МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

Национальный технический университет
«Харьковский политехнический институт»

В.А. Крылова

ПОМЕХОУСТОЙЧИВОЕ КОДИРОВАНИЕ. МЕТОДЫ И АЛГОРИТМЫ ЦИКЛИЧЕСКИХ БЧХ КОДОВ

УЧЕБНОЕ ПОСОБИЕ

Утверждено
редакционно-издательским
советом университета,
протокол № от .12.16 г.

Харьков
НТУ «ХПИ»
2016

ББК 32.988-5
УДК 681.5 (004)
К 19

Рецензенты:

М.А. Мирошник, д-р техн. наук, профессор, Украинская государственная академия железнодорожного транспорта, г. Харьков
О.С. Шкиль, канд. техн. наук, доцент, Харьковский национальный университет радиоэлектроники, г. Харьков

У посібнику викладено основи теорії лінійних кодів та дослідження циклічних двійкових БЧХ кодів і недвійкових кодів Рида-Соломона. Розглянуті основні методи та алгоритми кодування/декодування циклