактам Таблица 2.3

№ такта	Вход	Выход
1	a_n	$a_n b_k$
2	a_{n-1}	$a_{n\text{-}1}b_k+a_n b_{k-1}$
3	a_{n-2}	$a_{n\text{-}2}b_k+a_{n-1}b_{k-1}+a_n b_{k-2}$
4	a_{n-3}	$a_{n\text{-}3}b_k+a_{n-2}b_{k-1}+a_{n-1}b_{k-2}+a_n b_{k-3}$
……	…….	…….
$n+k-1$	0	$a_0b_1+a_1b_0$
$n+k$	0	a_0b_0

Следовательно, произведение будет равно:

$$f_1(x)\cdot f_2(x) = = a_0b_0 + (a_0b_1 + a_1b_0)x + (a_0b_2 + a_1b_1 + a_2b_0)x^2 + \ldots +$$
$$+(a_{n-2}b_k + a_{n-1}b_{k-1} + a_n b_{k-2})\, x^{n+k-2} + (a_{n-1}b_k + a_n b_{k-1})\, x^{n+k-1} + a_n b_k\, x^{n+k}.$$

Схема, изображенная на рис. 2.3,б и представляющая собой регистр с встроенными сумматорами, работает аналогично предыдущей. При поступлении на вход элемента a_n на выходе схемы появится коэффициент $a_n b_k$. Одновременно с этим в первую ячейку памяти запишется коэффициент $a_n b_0$, во вторую – $a_n b_1$ и т. д., в последнюю ячейку памяти запишется величина, равная $a_n b_{k-1}$. После второго такта на выходе появится сумма входной величины $a_{n\text{-}1}$, умноженной на b_k, и содержимого последней ячейки памяти, т.

е. ($a_{n-1}b_k+a_nb_{k-1}$). При этом в первую ячейку памяти запишется величина $a_{n-1}b_0$, во вторую – сумма $a_{n-1}b_1$ и предшествовавшего содержимого первой ячейки памяти, т. е. ($a_{n-1}b_1 + a_nb_0$) и т. д.

Таким образом, обе схемы равноценны.

Рассмотрим некоторые характерные особенности построения этих схем для многочленов с двоичными коэффициентами, а именно с коэффициентами 0 и 1.

Умножение на величину b_i производим по правилу:
для b_i=0: $a_j \cdot b_i = a_j \cdot 0 = 0$ и для b_i = 1: $a_j \cdot b_i = a_j \cdot 1 = a_j$.

Таким образом, умножение на 0 соответствует разорванной цепи, а умножение на 1 – короткому замыканию.

Пусть, например, $f_2(x) = 1 + x + x^3 + x^5$, т. е. $b_0 = 1$, $b_1 = 1$, b_2=0,b_3= 1 $b_4 = 0$, $b_5 = 1$.

Схема умножения на многочлен $f_2(x)$ реализуется с помощью регистра с встроенными сумматорами, имеющего вид, представленный на рис. 2.4. Как видно из рис. 2.4, сумматор, на который подан только один вход, может быть устранен.

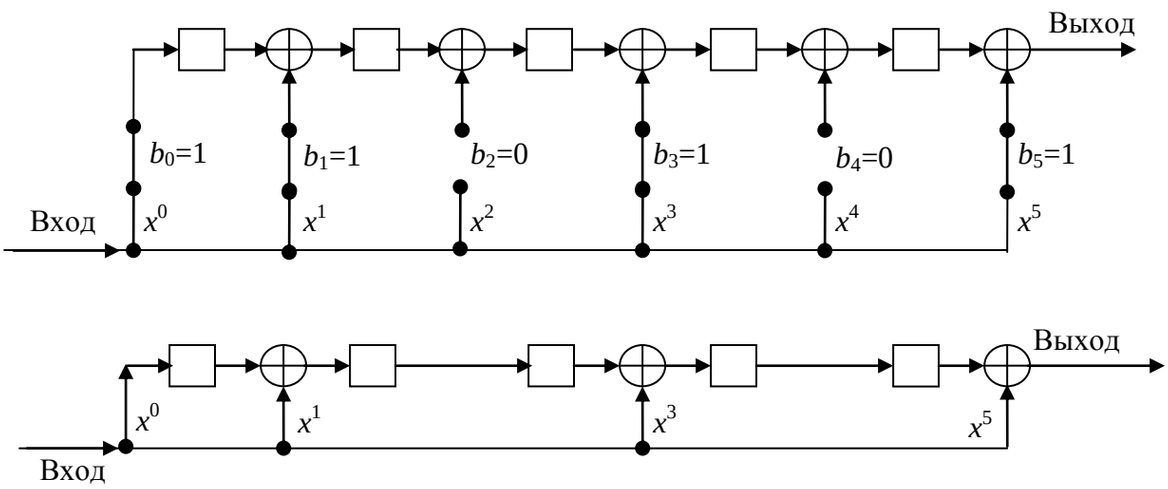


Рис. 2.4. Устройство умножения на многочлен $f(x) = 1 + x + x^3 + x^5$

Аналогично строится регистр с вынесенными сумматорами.

Реализация деления многочленов

Операция деления многочленов может быть реализована с помощью регистра сдвига с обратными связями. Схема деления многочлена $f_1(x) = a_0 + a_1x^2 + a_2x^2 + ... + a_{n-1}x^{n-1} + a_nx^n$ на многочлен $f_2(x) = b_0 + b_1x + \ldots + b_{k-1}x^{k-1} + b_kx^k$ ($k < n$) в общем виде представлена на рис. 2.5.

До поступления многочлена $f_1(x)$ на вход схемы запоминающие элементы должны находиться в нулевом состоянии. На выходе схемы будет 0 до тех пор, пока первый элемент a_n многочлена $f_1(x)$ не достигнет конца регистра, т. е. в течение первых k сдвигов. После этого на выходе, в соответствии со схемой на рис. 2.5, появится первый элемент частного, а именно $a_nb_k^{-1}$, получаемый умножением элемента a_n, следующего из последней ячейки памяти, на величину b_k^{-1}. Как видно из рис.2.5, при каждом ненулевом коэффициенте частного c_i из делимого вычитается произведение

$c_i f_2(x)$. Вычитание осуществляется в сумматорах по модулю два после умножения коэффициента частного на множители $-b_0,-b_1,...,-b_{k-1}$. В двоичном поле коэффициенты $-b_0$, $-b_1,...,-b_{k-1}$ могут принимать значения 0 или 1, а значение b_k – только 1. Учитывая, что при сложении по модулю 2 знак минус может быть заменен плюсом, схема на рис. 2.5 значительно упростится, если устройства умножения заменить, как и в регистрах умножения, короткими замыканиями для $b_i = 1$ и обрывом цепи для $b_i = 0$. При этом число встроенных в регистр сумматоров уменьшится и будет равно числу единичных коэффициентов $b_i = 1$ ($i = 0, 1, ..., k - 1$).

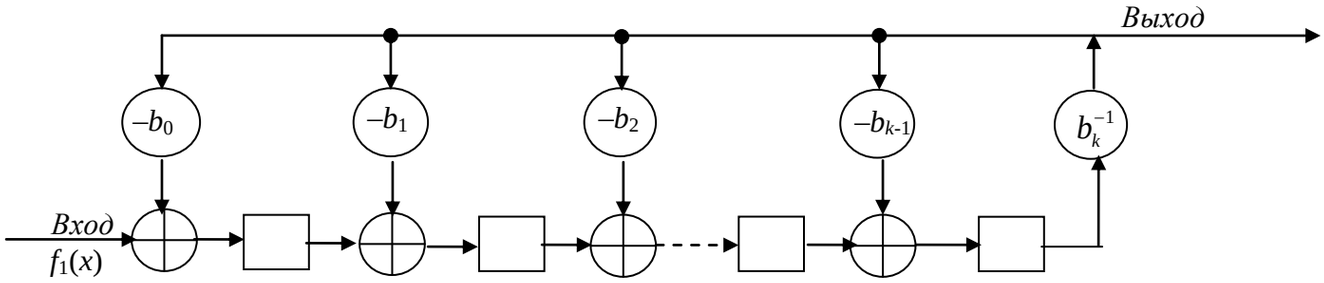


Рис. 2.5. Реализация операции деления с помощью регистра сдвига с обратными связями

Пример 2.4. Пусть $f_2(x) = 1 + x^2 + x^4 + x^5$, т. е. $b_0 = 1$, $b_1 = 0$, $b_2 = 1$, $b_3 = 0$, $b_4 = 1$, $b_5 = 1$.

На рис. 2.6 представлена схема деления на заданный многочлен.

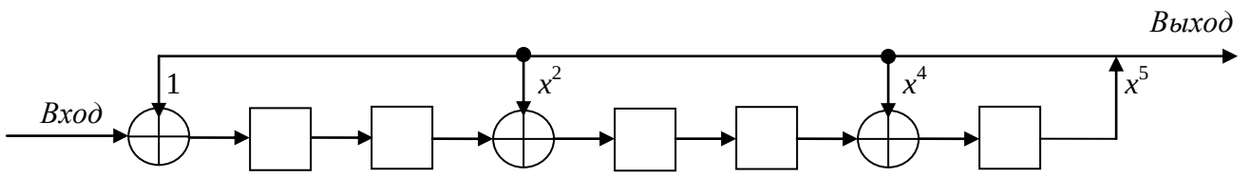


Рис. 2.6. Схема деления на многочлен $f_2(x) = 1 + x^2 + x^4 + x^5$

На основе регистра с встроенными сумматорами может быть построена схема (рис. 2.7), реализующая одновременное умножение на многочлен $h(x) = c_0 + c_1x + .. + c_kx^k$ и деление на многочлен $g(x) = b_0 + b_1x + b_2x^2 + ... + b_kx^k$.

Отметим, что степени многочленов $h(x)$ и $g(x)$ могут быть различными. Примеры приведены на рис. 2.8.

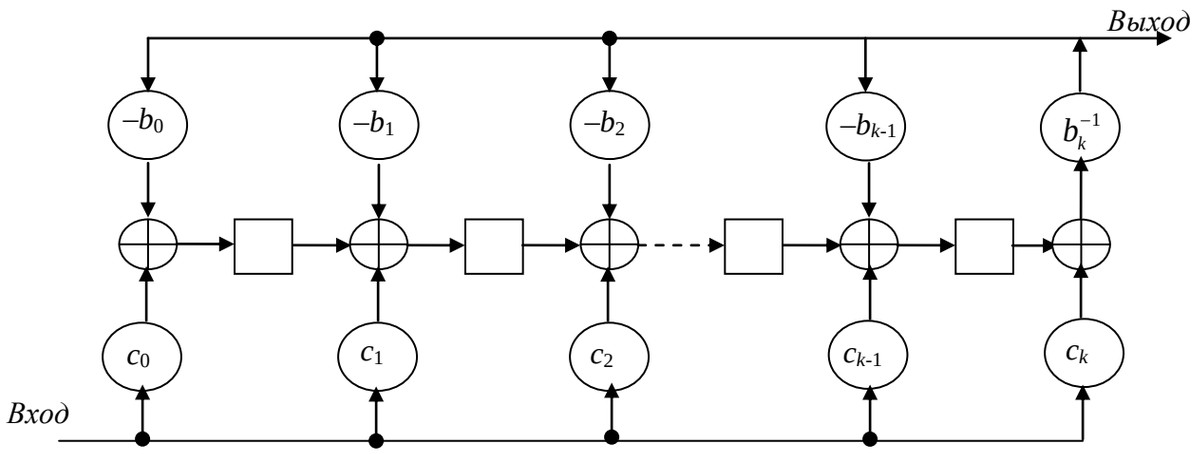
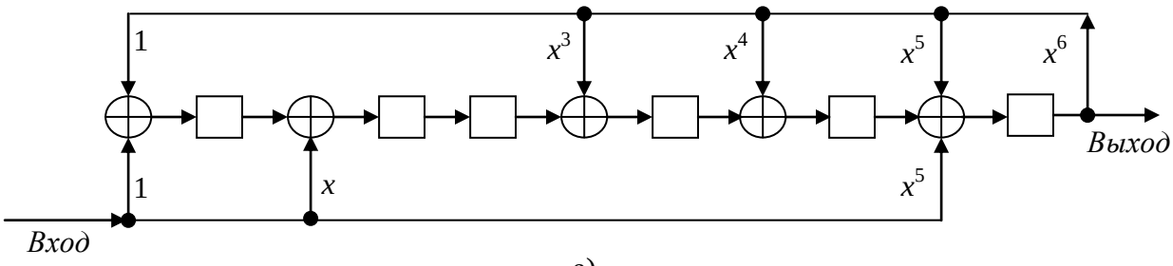
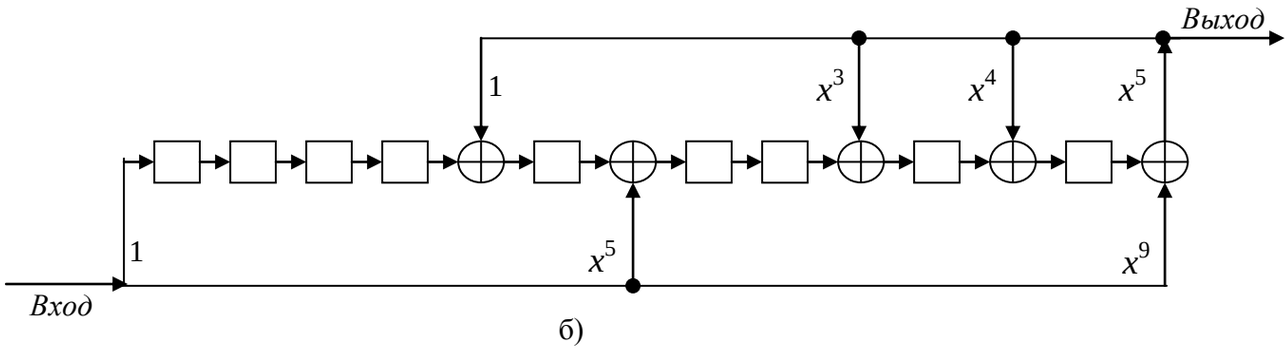


Рис. 2.7. Схема, реализующая одновременное умножение на многочлен $h(x) = c_0 + c_1x + .. + c_kx^k$ и деление на многочлен $g(x) = b_0 + b_1x + b_2x^2 + ... + b_kx^k$



а)



б)

Рис. 2.8. Примеры схем, реализующих одновременное умножение на многочлен $h(x)$ и деление на многочлен $g(x)$ для:
а) $h(x)= 1+ x + x^5$ и $g(x)=1+x^3+x^4+ x^5+ x^6$;
б) $h(x)= 1+ x^5 + x^9$ и $g(x)=1+x^3+x^4+ x^5$

2.3. Реализация действий над элементами расширенного поля

При построении кодирующих и декодирующих устройств циклических кодов, особенно БЧХ и Рида–Соломона, должны быть реализованы операции умножения и деления в поле GF(2^k). Рассмотрим реализационные основы таких действий над элементами конечного поля Галуа.

Пусть задан примитивный многочлен k-й степени над простым полем GF(2) :

$$g(x)= x^k+p_1 x^{k-1} + ... +p_{k-1}x+p_k; \quad p_i \in GF(2). \tag{2.8}$$

Сопровождающая матрица F такого многочлена имеет вид

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 1 \\ p_k & p_{k-1} & p_{k-2} & p_{k-3} & \cdots & p_1 \end{bmatrix} \tag{2.9}$$

Квадратной матрице F соответствует характеристический многочлен $f(\lambda)$, представляющий собой определитель характеристической матрицы $[\lambda E - F]$, где E – квадратная единичная матрица k-го порядка, т. е. $f(\lambda) = |\lambda E - F|$. Корни многочлена $f(\lambda)$ λ_1, λ_2, ... λ_k, называемые характеристическими числами, являются решением векового [9] уравнения $|\lambda E - F| = 0$.

Как видно из этого уравнения, одним из корней многочлена $f(\lambda)$ является сопровождающая матрица F. Действительно, если в характеристический многочлен $f(\lambda)$ вместо λ подставить F, то получим $f(F) = |FE - F| = 0$. Тем самым доказана теорема Гамильтона–Кэли [9], в соответствии с которой

всякая квадратная матрица F является корнем своего характеристического многочлена $f(\lambda)$.

С другой стороны, легко проверить, что заданный примитивный многочлен $g(x)$ в точности совпадает с характеристическим многочленом $f(\lambda)$ при подстановке $x = \lambda$, т. е. $f(\lambda) = g(\lambda)$. Отсюда следует, что матрица F является также корнем многочлена $g(x)$, т. е. $g(F) = 0$. Последнее условие приводит к важному следствию, заключающемуся в том, что множество элементов поля $GF(2^k)$, построенного по примитивному многочлену $g(x)$, изоморфно множеству многочленов – вычетов по модулю $g(F)$. Другими словами, любому элементу ε^m поля $GF(2^k)$, выраженному через степенной базис 1, ε, ε^2, ..., ε^{k-1},

т. е. $$\varepsilon^m = a_0 + a_1\varepsilon + a_2\varepsilon^2 + \ldots + a_{k-1}\varepsilon^{k-1},$$

где ε – первообразный элемент поля $GF(2^k)$, а коэффициенты $a_i \in GF(2)$, взаимнооднозначно соответствует многочлен

$$F^m = a_0E + a_1F + a_2F^2 + \ldots + a_{k-1}F^{k-1},$$

приведенный по модулю $g(F)$. Указанное свойство позволяет достаточно просто реализовать умножение и деление элементов поля $GF(2^k)$ в векторной форме. Рассмотрим эти действия.

2.3.1. Умножение элементов поля

Пусть необходимо умножить элемент ε^i на элемент ε^j. С этой целью выразим эти элементы поля $GF(2^k)$ через степенной базис следующим образом:

$$\varepsilon^i = a_0 + a_1\varepsilon + a_2\varepsilon^2 + \ldots + a_{k-1}\varepsilon^{k-1};\ a_i \in GF(2)$$
$$\varepsilon^j = b_0 + b_1\varepsilon + b_2\varepsilon^2 + \ldots + b_{k-1}\varepsilon^{k-1}.\ b_i \in GF(2)$$

Тогда произведение $\varepsilon^i\varepsilon^j = \varepsilon^m = c_0 + c_1\varepsilon + c_2\varepsilon^2 + \ldots + c_{k-1}\varepsilon^{k-1}$ может быть реализовано в векторной форме следующим образом:

$$(\varepsilon_0\ \varepsilon_1 \ldots\ \varepsilon_k)\,F^j = (\varepsilon_0\ \varepsilon_1 \ldots\ \varepsilon_k)\,[b_0E + b_1F + \ldots + b_{k-1}F^{k-1}] = b_0\,[(\varepsilon_0\ \varepsilon_1 \ldots\ \varepsilon_{k-1})E +$$
$$+ b_1[(\varepsilon_0\ \varepsilon_1 \ldots\ \varepsilon_{k-1})F] + \ldots + b_{k-1}[(\varepsilon_0\ \varepsilon_1 \ldots\ \varepsilon_{k-1})\,F^{k-1}] = (c_0\ c_1 \ldots\ c_{k-1}) \qquad (2.10)$$

Напомним, что для двоичных кодов сложение производится по mod 2.

Структурная схема умножителя в соответствии с алгоритмом (2.10) представлена на рис. 2.9.

Рассмотрим конкретный пример.

Пусть поле GF (2^4) образовано примитивным многочленом $g(x) = 1 + x + x^4$. Сопровождающая матрица F этого многочлена и ее степени F^2 и F^3 имеют вид

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix};\quad F^2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix};\quad F^3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Необходимо произвести умножение двух элементов ε^i и ε^j, принадлежащих полю $GF(2^4)$.

Выразим ε^i и ε^j через базис $[1, \varepsilon, \varepsilon^2, \varepsilon^3]$ следующим образом:

$$\varepsilon^i = a_0 + a_1\varepsilon + a_2\varepsilon^2 + a_3\varepsilon^3;$$
$$\varepsilon^j = b_0 + b_1\varepsilon + b_2\varepsilon^2 + b_3\varepsilon^3.$$

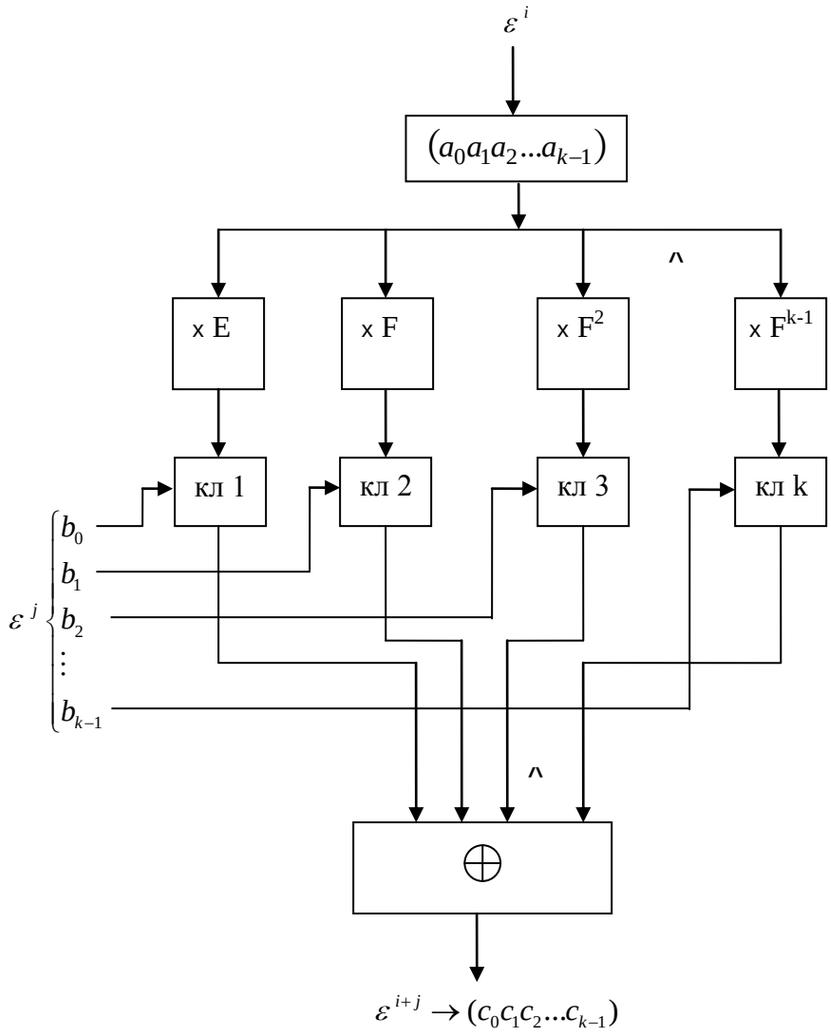


Рис. 2.9. Структурная схема умножителя произвольных элементов поля GF (2^k)

Таким образом, векторная форма записи элементов ε^i и ε^j будет соответственно $[a] = [a_0, a_1, a_2, a_3]$ и $[b] = [b_0, b_1, b_2, b_3]$. Заменив элемент поля ε^j на F^j равный $F^j = b_0E + b_1F + b_2F^2 + b_3F^3$, найдем произведение $(\varepsilon^i \varepsilon^j) = c_0 + c_1\varepsilon + c_2\varepsilon^2 + c_3\varepsilon^3$ в векторной форме, используя алгоритм (2.10), а именно

$$[c]= [c_0, c_1, c_2, c_3] = [a] F^j = b_0[a]E + b_1[a]F + b_2[a]F^2 + b_3[a]F^3$$

Учтем, что для данного многочлена $g(x)$ и его матрицы F имеют место равенства:

$$\begin{aligned} &[a]E = [a_0, a_1, a_2, a_3];\\ &[a]F = [a_3, (a_0 + a_3), a_1, a_2];\\ &[a]F^2= [a_2, (a_2 + a_3), (a_0 + a_3), a_1];\\ &[a]F^3= [a_1, (a_1 + a_2), (a_2 + a_3), (a_0 + a_3)]. \end{aligned} \qquad (2.11)$$

Тогда вектор $[c] = [c_0, c_1, c_2, c_3]$, соответствующий произведению $\varepsilon^i \varepsilon^j$, будет равен поэлементной сумме по модулю 2:

$$[c]= b_0 [a_0, a_1, a_2, a_3] + b_1[a_3, (a_0 + a_3), a_1, a_2] + b_2 [a_2, (a_2 + a_3), (a_0 + a_3), a_1] + + b_3 [a_1, (a_1 + a_2), (a_2 + a_3), (a_0 + a_3)].$$

Функциональная схема, реализующая рассмотренный алгоритм умножения в поле GF(2^4), представлена на рис 2.10.

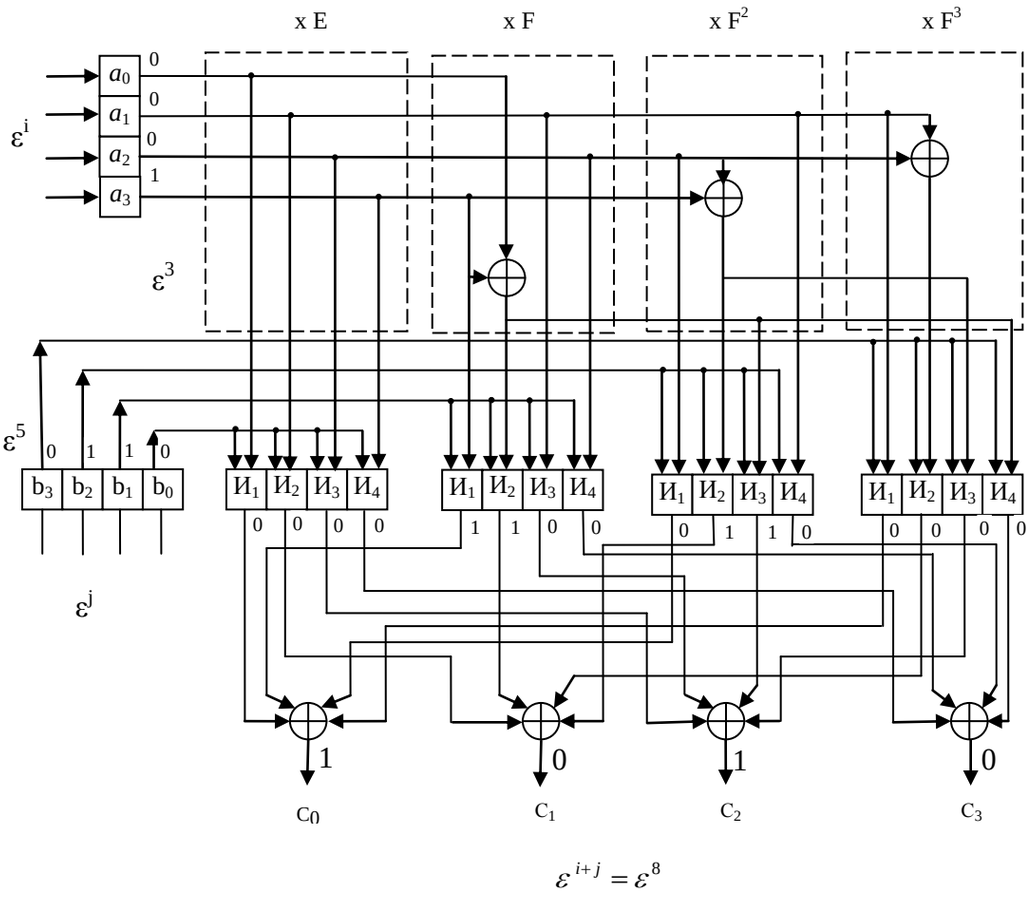


Рис. 2.10. Функциональная схема умножения произвольных элементов поля GF (2^4) с образующим многочленом $g(x) = 1 + x + x^4$

Схема существенно упрощается, если необходимо произвольный элемент поля ε^i умножить на константу ε^j . Пусть для рассматриваемого выше примера необходимо производить умножение на ε^5. Тогда произведение $\varepsilon^i \varepsilon^5$ в векторной форме будет выражаться следующим образом:

$$[a]F^5 = [a_0, a_1, a_2, a_3]\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} = \left[(a_2 + a_3), (a_0 + a_2), (a_0 + a_1 + a_3), (a_1 + a_2)\right].$$

Функциональная схема такого умножителя представлена на рис. 2.11.

Другой распространенный в настоящее время алгоритм умножения элементов поля GF(2^k) основан на применении операций логарифмирования и антилогарифмиравания в полях GF(2^k) [4, 6].

Если некоторый элемент поля $\beta = \varepsilon^i$ принадлежит полю GF(2^k), то логарифм элемента β по основанию $\varepsilon \in$ GF(2^k) есть показатель степени i, а именно: $\log_a \beta = \log_a \varepsilon^i = i$, $i \geq 0$. Тогда умножение $\varepsilon^i \varepsilon^j = \varepsilon^m$ может быть реализовано по следующему алгоритму, представленному на рис. 2.12.

Особенностью данного алгоритма является то, что комбинациям из k нулей или k единиц на выходе арифметического сумматора должно соответствовать появление на выходе антилогарифматора одинакового элемента, а именно $\varepsilon^0 = 1$.

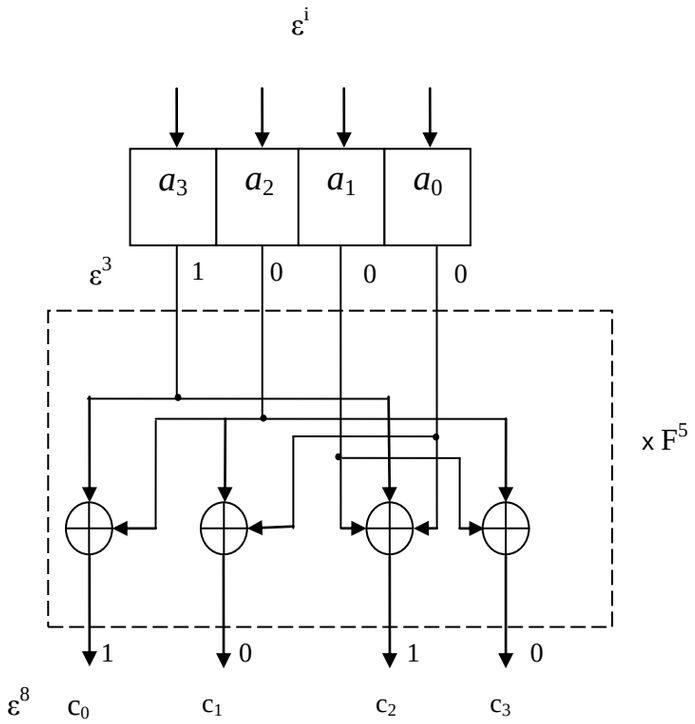


Рис. 2.11. Функциональная схема умножения произвольного элемента ε^i поля GF(2^4) на константу ε^5

Логарифмирование и антилогарифмирование реализуется чаще всего табличным способом.

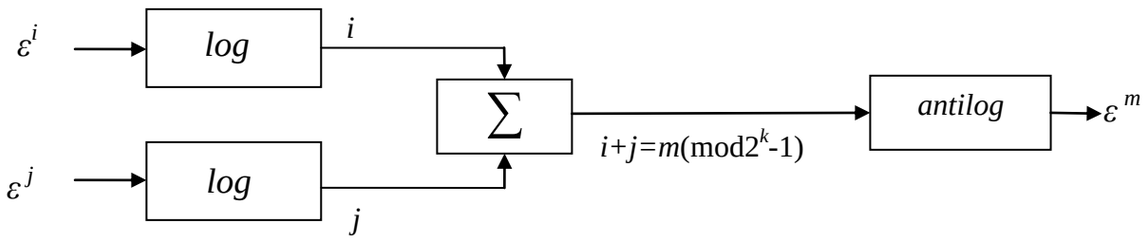


Рис. 2.12. Структурная схема умножителя произвольных элементов ε^i и ε^j в поле GF(2^k) с использованием логарифмировании и антилогарифмирования

2.3.2. Деление элементов поля

В теории циклических кодов часто встречающейся операцией является деление произвольного элемента ε^i на некоторый элемент ε^j, где ε^i , $\varepsilon^j \in$ GF(2^k). Вместе с тем на практике деление элементов заменяется умножением элемента ε^i на ε^{-j}, т. е.

$$\varepsilon^i / \varepsilon^j = \varepsilon^i \varepsilon^{-j},$$

где ε^{-j} – элемент поля, обратный элементу ε^j. Напомним, что для любого элемента ε^j в поле GF(2^k) всегда существует и обратный ему ε^{-j}, которые связаны равенством $\varepsilon^j \varepsilon^{-j} = 1$. Таким образом, реализация деления осуществляется по той же схеме, что и умножение (рис. 2.11) с той разницей, что один из сомножителей является обратным элементом ε^{-j}. Рассмотрим способы *обращения элементов* поля GF(2^k).

Один из способов основан на свойстве полей Галуа, которое заключается в том, что ненулевые элементы поля GF(2^k) образуют циклическую мультипликативную группу порядка 2^k–1, т. е. для любого ненулевого

элемента ε^j справедливо равенство $(\varepsilon^j)^{2^k-1}=1.$ Отсюда получается $(\varepsilon^j)^{2^k-2}=(\varepsilon^j)^{-1}=\varepsilon^{-j}.$

Для вычисления (2^k-2)-й степени элемента β можем воспользоваться одним из следующих двух равенств:

$$(\beta)^{2^k-2}=(\beta)^2(\beta)^{2^2}(\beta)^{2^3}\ldots(\beta)^{2^{k-1}}=\beta^{-1} \tag{2.12}$$

или

$$(\beta)^{2^k-2}=\left\{\left[\left(\beta^2\cdot\beta\right)^2\cdot\beta\right]^2\ldots\beta\right\}^2=\beta^{-1} \tag{2.13}$$

Характерной особенностью равенства (2.12) является то, что обращение элемента β может быть реализовано аппаратным путем за один шаг (такт), тогда как равенство (2.13) может быть реализовано за (k–1) последовательных шагов. Вместе с тем, реализация последнего значительно проще, так как на каждом шаге необходимо осуществить умножение текущего значения на β и возведение результата в квадрат. Если элемент β выразить через базисные элементы поля 1, ε, ε^2, ..., ε^{k-1} в виде $\beta = a_0 + a_1\varepsilon + a_2\varepsilon^2 + \ldots + a_{k-1}\varepsilon^{k-1}$, то на основании свойств (1.4) и (1.5) возведение в квадрат в поле GF(2^k) соответствует выражению

$$\beta^2 = (a_0 + a_1\varepsilon + a_2\varepsilon^2 + \ldots + a_{k-1}\varepsilon^{k-1})^2 = a_0 + a_1\varepsilon^2 + a_2(\varepsilon^2)^2 + \ldots + a_{k-1}(\varepsilon^{k-1})^2.$$

Тогда операцию возведения в квадрат в векторной форме можно записать

$$(a_0a_1a_2\ldots a_{k-1})\,[X] = (b_0b_1b_2\ldots b_{k-1}),$$

где [X] – квадратная матрица k-го порядка, строки которой представляют собой двоичные векторы элементов поля

$$\varepsilon^0, \varepsilon^2, \varepsilon^4, \ldots, \varepsilon^{2(k-1)}.$$

Пример 2.5. Построить матрицу [X] и схему возведения в квадрат, если конечное поле GF(2^4) образовано примитивным многочленом $g(x) = 1+x+x^4$.

Выразим элементы поля ε^0, ε^2, ε^4, ε^6 через левый степенной базис {1, ε, ε^2, ε^3} в виде $\varepsilon^i = c_0 + c_1\varepsilon + c_2\varepsilon^2 + c_3\varepsilon^3$, где $c_i \in$ GF(2):

$$(\varepsilon^0)^2 = \varepsilon^0 = 1;$$
$$(\varepsilon^1)^2 = \varepsilon^2;$$
$$(\varepsilon^2)^2 = \varepsilon^4 = 1+\varepsilon;$$
$$(\varepsilon^3)^2 = \varepsilon^6 = \varepsilon^2+\varepsilon^3.$$

Из векторов ($c_0\,c_1\,c_2\,c_3$) указанных элементов составим матрицу [X]:

$$[X]=\begin{bmatrix}1&0&0&0\\0&0&1&0\\1&1&0&0\\0&0&1&1\end{bmatrix}.$$

Пусть элементу ε^i соответствует вектор ($a_0\,a_1\,a_2\,a_3$). Для возведения ε^i в квадрат произведем умножение вектора ($a_0\,a_1\,a_2\,a_3$) на матрицу [X]:

$$(a_0, a_1, a_2, a_3)\cdot\begin{bmatrix}1&0&0&0\\0&0&1&0\\1&1&0&0\\0&0&1&1\end{bmatrix}=(a_0+a_2, a_2, a_1+a_3, a_3)=(b_0, b_1, b_2, b_3)$$

Логическая схема возведения в квадрат в поле GF(2^4) с образующим многочленом $g(x) = 1 + x + x^4$ представлена на рис. 2.13.

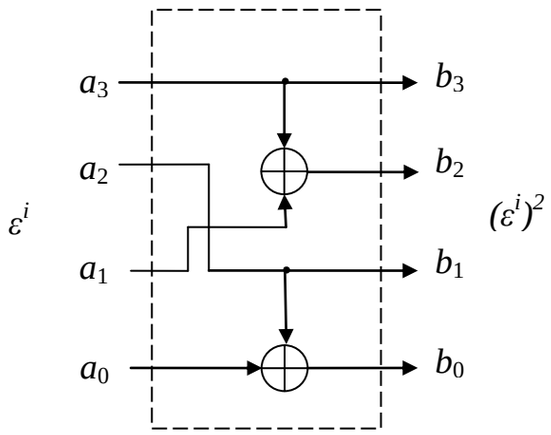


Рис. 2.13. Функциональная схема возведения в квадрат произвольного элемента ε^i в поле GF(2^k)

Для проверки возведем элемент ε^5 в квадрат. Так как $\varepsilon^5 = \varepsilon + \varepsilon^2$, для получения элемента $(\varepsilon^5)^2$ выполним умножение вектора элемента ε^5 на матрицу [X], т. е.

$$(0\,1\,1\,0)\cdot\begin{bmatrix}1&0&0&0\\0&0&1&0\\1&1&0&0\\0&0&1&1\end{bmatrix}=(1\,1\,1\,0)$$

Полученный вектор (1110) соответствует элементу $(\varepsilon^5)^2 = \varepsilon^{10}$ поля GF(2^k).

2.4.Общие принципы построения циклических кодов

Циклический код относится к числу блоковых равномерных кодов, он образуется из равномерного k-элементного кода, множество комбинаций которого представляет собой конечную группу порядка p^k, где p – основание кода.

Характерной особенностью циклических кодов является полиномиальное представление кодовых комбинаций длины n многочленами степени (n – 1):

$$f(x) = a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^{n-1}. \qquad (2.14)$$

Другой распространенной формой представления комбинаций циклического кода является векторная форма (a_0 a_1 a_2 … $a_{n\text{-}1}$), где элементы кода a_i являются коэффициентами многочлена $f(x)$, принадлежащими простому полю GF(p). Для двоичных циклических кодов коэффициенты a_i принадлежат полю GF(2), т.е. принимают значения 0 и 1.

Таким образом, обе формы представления комбинаций связаны между собой тем, что одна вытекает из другой. Например, комбинации (1010001) соответствует многочлен $f(x) = 1 + x^2 + x^6$.

Приведем основные свойства циклических кодов, которые определяют их построение.

Циклический код образуется путем умножения каждой комбинации равномерного k-элементного кода, записанной в виде многочлена Q(x) = b_0 + b_1x + ... + $b_{k-1}x^{k-1}$ степени (k – 1) и с коэффициентами b_i – вычетами по модулю p, т. е., принимающими значения 0,1, …, (p – 1), на некоторый многочлен $G(x)$

степени $n–k$. Умножение производится по обычным правилам алгебры с приведением подобных членов по модулю p.

Следовательно, любая комбинация циклического кода может быть записана следующим образом:

$$f(x) = Q(x)\,G(x) = (b_0 + b_1 x + b_2 x^2 + \ldots + b_{k-1} x^{k-1})G(x) = \\ = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} \qquad (2.15)$$

Отсюда выявляется важное свойство циклических кодов:

Свойство 2.1. *Циклический (n,k)-код в полиномиальном его представлении можно определить как множество многочленов степени (n – 1) и меньше, каждый из которых делится без остатка на некоторый многочлен G(x) степени (n–k), называемый образующим или порождающим многочленом кода.*

Из этого свойства вытекает, что для каждой комбинации циклического кода, представленной многочленом $f(x)$, справедливо сравнение

$$f(x) \equiv 0 [\mathrm{mod}\ G(x)]. \qquad (2.16)$$

Следующее свойство циклических кодов, определяющее их название, формулируется так:

Свойство 2.2. *Если некоторая комбинация (a_0, a_1, a_2, …, $a_{n–1}$) принадлежит какому-либо циклическому коду, то и комбинация, полученная из предыдущей в результате циклического сдвига ее элементов, например ($a_{n–1}$, a_0, a_1, …, $a_{n–2}$), также принадлежат этому циклическому коду.*

Это свойство основывается на свойстве полей Галуа, согласно которому любой неприводимый в поле двоичных чисел многочлен $G(x)$ степени m является делителем двучлена $(x^{2^m-1}-1)$, т.е.

$$x^{2^m-1} - 1 \equiv 0 [\mathrm{mod}\, G(x)]. \qquad (2.17)$$

Таким образом, выбрав длину комбинации n, удовлетворяющую равенству $n= 2^m–1$, получим

$$x^n - 1 \equiv 0 [\mathrm{mod}\, G(x)]. \qquad (2.18)$$

При таком значении n, умножив на x многочлен комбинации циклического кода $f(x)$, представленного в виде (2.14), получим: $x\,f(x) = a_0\,x + a_1 x^2 + a_2 x^3 + \ldots + a_{n-1} x^n$. В силу (2.18) справедливо сравнение: $a_{n-1} x^n \equiv a_{n-1} [\mathrm{mod}\ G(x)]$. Тогда получаем:

$$xf(x) \equiv a_{n-1} + a_0\,x + a_1 x^2 + a_2 x^3 + \ldots + a_{n-2} x^{n-1}. \qquad (2.19)$$

Так как, согласно свойству 2.1, многочлен $f(x)$ делится на порождающий многочлен $G(x)$ без остатка, то и многочлен $x\,f(x)$ также делится на $G(x)$ без остатка. Отсюда, вектор ($a_{n–1}$, a_0, a_1, …, $a_{n–2}$), полученный из комбинации циклического кода (a_0, a_1, a_2, …, $a_{n–1}$) путем циклического сдвига, также будет принадлежать циклическому коду. Из доказательства свойства 2.2 вытекает следствие, которое заключается в том, что для большинства циклических кодов принимают $n=2^m–1$, где m – степень неприводимого порождающего многочлена $G(x)$. Но на практике иногда выбирают длину комбинации n_1 меньше $2^m–1$. Такой циклический код называют укороченным.

Из предыдущих свойств вытекает еще одно важное свойство 2.3 циклических кодов.

Свойство 2.3. *Так как любая комбинация циклического кода, представленная в виде многочлена f(x), делится на порождающий многочлен G(x) без остатка, а G(x), в силу (2.18-++), является делителем двучлена (x^n – 1), циклический код в алгебре многочленов A_n по модулю F(x)= (x^n – 1) будет являться идеалом I, порожденным многочленом G(x).*

Тогда, по определению, идеалу будут принадлежать все многочлены, кратные $G(x)$, а именно: $G(x)$, $x \cdot G(x)$, $x^2 \cdot G(x)$, $(1 + x)\, G(x)$,..., $f_1(x)\, G(x)$.

Наконец, сформулируем свойство *линейности* циклических кодов.

Свойство 2.4. *Так как множество комбинаций циклического кода представляют собой циклическую группу, по условию замкнутости группы, сумма любых комбинаций циклического кода порождает комбинацию, также принадлежащую этому же циклическому коду.*

Рассмотрим теперь общие принципы построения циклического кода. Задачей кодирующего устройства является преобразование k-элементной комбинации исходного кода в n-элементную комбинацию (a_0, a_1, a_2,…, a_{n-1}) избыточного циклического (n, k)-кода. Таким образом, каждая комбинация циклического кода будет содержать ($n-k$) избыточных элементов.

Задачей декодирующего устройства является обратное преобразование принятой n-элементной комбинации циклического (n, k)-кода в исходную k-элементную комбинацию. При этом эффективность циклического кода оценивается его способностью корректировать возникающие при передаче по каналу связи ошибки.

Применяют два основных способа кодирования. Один из них заключается в том, что многочлен (2.14), соответствующий комбинации циклического кода, получают путем умножения многочлена $\varphi(x)$ степени ($k-1$), соответствующего исходной комбинации, на порождающий многочлен $G(x)$ степени m, т.е.

$$f(x) = \varphi(x)\, G(x). \qquad (2.20)$$

Отсюда видно, что для циклического (n, k)-кода степень порождающего многочлена $m = (n-k)$, т. е. равна числу избыточных элементов комбинации.

Циклический код, построенный в соответствии с (2.20), будет *несистематическим*, особенностью которого является то, что среди элементов комбинации (a_0, a_1, a_2, …, a_{n-1}) не могут быть выделены отдельно информационные и проверочные элементы.

Другой способ кодирования соответствует *систематическому* циклическому (n, k)-коду. Построение систематического циклического кода сводится к тому, что к информационным элементам исходного k-элементного кода добавляются ($n-k$) полученных определенным образом избыточных элементов. Причем в комбинации n-элементного циклического кода информационные и избыточные (проверочные) элементы будут занимать определенные позиции. Значения избыточных элементов должны определяться таким образом, чтобы выполнялось свойство 2.1 циклических кодов, в соответствии с которым комбинация циклического кода $f(x)$ должна делиться на порождающий многочлен $G(x)$ без остатка.

Чтобы построить систематический циклический код, комбинацию исходного k-элементного кода записывают в виде многочлена $\varphi(x)$ степени ($k-1$). Затем многочлен $\varphi(x)$ умножается на x^{n-k}, что равносильно

приписыванию к исходной k-элементной комбинации $(n-k)$ нулей со стороны младшего разряда.

Полученное таким образом произведение $x^{n-k}\cdot\varphi(x)$ делим на производящий многочлен $G(x)$ степени $(n-k)$. При делении на порождающий многочлен $G(x)$ образуется частное $Q(x)$ и возможный остаток $R(x)$, т. е.

$$\frac{x^{n-k}\varphi(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}.$$

Полученное выражение преобразуется к следующему виду:

$$Q(x)G(x) = x^{n-k}\varphi(x) - R(x). \qquad (2.21)$$

Очевидно, что многочлен (2.21) удовлетворяет свойству 2.3 и соответствует комбинации циклического кода.

Так как в двоичном поле знак «−» равносилен знаку «+», то только для двоичных циклических кодов выражение (2.21) будем записывать так:

$$f(x) = Q(x)G(x) = x^{n-k}\varphi(x) + R(x). \qquad (2.22)$$

Степень остатка $R(x)$ меньше степени производящего многочлена $G(x)$, равной $(n-k)$. Поэтому число разрядов в комбинации, отображающих многочлен $R(x)$, не превышает $n-k$.

Пример 2.6. Рассмотрим пример построения систематического циклического кода $(n, k)=(15,11)$. В качестве порождающего выберем многочлен $G(x) = 1 + x + x^4$.

Для кодирования комбинации (10100100001), которой соответствует многочлен $\varphi(x) = 1 + x^2 + x^5 + x^{10}$, разделим $x^4\cdot\varphi(x)$ на порождающий многочлен $G(x)$ и найдем остаток $R(x)$. В результате находим, что $R(x) = x^2 + x^3$, тогда в соответствии с (2.22) многочлен кодовой комбинации систематического кода будет иметь вид:

$$f(x) = R(x) + x^4\varphi(x) = x^2 + x^3 + x^4 + x^6 + x^9 + x^{14}.$$

Этому многочлену соответствует двоичная кодовая комбинация (старший разряд справа):

0011	10100100001
проверочные элементы	информационные элементы

Матричное представление циклических кодов

Систематический циклический код может быть полностью определен своей порождающей матрицей, которая строится по следующему правилу. Сначала записывается единичная матрица E_k для исходного k-элементного кода. Далее многочлен, соответствующий каждой из комбинаций E_k, умножается на x^{n-k} и делится на производящий многочлен $G(x)$. В результате деления получим остатки, каждый из которых соответствует определенной строке матрицы E_k. Из этих остатков составляется дополнительная матрица R, которая приписывается к матрице E_k со стороны младших разрядов. Таким образом, порождающая матрица для систематического циклического (n, k)-кода будет иметь вид $G = [R\ E_k]$, где R − приписанная к E_k матрица остатков или матрица проверочных элементов размерности $k\times(n-k)$.

Все остальные комбинации циклического кода, в силу свойства линейности, получаются путем сложения по модулю 2 строк матрицы G в различных сочетаниях.

Пример 2.7. Пусть необходимо построить циклический код, для которого $n = 7$, $k = 4$, $n - k = 3$, а $G(x) = 1 + x + x^3$.

Прежде всего, запишем единичную матрицу E_k:

$$E_k = \begin{bmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \end{bmatrix},$$

строкам которой соответствуют многочлены $1, x, x^2, x^3$.

Каждая строка умножается на $x^{n-k} = x^3$, тогда получим новую матрицу, в которой каждая строка будет сдвинута на $n - k$ нулей:

$$\begin{bmatrix} 0001000 \\ 0000100 \\ 0000010 \\ 0000001 \end{bmatrix} \quad \textit{или} \quad \begin{bmatrix} x^3 \\ x^4 \\ x^5 \\ x^6 \end{bmatrix}.$$

Каждый из полученных многочленов делится на порождающий многочлен $G(x)$ и находится соответствующий остаток, который записывается на месте первых $n - k$ нулей в новой матрице.

Найденные остатки от деления многочленов x^3, x^4, x^5, x^6 на $G(x) = 1 + x + x^3$ будут соответственно равны $R_1(x) = 1 + x$; $R_2(x) = x + x^2$; $R_3(x) = 1 + x + x^2$; $R_4(x) = 1 + x^2$.

Теперь составляем матрицу проверочных элементов R, элементами которой являются остатки в двоичной форме записи:

$$R = \begin{bmatrix} 110 \\ 011 \\ 111 \\ 101 \end{bmatrix}$$

Приписывая эту матрицу к единичной матрице E_k, получим порождающую матрицу для циклического кода

$$G = [R\, E_k] = \begin{bmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{bmatrix}.$$

Отметим, что порождающая матрица G позволяет закодировать комбинацию исходного k-элементного кода циклическим n-элементным кодом путем умножения на матрицу G. Пусть для рассматриваемого примера кода исходная комбинация будет (1110), т. е. $\varphi(x) = 1 + x + x^2$. Умножив ее на матрицу G, получим циклическую кодовую комбинацию (0101110).

В [7] доказана следующая теорема.

«Если V − пространство строк матрицы $G = [E_k R]$, где E_k − единичная матрица размерности $k \times k$, а R− матрица остатков размерности $k \times (n - k)$, то V является нулевым пространством матрицы $H=[R^T E_{n-k}]$, где E_{n-k} − единичная матрица размерности $(n - k) \times (n - k)$, *а* R^T − транспонированная матрица R.»

По этой теореме полученный циклический (n, k)-код, порождающая матрица которого имеет вид $G = [R E_k]$, является нулевым пространством матрицы $H= [E_{n-k} R^T]$ и, следовательно, $G \cdot H^T = 0$.

Матрицу H называют проверочной матрицей систематического циклического (n, k)-кода, так как в результате умножения вектора любой разрешенной комбинации циклического кода на матрицу H^T, всегда будет получен 0.

Так, для рассматриваемого выше примера матрица H, с учетом найденной ранее матрицы R, будет иметь вид:

$$H = \left[E_{n-k} R^T \right] = \begin{bmatrix} 1001011 \\ 0101110 \\ 0010111 \end{bmatrix}$$

Легко проверить, что $G \cdot H^T = 0$.

Для несистематического кода порождающая матрица G строится в соответствии с правилом построения несистематического циклического кода, а именно, путем умножения многочлена исходной комбинации $\varphi(x)$ на порождающий многочлен $G(x)$. Для построения порождающей матрицы G составляют единичную матрицу E_k и каждую её строку, представленную одночленом x^i, умножают на многочлен $G(x)$. Из двоичных коэффициентов полученных многочленов $x^i G(x)$ составляют порождающую матрицу G несистематического кода. Для рассматриваемого выше примера циклического кода с $n = 7$, $k = 4$ и $G(x) = 1 + x + x^3$, записав произведения $x^i \, G(x)$, где i=0,1,2,3, получим следующую матрицу G:

$$G = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix}.$$

Так же как и для систематического кода, несистематический циклический код должен представлять собой нулевое пространство по отношению к некоторой проверочной матрице H, которая может быть построена по проверочному многочлену

$$H(x) = \frac{x^n + 1}{G(x)}. \qquad (2.23)$$

Выражение (2.23) следует из свойства 2.2 циклических кодов и сравнения (2.18), согласно которому многочлен $G(x)$ является делителем двучлена x^n+1, где $n=2^{n-k} - 1$. Тогда строки проверочной матрицы H будут состоять из двоичных коэффициентов многочленов $x^i \cdot H(x)$, i=0,1,…,(n–k–1). Так, для рассматриваемого выше кода (7,4) имеем $H(x) = \dfrac{x^7 + 1}{G(x)} = 1 + x + x^2 + x^4$. Записав в двоичном виде многочлены $H(x)$, $x \cdot H(x)$, $x^2 \cdot H(x)$, составим проверочную матрицу H:

$$H = \begin{bmatrix} 0010111 \\ 0101110 \\ 1011100 \end{bmatrix},$$

в которой строки записаны в обратном порядке, т. е. со старшей степени слева.

Легко проверить, что произведение $G \cdot H^T = 0$.

Следует отметить, что исходя из свойства 2.1 циклических кодов, может быть построена проверочная матрица H, единая как для систематического, так и для несистематического кодов. Согласно этому свойству, если некоторый элемент ε поля GF(2^n) является корнем порождающего многочлена $G(x)$, то он является также корнем и многочлена $f(x)$. Отсюда следует, что любая кодовая комбинация циклического кода, представленная многочленом (2.14), удовлетворяет равенству

$$f(\varepsilon) \equiv 0 \ [\text{mod } G(x)], \qquad (2.24)$$

которое может быть получено как произведение $[a_0, a_1, a_2, \ldots, a_{n-1}]\, H^T$, где H – проверочная матрица вида

$$H = [1 \varepsilon \varepsilon^2 \ldots \varepsilon^{n-1}]. \qquad (2.25)$$

Учитывая, что для рассматриваемого выше примера 2.7 кода (7,4) порождающий многочлен $G(x) = 1 + x + x^3$ является примитивным и его корень ε будет первообразным, проверочная матрица H будет состоять из всего множества ненулевых элементов поля GF(2^3), т.е. $H = [1 \varepsilon \varepsilon^2 \varepsilon^3 \varepsilon^4 \varepsilon^5 \varepsilon^6]$.

Заменив элементы ε^i двоичными векторами в виде столбцов, получим проверочную матрицу

$$H = \begin{bmatrix} 1001011 \\ 0101110 \\ 0010111 \end{bmatrix}.$$

Легко проверить, что полученная проверочная матрица H удовлетворяет равенству $G \cdot H^T = 0$ как для систематического, так и для несистематического циклического (7,4)-кода с порождающим многочленом $G(x) = 1 + x + x^3$.

Принципы обнаружения и исправления ошибок циклическими кодами

Принцип обнаружения ошибок в классических циклических кодах основывается на делении принятой кодовой комбинации на порождающий многочлен $G(x)$. В общем виде принятая комбинация $h(x)$ представляет собой сумму $h(x) = f(x) + e(x)$, где $f(x)$ – многочлен, соответствующий переданной кодовой комбинации, $e(x)$ – многочлен, соответствующий вектору ошибок.

Многочлен $e(x)$ имеет ненулевые коэффициенты у членов, соответствующих ошибочным элементам в принятой комбинации.

Если при передаче сообщения ошибок не было, то принятая комбинация равна переданной кодовой комбинации, т.е. $h(x) = f(x) = Q(x)G(x)$, поэтому многочлен $h(x)$ делится на $G(x)$ без остатка. Однако из-за помех могут произойти ошибки, многочлен которых $e(x)$ не равен нулю и делится на $G(x)$ без остатка. В таком случае комбинация $h(x)$ принимается как разрешенная кодовая комбинация, а ошибки считаются необнаруженными. При всех

других ошибках многочлен $e(x)$ не делится на $G(x)$ без остатка, благодаря чему ошибки обнаруживаются.

Следовательно, ошибка обнаруживается, если остаток от деления многочлена $h(x)$ на $G(x)$ не равен нулю. Если остаток равен нулю, то ошибка или отсутствует или не обнаруживается кодом.

Пусть, например, $G(x) = 1 + x + x^4$, $f(x) = x^2 + x^3 + x^4 + x^6 + x^9 + x^{14} =$ 001110100100001;

$e(x) = x^6 + x^9 =$ 000000100100000;

$h(x) = f(x) + e(x) = x^2 + x^3 + x^4 + x^6 + x^{14} =$ 001110100000001.

В результате деления $h(x)$ на $G(x)$ получается отличный от нуля остаток, равный

$$R(x) = x + x^2,$$

что говорит о наличии ошибок в принятой комбинации.

Такой же остаток получается, если на $G(x)$ делить $e(x)$, так как $f(x)$ делится на $G(x)$ без остатка.

Исправление ошибок является более трудной задачей, чем обнаружение. Реализация такой операции возможна, если многочлены различных комбинаций исправляемых ошибок будут давать отличающиеся друг от друга остатки от деления на порождающий многочлен $G(x)$. Поэтому исправление ошибки на приеме может быть осуществлено с помощью следующих операций.

1. Принятое сообщение делится на $G(x)$ и вычисляется остаток.
2. Из таблиц или путем некоторых вычислений находится многочлен ошибок $e(x)$, соответствующий остатку.
3. Найденный многочлен $e(x)$ вычитается из $h(x)$, в результате чего получается $f(x)$, т. е. переданное сообщение.

По другому реализуется процедура обнаружения и исправления ошибок циклическими кодами, основанная на том свойстве, что множество разрешенных комбинаций циклического кода является нулевым пространством по отношению к его проверочной матрице H. Из этого свойства следует, что произведение вектора любой комбинации, принадлежащей циклическому коду, на матрицу H^T равно нулю. Если же в принятой комбинации содержатся ошибки и эта ошибочная комбинация попадет в число запрещенных, то результат умножения, называемый *синдромом*, не будет равен нулю.

Таким образом, первым шагом декодирования является получение синдрома $S=(s_0, s_1,\ldots,s_{n-k-1})$, вычисляемого путем умножения принятого вектора на транспонированную проверочную матрицу, т.е.

$$(h_0 h_1 h_2 \ldots h_{n-1}) \cdot H^T = (s_0 s_1 \ldots s_{n-k-1}). \qquad (2.26)$$

Исправить можно только те конфигурации ошибок, которые дают различные друг от друга ненулевые синдромы.

В дальнейшем процедура исправления ошибок будет подробно рассмотрена на конкретных примерах как простых, так и сложных циклических кодов. Рассмотрены также будут различные алгоритмы декодирования циклических кодов.

Циклические коды, исправляющие однократные ошибки

Как было показано, любая однократная ошибка обнаруживается по ненулевому остатку от деления многочлена $h(x)=f(x)+e(x)$, соответствующего принятой комбинации, на порождающий многочлен $G(x)$. Для исправления ошибок необходимы дополнительные сведения о местоположении ошибки. С этой целью потребуем, во-первых, чтобы любой однократной ошибке соответствовал определенный отличный от нуля остаток и, во-вторых, чтобы все эти остатки были различны между собой, т. е. чтобы выполнялось взаимнооднозначное соответствие между местоположением ошибки и ее остатком. Так как длина комбинации остатка равна степени порождающего многочлена $(n–k)$, а длина комбинации двоичного циклического (n,k)-кода равна n, то для исправления однократных ошибок в такой комбинации необходимо иметь не менее чем n ненулевых остатков. Тогда, с учетом нулевого остатка при отсутствии ошибок, должно выполняться неравенство $2^{n-k} \geq n+1$. Таким образом, все ненулевые остатки будут представлять собой вычеты по модулю примитивного многочлена $G(x)$, и, следовательно, ненулевые элементы поля Галуа GF(2^m), где $m=(n-k)$. Очевидно, что для двоичных циклических кодов совокупность всех остатков, в том числе и нулевого, представляет собой алгебру многочленов по двойному модулю [mod 2; $G(x)$].

Рассмотрим процедуру исправления однократной ошибки. Пусть в комбинаций длины $n = 2^m - 1$ на i-й позиции возникла однократная ошибка, которой соответствует одночлен $e(x)= x^i$. В результате деления принятой комбинации на порождающий многочлен $G(x)$ образуется остаток $R(x)$, которому будет соответствовать элемент ε^i поля G$F(2^m)$, где ε – первообразный элемент поля и корень многочлена $G(x)$. Этот остаток будет содержаться в регистре сдвига, который реализует операцию деления на многочлен $G(x)$. При этом сдвиг содержимого регистра на один шаг соответствует умножению на x с последующим приведением результата по модулю $G(x)$. Следовательно, произведя $(n - i)$ сдвигов остатка $R(x)$, мы тем самым как бы умножили принятую комбинацию $h(x)$ на x^{n-i} и разделили результат на $G(x)$. В регистре при этом останется некоторый новый остаток $R_1(x) = x^{n-i} R(x)$ [mod$G(x)$].

Состояние регистра определим из преобразования:

$$x^{n-i} h(x)= x^{n-i} [f(x)+e(x)] = x^{n-i} \cdot f(x) + x^{n-i} e(x) \equiv x^{n-i} x^i = x^n \equiv 1[\text{mod } G(x)]. \quad (2.27)$$

Следовательно, состояние регистра после $(n-i)$ сдвигов будет (1000), что соответствует элементу поля $\varepsilon^0 = \varepsilon^n$. Обнаружив это состояние, можно легко исправить одиночную ошибку.

Пример 2.8. Рассмотрим процесс исправления однократных ошибок декодирующим устройством (рис. 2.14) циклического (15,11) кода, порождающий многочлен которого имеет вид $G(x) = 1 + x + x^4$. Комбинация циклического кода поступает на вход буферного накопителя и одновременно с этим на регистр с сумматорами по mod2, реализующим деление на порождающий многочлен $G(x)$. Емкость буферного накопителя равна длине кодовой комбинации $n = 15$.

Когда элементы комбинации циклического кода заполнят буферный накопитель, в регистре сдвига с обратными связями будет содержаться остаток от деления принятой комбинации на порождающий многочлен $G(x)$.

Как было показано, определение ошибочной i-й позиции для рассматриваемого кода осуществляется путем обнаружения состояния (1000) через $(n-i)$ сдвигов остатка после приема комбинации. Чтобы предотвратить ложную работу декодирующего устройства во время заполнения комбинацией буферного накопителя после дешифратора состояния регистра (1000), поставлен ключ Кл. Этот ключ замыкается в момент поступления на выходной сумматор 3 первого кодового элемента, соответствующего коэффициенту при x^{14} в принятой комбинации, а размыкается в момент сброса регистра сдвига с обратными связями, т. е. в момент установления состояния 0000. Сброс можно осуществить, например, сигналом из дешифратора в момент исправления ошибки подачей единицы в сумматор 2.

Пусть была передана комбинация циклического кода 111100000000100, которой соответствует многочлен $f(x) = 1 + x + x^2 + x^3 + x^{12}$. Пусть далее произошла одиночная ошибка, которой соответствует одночлен $e(x) = x^0 = 1$. Тогда будет принята следующая комбинация: 011100000000100, которой соответствует многочлен $h(x) = f(x) + e(x) = x + x^2 + x^3 + x^{12}$.

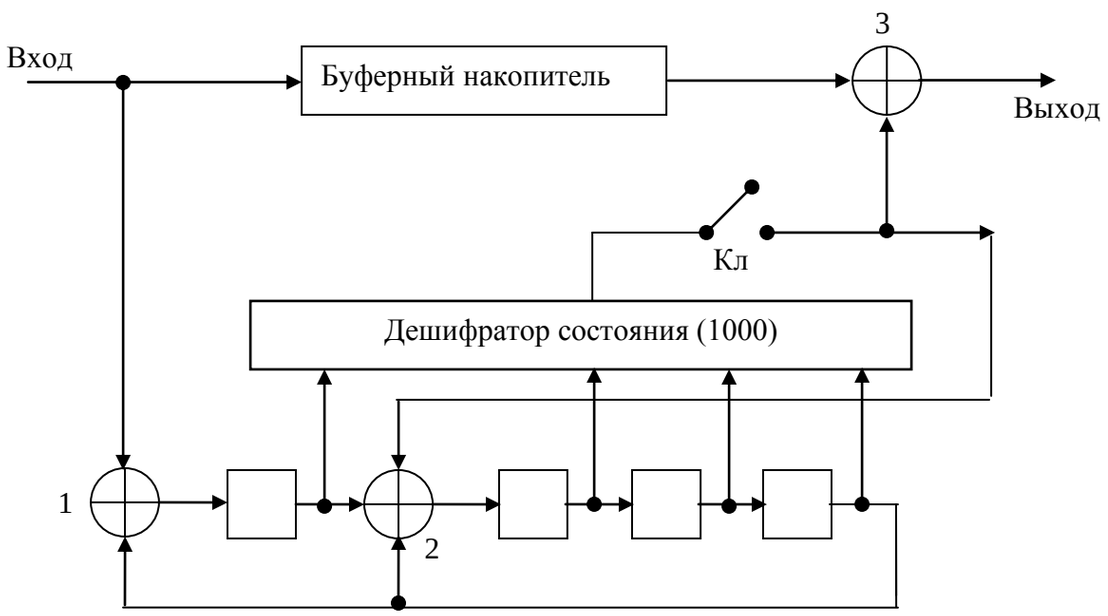


Рис. 2.14. Декодирующее устройство циклического кода (15,11), исправляющего однократные ошибки

После заполнения принятой комбинацией буферного накопителя в регистре сдвига будет содержаться остаток от деления многочлена $h(x)$ на $G(x)$. Как было показано, каждой одиночной ошибке изоморфно соответствует ненулевой элемент поля Галуа $GF(2^4)$. В данном случае ошибке x^0 соответствует остаток, являющийся элементом ε^0 поля. Этому элементу поля соответствует состояние регистра 1000, которое вызывает срабатывание блока

обнаружения. Однако сигнал с выхода последнего не попадает на выходной сумматор 3 и ложной работы не произойдет, так как ключ Кл разомкнут. С первым тактом сдвига регистр установится в состояние 0100, ключ Кл перейдет в замкнутое состояние, а старший разряд принятой комбинации поступает через сумматор 3 на выход. Как было показано, исправление ошибочной i-й позиции должно происходить после $(n - i)$ сдвигов. В нашем случае $i = 0$, поэтому исправление должно произойти через $n = 15$ сдвигов; 15-му сдвигу будет соответствовать обнаруживаемое состояние регистра 1000, при этом последний ошибочный элемент принятой комбинации поступит в сумматор 3, в котором и произойдет исправление ошибки импульсом с дешифратора состояния (1000). Одновременно единица поступает в сумматор 2 и устанавливает регистр в исходное состояние 0000.

Недостатком этих и подобных им схем является то, что после заполнения буферного накопителя в течение следующих n тактов (максимально) информация на вход декодирующего устройства не должна поступать. В некоторых случаях, в зависимости от требований к системе передачи данных, это время простоя можно сократить до k тактов, если в систематическом циклическом коде производить исправление однократных ошибок только среди k информационных элементов принятой комбинации.

Если к сумматору 3 параллельно подключить два идентичных декодера, каждый из которых будет состоять из регистра сдвига с обратными связями, обнаружителя и ключа, и которые будут работать поочередно, то это обеспечит непрерывное поступление элементов на вход декодирующего устройства и исправление однократных ошибок среди всех элементов комбинации.

Для укороченных циклических кодов, у которых длина комбинации $n_1 < 2^m - 1$, дешифратор декодера должен быть настроен на состояние регистра, соответствующее элементу $x^{n_1}[\mathrm{mod}\, 2, G(x)]$.

В заключение этого раздела отметим, что в декодере должны быть применены меры по согласованию тактовой частоты и моментов принятия решений по исправлению однократной ошибки.

3. ЦИКЛИЧЕСКИЕ КОДЫ БЧХ

3.1. Построение циклических кодов БЧХ

Коды Боуза–Чоудхури–Хоквингема (БЧХ) представляют собой большой класс циклических кодов, исправляющих независимые ошибки кратности t и менее. Для кодов БЧХ характерны все основные свойства циклических кодов. Чаще всего коды БЧХ описывают с помощью корней порождающего многочлена $G(x)$. В качестве корней $G(x)$ выбирают $2t$ последовательных элементов ε^{j}, ε^{j+1},…, ε^{j+2t-1} поля Галуа GF(p^m), где $1 \le j \le p^m - 1$. Если при этом элемент ε

является примитивным (первообразным) в поле GF(p^m), то такой код БЧХ называют примитивным с длиной кодовой комбинации $n = p^m - 1$ над полем GF(p). Для двоичных примитивных кодов БЧХ $n = 2^m - 1$ над полем GF(2). В случае, если ряд корней многочлена $G(x)$ для кода БЧХ начинается с j= l, т. е. $\varepsilon, \varepsilon^2, \ldots, \varepsilon^{2t}$, то такой код называют кодом БЧХ в узком смысле.

Код БЧХ, у которого корень ε не является примитивным элементом поля GF(p^m), т. е. имеет порядок $d < p^m - 1$, называют непримитивным с длиной комбинации $n = d$.

Пусть ε^i – элемент расширения простого числового поля. Тогда по определению, данному в [7], некоторый нормированный многочлен $m(x)$ наименьшей степени, для которого $m(\varepsilon^i) = 0$, называют минимальной функцией для ε^i.

Отметим, что минимальные функции $m(x)$ являются неприводимыми многочленами. Если предположить, что каждому из корней ε^i порождающего многочлена $G(x)$ соответствует своя минимальная функция $m_i(x)$, то порождающий многочлен $G(x)$ должен быть наименьшим общим кратным всех минимальных функций, т. е.

$$G(x) = \text{НОК}[m_1(x), m_2(x), \ldots, m_{2t}(x)] \qquad (3.1)$$

Таким образом, вектор, представленный многочленом $f(x)$, будет кодовым тогда и только тогда, когда он делится без остатка как на каждую из минимальных функций $m_1(x)$, $m_2(x)$, ..., $m_{2t}(x)$ так и на их наименьшее общее кратное.

Тогда для любого из корней $\varepsilon, \varepsilon^2, \ldots, \varepsilon^{2t}$ справедливо уравнение $f(\varepsilon) = c_0 + c_1 \varepsilon^i + c_2(\varepsilon^i)^2 + \ldots + c_{n-1}(\varepsilon^i)^{n-1} = 0$, которое можно записать в виде произведения двух матриц $[c_0\ c_1\ c_2\ \ldots\ c_{n-1}] \cdot [1\varepsilon^i\ (\varepsilon^i)^2 \ldots (\varepsilon^i)^{n-1}]^T = 0$. Но так как корнями $f(x)$ должны быть все элементы $\varepsilon, \varepsilon^2, \ldots, \varepsilon^{2t}$, то можно сделать вывод, что вектор $[c_0\ c_1 \ldots c_{n-1}]$ будет кодовым тогда и только тогда, когда он принадлежит нулевому пространству матрицы

$$H = \begin{bmatrix} 1 & \varepsilon & \varepsilon^2 & \ldots & \varepsilon^{n-1} \\ 1 & \varepsilon^2 & (\varepsilon^2)^2 & \ldots & (\varepsilon^2)^{n-1} \\ 1 & \varepsilon^3 & (\varepsilon^3)^2 & \ldots & (\varepsilon^3)^{n-1} \\ . & . & . & \ldots & . \\ 1 & \varepsilon^{2t} & (\varepsilon^{2t})^2 & \ldots & (\varepsilon^{2t})^{n-1} \end{bmatrix}. \qquad (3.2)$$

Может оказаться, что элементы ε^i и ε^j из совокупности $\varepsilon, \varepsilon^2, \ldots, \varepsilon^{2t}$ соответствуют одной и той же минимальной функции, т. е. $m_i(x) = m_j(x)$. Вследствие того, что порождающий многочлен $G(x)$ равен наименьшему общему кратному всех минимальных функций $m_i(x)$, в качестве сомножителя в разложении многочлена $G(x)$ следует взять только одну из нескольких равных между собой минимальных функций.

Из свойств полей следует, что если ε^i корень какой-либо минимальной неприводимой по mod2 функции $m_i(x)$ степени k, то остальными корнями будут $(\varepsilon^i)^2, (\varepsilon^i)^{2^2}, \ldots, (\varepsilon^i)^{2^{k-1}}$. Следовательно, в разложение многочлена $G(x)$ каждая из различных минимальных функций $m_i(x)$ должна входить только один раз, а для построения матрицы (3.2) нужно использовать только по одному корню каждой из минимальных функций, входящих в разложение многочлена $G(x)$. Таким

образом, если в качестве совокупности корней многочлена $G(x)$ выбраны элементы поля Галуа GF(2^m) $\varepsilon, \varepsilon^2, \varepsilon^3, ..., \varepsilon^{2t}$, то при построении матрицы H должны быть использованы только нечетные степени $\varepsilon, \varepsilon^3, ..., \varepsilon^{2t-1}$. Тогда многочлен $G(x)$ будет иметь вид

$$G(x) = m_1(x)\, m_3(x) \ldots m_{2t-1}(x), \tag{3.3}$$

а проверочная матрица H (3.2) преобразуется к виду:

$$H = \begin{bmatrix} 1 & \varepsilon & \varepsilon^2 & \ldots & \varepsilon^{n-1} \\ 1 & \varepsilon^3 & (\varepsilon^3)^2 & \ldots & (\varepsilon^3)^{n-1} \\ 1 & \varepsilon^5 & (\varepsilon^5)^2 & \ldots & (\varepsilon^5)^{n-1} \\ . & . & . & \ldots & . \\ 1 & \varepsilon^{2t-1} & (\varepsilon^{2t-1})^2 & \ldots & (\varepsilon^{2t-1})^{n-1} \end{bmatrix}.$$

Пример 3.1. Пусть имеем двоичный циклический код БЧХ, к которому вектор $\{f(x)\}$ будет принадлежать только тогда, когда элементы $\varepsilon, \varepsilon^2, \varepsilon^3, \varepsilon^4, \varepsilon^5, \varepsilon^6$ будут корнями многочлена $f(x)$. Кроме того, предполагается, что ε – примитивный элемент поля Галуа GF(2^m), где m=4. Тогда ε^{15}=1 и $m_i(x)$ обозначает минимальную функцию для ε^i.

В соответствии со свойством 1.6 элементы $\varepsilon, \varepsilon^2, \varepsilon^4$ и ε^8 – корни одной и той же минимальной функции четвертой степени, следовательно, $m_1(x) = m_2(x) = m_4(x) = m_8(x)$. Аналогично, $\varepsilon^3, \varepsilon^6, \varepsilon^{12}$ и $\varepsilon^{24} = \varepsilon^9$ – корни минимальной функции четвертой степени и $m_3(x) = m_6(x) = m_9(x) = m_{12}(x)$, а элементы ε^5 и ε^{10} являются корнями минимальной функции второй степени и, следовательно, $m_5(x) = m_{10}(x)$. Отсюда $G(x)$ является многочленом

$$G(x) = m_1(x) \cdot m_3(x) \cdot m_5(x), \tag{3.4}$$

степень которого равна 10.

Таким образом, к искомому коду БЧХ будет принадлежать любой вектор $\{f(x)\}$, если $f(x)$ делится на $G(x)$ без остатка. В то же время циклический код будет нулевым пространством матрицы

$$H = \begin{bmatrix} 1 & \varepsilon & \varepsilon^2 & \ldots & \varepsilon^{14} \\ 1 & \varepsilon^3 & \varepsilon^6 & \ldots & \varepsilon^{12} \\ 1 & \varepsilon^5 & \varepsilon^{10} & \ldots & \varepsilon^{10} \end{bmatrix}. \tag{3.5}$$

Так как ε^i является элементом ноля GF(2^m) и представляет собой вектор из m двоичных элементов 0 и 1, то матрица H^T имеет размерность $n \times mt$. Из свойств циклических кодов следует, что многочлен $G(x)$ является делителем многочлена $F(x) = x^n - 1$, где $n = 2^m - 1$. В то же время многочлен $G(x)$ для кодов БЧХ равен произведению минимальных функций. Следовательно, любая из минимальных функций, входящих в разложение многочлена $G(x)$, должна быть делителем функции $F(x) = x^n - 1 = x^{2^m-1} - 1$. При этом, как следует из высшей алгебры [4, 7], степень каждой минимальной функции не может быть больше m. А так как таких функций t, то степень многочлена $G(x)$ не превосходит mt. Отсюда каждая комбинация примитивного циклического кода БЧХ при длине, равной n=2^m–1, имеет число информационных разрядов, равное $k \geq 2^m - 1 - mt$.

Рассмотрим конкретный пример построения циклического кода БЧХ.

Пример 3.2. Построить двоичный примитивный циклический код БЧХ для $m = 4$ и $t = 3$.

В этом случае длина кодовой комбинации равна $n = 2^m - 1 = 15$, а вектор $\{f(x)\}$ будет принадлежать этому циклическому коду, если элементы поля $GF(2^4)$ ε, ε^2, ..., ε^6 будут корнями многочлена $f(x)$. Кроме того, отметим, что ε – примитивный элемент поля, а минимальной функцией для него пусть будет примитивный неприводимый многочлен $m_1(x) = 1+x + x^4$.

Как видим, этот пример является непосредственным продолжением примера 3.1, из которого следует, что порождающий многочлен $G(x) = m_1(x)\ m_3(x)\ m_5(x)$, имеет степень, равную 10. Кроме того, было показано, что матрица H имеет вид (3.5).

Так как примитивный элемент ε – корень минимального многочлена $m_1(x) = 1+x + x^4$, то, подставив вместо каждого элемента матрицы H его двоичную комбинацию, получим транспонированную матрицу H^T:

$$H^T = \begin{bmatrix} 1000 & 1000 & 1000 \\ 0100 & 0001 & 0110 \\ 0010 & 0011 & 1110 \\ 0001 & 0101 & 1000 \\ 1100 & 1111 & 0110 \\ 0110 & 1000 & 1110 \\ 0011 & 0001 & 1000 \\ 1101 & 0011 & 0110 \\ 1010 & 0101 & 1110 \\ 0101 & 1111 & 1000 \\ 1110 & 1000 & 0110 \\ 0111 & 0001 & 1110 \\ 1111 & 0011 & 1000 \\ 1011 & 0101 & 0110 \\ 1001 & 1111 & 1110 \end{bmatrix} \quad \text{.....(3.6)}$$

В [7] показано, что $m_3(x) = 1 + x + x^2 + x^3 + x^4$, $m_5(x) =1 + x + x^2$, тогда степень многочлена $G(x)$, равная 10, не превышает mt.

В рассмотренном примере $G(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$.

Тогда порождающая матрица G искомого систематического кода БЧХ имеет вид

$$G = \begin{bmatrix} 111011001010000 \\ 011101100101000 \\ 110101111000100 \\ 011010111100010 \\ 110110010100001 \end{bmatrix}$$

Образованный таким образом циклический $(n,k) = (15, 5)$ код БЧХ позволяет исправить любую совокупность ошибок кратности $t = 3$ или менее.

3.2. Принципы исправления ошибок кодами БЧХ на основе алгоритма Питерсона–Горенштейна–Цирлера

Предположим, что был передан кодовый вектор $\{f(x)\}$, представление которого в виде многочлена будет иметь вид: $f(x) = c_0 + c_1x + ... + c_{n-1}x^{n-1}$. Пусть далее, вследствие ошибок, вместо вектора $\{f(x)\}$ принят вектор $\{f(x) + e(x)\} = \{f(x)\} + \{e(x)\}$, где $\{e(x)\}$ – вектор ошибок.

Обозначим принятый с ошибками вектор $\{f(x) + e(x)\}$ через вектор $\{h(x)\} = (h_0h_1h_2\ldots h_{n-1})$. Принятую комбинацию можно представить в виде многочлена степени $(n - 1)$, т. е. $h(x) = h_0 + h_1x + h_2x^2 + ... + h_{n-1}x^{n-1}$. В результате умножения вектора $\{h(x)\}$ на матрицу (3.2) получим вектор из t компонент $[h(\varepsilon), h(\varepsilon^3), ..., h(\varepsilon^{2t-1})]$, где $h(\varepsilon) = h_0 + h_1\varepsilon + h_2\varepsilon^2 + ... + h_{n-1}\varepsilon^{n-1}$;

$$h(\varepsilon^3) = h_0 + h_1\varepsilon^3 + h_2(\varepsilon^3)^2 + ... + h_{n-1}(\varepsilon^3)^{n-1} \text{ и т. д.}$$

В то же время, $h(x) = f(x)+e(x)$. Тогда $h(\varepsilon^i) = f(\varepsilon^i) + e(\varepsilon^i)$, где $i = 1, 3,\ldots, (2t–1)$, но так как $f(x)$ делится на $G(x)$ без остатка, то $h(\varepsilon^i) \equiv e(\varepsilon^i) \bmod G(x)$. Таким образом, получаем тождественное равенство

$$[h(\varepsilon), h(\varepsilon^3), ..., h(\varepsilon^{2t-1})] \equiv [e(\varepsilon), e(\varepsilon^3), .., e(\varepsilon^{2t-1})\}, \quad (3.7)$$

где $e(\varepsilon^i) = e_0 + e_1\varepsilon^i + e_2(\varepsilon^i)^2 + ... + e_{n-1}(\varepsilon^i)^{n-1}$.

При умножении вектора $\{h(x)\}$ на первый столбец матрицы H^T, образованной из (3.5), получаем элемент

$$h(\varepsilon) = e(\varepsilon) = e_0 + e_1\varepsilon + e_2\varepsilon^2 + ... + e_{n-1}\varepsilon^{n-1}. \quad (3.8)$$

Отсюда следует, что имеется взаимно однозначное соответствие между элементами вектора ошибок и элементами поля $GF(2^m)$. Каждому ошибочному элементу e_i соответствует элемент i-й строки ($i = 0, 1, 2, .., n-1$) первого столбца транспонированной матрицы H^T, т. е. элемент поля ε^i.

Предположим, что ошибки произошли на позициях $i_1, i_2,\ldots, i_t$, для которых $e_i = 1$, а для всех остальных $e_i = 0$, тогда выражение (3.8) может быть переписано следующим образом:

$$h(\varepsilon) = e(\varepsilon) = \sum_{k=1}^{t} \varepsilon^{i_k}. \quad (3.9)$$

Обозначим каждый из ошибочных элементов ε^{i_k} через X_k ($k = 1, 2, \ldots, t$) и назовем их *локаторами ошибок*, тогда выражение (3.9) может быть записано как:

$$h(\varepsilon) = e(\varepsilon) = \sum_{k=1}^{t} \varepsilon^{i_k} = \sum_{k=1}^{t} X_k$$

Умножив вектор $\{h(x)\}$ на какой-либо другой j-й столбец матрицы H^T, получим

$$h(\varepsilon^j) = e(\varepsilon^j) = (\varepsilon^j)^{i_1} + (\varepsilon^j)^{i_2} + \ldots + (\varepsilon^j)^{i_t} = \sum_{k=1}^{t} (\varepsilon^{i_k})^j = \sum_{k=1}^{t} X_k^j, \quad (3.10)$$

где $j = 1, 3, \ldots, 2t–1$.

Учитывая, что в качестве корней порождающего многочлена $G(x)$ выбраны элементы поля Галуа $GF(2^m)$ $\varepsilon, \varepsilon^2, \varepsilon^3, ..., \varepsilon^{2t}$, а также то, что для этих элементов $f(\varepsilon^j)=0$ запишем совокупность синдромов S_j принятой комбинации так:

$$S_1 = h(\varepsilon) = e(\varepsilon) = \sum_{k=1}^{t} \varepsilon^{i_k} = \sum_{k=1}^{t} X_k ,$$

$$S_2 = h(\varepsilon^2) = e(\varepsilon^2) = \sum_{k=1}^{t} (\varepsilon^2)^{i_k} = \sum_{k=1}^{t} X_k^2 , \qquad (3.11)$$

……

$$S_{2t} = h(\varepsilon^{2t}) = e(\varepsilon^{2t}) = \sum_{k=1}^{t} (\varepsilon^{2t})^{i_k} = \sum_{k=1}^{t} X_k^{2t}$$

Выражения (3.11) представляют $2t$ симметрических функций S_j от степеней элементов $X_1, X_2, ..., X_t$.

Так как суммы степеней j элементов $X_1, X_2,..., X_t$ представляют собой симметрические функции S_j, то эти элементы могут рассматриваться [8] как корни некоторого многочлена степени t

$$\Psi(X) = X^t + p_1 X^{t-1} + p_2 X^{t-2} + \ldots + p_t , \qquad (3.12)$$

разложение которого на множители дает уравнение

$$(X - X_1)(X - X_2) \ldots (X - X_t) = 0. \qquad (3.13)$$

Многочлен (3.12) называют *многочленом локаторов ошибок*.

Коэффициенты $p_1, p_2, ..., p_t$ многочлена локаторов ошибок (3.12) являются простейшими симметрическими функциями, которые связаны с симметрическими функциями S_j тождествами Ньютона [1,7]:

$$\begin{aligned}
&S_1 = p_1; \\
&S_2 = p_1 S_1; \\
&S_3 = p_1 S_2 + p_2 S_1 + p_3; \\
&S_4 = p_1 S_3 + p_2 S_2 + p_3 S_1; \\
&\ldots\ldots\ldots\ldots\ldots\ldots \\
&S_t = p_1 S_{t-1} + p_2 S_{t-2} + \ldots + p_{t-1} S_1 + \delta p_t ,
\end{aligned} \qquad (3.14)$$

где $\delta = 0$ при j – четном и $\delta = 1$ при j – нечетном.

Если тождества (3.14) решить относительно простейших симметрических функций p_i и найденные значения p_i подставить в (3.12), то корни этого многочлена $X_1, X_2, ..., X_t$ можно определить последовательной подстановкой в него каждого из $n=2^m-1$ элементов поля. Такую процедуру называют *процедурой Ченя*. Если подставленный вместо X_i элемент ε^i не является корнем многочлена (3.12), то соответствующий символ вектора $\{h(x)\}$ принят правильно. Если же элемент ε^i является корнем, то соответствующий i-й символ вектора $\{h(x)\}$ принят ошибочным и должен быть исправлен.

Однако в силу свойства 1.2 поля для двоичных кодов справедливо равенство

$$(S_j)^2 = (\sum_{k=1}^{t} X_k^j)^2 = \sum_{k=1}^{t} X_k^{2j} = S_{2j} ,$$

поэтому следует рассматривать не все тождества (3.14), а лишь те, которые определяют S_j для нечётных j, т. е.

$$\begin{aligned}
&S_1 = p_1; \\
&S_3 = p_1 S_1 + p_2 S_2 + p_3; \\
&S_5 = p_1 S_4 + p_2 S_3 + p_3 S_2 + p_4 S_1 + p_5;
\end{aligned} \qquad (3.15)$$

……

$S_{t-1} = p_1 S_{t-2} + p_2 S_{t-3} + \ldots + p_{t-2} S_1 + p_{t-1}$ при t – чётном;

или $S_t = p_1 S_{t-1} + p_2 S_{t-2} + \ldots + p_{t-1} S_1 + p_t$ при t – нечётном.

Отсюда видно, что имеется $t/2$ (при t – четном) или $(t+1)/2$ (при t – нечетном) уравнений, которых недостаточно для определения t неизвестных $p_1, p_2, ..., p_t$. Однако в результате умножения $\{h(x)\} \cdot H^T$ можно определить все симметрические функции $S_1, S_2, ..., S_{2t}$. Знание симметрических функций S_j со значениями j от 1 до $2t$ включительно (3.11) позволяет составить систему уравнений, которые можно уже решить относительно p_i.

Выше было показано, как составить тождества Ньютона для j при S_i, не превосходящих t, т. е. степени многочлена локаторов ошибок (3.12). Можно показать, что аналогично можно составить тождества Ньютона для значений $j > t$.

Действительно, если в многочлен (3.12) подставить вместо X какой-либо из корней $X_1, X_2, ..., X_t$, то получим уравнение

$$\Psi(X_i) = 0 \ . \tag{3.16}$$

Умножение обеих частей уравнения (3.16) на $X_i^{\,c-t}$ дает

$$X_i^{\,c-t} \ \Psi(X_i) = 0 \ , \tag{3.17}$$

где $2t \geq c > t$.

Если теперь просуммируем (3.17) по всем корням, т. е.

$$\sum_{i=1}^{t} X_i^{\,c-t} \Psi(X_i) = 0, \tag{3.18}$$

то получим

$$S_c + p_1 S_{c-1} + \ldots + p_{t-1} S_{c-t+1} + p_t S_{c-t} = 0 \ . \tag{3.19}$$

По формуле (3.19) составим тождества Ньютона для S_c, где $2t \geq c > t$, и получим следующую систему уравнений в матричной форме :

$$\begin{bmatrix} S_{t+1} \\ S_{t+2} \\ \ldots \\ S_{2t} \end{bmatrix} = \begin{bmatrix} S_t & S_{t-1} & S_{t-2} & \ldots & S_1 \\ S_{t+1} & S_t & S_{t-1} & \ldots & S_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ S_{2t-1} & S_{2t-2} & S_{2t-3} & \ldots & S_t \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \ldots \\ p_t \end{bmatrix}. \tag{3.20}$$

Уравнение (3.20) в матричной форме в литературе называют *ключевым уравнением*, которое необходимо решить относительно неизвестных коэффициентов p_i многочлена локаторов ошибок.

Найденные значения p_i подставляем в многочлен локаторов ошибок (3.12) и, применяя процедуру Ченя, последовательной подстановкой вместо X_i элементов поля GF(2^m), находим корни этого многочлена. Этим самым мы устанавливаем ошибочные позиции принятого вектора $\{h(x)\}$

Если произошло в действительности $t - 1$ ошибок, то один из корней $X_1, X_2, ..., X_t$ должен быть равен нулю. Следовательно, при решении уравнений (3.20) мы должны получить $p_t = 0$. Если произошло $t - 2$ ошибки, то два корня равны нулю и, следовательно, $p_t = p_{t-1} = 0$ и так далее.

Таким образом, при декодировании циклических кодов БЧХ должны быть последовательно выполнены следующие общие задачи:

- вычислены синдромы в соответствии с (3.11);
- в результате решения ключевого уравнения (3.20) найдены коэффициенты $p_1, p_2, \ldots, p_t$ многочлена локаторов ошибок (3.12);
- определены корни многочлена локаторов ошибок $\Psi(X)$, т.е. ошибочные позиции;
- исправлены ошибки на вычисленных позициях.

Специалистами в области теории циклических кодов предложено несколько алгоритмов декодирования кодов БЧХ как во временной, так и в частотной областях. Наиболее трудоемким является решение ключевого уравнения. Известно несколько методов его решения.

Рассмотрим на конкретном примере алгебраический алгоритм декодирования кодов БЧХ во временной области, предложенный Питерсоном [7]. В основе этого алгоритма лежит *прямое решение* системы уравнений (3.20). В литературе его часто называют алгоритмом (декодером) Питерсона–Горенштейна–Цирлера (PGZ).

Пример 3.3. Рассмотрим процесс исправления тройных ошибок (t = 3) циклическим кодом БЧХ, построение которого рассматривалось в примере 3.2. Поле GF(2^4) образовано примитивным многочленом $m_1(x) = 1 + x + x^4$ с первообразным корнем ε. Тогда многочлен локаторов ошибок (3.12) для такого кода будет иметь вид $\Psi(X) = X^3 + p_1X^2 + p_2X + p_3$.

Ключевое уравнение в матричном представлении (3.20) будет иметь вид:

$$\begin{bmatrix} S_4 \\ S_5 \\ S_6 \end{bmatrix} = \begin{bmatrix} S_3 & S_2 & S_1 \\ S_4 & S_3 & S_2 \\ S_5 & S_4 & S_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}. \qquad (3.21)$$

Рассмотрим случай, когда ошибки появляются на 2, 5 и 7-м местах, т. е. $\{e(x)\}$ = (010010100000000), тогда, в соответствии с (3.11), вычислим $2t$=6 синдромов, которые в элементах поля GF(2^4) будут следующими: $S_1 = \varepsilon^{13}$, $S_2 = \varepsilon^{11}$, $S_3 = \varepsilon^{12}$, $S_4 = \varepsilon^7$, $S_5 = 1$ и $S_6 = \varepsilon^9$.

Ключевое уравнение в соответствии с (3.20) приобретает вид:

$$\begin{bmatrix} \alpha^7 \\ 1 \\ \alpha^9 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^{11} & \alpha^{13} \\ \alpha^7 & \alpha^{12} & \alpha^{11} \\ 1 & \alpha^7 & \alpha^{12} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}.$$

Определитель $\Delta = \begin{vmatrix} \alpha^{12} & \alpha^{11} & \alpha^{13} \\ \alpha^7 & \alpha^{12} & \alpha^{11} \\ 1 & \alpha^7 & \alpha^{12} \end{vmatrix} = \alpha^{12}$ не равен 0, следовательно, система уравнений (3.21) имеет решение относительно коэффициентов многочлена локаторов ошибок.

В результате решения системы уравнений искомые коэффициенты будут иметь значения:

$$p_1 = \varepsilon^{13}, \quad p_2 = \varepsilon^9, \quad p_3 = \varepsilon^{11}.$$

Теперь путем подстановки элементов поля в уравнение

$$X^3 + \varepsilon^{13}X^2 + \varepsilon^9X + \varepsilon^{11} = 0$$

находим, что корни этого уравнения будут ε, ε^4, ε^6. Следовательно, в принятом векторе $\{h(x)\}$ ошибки находятся на 2, 5 и 7-м местах. Если произошла только одна ошибка, то $p_2 = p_3 = 0$. Следовательно, номер ошибочной позиции можно определить, решая уравнение $X + p_1 = 0$.

Метод прямого решения ключевого уравнения требует обращения матрицы, сложность которого растет как куб корректирующей способности кода [10]. Поэтому прямой алгоритм рекомендуется использовать только для малых значений *t*.

Для кодов БЧХ имеются другие, более эффективные алгебраические алгоритмы декодирования. К ним относятся алгоритмы Берлекэмпа-Мэсси (BMA) и алгоритм Евклида (ЕА). Алгоритм Берлекэмпа-Мэсси чаще всего используется для программной реализации декодеров кодов БЧХ и Рида-Соломона, а алгоритм Евклида – для аппаратной реализации таких декодеров. Эти алгоритмы будут рассмотрены ниже.

4. НЕДВОИЧНЫЕ ЦИКЛИЧЕСКИЕ КОДЫ БЧХ (РИДА-СОЛОМОНА)

Важный подкласс кодов БЧХ составляют циклические коды Рида-Соломона. Приведем основные особенности кодов Рида-Соломона (РС).

Прежде всего, коды РС − это недвоичные коды, комбинации которых, представляются многочленами

$$f(x)=c_0 + c_1 x^n + c_2 x^2 +\ldots+ c_{n-1}x^{n-1}, \qquad (4.1)$$

где коэффициенты c_i принадлежат, полю G$F(2^m)$, которое порождается многочленом $g(x)$ степени m.

Коды РС являются циклическими, чаще всего систематическими, с порождающим многочленом

$$G(x) = (x +1)(x + \varepsilon)(x + \varepsilon^2)\ldots(x + \varepsilon^{r-1}),$$

где ε − примитивный элемент поля G$F(2^m)$, представляющий корень многочлена $g(x)$, образующего поле.

Корректирующие свойства кода РС определяются минимальным кодовым расстоянием $d_{\min}$, которое для систематических кодов РС является максимально достижимым и равным $d_{\min}= r + 1$.

Более широкое применение на практике находят систематические коды РС, так как они обеспечивают наибольшее минимальное расстояние $d_{\min}$. В то же время эти коды допускают более простую техническую реализацию, поэтому ограничимся рассмотрением только систематических циклических кодов РС.

4.1. Принципы кодирования и декодирования кодов Рида–Соломона

Процесс кодирования для систематического кода РС осуществляется по тем же правилам, что и для любого циклического систематического кода − путем деления на порождающий многочлен $G(x)$ и получения остатка, представляющего собой проверочные элементы.

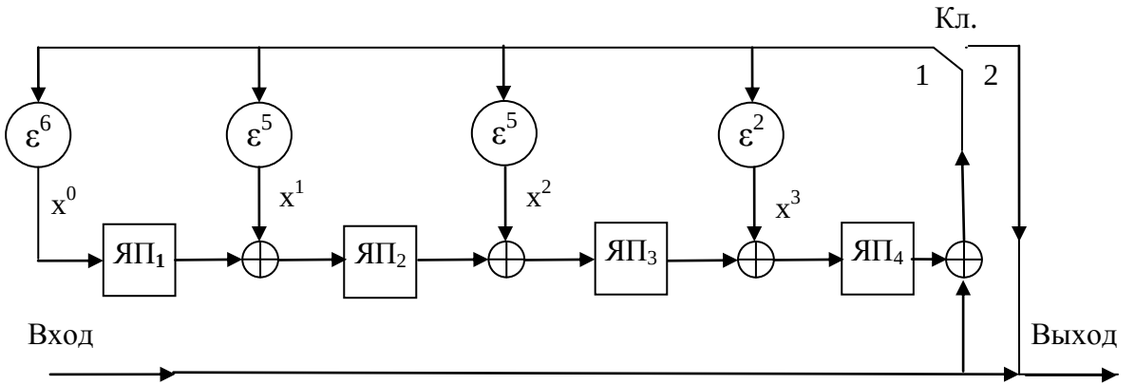


Рис 4.1. Кодирующее устройство кода Рида–Соломона с порождающим многочленом $G(x)=(x+1)(x+\varepsilon)(x+\varepsilon^2)(x+\varepsilon^3)$ в поле $GF(2^3)$

Пример построения кодирующего устройства кода РС представлен на рис. 4.1. Кодер содержит $r = 4$ ячейки памяти, способных хранить элементы поля $GF(2^m)$, представленные двоичными векторами длины $m = 3$. Таким образом, каждая ячейка памяти может быть построена из m триггеров. Кроме того, кодер содержит по $r = 4$ сумматоров и умножителей на фиксированные элементы поля $GF(2^3)$. При этом каждый сумматор состоит из двухвходовых сумматоров по модулю два, на входы которых поступают параллельно m-элементные двоичные комбинации. Умножители на фиксированные элементы ε^i поля $GF(2^m)$ реализуются как умножители на матрицу F^i с помощью комбинаторных схем или ПЗУ с m входами и m выходами. Практическая реализация умножения приведена в разд. 2.3.

Алгоритм работы кодера ничем не отличается от работы обычного кодера для циклических кодов.

Декодирование кодов РС включает более сложные процедуры. Рассмотрим наиболее широко применяемый принцип алгебраического декодирования.

Пусть некоторая разрешенная кодовая комбинация длины n представлена в общем виде многочленом $f(x)$ (4.1).

Коды РС обладают всеми свойствами циклических кодов, одним из которых является делимость многочлена $f(x)$ на порождающий многочлен $G(x)$ без остатка. Отсюда следует, что корни $1, \varepsilon, \varepsilon^2, .., \varepsilon^{r-1}$ многочлена $G(x)$ являются также корнями многочленов $f(x)$, т. е. имеют место равенства:

$$f(1)=c_0+c_1+c_2+\ldots+c_{n-1}=0,$$
$$f(\varepsilon)=c_0+c_1\varepsilon+c_2\varepsilon^2+\ldots+c_{n-1}\varepsilon^{n-1}=0,$$
$$f(\varepsilon^2)=c_0+c_1\varepsilon^2+c_2(\varepsilon^2)^2+\ldots+c_{n-1}(\varepsilon^2)^{n-1}=0,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$f(\varepsilon^{r-1})=c_0+c_1\varepsilon^{r-1}+c_2(\varepsilon^{r-1})^2+\ldots+c_{n-1}(\varepsilon^{r-1})^{n-1}=0.$$

Приведенную систему уравнений можно записать в виде

$$[f]^* H^T = 0, \qquad (4.2)$$

где $[f] = [c_0\ c_1\ c_2\ \ldots\ c_{n-1}]$, $c_i \in GF(2^m)$, а матрица H является проверочной матрицей кода РС и имеет вид

$$H=\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \varepsilon & \varepsilon^2 & \dots & \varepsilon^{n-1} \\ 1 & \varepsilon^2 & \left(\varepsilon^2\right)^2 & \dots & \left(\varepsilon^2\right)^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \varepsilon^{r-1} & \left(\varepsilon^{r-1}\right)^2 & \dots & \left(\varepsilon^{r-1}\right)^{n-1} \end{bmatrix}$$

Комбинация, поступающая на вход декодера, представленная в виде многочлена $h(x)=h_0+h_1x + c_2x^2 + ... + h_{n-1}x^{n-1}$, равна сумме $h(x) = f(x) + e(x)$, где $f(x)$ − разрешенная кодовая комбинация и $e(x)$ − многочлен ошибок, равный $e(x) = e_0+e_1x + e_2x^2 + ... + e_{n-1}x^{n-1}$, где $e_i \in GF(2^m)$. Очевидно, что $h_i = c_i + e_i$ (mod 2).

В декодере осуществляется умножение входного вектора $\{h\} = \{ h_0\ h_1\ h_2 ... h_{n-1}\}$ на проверочную матрицу H, в результате чего, с учетом (4.2), получим

$$\{h\}H^T = \{f\}H^T + \{e\}H^T = \{e\}H^T = \{ S_0\ S_1\ S_2 ... S_{r-1}\}.$$

Таким образом, первым этапом декодирования является получение синдромов $S_0, S_1, S_2,..., S_{r-1}$, значения которых определяются выражениями:

$S_0=e_0+e_1 + e_2 + ... + e_{n-1} = 0,$

$S_1=e_0+e_1\varepsilon + e_2\varepsilon^2 + ... + e_{n-1}\varepsilon^{n-1} = 0,$

$S_2=e_0+e_1\,\varepsilon^2 + e_2(\varepsilon^2)^2 + ... + e_{n-1}(\varepsilon^2)^{n-1} = 0$

…….

$S_{r-1}=e_0+e_1\,\varepsilon^{r-1} + e_2(\varepsilon^{r-1})^2 + ... + e_{n-1}(\varepsilon^{r-1})^{n-1} = 0.$

При этом отличие от нуля хотя бы одного синдрома S_i свидетельствует о наличии в принятой комбинации ошибок. Итак, для обнаружения ошибок кодом РС достаточно проверить на нуль вычисленные синдромы.

Рассмотрим теперь процедуру исправления ошибок. Пусть требуется построить код РС, исправляющий ошибки кратности t и менее. Очевидно, что в этом случае минимальное кодовое расстояние должно быть равно $d_{\min} = 2t + 1$. Следовательно, для кодов РС с исправлением t и менее ошибок степень порождающего многочлена $G(x)$ будет равна $r = 2t$.

Многочлен t-кратной ошибки можно записать:

$$e(x) = e_{i_1}x^{i_1} + ... + e_{i_t}x^{i_t},$$

где коэффициенты ошибок $e_{i_1}, e_{i_2}, ..., e_{i_t}$ отличны от нуля и принадлежат полю $GF(2^m)$.

На первом этапе декодирования, в результате умножения вектора $\{h\}$ или, что то же самое, вектора ошибок $\{e\}$ на проверочную матрицу H^T будут получены синдромы

$$S_i = e\left(\varepsilon^j\right) = \sum_{i=1}^{t} Y_i \cdot X_i^j \quad , \qquad j = 0, 1, \dots, (r-1), \tag{4.3}$$

где величины

$$X_1 = \varepsilon^{i_1}\ ,\ X_2 = \varepsilon^{i_2}\ ,\ \dots,\ X_t = \varepsilon^{i_t},$$

указывающие место расположения ошибок, называют локаторами ошибок; а величины

$$Y_1 = e^{i_1}\ ,\ Y_2 = e^{i_2}\ ,\ \dots,\ Y_t = e^{i_t}$$

представляют значения самих ошибок.

Таким образом, для исправления t ошибок необходимо на втором этапе декодирования найти значения t пар (X_i , Y_i) путем решения системы из $2t$ уравнений (4.3), в которых известными величинами являются полученные ранее синдромы.

Как и в двоичных кодах БЧХ, будем считать, что искомые ошибочные позиции X_1, X_2, ..., X_t являются корнями некоторого многочлена локаторов ошибок

$$\Psi(X)= X^t + p_1 X^{t-1}+ p_2 X^{t-2}+\ldots+ p_{t-1} X+ p_t \qquad (4.4)$$

Очевидно, что при $X = X_i$ имеем

$$\Psi (X_i)= X_i{}^t + p_1 X_i{}^{t-1}+ p_2 X_i{}^{t-2}+\ldots+ p_{t-1} X_i + p_t =0. \qquad (4.5)$$

Умножив уравнение (4.5) на $Y_i X_i{}^j$, получим

$$Y_i X_i^{t+j} + p_1 Y_i X_i{}^{t+j-1} + p_2 Y_i X_i{}^{t+j-2} + \ldots + p_{t-1} Y_i X_i{}^{j+1} + p_t Y_i X_i{}^j = 0.$$

Составив такие уравнения для всех локаторов ошибок X_1, X_2, ..., X_t и просуммировав их, получим t уравнений вида

$$S_{j+t} + p_1 S_{j+t-1} + p_2 S_{j+t-2} +\ldots + p_{t-1} S_{j+1} + p_t S_j=0, \qquad (4.6)$$

где $j = 0, 1, ..., (t-1)$.

Решив эту систему уравнений относительно неизвестных p_1 p_2, ..., p_t и подставив их в многочлен локаторов ошибок (4.4), определим ошибочные позиции X_1 X_2, ..., X_t, используя процедуру Ченя и уравнения (4.5). Подставим найденные значения локаторов ошибок X_i в уравнения (4.3) и решим эту систему относительно неизвестных значений ошибок Y_1, Y_2, ..., Y_t .

Пример 4.1. Рассмотрим пример алгебраического декодирования кода РС с порождающим многочленом $G(x) = (x + 1)(x + \varepsilon)(x + \varepsilon^2)(x + \varepsilon^3)$, где ε – примитивный элемент поля $GF(2^m)$, образованного многочленом $g(x) =1+ x + x^3$, т.е. m=3. Многочлен $G(x)$ образует систематический (n, k) = (7, 3) код Рида–Соломона с $d_{\min}$=5 (рис. 4.1). Легко проверить, что исходной комбинации, представленной, например, многочленом $\varphi(x) = \varepsilon + \varepsilon^3 x + x^2$, соответствует закодированная комбинация, выраженная многочленом

$$f(x) = \varepsilon + \varepsilon^5 x + \varepsilon^5 x^2 + \varepsilon x^3 + \varepsilon x^4 + \varepsilon^3 x^5 + x^6,$$

где $n - k = 4$ младших разрядов являются проверочными элементами кода.

Рассмотрим пример исправления двух ошибок, многочлен которых имеет вид $e(x) = \varepsilon^5 x^3 + \varepsilon x^5$. Здесь в соответствии с введенными выше обозначениями $e(x) = Y_1 X_1 + Y_2 X_2$ имеем:

$X_1 = x^3$ и $X_2 = x^5$ − места расположения ошибок (локаторы);

$Y_1 = \varepsilon^5$ и $Y_2 = \varepsilon$ − соответствующие значения ошибок.

На первом этапе декодирования необходимо определить синдромы S_0, S_1, S_2, S_3, которые получаются путем умножения входной комбинации, соответствующей многочлену $h(x) = f(x) + e(x)$, на транспонированную проверочную матрицу H, имеющую вид:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \varepsilon & \varepsilon^2 & \varepsilon^3 & \varepsilon^4 & \varepsilon^5 & \varepsilon^6 \\ 1 & \varepsilon^2 & \left(\varepsilon^2\right)^2 & \left(\varepsilon^2\right)^3 & \left(\varepsilon^2\right)^4 & \left(\varepsilon^2\right)^5 & \left(\varepsilon^2\right)^6 \\ 1 & \varepsilon^3 & \left(\varepsilon^3\right)^2 & \left(\varepsilon^3\right)^3 & \left(\varepsilon^3\right)^4 & \left(\varepsilon^3\right)^5 & \left(\varepsilon^3\right)^6 \end{bmatrix}$$

Для определения синдромов воспользуемся формулой (4.3), при этом получим:

$$\begin{aligned} S_0 &= e(1) = Y_1 + Y_2 = \varepsilon^5 + \varepsilon = \varepsilon^6, \\ S_1 &= e(\varepsilon) = Y_1 X_1 + Y_2 X_2 = \varepsilon^5 \varepsilon^3 + \varepsilon \varepsilon^5 = \varepsilon^5, \\ S_2 &= e(\varepsilon^2) = Y_1 X^2_1 + Y_2 X^2_2 = \varepsilon^5 (\varepsilon^3)^2 + \varepsilon (\varepsilon^5)^2 = 0, \\ S_3 &= e(\varepsilon^3) = Y_1 X^3_1 + Y_2 X^3_2 = \varepsilon^5 (\varepsilon^3)^3 + \varepsilon (\varepsilon^5)^3 = \varepsilon^6. \end{aligned} \qquad (4.7)$$

Практическая реализация блока вычисления синдромов для рассматриваемого примера представлена на рис. 4.2, где представлены четыре регистра с сумматорами по модулю 2, предназначенные соответственно для определения синдромов. Верхний регистр реализует суммирование по модулю два всех элементов принятой комбинации, формируя синдром S_0. Остальные синдромы формируются тремя другими регистрами по схеме Горнера.

Рассмотрим подробнее формирование синдрома S_1. Пусть принятый многочлен имеет вид $h(x) = h_0 + h_1 x + h_2 x^2 + ... + h_5 x^5 + h_6 x^6$.

Тогда синдром S_1 будет получен путем умножения входного вектора $\{h_0\ h_1\ h_2\ h_3\ h_4\ h_5\ h_6\}$ на транспонированную строку проверочной матрицы $[1\ \varepsilon\ \varepsilon^2\ \varepsilon^3\ \varepsilon^4\ \varepsilon^5\ \varepsilon^6]$, т. е.

$$S_1 = h(\varepsilon) = h_0 + h_1\varepsilon + h_2\varepsilon^2 + h_3\varepsilon^3 + h_4\varepsilon^4 + h_5\varepsilon^5 + h_6\varepsilon^6.$$

Очевидно, что это же выражение может быть записано по схеме Горнера следующим образом:

$$S_1 = ([([(h_6\varepsilon + h_5)\varepsilon + h_4]\varepsilon + h_3)\varepsilon + h_2\ ]\varepsilon + h_1)\varepsilon + h_0.$$

Аналогично могут быть записаны выражения для S_2 и S_3:

$$S_2 = h(\varepsilon^2) = (\ldots[(h_6\,\varepsilon^2 + h_5)\,\varepsilon^2 + h_4]\,\varepsilon^2 + \ldots + h_1)\,\varepsilon^2 + h_0;$$

$$S_3 = h(\varepsilon^3) = (\ldots[(h_6\,\varepsilon^3 + h_5)\,\varepsilon^3 + h_4]\,\varepsilon^3 + \ldots + h_1)\,\varepsilon^3 + h_0.$$

Таким образом, в процессе получения синдрома S_1, реализуется умножение на ε, синдрома S_2 − на элемент ε^2 и синдрома S_3 — на элемент ε^3. Как было показано в разд. 2.3, умножениям входного вектора на постоянные элементы поля ε, ε^2, ε^3 соответствуют умножения на матрицы F, F^2, F^3, которые для многочлена $g(x) = 1 + x + x^3$ имеют вид:

$$F = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}; \quad F^2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}; \quad F^3 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Как следует из (4.4), при исправлении $t = 2$ ошибок искомые локаторы ошибок X_1 и X_2 являются корнями многочлена $\Psi(X) = X^2 + p_1 X + p_2$. Для определения коэффициентов p_1 и p_2 многочлена локаторов ошибок составим в соответствии с (4.6) следующую систему уравнений:

$$S_2 + p_1 S_1 + p_2 S_0 = 0;$$
$$S_3 + p_1 S_2 + p_2 S_1 = 0.$$

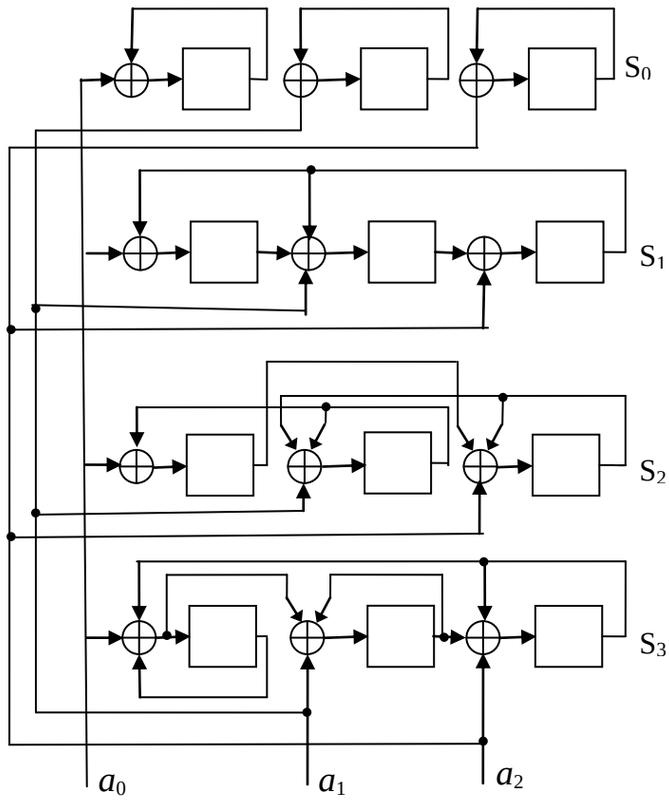


Рис.4.2. Схема реализации блока вычисления синдромов S_0, S_1, S_2, S_3 для многочлена $g(x) = 1 + x + x^3$

Решая систему уравнений относительно неизвестных p_1 и p_2, получим их выражения в общем виде:

$$p_1 = (S_0 S_3 + S_1 S_2)/(S_1^2 + S_0 S_2);$$
$$p_2 = (S_1 S_3 + S_2^2)/(S_1^2 + S_0 S_2), \qquad (4.8)$$

при условии, что $S_1^2 + S_0 S_2 \neq 0$.

Подставив в эти выражения значения синдромов, получим решения: $p_1 = \varepsilon^2$ и $p_2 = \varepsilon$.

Таким образом, многочлен локаторов ошибок будет иметь следующий вид:

$$\Psi(X) = X^2 + \varepsilon^2 X + \varepsilon. \qquad (4.9)$$

Корни этого многочлена, являющиеся локаторами ошибок X_1 и X_2, могут быть практически найдены на основе процедуры Ченя [6], суть которой сводится не к явному решению уравнения $\Psi(\varepsilon^i) = 0$, а к последовательной подстановке элементов поля $GF(2^m)$ в многочлен $\Psi(X)$ вместо переменной X.

В общей теории циклических кодов используются две формы разложения многочлена $\Psi(X)$ на множители. Одна из них была применена в разделе кодов БЧХ и имела вид (3.13), из которого следует, что локаторами ошибок являются непосредственно корни многочлена $\Psi(X)$. Для циклических кодов БЧХ и Рида–Соломона, исправляющих многократные ошибки (кратность $t \geq 3$), с точки зрения более простой реализации целесообразнее использовать вторую форму разложения на множители многочлена локаторов $\Psi(X)$ [2, 4, 6, 10]. Эта форма имеет вид: $\Psi(X) = \prod_{k=1}^{t} (1 - X_k X)$. В данном случае корнями многочлена $\Psi(X)$ будут элементы поля, обратные локаторам ошибок, а

именно: X_k^{-1}. Таким образом, ошибочные позиции (локаторы) будут определяться не самими корнями многочлена $\Psi(X)$, а их обратными элементами. Так, например, если корнем многочлена $\Psi(X)$ будет некоторый элемент поля ε^i, то соответствующий локатор ошибки будет $X_i = \varepsilon^{-i} = \varepsilon^{2^m-1-i}$. Следовательно, ошибочным будет элемент h_{2^m-1-i} в принятой комбинации. Поэтому при такой форме разложения многочлена $\Psi(X)$ процедура Ченя в декодере должна начинаться с элемента поля ε^{n-1}, где n – длина комбинации кода.

Далее мы будем рассматривать декодирование кодов Рида–Соломона применительно ко второй форме разложения многочлена локаторов ошибок на множители.

Таким образом, для рассматриваемого примера, локаторами ошибок X_1 и X_2 будут те элементы поля G$F(2^3)$, для которых $\Psi(X_1^{-1}) = \Psi(X_2^{-1}) = 0$. Подставив все ненулевые элементы поля G$F(2^3)$ поля ε^{-1}, ε^{-2}, ε^{-3}, ... в $\Psi(X)$, получим:

$$\Psi(\varepsilon^{-2}) = \Psi(\varepsilon^5) = 0; \qquad \Psi(\varepsilon^{-4}) = \Psi(\varepsilon^3) = 0,$$

т. е. ошибочными в принятой комбинации будут элементы h_3 и h_5, так как локаторы ошибок равны $X_1 = \varepsilon^3$, $X_2 = \varepsilon^5$.

Удобство процедуры Ченя заключается в том, что вычисление локаторов ошибок производится параллельно с выводом принятой комбинации (h_0 h_1 h_2 h_3 h_4 h_5 h_6) из буферного накопителя, начиная со старшего разряда. В частности, при выводе элемента h_6 вычисляется значение локатора ошибок $\Psi(\varepsilon^{-1}) = \Psi(\varepsilon^6)$, при выводе h_5 вычисляется $\Psi(\varepsilon^{-2}) = \Psi(\varepsilon^5)$ и т. д. В те моменты, когда многочлен локаторов ошибок становится равным нулю, т. е. $\Psi(\varepsilon^j) = 0$, из буферного накопителя выводится ошибочный элемент кода h_j и к нему добавляется вычисленное значение ошибки Y_j. При этом будет получен правильный элемент $c_j = h_j + Y_j$, т. е. происходит исправление ошибок.

Рассмотрим теперь практическую реализацию вычислителя локаторов ошибок. Заметим при этом, что если на i-м шаге было вычислено значение многочлена локаторов ошибок (4.4) при $X = \varepsilon^{-i}$; т. е. $\Psi(\varepsilon^{-i}) = (\varepsilon^{-i})^t + p_1 (\varepsilon^{-i})^{t-1} + ... + p_{t-1} \varepsilon^{-i} + p_t$,то на $(i + 1)$-м шаге будет вычислено следующее значение, а именно:

$$\Psi(\varepsilon^{-i-1}) = \varepsilon^{-(i+1)t} + p_1 \varepsilon^{-(i+1)(t-1)} + ... + p_{t-1} \varepsilon^{-(i+1)} + p_t.$$

Если последнее выражение записать следующим образом:

$$\Psi(\varepsilon^{-i-1}) = (\varepsilon^{-it})\varepsilon^{-t} + (p_1 \varepsilon^{-(t-1)})\varepsilon^{-(t-1)} + ... + (p_{t-1} \varepsilon^{-i})\varepsilon^{-1} + p_t,$$

то становится очевидным, что первый член многочлена $\Psi(\varepsilon^{-i})$ умножен на ε^{-t}, второй − на $\varepsilon^{-(t-1)}$, предпоследний − на ε^{-1}, а последний остается без изменения. Отсюда следует, что блок вычисления локаторов ошибок может быть реализован с помощью $t + 1$ регистров, первый из которых (R_1) реализует умножение содержимого на F^{-t}, второй регистр (R_2) − умножение на $F^{-(t+1)}$ и т. д. Последний регистр не меняет своего значения.

В исходном состоянии в регистры $R1$, $R2$, ... записываются коэффициенты многочлена $\Psi(X)$ соответственно 1, p_1,..., p_t. Схема, реализующая процедуру Ченя вычисления локаторов ошибок в общем виде, представлена на рис. 4.3.

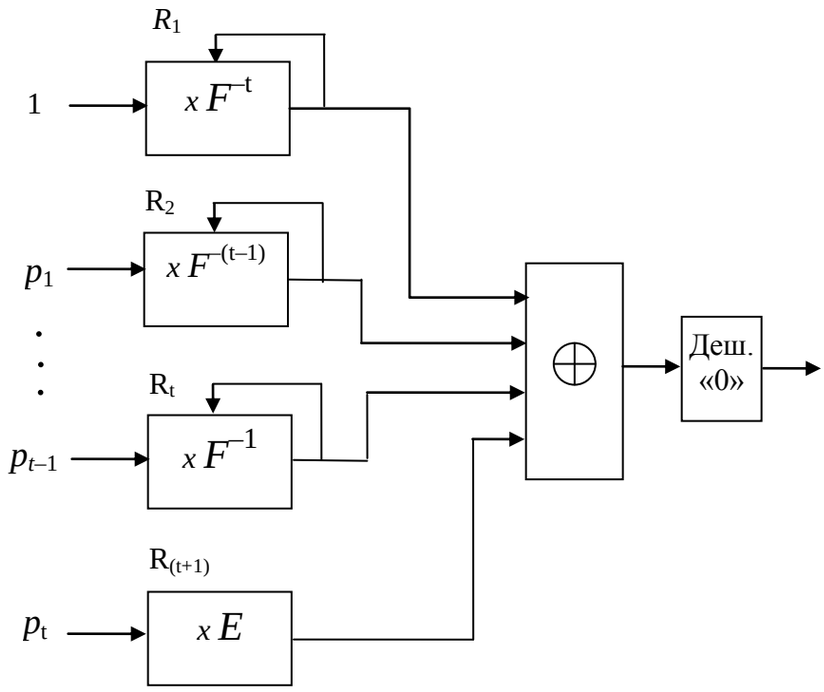


Рис. 4.3. Схема вычисления локаторов ошибок

Для рассматриваемого примера кода (7,3) с исправлением двукратных ошибок в поле $GF(2^3)$ с примитивным многочленом $g(x) = 1 + x + x^3$ многочлен $\Psi(X)$ имеет вид (4.9) с коэффициентами $p_1 = \varepsilon^2$ и $p_2 = \varepsilon$. Для данного многочлена $g(x)$ матрицы F^{-1} F^{-2} являются обратными по отношению к матрицам F и F^2 и имеют следующий вид:

$$F^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; \qquad F^{-2} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

В результате умножения некоторого вектора (a_0, a_1, a_2) на эти матрицы будем иметь

$$(a_0, a_1, a_2)\, F^{-1} = [(a_0 + a_1), a_2, a_0];$$
$$(a_0, a_1, a_2)\, F^{-2} = [(a_0 + a_1 + a_2), a_0, (a_0 + a_1)].$$

Схема вычисления локаторов ошибок X_1 и X_2 для данного примера приведена на рис. 4.4. На рис.4.4 верхний регистр реализует умножение на F^{-2}, средний регистр – на F^{-1} а нижний регистр представляет собой память, где запоминается коэффициент p_2 многочлена $\Psi(X)$.

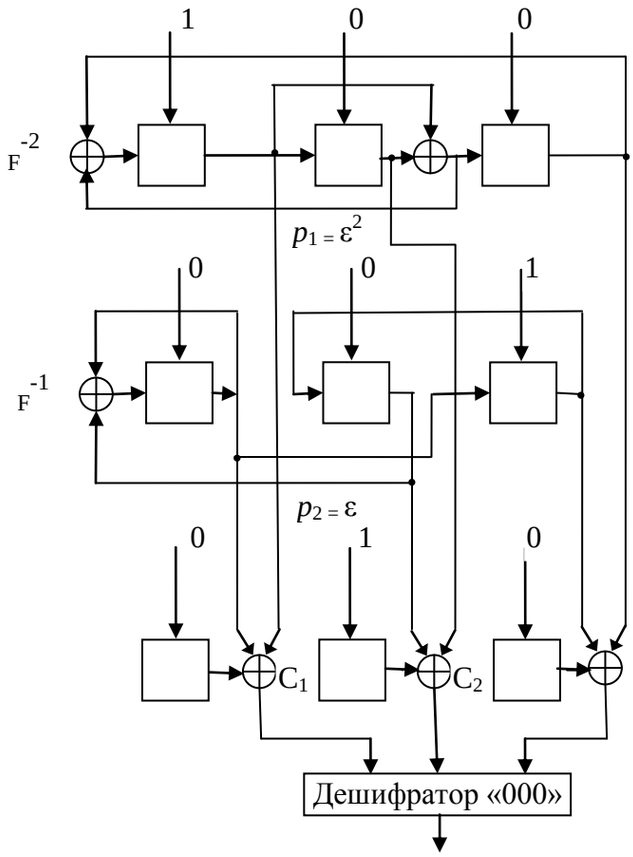


Рис. 4.4. Схема вычисления локаторов ошибок
для многочлена $\Psi(X) = X^2 + p_1X + p_2$, где $p_1 = \varepsilon^2$, $p_2 = \varepsilon$.

Рассмотрим теперь один из алгоритмов исправления ошибок. Как было показано, дешифратор нуля в схеме вычисления локаторов ошибок (рис. 4.4) срабатывает в моменты вывода ошибочных позиций X_1 и X_2 из буферного накопителя. Используя этот факт, легко определить значения X_1 и X_2, запустив с началом вывода кодовой комбинаций из буферного накопителя генератор обратных элементов поля Галуа $\varepsilon^{-(n-j)}$, где $n = 2^m - 1$. При первом срабатывании дешифратора нуля (рис. 4.4), т. е. когда из буфера выводится первый ошибочный элемент, генератор будет формировать элемент $X_2 = \varepsilon^j$. Для исправления этого ошибочного элемента к нему необходимо добавить по модулю2 значение ошибки Y_2, которое можно легко вычислить, зная значение X_2. Действительно, решая уравнения (4.7) для синдромов S_0 и S_1 относительно Y_1 и Y_2, находим, что

$$Y_1 = (S_1 + X_2 S_0)/(X_1 + X_2). \qquad (4.10)$$

При этом учтем, что сумма $(X_1 + X_2)$ была вычислена раньше как коэффициент p_1 многочлена локаторов ошибок по (4.8). Определив Y_1 и учитывая выражение для синдрома S_0, находим, что $Y_2 = S_0 + Y_1$.

Для рассматриваемого примера было определено, что

$$S_0 = \varepsilon^6, \quad S_1 = \varepsilon^5, \quad X_1 = \varepsilon^3, \quad X_2 = \varepsilon^5, \; p_1 = \varepsilon^2,$$

учитывая (4.10), находим значения ошибок:

$$Y_1 = (\varepsilon^5 + \varepsilon^6 \varepsilon^5)/ \varepsilon^2 = \varepsilon^5; \; Y_2 = S_0 + Y_1 = \varepsilon^6 + \varepsilon^5 = \varepsilon.$$

Схема алгоритма декодирования в общем виде для кода Рида–Соломона, исправляющего двукратные ошибки, представлена на рис. 4.5.

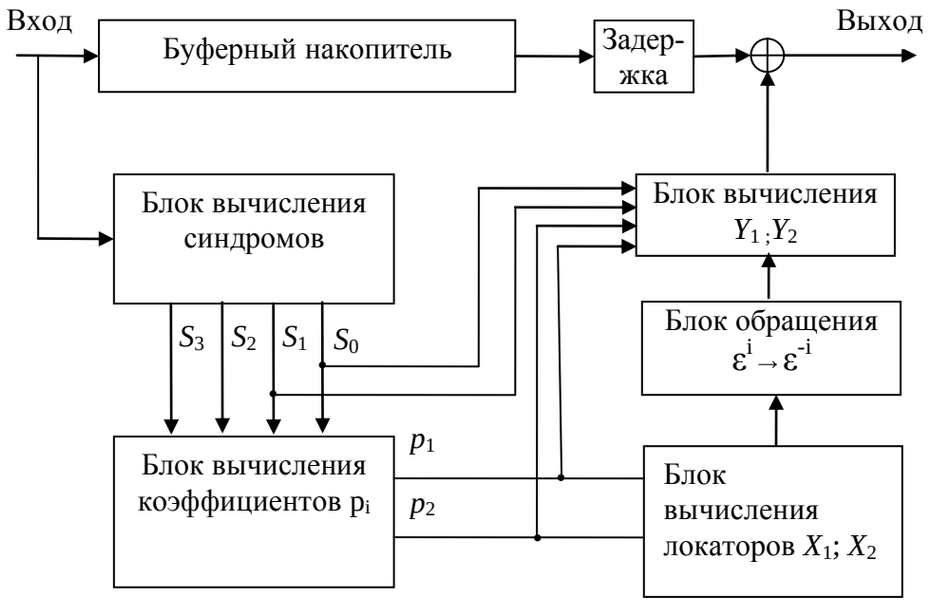


Рис. 4.5. Обобщенная схема декодера кода Рида–Соломона, исправляющего двукратные ошибки

Рассмотренный алгоритм является классическим алгоритмом алгебраического декодирования кодов Рида–Соломона. Вместе с тем существуют и другие алгоритмы алгебраического декодирования. В частности в [4] предложен более простой в реализации алгоритм исправления двукратных ошибок кодами Рида–Соломона. Другие обобщенные алгоритмы декодирования кодов Рида–Соломона рассмотрены в [2].

4.2. Построение кодов Рида–Соломона, исправляющих однократные ошибки

Систематические коды Рида-Соломона, исправляющие однократные ошибки, отличаются более простой реализацией. Эти (n, k) коды характеризуются минимальным кодовым расстоянием $d_{\min} = 3$, порождающим многочленом $G(x) = (x + 1)(x + \varepsilon)$ и проверочной матрицей

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \varepsilon & \varepsilon^2 & \varepsilon^3 & \dots & \varepsilon^{n-1} \end{bmatrix}. \qquad (4.11)$$

Учитывая (4.3), выражения для синдромов S_0 и S_1 в случае однократной ошибки с многочленом $e(x) = YX$ будут следующие:

$$S_0 = e(1) = Y; \qquad S_1 = e(\varepsilon) = Y\varepsilon^i, \qquad (4.12)$$

где $X = \varepsilon^i$ – локатор ошибки, указывающий на i-ю ошибочную позицию в комбинации, а Y – значение ошибки.

Из уравнений (4.12) видно, что

$$S_1 \varepsilon^{-i} + S_0 = 0. \qquad (4.13)$$

Уравнение (4.13) позволяет легко осуществить исправление ошибки. Для этого при выводе из буферного накопителя i-го элемента кодовой комбинации h_i каждый раз будем умножать синдром S_1 на ε^{-i} и сравнивать с S_0. В тот момент, когда будет выполняться уравнение (4.13), к выходному элементу h_i

следует добавить по модулю два синдром S_0, который представляет собой значение ошибки. Тем самым ошибка будет исправлена.

Функциональная схема декодера, исправляющего однократные ошибки, представлена на рис. 4.6.

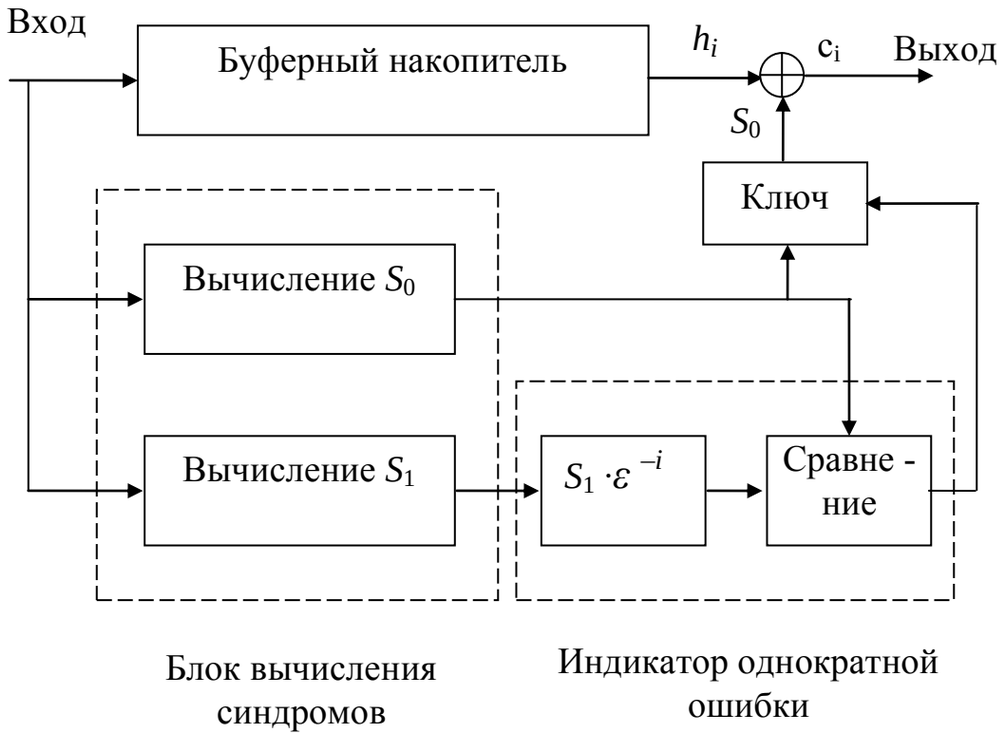


Рис. 4.6. Функциональная схема декодера кода Рида–Соломона, исправляющего однократные ошибки

При решении некоторых задач целесообразно построить код Рида-Соломона с $d_{\min} > 3$, который исправлял бы однократные ошибки и обнаруживал ошибки более высокой кратности. Очевидно, принцип построения кодирующего устройства не будет отличаться от обычного кодирующего устройства систематического (несистематического) циклического кода.

Рассмотрим процедуру декодирования, т. е. исправления однократных ошибок и обнаружения ошибок более высокой кратности для того же примера кода, который был ранее выбран для исправления двукратных ошибок, а именно: для кода $(n, k) = (7, 3)$ с $d_{\min}= 5$; с многочленом, образующим поле, $g(x) = 1 + x + x^3$ и порождающим многочленом $G(x)= (x + 1)(x + \varepsilon)(x + \varepsilon^2)(x + \varepsilon^3)$.

Синдромы этого кода в случае однократной ошибки будут иметь вид:

$$S_0 = e(1) = Y, \quad S_1 = e(\varepsilon) = YX = Y\varepsilon^i ; \qquad (4.14)$$
$$S_2 = e(\varepsilon^2) = YX^2 = Y\varepsilon^{2i}, \quad S_3 = YX^3 = e(\varepsilon^3) = Y\varepsilon^{3i} .$$

Задачей декодера является исправление однократной ошибки и обнаружение всех ошибок второй и третьей кратности. Очевидно, в состав декодера должен входить индикатор однократной ошибки. Проанализировав выражения (4.14) для синдромов при однократной ошибке, легко заметить, что

$$S_1^2 = S_0 S_2;$$
$$S_2^2 = S_1 S_3; \qquad (4.15)$$
$$S_0 S_3 = S_1 S_2.$$

Отсюда следует вывод, что индикатором однократной ошибки будет одновременное выполнение уравнений (4.15) при ненулевых синдромах. Если все четыре синдрома равны нулю, то это свидетельствует об отсутствии или о возникновении необнаруживаемых ошибок. Если же все или часть синдромов не равны нулю и хотя бы одно из уравнений (4.15) не выполняется, то это говорит о возникновении ошибок второй или большей кратности, которые будут обнаружены кодом.

Процедура непосредственного исправления однократной ошибки не отличается от рассмотренной ранее для кодов Рида–Соломона с $d_{\min} = 3$.

Схема декодера кода РС, исправляющего однократные ошибки с обнаружением ошибок более высокой кратности, представлена на рис. 4.7.

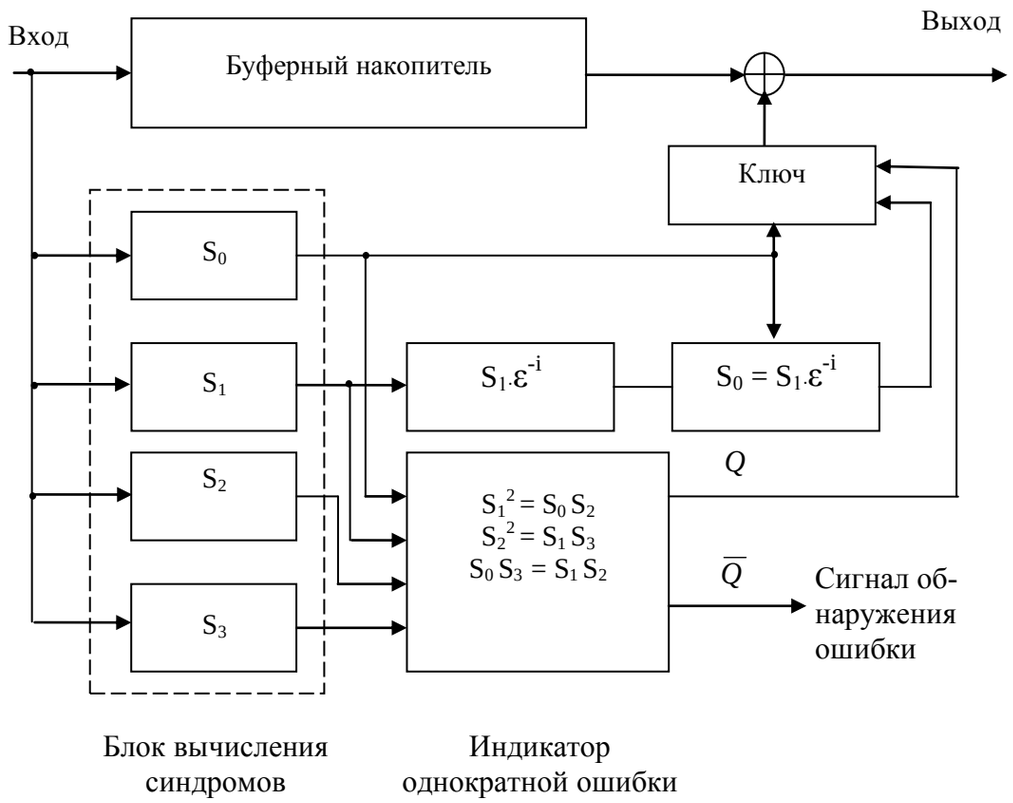


Рис. 4.7. Обобщенная схема декодера кода Рида–Соломона, исправляющего однократные ошибки и обнаруживающего часть ошибок более высокой кратности

В случае однократной ошибки появится сигнал на выходе Q индикатора однократной ошибки и будет удерживаться в течение считывания принятой комбинации кода РС из буферного накопителя. Этот сигнал Q совместно с сигналом со схемы сравнения откроют ключ, который пропустит синдром S_0 на выходной сумматор в момент считывания из накопителя ошибочного элемента. Таким образом, ошибка будет исправлена.

В случае появления обнаруживаемых ошибок второй или большей кратности на другом выходе индикатора однократной ошибки появится сигнал $\overline{Q}$, с помощью которого в случае необходимости можно стереть принятую с ошибками кодовую комбинацию.

Рассмотрим пример исправления однократной ошибки для кода (7, 3) с теми же, что и ранее, порождающим многочленом $G(x)$ и многочленом $g(x)$.

Пусть многочлен ошибки равен $e(x) = YX$, где $Y = \varepsilon^4$ − значение ошибки, а $X = \varepsilon^3$ − локатор ошибки.

Тогда на основании (4.14) синдромы будут иметь следующие значения:

$$S_0 = Y = \varepsilon^4;$$
$$S_1 = e(\varepsilon) = Y\varepsilon^3 = 1;$$
$$S_2 = e(\varepsilon^2) = \varepsilon^4\varepsilon^6 = \varepsilon^3;$$
$$S_3 = e(\varepsilon^3) = \varepsilon^4\varepsilon^9 = \varepsilon^6.$$

Подставив значения синдромов в уравнения (4.15), заметим, что все три сравнения выполняются, т. е. имеет место однократная ошибка.

Аналогично можно показать, что при двукратных и трехкратных ошибках уравнения (4.15) не выполняются, что свидетельствует об обнаружении ошибок большей кратности.

4.3. Применение кодов Рида-Соломона для исправления стираний

Код Рида–Соломона с минимальным расстоянием d может исправлять $d-1$ или менее стираний. При этом под стиранием понимается такое состояние приемника, когда значение сигнала попадает в так называемую зону неопределенности. Таким образом, при передаче кодового вектора по стирающему каналу на приемной стороне становятся известными номера позиций $X_1, X_2, .., X_t$ со стираниями, в некоторых системах эти позиции могут быть помечены специальной «маской». Поэтому задачей декодирования является определение неизвестных кодовых элементов, расположенных на этих позициях. Алгоритм исправления стираний мало чем отличается от исправления ошибок и сводится к решению системы уравнений (4.3) относительно неизвестных Y_i , которые являются значениями стертых символов комбинации либо значениями вектора ошибок в позициях, помеченных «масками».

Пример 4.2. Рассмотрим пример исправления стираний. Пусть (n, k)-код Рида-Соломона с порождающим многочленом $G(x) = (x + 1)(x + \varepsilon)$ имеет минимальное расстояние $d_{\min} = 3$. Тогда такой код исправляет одно или два стирания. Проверочная матрица имеет вид (4.11).

Пусть ε − примитивный элемент поля GF(2^3) с образующим его многочленом $g(x) = 1 + x + x^3$. Тогда, выбрав длину комбинации $n = 2^3-1$, мы построим код (7, 5), который может исправлять одно или два стирания или однократную ошибку без стираний.

Рассмотрим передачу по стирающему каналу кодового вектора

$$(c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ c_6) = (\varepsilon^6\ 1\ \varepsilon\ \varepsilon^2\ \varepsilon^3\ \varepsilon^4\ \varepsilon^5),$$

где элементы c_i являются коэффициентами разрешенной кодовой комбинации кода (7,5), представленной многочленом (4.1).

Предположим, что в результате приема были стерты элементы c_2 и c_4 и заменены нулями. Тогда на декодер поступит комбинация $(\varepsilon^6\ 1\ 0\ \varepsilon^2\ 0\ \varepsilon^4\ \varepsilon^5)$. Таким образом стертые позиции $X_1 = x^2$ и $X_2 = x^4$ известны. Умножив принятый вектор со стираниями на H^T в соответствии с (4.11), получим значения синдромов

$$S_0 = Y_1 + Y_2 = 1; \qquad S_1 = Y_1 X_1 + Y_2 X_2 = \varepsilon.$$

Решая полученную систему уравнений относительно значений Y_1 и Y_2 стертых символов, находим:

$$Y_1 = (S_1 + X_2 S_0)/(X_1 + X_2) = \varepsilon; \qquad Y_2 = S_0 + Y_1 = \varepsilon^3.$$

Таким образом, зная номера X_1 и X_2 стертых символов, может быть осуществлено исправление стираний путем замены нулей найденными значениями Y_1 и Y_2.

4.4. Быстрое декодирование кодов БЧХ

4.4.1. Ключевое уравнение

Рассмотрим процедуру быстрого декодирования кодов БЧХ применительно к кодам Рида–Соломона (РС). Такой подход не влияет на общность полученных результатов, но позволяет учесть все нюансы декодирования как двоичных, так и недвоичных циклических кодов. При этом будут использованы, более подробно рассмотрены и обобщены некоторые положения и понятия, приведенные в разд. 4.2.

Для РС кодов справедлива граница Синглтона:

$$n - k = d - 1 = 2t,$$

где $n - k$ — избыточность кодовой комбинации, d – минимальное кодовое расстояние, t –кратность гарантированно исправляемых ошибок.

Будем полагать, что код используется в режиме исправления ошибок. Пусть в приемник АПД поступила кодовая комбинация РС-кода

$$C(x)=f(x)+e(x),$$

где $f(x)$ – переданная передатчиком АПД кодовая комбинация, в которой в процессе передачи по каналу связи произошло v ошибок, отображаемых многочленом $e(x)$; здесь $0 \le v \le t$.

Многочлен ошибок можно представить в виде

$$e(x)=e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \ldots + e_{i_v} x^{i_v},$$

где e_{i_k} – значение ошибки, i_k – номер позиции кодовой комбинации, в которой произошла ошибка.

Для исправления ошибок необходимо определить e_{i_k} и i_k. Это возможно выполнить на основе вычисления синдрома. В процедуре быстрого декодирования кодов БЧХ [1] обобщением понятия синдрома является *синдромный многочлен*

$$S(x)=S_1 x^0 + S_2 x^1 + \ldots + S_{n-k} x^{n-k-1}. \qquad (4.16)$$

Коэффициенты S_i определяются подстановкой в $C(x)$ корней a^i порождающего многочлена кода РС:

$$S_1=C(x=\alpha^i)= f(x=\alpha^i)+e(x=\alpha^i)=e(\alpha^i),$$

где в общем случае для кодов БЧХ $m_0 \le i \le m_0+2t-1$ (m_0 – младшая степень корня в последовательности корней порождающего многочлена кода).

Для РС-кодов обычно $m_0=1$ и степени корней порождающего многочлена изменяются в пределах: $1 \le i \le 2t$. При этом для РС-кодов как подкласса кодов БЧХ значение m_0 может быть выбрано иным, если это связано с уменьшением сложности кодирующих и декодирующих устройств. Учет выбора значения m_0 на результат декодирования в процедуре быстрого декодирования будет показан при вычислении значения ошибки.

Теперь получим $S_1 = e_{i_1}\alpha^{i_1} + e_{i_2}\alpha^{i_2} + \ldots + e_{i_\nu}\alpha^{i_\nu}$ и т. д.

Для упрощения записи компонентов синдромного многочлена определим для всех $j=1,\ldots,\nu$ значения ошибки $Y_j = e_{i_j}$, и локаторы ошибок $X_j = \alpha^{i_j}$, где $i_j = j$ – истинное положение j-й ошибки.

В этих новых обозначениях S_1 запишется в виде

$$S_1 = Y_1X_1 + Y_2X_2 + \ldots + Y_\nu X_\nu .$$

Здесь Y_l и X_l – элементы поля $GF(q)$ над которым построен РС-код, а α – примитивный элемент этого поля.

В результате получим следующую систему из $2t$ уравнений относительно ν неизвестных локаторов $X_1,\ldots, X_\nu$ и ν неизвестных значений ошибок $Y_1,\ldots, Y_\upsilon$:

$$\begin{aligned} S_1 &= Y_1X_1 + Y_2X_2 + \ldots + Y_\nu X_\nu, \\ S_2 &= Y_1X_1^2 + Y_2X_2^2 + \ldots + Y_\nu X_\nu^2, \\ &\vdots \\ S_{2t} &= Y_1X_1^{2t} + Y_2X_2^{2t} + \ldots + Y_\nu X_\nu^{2t}. \end{aligned}$$

Итак, $S_j = \sum_{i=1}^{\upsilon} Y_i X_i^j = e(\alpha^j)$ – компонент синдрома и $S(x) = \sum_{j=1}^{2t} S_j x^j = \sum_{j=1}^{2t}\sum_{i=1}^{\upsilon} Y_i X_i^j x^j$ – многочлен в принципе бесконечной степени, для которого известны только младшие $2t$ коэффициентов – синдромный многочлен, вычисленный декодером для поступившей из канала в приемник АПД комбинации РС-кода.

В силу определения синдрома эта система уравнений должна иметь хотя бы одно решение. В теории кодирования доказано, что это решение единственно. Изложение известных алгоритмов поиска этого решения базируется на понятии ключевого уравнения и методах его решения.

Для формирования ключевого уравнения вводятся еще два многочлена.

Многочлен локаторов ошибок $\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2 + \Lambda_\nu x^\nu$ – это многочлен, степень которого равна числу выявленных в процессе декодирования ошибок $\nu \le t$, а корни обратны локаторам ошибок $X_i^{-1} = \alpha^{-i}$, $i = 1, \ldots, \nu$, т. е. $X_i^{-1} \cdot \alpha^i = 1$, что позволяет представить $\Lambda(x)$ в следующем виде:

$$\Lambda(x) = \prod_{l=1}^{\nu} (1 - X_l x). \qquad (4.17)$$

Многочлен значений ошибок определяется через $S(x)$ и $\Lambda(x)$ в соответствии с видом введенного выше многочлена ошибок

$$\Omega(x) = S(x) * \Lambda(x) (mod x^{2t}). \qquad (4.18)$$

Фактически компоненты синдромного многочлена могут иметь ненулевые значения лишь до степени x^{2t-1}. Этим объясняется приведение произведения $S(x)*\Lambda(x)$ по модулю x^{2t}. Подставляя в выражение (4.18) для $\Omega(x)$ определенные выше через локаторы и значения ошибок многочлены $S(x)$ и $\Lambda(x)$, получаем

$$\Omega(x) = \left[\sum_{j=1}^{2t}\sum_{i=1}^{\upsilon} Y_i X_i^j x^{j-1}\right]\left[\prod_{i=1}^{\upsilon}(1 - X_i x)\right] \left(\mathrm{mod}\, x^{2t}\right) =$$

$$= \sum_{i=1}^{\upsilon} Y_i X_i \left[(1 - X_i x)\sum_{j=1}^{2t} (X_i x)^{j-1}\right] \prod_{l \ne i} (1 - X_l x) \left(\mathrm{mod}\, x^{2t}\right).$$

Сворачивая выражение в квадратных скобках, получаем окончательно

$$\Omega(x)=\sum_{i=1}^{\upsilon} Y_i X_i (1-X_i^{2t}x^{2t})\prod_{l\neq i}(1-X_l x)(\mathrm{mod}\, x^{2t})\,.$$

Приводя это выражение по модулю x^{2t}, получаем

$$\Omega(x)=\sum_{i=1}^{\upsilon} Y_i X_i \prod_{l\neq i}(1-X_l x)\,. \qquad (4.19)$$

Из (4) видно, что степень многочлена $\Omega(x)$ не старше $\nu-1$, и поэтому наряду с представленной (4.18) и (4.19) существует еще одна форма записи ключевого уравнения:

$$[\Lambda(x)\cdot S(x)]_{\nu}^{2t-1}=0\,, \qquad (4.20)$$

где $[a(x)]_m^n = a_m x^m + a_{m+1}x^{m+1}+\ldots+a_n x^n$.

Из (4.20) можно получить ν уравнений для ν неизвестных коэффициентов Λ_k. Эти уравнения являются линейными. Они могут быть решены обычными методами либо с помощью итерационных процедур. После нахождения $\Lambda(x)$ ключевое уравнение позволяет найти неизвестные компоненты многочлена $e(x)$ и по ним переданную кодовую комбинацию $f(x)=C(x)+e(x)$.

Из изложенного можно сделать вывод, что декодирование кодов БЧХ на основе решения ключевого уравнения распадается на два этапа.

Этап I – вычисление многочлена локаторов ошибок $\Lambda(x)$, его корней и связанных с ними локаторов ошибок. Для двоичных кодов БЧХ этим этапом декодирование завершается.

Этап II – для недвоичных кодов БЧХ, каковыми являются и РС-коды, вычисление многочлена значений ошибок $\Omega(x)$, позволяющего вычислять значение каждой из ν ошибок в принятой комбинации.

Для нахождения многочлена локаторов ошибок из литературы[1,2,6,7] известны три алгоритма: алгоритм Питерсона, алгоритм Берлекэмпа–Месси и алгоритм Евклида - алгоритм СКХН.

Авторы первого и второго алгоритмов указаны в их названии. Известный из курса математики алгоритм Евклида, предназначенный для вычисления наибольшего общего делителя (НОД) двух целых чисел или двух многочленов от одной переменной, для целей решения ключевого уравнения был впервые использован четырьмя авторами: Сугияма, Касахара, Хирасава и Намекава. В дальнейшем для краткости будем называть алгоритмом Евклида.

Прежде, чем перейти к изучению алгоритмов решения ключевого уравнения, целесообразно рассмотреть пример, иллюстрирующий процедуры, на основе которых строится метод быстрого декодирования.

Пример 4.3. Пусть над полем GF(2^3) с элементами 0 = 000, α^0=1=100, α^1=010, α^2=001, α^3=110, α^4=011, α^5=111, α^6=101 построен код Рида–Соломона (7,3) с $d_{\min}=5$, способный исправлять все однократные и двукратные ошибки. Корнями порождающего многочлена являются следующие элементы $GF(2^3)$: α^1=010, α^2=110, α^3=110, α^4=011.

Порождающий многочлен кода имеет вид

$$g(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)=\alpha^3+\alpha^1 x+\alpha^0 x^2+\alpha^3 x^3+x^4$$

Предположим, что по каналу связи была передана комбинация $f(x)$ =(0000000), а на вход декодера поступила ошибочная комбинация $C(x) = e(x) = \alpha^2 x^3 + \alpha^5 x^4$.

Схема вычисления синдрома определила компоненты синдромного многочлена:

$$S_1=f(x=\alpha)=\alpha^5+\alpha^2=\alpha^3,$$
$$S_2=f(x=\alpha^2)=\alpha^1+\alpha^6=\alpha^5,$$
$$S_3=f(x=\alpha^3)=\alpha^4+\alpha^3=\alpha^6,$$
$$S_4=f(x=\alpha^4)=\alpha^0+\alpha^0=0.$$

При вычислении значений элементов S_i, показатели степеней элементов поля приводятся по mod7, так как для $GF(2^3)$ $\alpha^0=\alpha^7=1$.

Итак, для принятой комбинации синдромный многочлен имеет вид: $S(x)=\alpha^3+\alpha^5x+\alpha^6x^2$.

Определим для принятой комбинации многочлен локаторов ошибок $\Lambda(x)$, используя известный из исходных данных примера многочлен ошибок $e(x)$. Не составляет труда определить локаторы ошибок: $X_3=\alpha^3$, $X_4=\alpha^4$ и соответствующие им корни $\Lambda(x)$: $\alpha^{-3}=\alpha^4$ и $\alpha^{-4}=\alpha^3$. По определению многочлена локаторов ошибок (4.17) находим $\Lambda(x)=(1+\alpha^3x)(1+\alpha^4x)=1+\alpha^6x+x^2$. По выражению (4.18) вычисляем многочлен значений ошибок $\Omega(x)=S(x)*\Lambda(x)\ (\mathrm{mod}\ x^{2t})=(\alpha^3+\alpha^5x+\alpha^6x^2)(1+\alpha^6x+x^2)\ \mathrm{mod}\ x^4=\alpha^3+\alpha^3x$.
Далее решим уравнение (4.19) относительно первого локатора ошибок: $\Omega(x=\alpha^{-3})=\alpha^3+1==Y_3\alpha^3(1+\alpha^4\alpha^{-3})$, откуда находим $Y_3=(\alpha^3+1)/\alpha^3(1+\alpha^4\alpha^{-3})=\alpha^2$. Аналогично для второго локатора: $\Omega(x=\alpha^{-4})=\alpha^3\alpha^{-4}+\alpha^3=Y_4\alpha^4(1+\alpha^3\alpha^{-4})$, откуда находим $Y_4=\alpha^4/\alpha^4(1+\alpha^6)=\alpha^5$. По результатам вычислений определяем многочлен ошибок: $e(x)=\alpha^2x^3+\alpha^5x^4$. Сложение вычисленного многочлена ошибок с принятой комбинацией дает переданную комбинацию: $C(x)+e(x)=\alpha^2x^3+\alpha^5x^4+\alpha^2x^3+\alpha^5x^4=(0000000)$.

Из примера следует вывод о необходимости дополнения рассмотренного метода декодирования процедурами вычисления корней $\Lambda(x)$ и значений ошибок Y_j .

Обычно для нахождения корней многочлена $\Lambda(x)$ используется метод проб и ошибок, известный как *процедура Ченя*. Эта процедура состоит в последовательном вычислении $\Lambda(\alpha^j)$ для каждого возможного j и проверки полученных значений на ноль. Если величина $\Lambda(\alpha^{-k})$ равна нулю, то α^k является локатором ошибки, взаимным к корню многочлена локаторов ошибок, и k-й элемент кодовой комбинации содержит ошибку.

Простым способом вычисления значения $\Lambda(x)$ в точке β является *схема Горнера*:

$$\Lambda(\beta)=(\ldots(((\Lambda_\upsilon\beta+\Lambda_{\upsilon-1})\beta+\Lambda_{\upsilon-2})\beta+\Lambda_{\upsilon-3})\beta+\ldots\Lambda_0).$$

Для вычисления $\Lambda(\beta)$ по схеме Горнера требуется только ν умножений и ν сложений, где ν – степень $\Lambda(x)$.

Приведенный в примере 4.3 способ вычисления значений ошибок обобщен в литературе [1,2] теоремой Форни и под названием алгоритма Форни будет рассмотрен ниже.

Алгоритм Питерсона может быть использован как самостоятельная процедура декодирования недвоичных циклических кодов и на этапе II.

Алгоритм Берлекэмпа–Месси и алгоритм Евклида на этапе II декодирования для недвоичных циклических кодов дополняют алгоритмом Форни, позволяющим по корням $\Lambda(x)$ и многочлену $\Omega(x)$ найти значения ошибок. Сочетание алгоритмов решения ключевого уравнения и алгоритма Форни получило название *быстрого декодирования кодов БЧХ.*
Рассмотрим сущность изложенных алгоритмов декодирования.

4.4.2. Решение ключевого уравнения

Алгоритм Питерсона

У. Питерсон [7] представил ключевое уравнение $[\Lambda(x)\cdot S(x)]_{\upsilon}^{2t-1}=0$ в матричной форме. Это можно сделать, перемножая многочлены $\Lambda(x)=1+\lambda_1 x+\lambda_2 x^2+\lambda_v x^v$ и $S(x)=S_1x^0+S_2x^1+\ldots+S_{n-\kappa}\,x^{n-k-1}$ и собирая коэффициенты в результирующем многочлене при степенях x от v до $2v-1$.

Для степени v: $1S_{v+1}+\lambda_1 S_v+\ldots+\lambda_v S_1=0$,

Для степени $v+1$: $1S_{v+2}+\lambda_1 S_{v+1}+\ldots+\lambda_v S_2=0$,

. . . . . .

Для степени $2v-1$: $1S_{2v}+\lambda_1 S_{2v+1}+\ldots+\lambda_v S_v=0$.

Представляя полученные равенства в виде системы уравнений относительно первых слагаемых и переходя к матричной форме записи, получаем

$$\begin{bmatrix} S_{v+1} \\ S_{v+2} \\ \bullet \\ \bullet \\ S_{2v} \end{bmatrix} = \begin{bmatrix} S_1 & S_2 & S_v \\ S_2 & S_3 & S_{v+1} \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ S_v & S_{v+1} & S_{2v-1} \end{bmatrix} \cdot \begin{bmatrix} \lambda_v \\ \lambda_{v-1} \\ \bullet \\ \bullet \\ \lambda_1 \end{bmatrix} \qquad (4.21)$$

Таким образом, задача определения коэффициентов многочлена локаторов ошибок в алгоритме Питерсона сводится к прямому решению системы v линейных уравнений с v неизвестными. Примеры использования алгоритма Питерсона для декодирования кодов БЧХ и оценка сложности реализации декодера приведены в предыдущих разделах.

Алгоритм Форни

Д. Форни получил выражение для вычисления значения ошибки по известным многочленам значения ошибок и локаторов ошибок.

Вычислением многочлена значений ошибок $\Omega(x)=\sum_{i=1}^{\upsilon} Y_i X_i \prod_{l\neq i}(1-X_l x)$ на позиции l, т. е. подстановкой $x=X_l^{-1}$, получаем

$$\Omega(X_l^{-1})=Y_l X_l \prod_{j\neq l}(1-X_j X_l^{-1}),$$

откуда находим

$$Y_l=\frac{\Omega(X_l^{-1})X_l^{-1}}{\prod_{j\neq l}(1-X_j X_l^{-1})}.$$

Найдем производную от многочлена локаторов ошибок $\Lambda(x)$:

$$\Lambda'(x)=-\sum_{i=1}^{\upsilon} X_i \prod_{j\neq i}(1-xX_j).$$

Для l-й позиции получаем

$$\Lambda'(X_l^{-1})=-X_l \prod_{j\neq l}(1-X_j X_l^{-1}).$$

С учетом последнего выражения Y_l значение ошибки на позиции l принимает вид:

$$Y_l = -\frac{\Omega(X_l^{-1})}{\Lambda'(X_l^{-1})}. \qquad (4.22)$$

Полученный результат справедлив для кодов БЧХ и РС, у которых последовательность корней порождающего многочлена начинается с первой степени, т. е. $m_0 = 1$.

В общем случае, когда младшая степень корня $m_0 > 1$, выражение (4.22) уточняется следующим образом [2]:

$$Y_l = -X_l^{m_0-1}\frac{\Omega(X_l^{-1})}{\Lambda'(X_l^{-1})}\ , \qquad (4.23)$$

где X_l – локатор ошибки;

m_0 – младшая степень в последовательности корней порождающего многочлена используемого кода БЧХ или Рида– Соломона;

$\Lambda(x)$ – многочлен локаторов ошибок;

X_l^{-1} – корень многочлена $\Lambda(x)$, обратный локатору ошибки X_l;

$\Omega(x)$ – многочлен значений ошибок;

$\Lambda'(x)$ – производная от многочлена $\Lambda(x)$.

Пример 4.4. Применим алгоритм Форни для вычисления значений ошибок в условиях предыдущего примера. В процессе декодирования декодер вычислил для принятой кодовой комбинация $C(x) = \alpha^2x^3 + \alpha^5x^4$ многочлены локаторов ошибок $\Lambda(x) = 1+\alpha^6x+x^2$ и значений ошибок $\Omega(x) = \alpha^3 + \alpha^3x$. Кроме того, вычислены корни многочлена $\Lambda(x)$ и определены локаторы ошибок: X_3 и X_4. Для вычисления значений ошибок находим производную многочлена локаторов ошибок $\Lambda'(x) = \alpha^6$. В соответствии с (8) вычисляем:

для локатора X_3 : $X_3^{\,m_0-1}=1$, $\Omega(x=\alpha^4) = \alpha^3 + \alpha^3\cdot\alpha^4 = 1 + \alpha^3 = \alpha$, $Y_3 = 1\cdot\alpha/\alpha^6 = \alpha^2$;

для локатора X_4 : $X_4^{\,m_0-1}=1$, $\Omega(x=\alpha^3) = \alpha^3 + \alpha^3\cdot\alpha^3 = \alpha^3 + \alpha^6 = \alpha^4$, $Y_4 = 1\cdot\alpha^4/\alpha^6 = \alpha^5$.

В результате вычислен многочлен ошибок $e(x) = \alpha^2x^3 + \alpha^5x^4$. Сумма принятой комбинации и многочлена ошибок $C(x) + e(x) = 0$ дает значение исправленной (переданной) комбинации: (0000000).

Алгоритм Берлекэмпа–Месси.

Этот алгоритм был предложен Э. Берлекэмпом и Дж. Месси [1,6] как итеративный процесс построения минимального линейного регистра сдвига с обратной связью, аналогичного схеме решения разностных уравнений, генерирующего все компоненты синдромного многочлена $S(x)$ по ν первым. Целью алгоритма является нахождение коэффициентов многочлена локаторов ошибок $\Lambda(x)$, значение которых определяется по виду обратных связей построенного регистра.

Алгоритм Берлекэмпа–Месси базируется на том факте, что в основе соотношений (4.20) и (4.21) лежит рекуррентное уравнение

$$\Lambda_0 S_j + \Lambda_1 S_{j-1} + \ldots + \Lambda_\nu S_{j-\nu} = 0.$$

Полагая, что многочлен $\Lambda(x)$ известен и известны значения $S_{1,\ldots,}\, S_\nu$, по приведенному рекуррентному уравнению последовательно определяются значения $S_{\nu+1}, \ldots, S_{2\nu}$:

$$S_j = \sum_{i=1}^{\nu} \Lambda_i S_{j-i}\ . \qquad (4.24)$$

Значения S_j, задаваемые равенством (4.24), полностью определяются известной схемой для решения разностных уравнений, представляющей

собой регистр сдвига с обратными связями, соответствующими коэффициентам многочлена Λ(x), приведенной на рис.4.8.

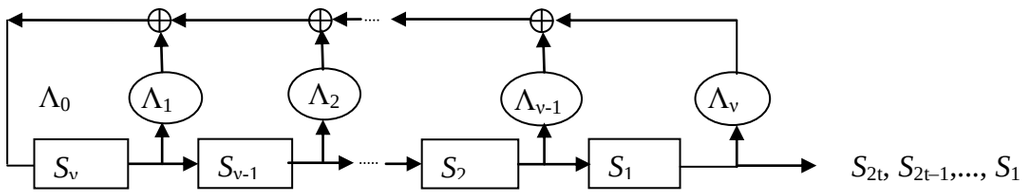


Рис. 4.8. Регистр сдвига с обратными связями,
соответствующими коэффициентам многочлена Λ(x),
определяющий значения $Sv_{+1},\ldots, S_{2t}$ по значениям $S_1,\ldots, Sv$

Таким образом, рекуррентное соотношение (4.24) с использованием схемы рис. 4.8 позволяет вычислить значение S_j, начиная с Sv_{+1} и до S_{n-k} по известному многочлену $\Lambda(x)$ и v младшим компонентам $S(x) – S_1,\ldots, Sv$.

При выполнении процедуры декодирования по алгоритму Берлекэмпа–Месси известным является синдромный многочлен $S(x)$, а многочлен локаторов ошибок $\Lambda(x)$ подлежит определению. Для реализации процедуры декодирования на основе рекуррентного соотношения (4.24) решаемая задача была переформулирована Берлеэмпом и Месси следующим образом: по $2t$ младшим компонентам синдромного многочлена $S(x)$, вычисленного декодером для принятой кодовой комбинации кода БЧХ, содержащей не более $v \leq t$ ошибок, построить регистр сдвига с обратными связями с минимальным (желательно равным $v \leq t$) числом ячеек, способный по v младшим компонентам синдромного многочлена $S(x)$ вычислить все $2t$ компонентов. Построение искомого регистра длины L предполагается итеративным по методу проб и ошибок. Общее число итераций r равно числу используемых компонентов синдромного многочлена $S(x)$: $r = \Lambda_r(x)$, даже если $2t<n–k$, что возможно для общего случая кодов БЧХ. Для РС кодов $2t=n–k$, и в процедуре построения минимального регистра используются все компоненты синдромного многочлена $S(x)$.

Многочлен $\Lambda(x)$ как многочлен обратных связей создаваемого регистра на каждом шаге итерации r вычисляется по формуле:

$$\Lambda_r(x) = \Lambda_{r-1}(x) + \Delta_r x B_{r-1}(x). \qquad (4.25)$$

Здесь Δ_r – невязка, вычисляемая в начале каждого шага итерации и численно равная отличию вычисленного декодером для принятой комбинации компонента синдромного многочлена S_r от ожидаемого на выходе сформированного по результатам предыдущего шага регистра того же компонента S_r :

$$\Delta_r = S_r - \sum_{j=1}^{v=L} \Lambda_j S_{r-j}\,, \qquad (4.26)$$

принимающая для двоичного случая значение $\Delta_r = \sum_{j=0}^{L} \Lambda_j S_{r-j}$; $B(x)$ – добавка, вычисляемая для каждого шага итерации на предыдущем шаге для получения нового значения $\Lambda_r(x)$

Исходными данными для итеративной процедуры вычисления многочлена локаторов ошибок $\Lambda(x)$ являются : $2t$ первых компонентов синдромного многочлена $S(x)$, $\Lambda(x)=1$, $B(x)=1$, $L=0$.

На каждом шаге итерации r находится самый короткий сдвигающий регистр, порождающий S_r по S_{r-1}. Далее проверяется, порождает ли данный регистр также и S_{r+1}. На некотором шаге очередной компонент синдромного многочлена уже не будет порождаться.

В этот момент необходимо модернизировать регистр таким образом, чтобы он удовлетворял следующим требованиям:

- правильно предсказывал очередной компонент синдромного многочлена $S(x)$,
- не изменял значение вычисленных ранее компонентов $S(x)$,
- увеличивал длину регистра на минимально возможную величину.

Требуемая длина регистра на шаге r итерации определяется так:

$$L_r = \max[L_{r-1}, r - L_{r-1}]. \qquad (4.27)$$

Анализ выражения (4.27) показывает, что увеличение длины регистра происходит на одну единицу на каждом нечетном шаге при ненулевом значении Δ_r. На этом шаге вычисляется добавка $B_r(x) = \Delta_r^{-1} \Lambda_{r-1}(x)$, используемая для вычисления $\Lambda(x)$ на следующем (четном) шаге итерации. На четном шаге r итерации вычисляется добавка $B_r(x) = x\, B_{r-1}(x)$, используемая для вычисления $\Lambda(x)$ на следующем, нечетном шаге итерации.

Таким образом, каждый очередной этап модернизации регистра охватывает два шага итерации – нечетный и четный.

На нечетном шаге выполняется модификация регистра, выражающаяся в увеличении длины регистра на одну ячейку и добавлении новой связи при сохранении предыдущего регистра:

$$\Lambda_r(x) = \Lambda_{r-1}(x) + \Delta_r x^2 B_{r-2}(x).$$

При этом добавленная связь не всегда позволяет получить на выходе регистра компонент синдромного многочлена S_j со значением $j > r$.

На четном шаге выполняется нормализация связей в регистре, построенном на предыдущем шаге:

$$\Lambda_r(x) = \Lambda_{r-1}(x) + \Delta_r x\, \Delta_{r-1}^{-1} \Lambda_{r-2}(x).$$

Нормализация на четном шаге определяет окончательный вид связей для сформированного на нечетном шаге регистра заданной длины и позволяет только за счет изменения значения связей при сохранении длины регистра получить на выходе регистра по меньшей мере еще один компонент синдромного многочлена S_j. После нормализации регистр генерирует все компоненты синдромного многочлена S_j со значением j по меньшей мере, равным $1 \ldots r$.

Итоговая блок-схема алгоритма представлена на рис. 4.10.

Представленная блок-схема реализует рассмотренный выше принцип решения ключевого алгоритма в соответствии с алгоритмом Берлекэмпа–Месси. Действуя по приведенному алгоритму, наряду с итеративной процедурой вычисления многочлена локаторов ошибок $\Lambda(x)$,можно аналогичным способом вычислить и многочлен значений ошибок $\Omega(x)$. Поскольку $\Omega(x)$ находится как произведение $S(x)$ и $\Lambda(x)$, можно показать [2], что последовательность значений $\Omega_r(x)$ вычисляется по тем же

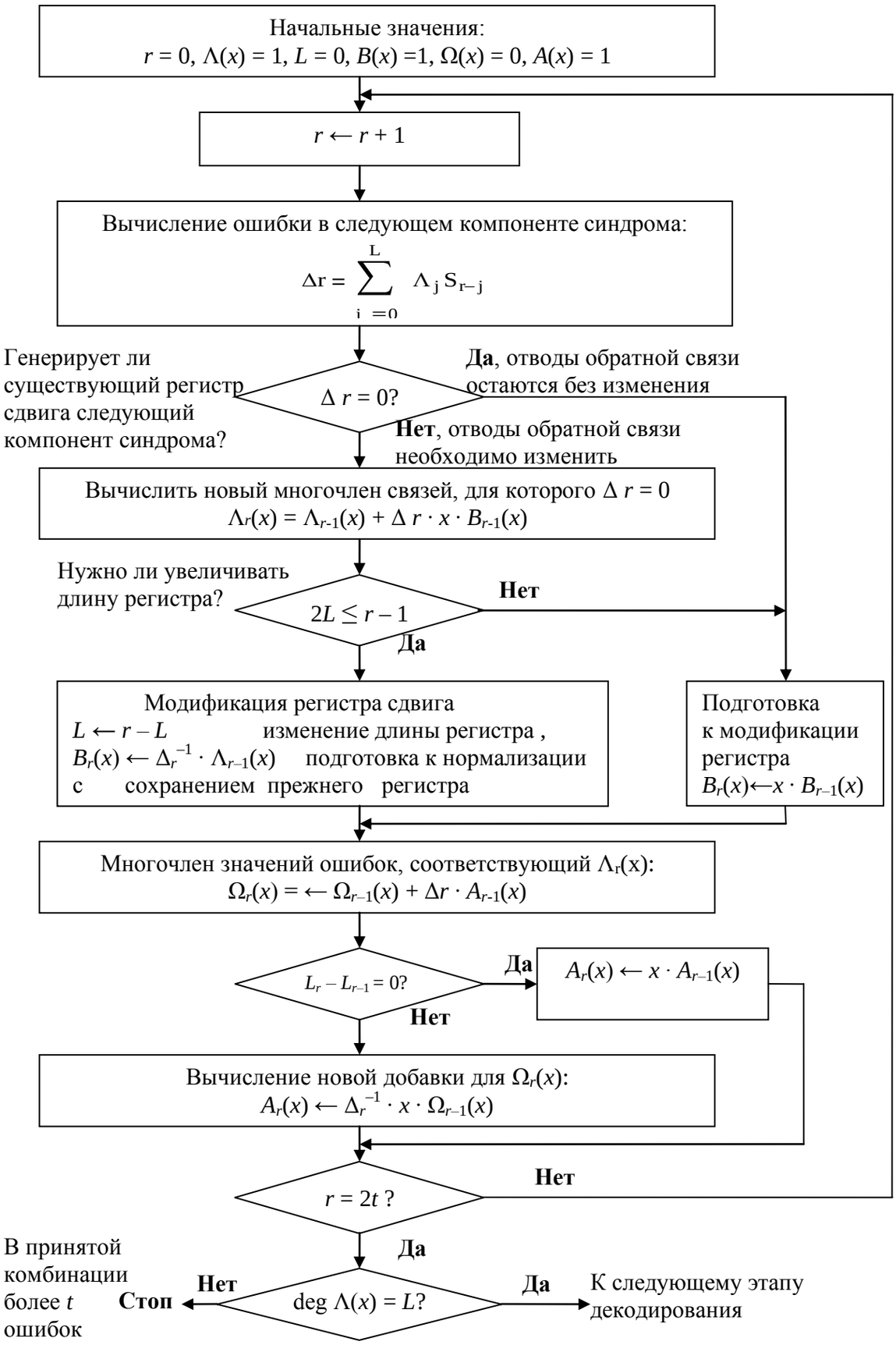


Рис. 4.10 Алгоритм Берлекэмпа-Месси

рекуррентным соотношениям, что и $\Lambda_r(x)$. При этом необходимо учитывает, что степень $\Omega(x)$ меньше степени $\Lambda(x)$.Общее выражение для $\Omega_r(x)$ имеет вид

$$\Omega_r(x)=\Omega_{r-1}(x)+\Delta_r A_{r-1}(x), \qquad (4.28)$$

где $A_{r-1}(x)$ – добавка, по значению аналогичная $B(x)$.

Пример 4.5. Рассмотрим процедуру декодирования кода Рида–Соломона (7,3) из примера 4.3 в соответствии с алгоритмом Берлекэмпа–Месси. Вычисление $\Lambda(x)$ и $\Omega(x)$ представим в виде табл. 4.1.

Таблица 4.1

r	S_r	Δr	$\Lambda(x)$	$B(x)$	L	$\Omega(x)$	$A(x)$
0			1	1	0	0	1
1	α^3	α^3	$1+\alpha^3 x$	α^4	1	α^3	0
2	α^5	α	$1+\alpha^2 x$	$\alpha^4 x$	1	α^3	0
3	α^6	α^2	$1+\alpha^2 x+ \alpha^6 x^2$	α^5+x	2	α^3	αx
4	0	α^2	$1+\alpha^6 x+x^2$	$\alpha^5 x+x^2$	2	$\alpha^3+ \alpha^3 x$	αx

Найденному значению $\Lambda(x)=1+\alpha^6 x+x^2$ соответствует регистр сдвига с обратными связями, определенными видом $\Lambda(x)$, длины $L=2$, способный генерировать все компоненты синдромного многочлена по двум младшим, представленный на рис. 4.9.

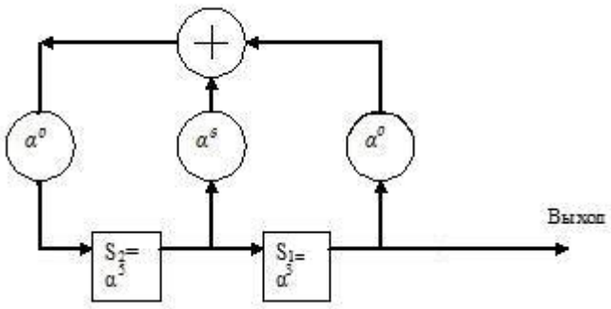


Рис. 4.9. Итоговый регистр сдвига, определяющий вид многочлена локаторов ошибок $\Lambda(x)=1+\alpha^6 x+x^2$ – результат использования алгоритма Берлекэмпа–Месси для решения задачи примера 4.3.

В исходном состоянии в ячейках памяти регистра записаны компоненты синдрома $S_1=\alpha^3$ и $S_2=\alpha^5$. По первому такту $S_1=\alpha^3$ поступает на выход, а в ячейках остаются $S_2=\alpha^5$ и $S_3=\alpha^6$. По второму такту $S_2=\alpha^5$ поступает на выход, а в ячейках остаются $S_3=\alpha^6$ и $S_4=0$, которые за два следующих такта поступают на выход.

Пример 4.6. Рассмотрим код БЧХ (15,7) с символами из поля GF(2^2), построенного по примитивному многочлену $\pi(x) = 1+ x + x^2$ и имеющего состав: $0 = 00$, $a^0 = 10$, $a^1 = 01$, $a^2 = 1 + a = 11$, где a – корень $\pi(x) = 1+ x + x^2$, т.е. $\pi(a) = 1+ a + a^2 = 0$. Так как многочлен $\pi(x) = 1+ x + x^2$ принадлежит показателю 3 (входит в состав двучлена $x^3 + 1$), то его корни a и a^2 имеют порядок 3, т. е. $a^3 = 1$. Корнями порождающего многочлена кода БЧХ (15,7) являются элементы поля GF(2^4): ε^2 ,ε^5 ,ε^6 ,ε^7 ,ε^8 ,ε^9 ,ε^{10},ε^{13} [12]. Поле GF(2^4) построено по примитивному многочлену $\pi(x) = 1+ x + x^4$. Поле GF(2^2) входит в состав поля GF(2^4) в качестве подполя. При этом элементам a и a^2 поля GF(2^2) соответствуют элементы ε^5 и ε^{10} поля GF(2^4) соответственно. Шесть подряд идущих степеней корней порождающего многочлена начинаются со степени $m_0 = \varepsilon^5$, а заканчиваются степенью $m_0+d–2 = \varepsilon^{10}$. Это означает, что код (15,7) имеет минимальное кодовое расстояние 7 и способен исправить все ошибки до третьей кратности включительно. Пусть по каналу связи передается нулевая кодовая комбинация этого кода, а в декодер поступила комбинация

$f'(x) = a+x^5+x^{10}$. Декодер вычисляет синдромный многочлен $S(x) = a^2 + ax + a^2x^2 + a^2x^3 + ax^4 + a^2x^5$.

Процесс вычисления $\Lambda(x)$ и $\Omega(x)$ по алгоритму Берлекэмпа–Месси представлен в табл. 4.2.

Таблица 4.2

r	S_r	Δr	$\Lambda(x)$	$B(x)$	L	$\Omega(x)$	$A(x)$
0			1	1	0	0	1
1	a^2	a^2	$1+a^2x$	a	1	a^2	0
2	a	0	$1+a^2x$	ax	1	a^2	0
3	a^2	a	$1+a^2x+a^2x^2$	a^2+ax	2	a^2	ax
4	a^2	0	$1+a^2x+a^2x^2$	a^2x+ax^2	2	a^2	ax^2
5	a	a	$1+a^2x+ax^2+a^2x^3$	$a^2+ax+ax^2$	3	$a^2+a^2x^2$	ax
6	a^2	1	$1+x^3$	$a^2x+ax^2+ax^3$	3	$a^2+ax+a^2x^2$	ax^2

Для нахождения локаторов и значений ошибок вычисляем корни $\Lambda(x)$.Полагая $\Lambda(x)=1+x^3 = 0$, определяем: $x_1 = \varepsilon^0 = 1/X_0$,$x_2 = \varepsilon^5 = 1/X_{10}$, $x_3 = \varepsilon^{10} = 1/X_5$. Это значит, что локаторы ошибок равны $X_0 = \varepsilon^0$, $X_5 = \varepsilon^5$, $X_{10} = \varepsilon^{10}$.

В соответствии с алгоритмом Форни вычисляем значения ошибок. Выражение (4.8) для рассматриваемого примера имеет вид: $Y_i = \varepsilon^{i(1-m_0)} \dfrac{\Omega(x=\varepsilon^{-i})}{\Lambda'(x=\varepsilon^{-i})}$.

Подставляя в это выражение необходимые данные, получаем :

$Y_0=a$, $Y_5=1$, $Y_{10}=1$. Значит многочлен ошибок имеет вид: $e(x) = a+x^5+x^{10}$, что в точности соответствует принятой комбинации. Следовательно, на выход декодера должна поступить комбинация из одних нулей.

Алгоритм Евклида

Известно, если существует наибольший общий делитель $d(x)$ двух многочленов $a(x)$ и $b(x)$, то существуют многочлены $f(x)$ и $g(x)$ такие, что справедливо: $a(x)f(x)+b(x)g(x)=d(x)$.

Многочлен $d(x)$ может быть найден по алгоритму Евклида, который состоит в последовательном делении с остатком $a(x)$ на $b(x)$, затем $b(x)$ на первый остаток $r_1(x)$, затем $r_1(x)$ на второй остаток $r_2(x)$ и т. д.

При декодировании кодов БЧХ интересуются не конечным результатом алгоритма Евклида, а промежуточными результатами, которые можно представить в виде:

$$a(x)f(x)+b(x)g(x)=r(x).$$

У. Сугияма и его соавторы [2] использовали этот результат для решения ключевого уравнения следующим образом: $b(x)g(x)= r_i(x) \pmod{a(x)}$, полагая,

$$a(x)=x^{2t},\ b(x)=S(x),\ g_l(x)=\Lambda_i(x),\ r_i(x)=\Omega_i(x),$$

где $S(x)$, $\Lambda_i(x)$ и $\Omega_i(x)$– введенные выше многочлены синдромный, локаторов ошибок и значений ошибок соответственно – компоненты ключевого уравнения.

При этом используется свойство алгоритма Евклида:

$$\deg[g_i(x)]+\deg[r_{i-l}(x)]=\deg[a(x)].$$

Если $a(x)=x^{2t}$, то $\deg[\Lambda_i(x)]+\deg[\Omega_{i-l}(x)]=2t$, $\deg[\Lambda_i(x)]+\deg[\Omega_i(x)]<2t$.
При появлении $\nu \le t$ ошибок имеем: $\deg[\Omega(x)]< \deg[\Lambda(x)] \le t$.

Существует единственный с точностью до постоянного множителя (т. е. элемента поля) многочлен $\Lambda(x)$ степени $\le t$, удовлетворяющий уравнению:

$$\Omega_i(x) = S(x)\,\Lambda_i(x) \pmod{x^{2t}},$$

если $\deg[\Omega_{i-1}(x)] \ge t$ при $\deg[\Lambda_i(x)] \le t$ и $\deg[\Omega_i(x)]< t$ при $\deg[\Lambda_{i+1},(x)] > t$.

Поэтому промежуточные результаты на i-м шаге дают единственное интересующее нас решение ключевого уравнения.

Таким образом, для решения ключевого уравнения следует применять алгоритм Евклида до тех пор, пока не будет выполнено условие

$$\deg[\Omega_i(x)]< t.$$

Алгоритм Евклида решения ключевого уравнения по методу У. Сугиямы и др. сводится к следующему:

1. Применить алгоритм Евклида к $a(x)=x^{2t}$ и $b(x)=S(x)$.

2. Использовать начальные условия: $\Lambda_{-1}(x)=0$, $\Lambda_0(x)=1$, $\Omega_{-1}(x)=x^{2t}$, $\Omega_0(x) = S(x)$.

3. Остановиться, если $\deg[\Omega_i(x)]< t$.

4. Положить $\Lambda(x)= \Lambda_i(x)$ и $\Omega(x)= \Omega_i(x)$.

При вычислении значений $\Lambda_i(x)$ и $\Omega_i(x)$ следует использовать свойство алгоритма Евклида: каждое из значений $\Lambda_i(x)$ и $\Omega_i(x)$ получается из двух предшествующих значений по следующей общей формуле:

$$E_i(x)=E_{i-2}(x) - q_i(x)E_{i-1}(x)\ ,\ \text{где}\ q_i(x)=\left[\frac{\Omega_{i-2}(x)}{\Omega_{i-1}(x)}\right]_0^{2t}.$$

В квадратных скобках записано частное от деления указанных многочленов т. е. многочлен, степень которого может находиться в пределах $0 \ldots 2t$.

Поиск значений $\Lambda_i(x)$ в $\Omega_i(x)$ по алгоритму Евклида, удовлетворяющих приведенным выше критериям, удобно представить в виде табл. 4.3.

Таблица 4.3

Функция	Шаг				
	-1	0	1	…	i
$\Lambda_i(x)$	0	1	$\Lambda_1(x)= \Lambda_{-1}(x)+ q_1(x)\cdot \Lambda_0(x)$	⋯	$\Lambda_i(x)= \Lambda_{i-2}(x)+ q_i(x)\cdot \Lambda_{i-1}(x)$
$\Omega_i(x)$	x^{2t}	$S(x)$	$\Omega_1(x)= \Omega_{-1}(x)+ q_1(x)\cdot \Omega_0(x)$	⋯	$\Omega_i(x)= \Omega_{i-2}(x)+ q_i(x)\cdot \Omega_{i-1}(x)$
$q_i(x)$	—	—	$q_1(x)=\left[\frac{\Omega_{-1}(x)}{\Omega_0(x)}\right]$	⋯	$q_i(x)=\left[\frac{\Omega_{i-2}(x)}{\Omega_{i-1}(x)}\right]$

Пример 4.7. Рассмотрим процедуру декодирования кода БЧХ (15,7) в условиях предыдущего примера с использованием алгоритма Евклида для вычисления значений $\Lambda_i(x)$ в $\Omega_i(x)$. Результаты пошагового вычисления декодером значений $\Lambda_i(x)$ в $\Omega_i(x)$ представлены в табл. 4.4.

Процедура вычисления декодером значений $\Lambda_i(x)$ в $\Omega_i(x)$ завершается на том шаге, когда степень многочлена значений ошибок $\Omega_i(x)$ станет меньше, чем степень гарантийно исправляемых кодом ошибок *t*.

Таблица 4.4

Функция	Шаг				
	-1	0	1	2	3
$\Lambda_i(x)$	0	1	$1+ ax$	$1 + x + ax^2$	$a+ax^3$
$\Omega_i(x)$	x^6	$S(x)$	$a^2+a^2x+ax^3+a^2x^4$	$a^2+ x+a^2x^3$	$1+ a^2x+x^2$
$q_i(x)$	_	_	$[x^6/S(x)] = 1+ ax$	x	$a^2 +x$

В рассматриваемом примере это произошло на шаге 3. Следует обратить внимание на известный из литературы факт [2] – вычисление декодером значений $\Lambda_i(x)$ в $\Omega_i(x)$ с использованием алгоритма Евклида осуществляется с точностью до постоянного множителя – элемента поля. Представленные в табл. 4.4 значения $\Lambda_3(x)$ и $\Omega_3(x)$ отличаются от соответствующих многочленов, вычисленных в примере 4.6 по алгоритму Берлекэмпа–Месси на постоянный множитель *a*. На результате декодирования это никак не отразится, так как корни многочленов $a+ax^3$ и $1+x^3$ совпадают, и результат деления $\Omega(x)$ на $\Lambda'(x)$ не изменяется, если каждый из них умножен на одно и то же число. Находим корни $\Lambda_i(x)$. Полагаем $a+ax^3 = 0$, тогда $x_1 = \varepsilon^0 = 1/X_0$, $x_2 = \varepsilon^5 = 1/X_{10}$, $x_3 = \varepsilon^{10} = 1/X_5$. Это значит, что локаторы ошибок равны $X_0=\varepsilon^0$, $X_5=\varepsilon^5$, $X_{10}=\varepsilon^{10}$. В соответствии с алгоритмом Форни вычисляем значения ошибок по формуле (8): $Y_0=a$, $Y_5=1$, $Y_{10}=1$. Вычисленные локаторы ошибок и значения ошибок полностью совпадают с приведенными для алгоритма Берлекэмпа–Месси. Представляет интерес сравнить результаты вычисления значений ошибок в примерах 4.6 и 4.7. Итоги расчетов приведены в табл. 4.5 и показывают полное совпадение результатов расчета локаторов и значений ошибок по этим алгоритмам.

Таблица 4.5

Алгоритм	Позиция ошибки *i*	$\varepsilon^{i(1-m_0)}$	$\Omega(z=\varepsilon^{-i})$	$\Lambda^1(z=\varepsilon^{-i})$	$Y_i = \varepsilon^{i(1-m_0)}\frac{\Omega(x=\varepsilon^{-i})}{\Lambda'(x=\varepsilon^{-i})}$
Берлекэмпа-Месси	0	1	$a^2+a+a^2=a$	1	$1\frac{a}{1}=a$
	5	$a^{-4} = a^2$	$a^2+aa^2+a^2a^4 = a^2$	$a^4=a$	$a^2\frac{a^2}{a}=1$
	10	$a^{-8} = a$	$a^2+aa+a^2a^2 = a$	a^2	$a\frac{a}{a^2}=1$
Евклида	0	1	$1 + a^2 +1= a^2$	a	$1\frac{a^2}{a}=a$
	5	$a^{-4} = a^2$	$1+a^2a^2+a^4= 1$	$aa^4=a^2$	$a^2\frac{1}{a^2}=1$
	10	$a^{-8} = a$	$1 +a^2a+a^2= a^2$	$aa^2=1$	$a\frac{a^2}{1}=1$

Представляет интерес сравнить рассматриваемые алгоритмы решения ключевого уравнения по сложности реализации. Сравнительный анализ алгоритма Берлекэмпа–Месси и алгоритма Евклида по наиболее важным общим характеристикам приведен в табл.4.6.

Таблица 4.6

Алгоритм	Коли-чество исход-ных данных	Число шагов итера-ций	Число вычисля-емых парамет-ров на каждом шаге	Число вычисляе-мых сложных функций на каждом шаге	Возможность получения в ходе одного вычисления двух различных параметров
Берлекэмпа–Месси	6	$2t$	6	5	Отсутствует
Евклида	2	$\le t$	3	3	В ходе одной процедуры деления $\Omega_{i-2}(x)$ на $\Omega_{i-1}(x)$ одновременно вычисляются $q_i(x)$ и $\Omega_i(x)$ как частное и остаток

Таблица 4.6 показывает преимущество алгоритма Евклида над алгоритмом Берлекэмпа–Месси в отношении сложности и времени вычислений.

Важным свойством этих алгоритмов является их возможность исправлять стирания. Эта возможность определяется следующим выражением:

$$2\nu + \rho + 1 = d_{\text{мин}} \ , \qquad (4.29)$$

где ν и ρ – кратности гарантийно исправляемых ошибок и стираний соответственно.

Ошибки и стирания могут исправляться либо в сочетании, либо раздельно. При раздельном исправлении число исправляемых стираний в 2 раза превосходит число исправляемых ошибок. Метод совместного исправления ошибок и стираний по алгоритму Берлекэмпа–Месси подробно изложен в работе [13].

Остановимся на процедуре декодирования с исправлением ошибок и стираний на основе алгоритма Евклида. Исправление стираний, а также ошибок и стираний осуществляется по той же схеме, что и исправление ошибок. Отличие состоит в необходимости ввода в исходные данные еще трех многочленов:

$\Gamma(x)$ – многочлен локаторов стираний ,

$\Psi(x) = \Gamma(x)\,\Lambda(x)$ – многочлен локаторов искажений ,

$T(x) = S(x)\,\Gamma(x) \bmod x^{2t}$ – видоизмененный синдромный многочлен.

Эти многочлены используются на этапе формирования исходных данных. В ходе вычислений в многочлене $\Psi(x)$ находится и уточняется лишь сомножитель $\Lambda(x)$. Производная $\Psi'(x)$ и корни полученного в итоге вычислений многочлена $\Psi(x)$ используется для расчета значений ρ стираний и ν ошибок в соответствии с алгоритмом Форни. Кроме того изменился и критерий завершения вычислений: итерации заканчиваются на шаге, когда

$$\deg[\Omega_i(x)] < t + \rho/2, \text{ если } \rho \text{ – четно или}$$
$$\deg[\Omega_i(x)] < t + (\rho-1)/2, \text{ если } \rho \text{ – нечетно} , \qquad (4.30)$$

где ρ – степень $\Gamma(x)$.

Рассмотрим пример декодирования по алгоритму Евклида с исправлением стираний и ошибок.

Пример 4.8. Выполним процесс декодирования с исправлением стираний и ошибок при использовании кода Рида–Соломона (7,3) из предыдущих примеров. Пусть переданная от АПД кодовая комбинация имеет вид: $f(x)=\alpha^3+\alpha^1x+\alpha^0x^2+\alpha^3x^3+x^4$, а в декодер поступила комбинация с ошибками и стираниями $f'(x)=\Theta_0+\Theta_1x+0x^2+\alpha^3x^3+x^4$, т. е. коэффициенты при нулевой и первой степенях x приняты стертыми, а коэффициент при второй степени превратился в 0. Таким образом принята комбинация $f'(x)$=(Θ_0 $\Theta_1$0$\alpha^3$1000), где Θ_0 и Θ_1 – стирания. Декодер вычислил известными методами синдромный многочлен $S(x)=\alpha^3+\alpha^4x+\alpha^4x^3$, многочлен локаторов стираний $\Gamma(x)=(1+x)(1+\alpha x)=1+\alpha^3x+\alpha x^2$ и видоизмененный синдромный многочлен $T(x) = S(x)\,\Gamma(x) \bmod x^4 = \alpha^3+\alpha^3x+\alpha^5x^2+x^3$. Результаты вычислений $\Omega(x)$ и $\Psi(x) = \Gamma(x)\,\Lambda(x)$ представлены в табл.4.7.

Таблица 4.7

Функция	Шаг		
	–1	0	1
$\Psi(x) = \Gamma(x)\,\Lambda(x)$	0	$\Gamma(x)$	$\Gamma(x)(x+\alpha^5)$
$\Omega(x)$	x^4	$T(x)$	$x+\alpha$
$q(x)$	-	-	$x+\alpha^5$

Степень $\Omega(x)$ равна 1< 3 и в соответствии с (4.30) $\Omega(x)$ и $\Lambda(x)$ найдены. Далее декодер вычисляет $\Psi(x) = \Gamma(x)\,\Lambda(x)= \alpha^5+\alpha^3x+\alpha^4x^2+\alpha\,x^3$, $\Psi'(x)= \alpha^3+\alpha x^2$. Корень $\Lambda(x)$ $x_1= \alpha^5$ и отсюда локатор ошибки $X_2 = \alpha^2$. По известным локаторам стираний определяем корни многочлена $\Gamma(x)$: $X_0^{-1}=1$ и $X_1^{-1} =\alpha^6$.

Значения стираний: $$Y_0 = \frac{\Omega(x=1)}{\Psi'(x=1)} = \frac{1+\alpha}{\alpha+\alpha^3} = \frac{\alpha^3}{1} = \alpha^3,$$

$$Y_1 = \frac{\Omega(x=\alpha^6)}{\Psi'(x=\alpha^6)} = \frac{\alpha^5}{\alpha^6+\alpha^3} = \frac{\alpha^5}{\alpha^4} = \alpha,$$

$$Y_2 = \frac{\Omega(x=\alpha^5)}{\Psi'(x=\alpha^5)} = \frac{\alpha^5+\alpha}{\alpha^4+\alpha^3} = \frac{\alpha^6}{\alpha^6} = 1.$$

Изложенная процедура исправления стираний и ошибок может быть применена к случаю исправления только одних стираний. Она может быть также успешно применена для реализации процедуры кодирования[14].

5. МАЖОРИТАРНОЕ ДЕКОДИРОВАНИЕ ЦИКЛИЧЕСКИХ КОДОВ

Широкому применению кодов, исправляющих любые сочетания ошибок кратности t и менее, способствовала возможность мажоритарного декодирования, основанного на принципе голосования, т. е. принятия решения «по большинству голосов» [3,4]. Использование мажоритарного декодирования для циклических кодов, особенно кодов БЧХ, позволяет значительно упростить реализацию декодера.

В основе мажоритарного декодирования двоичных циклических кодов лежит то, что для каждого кодового символа c_i можно составить S контрольных проверок на четность вида

$$c_i = \sum_{i=1}^{r} c_{ij} = c_{i1} + c_{i2} + \ldots + c_{ir} \ (mod\ 2) \qquad (5.1)$$

Если при этом элемент c_i не входит в правую часть ни одного из соотношений (5.1), а любой элемент $c_{ij} \neq c_i$ входит только к одно соотношение, то такие проверки называют разделенными или ортогональными. К разделенным проверкам (5.1) добавляют тривиальную проверку $c_i = c_i$, что позволяет для определения элемента c_i использовать $S + 1$ проверочных соотношений. Отсюда следует, что для циклического кода, исправляющего ошибки кратности t и менее, количество S контрольных соотношений (5.1) с разделенными проверками не должно быть меньше $2t$.

Тогда, с учетом тривиальной проверки, будем иметь $S + 1 \geq d_{\min} = 2t + 1$. Из этого, а также из свойств ортогональных проверок следует, что ошибки в t кодовых символах могут привести к неправильному определению значения c_i не более чем в t проверках. Именно последнее обстоятельство и позволяет для определения значения символа c_i применить мажоритарный принцип.

Наряду с разделенными проверками существует мажоритарное декодирование с квазиразделенными проверками и с λ-связанными проверками [3].Система квазиразделенных проверок отличается от системы разделенных проверок тем, что в каждую проверку входит одна и та же линейная комбинация символов, тогда как любой другой символ c_j, не входящий в эту комбинацию, входит только в одну из проверок.

Система λ-связанных проверочных соотношений вида (5.1), определяющих символ c_i, характеризуются тем, что любой другой символ $c_j \neq c_i$ входит не более чем в λ проверок, но существует хотя бы один символ c_j, который входит точно, в λ проверок.

На практике более широкое применение находит мажоритарное декодирование с разделенными проверками.

Не проводя подробный анализ эффективности того или иного метода мажоритарного декодирования циклических кодов, укажем лишь некоторые его достоинства.

Первое из них заключается в сравнительно простой технической реализации мажоритарного декодирования. Схема декодера двоичного циклического кода содержит мажоритарный элемент, регистр сдвига и сумматоры по модулю два.

Второе достоинство мажоритарного декодирования заключается в том, что система проверочных соотношений для символа c_i при сдвиге содержимого регистра переходит в систему проверок для очередного символа комбинации c_{i+1} .Эта особенность позволяет осуществить процедуру последовательного (один за другим) мажоритарного декодирования всех символов комбинации циклического кода.

Наконец, существенным достоинством мажоритарного декодирования является также и то, что кроме всех ошибок кратности $t \leq S/2$ могут исправляться и некоторые ошибки более высокой кратности.

Пример 5.1. Приведем пример мажоритарного декодирования циклического кода (7,3) с разделенными проверками и с порождающим многочленом $G(x) = (1 + x)(1 + x + x^3)$. Этот код имеет минимальное кодовое расстояние $d_{\min} = 4$. Очевидно, что такой код может исправлять однократные ошибки и обнаруживать все двукратные ошибки.

Порождающая матрица G систематического циклического кода (n, k) =(7, 3) с указанным выше многочленом $G(x) = 1 + x^2 + x^3 + x^4$ имеет вид

$$G = [R\,E_k] = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (5.2)$$

Проверочная матрица H образуется из единичной матрицы размерности ($n - k$) и транспонированной матрицы остатков R^T, т. е.

$$H = \left[E_{n-k} R^T \right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (5.3)$$

Так как циклический код является нулевым пространством по отношению к проверочной матрице, то для любой разрешенной кодовой комбинации ($c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ c_6$) справедливо равенство

$$(c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ c_6)H^T = (0000).$$

Таким образом, умножив вектор-строку ($c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ c_6$) на транспонированную проверочную матрицу (5.3), получим следующие уравнения:

$$\begin{aligned} &c_0 + c_4 + c_5 = 0; \\ &c_1 + c_5 + c_6 = 0; \\ &c_2 + c_4 + c_5 + c_6 = 0; \\ &c_3 + c_4 + c_6 = 0. \end{aligned} \quad (5.4)$$

Взяв первое из уравнений системы (5.4), а также сложив по модулю два первое и третье, первое, второе и четвертое уравнения получим следующую систему разделенных проверок для мажоритарного определения элемента c_0:

$$\begin{aligned} &c_0 = c_4 + c_5; \\ &c_0 = c_2 + c_6; \\ &c_0 = c_1 + c_3 \end{aligned} \quad (5.5)$$

Как видно из системы (5.5), любой элемент $c_i \neq c_0$ входит только в одну из трех проверок, что позволяет правильно определить значение c_0 по большинству при любой однократной ошибке.

Напомним свойство циклических кодов, заключающееся в том, что если вектор ($c_0\ c_1\ c_2\ c_3\ c_4\ c_5\ c_6$) соответствует разрешенной комбинации не укороченного циклического (n, k)-кода, то ее циклический сдвиг влево, т. е. ($c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_0$), также является разрешенной кодовой комбинацией, для которой система разделенных проверок (5.5) превратится в систему для определения уже элемента c_1:

$$\begin{aligned} &c_1 = c_5 + c_6; \\ &c_1 = c_3 + c_0; \\ &c_1 = c_2 + c_4 \end{aligned} \quad (5.6)$$

Таким образом, мажоритарный элемент, определяющий любой элемент кода c_i и позволяющий исправлять однократные ошибки, является одним и тем же.

Добавив систему разделенных проверок (5.5) тривиальной проверкой $c_0 = c_0$, получим систему, позволяющую не только исправлять любую однократную ошибку «по большинству», но и обнаруживать любую двукратную ошибку. Например, если ошибки возникли в элементах c_4 и c_5 то при мажоритарном декодировании значение элемента c_0 будет определено в

соответствии с уравнениями (5.5) правильно, а при определении значения элемента c_1 по уравнениям (5.6) и тривиальной проверке $c_1 = c_1$ возникнет неопределенность, так как два значения c_1 будут правильными, а два – нет. При такой ситуации мажоритарный декодер будет формировать сигнал обнаружения ошибок кратностью больше единицы.

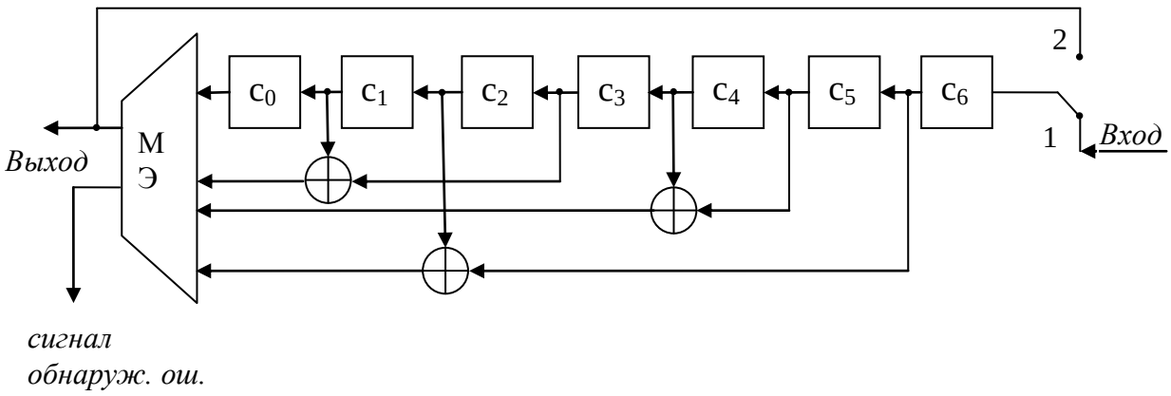


Рис. 5.1. Функциональная схема мажоритарного декодирования циклического кода с образующим многочленом $G(x) = 1 + x^2 + x^3 + x^4$

Функциональная схема декодера, соответствующего рассмотренному примеру, показана на рис. 5.1. Из описанного выше алгоритма следует, что при поступлении на вход декодера комбинации в течение n тактов ключ находится в положении 1, а в течение следующих n тактов, пока осуществляется считывание комбинации, ключ находится в положении 2.

ЗАКЛЮЧЕНИЕ

Рассмотрены алгебраические основы циклических кодов, принципы обнаружения и исправления ошибок. Приведены примеры построения наиболее широко используемых циклических кодов, а именно: циклических кодов БЧХ и Рида–Соломона. Особое внимание уделено алгоритмам их декодирования.

Ввиду ограниченного объёма учебного пособия в нем не нашли отражения такие важные вопросы, как оценка эффективности применения циклических кодов в системах передачи данных с учетом распределения ошибок в реальных дискретных каналах. Вместе с тем, изучив основы циклических кодов, будущие специалисты в области телекоммуникаций смогут успешно справиться с такими задачами.

Контрольные вопросы

1. Что такое группа? Какие виды групп существуют?
2. Что такое циклический код?
3. Назовите основные свойства циклических кодов.
4. Какие из образующих смежных классов могут составить циклический код?
5. На чём основывается принцип обнаружения ошибок?
6. В чём состоит принцип исправления ошибок?
7. Опишите процесс исправления одиночных ошибок декодирующим устройством циклического (15, 11)- кода?
8. В чём недостаток схем декодирующего устройства?
9. Что представляют из себя коды БЧХ?
10. В чём состоит принцип исправления ошибок кодами БЧХ?
11. Опишите работу схемы декодера кода Рида–Соломона, исправляющего однократные ошибки.
12. Опишите принцип работы обобщающей схемы декодера кода РС, исправляющего часть ошибок более высокой кратности.
13. Дайте определение:
   – синдромного многочлена,
   – многочлена локаторов ошибок,
   – многочлена значений ошибок.
14. Поясните назначение и формы ключевого уравнения.
15. Дайте характеристику:
   –метода быстрого декодирования кодов БЧХ,
   –алгоритма Питерсона,
   –алгоритма Берлекэмпа–Месси,
   –алгоритма Евклида (Сугияма, Касахара, Хирасава и Намекава),
   –алгоритма Форни.

Список литературы

1. Блейхут, Р. Теория и практика кодов, контролирующих ошибки/ Р. Блейхут; пер. с англ. М.: Мир, 1986.– 576 с.
2. Кларк, Дж.К. Кодирование с исправлением ошибок в системах цифровой связи / Дж.К. мл. Кларк, Дж. Б. Кейн ; пер. с англ. – М.: Радио и связь, 1987. – 392 с.
3. Колесник, В. Д. Декодирование циклических кодов / В.Д. Колесник, Е. Т. Мирончиков.– М.: Связь, 1968. – 251 с.
4. Касами, Т. Теория кодирования/ Касами Т., Токура Н., Ивадари Е., Инагаки Я.; пер. с япон. – М.: Мир, 1978. – 576 с.
5. Рид, И. Полиномиальные коды над некоторыми конечными полями/ И. Рид, Г. Соломон. Кибернетический сборник. – М. – 1963. – Вып. 7. – с. 74 – 79.
6. Берлекэмп, Э. Алгебраическая теория кодирования/ Э. Берлекэмп ; пер. с англ. – М.: Мир, 1971. – 478 с.
7. Питерсон, У. Коды, исправляющие ошибки/У. Питерсон; пер. с англ. – М.: Мир, 1964. – 338 с.
8. Курош, А. Г. Курс высшей алгебры / А.Г. Курош – М.: Наука, 1975. – 431 с.

9. Гантмахер, Ф. Р. Теория матриц /Ф.Р. Гантмахер – М.: Наука, 1967. – 575 с.
10. Морелос-Сарагоса, Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение/Р. Морелос-Сарагоса ; пер. с англ. – М.: Техносфера, 2006. – 319 с.
11. Мак-Вильямс, Ф. Дж.Теория кодов, исправляющих ошибки /Ф.Дж. Мак-Вильямс, Н. Дж. Слоэн ; пер. с англ. – М.: Связь, 1979. – 743 с.
12. Стародубцев, В.Г. Помехоустойчивые коды в телекоммуникационных и информационных системах/ В. Г. Стародубцев, О. А. Павлов. Вып. 1. – СПб.: ВКА им. А. Ф. Можайского, 2003. – 255 с.
13. Охорзин, В. М. Построение каскадных кодов на основе кодов Рида-Соломона и Боуза – Чоудхури – Хоквингема / В. М. Охорзин, Д.С. Кукунин , М. С. Новодворский; – СПб.: СПб ГУТ им. проф. М. А. Бонч-Бруевича, 2004.
14. Когновицкий, О. С. Теория помехоустойчивого кодирования: практикум/ О. С. Когновицкий, В.М. Охорзин; – СПб.: СПбГУТ им. проф. М. А. Бонч-Бруевича, 2013. – 72 с.
15.Когновицкий, О. С. Основы циклических кодов. Учебное пособие/ О.С. Когновицкий ;– Ленинград: ЛЭИС им. проф. М. А. Бонч-Бруевича, 1990. – 64 с.
16. Финк, Л. М. Теория передачи дискретных сообщений /Л.М. Финк; – М.: Советское радио, 1963. – 576 с.
17. Деев, В. В. Методы модуляции и кодирования в современных системах связи/ В.В. Деев; – СПб.: Наука, 2007. – 266 с.
18. Когновицкий, О. С. Двойственный базис и его применение в телекоммуникациях/ О.С. Когновицкий ;– СПб.: Линк, 2009. – 423 с.
19. Охорзин, В.М. Циклические коды: практикум/ В.М. Охорзин; – СПб. : СПбГУТ им. проф. М. А. Бонч-Бруевича, 2010. – 56 с.

Когновицкий Олег Станиславович,
Охорзин Виктор Михайлович

ТЕОРИЯ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

ЧАСТЬ 1
ЦИКЛИЧЕСКИЕ КОДЫ

УЧЕБНОЕ ПОСОБИЕ

Редактор ***Л.А. Медведева***

План 2013 г., п. 13

Подписано к печати
Объем 5 усл. печ. л. Тираж 100 экз. Зак.
Издательство СПбГУТ.
191186 СПб, наб. р. Мойки, 61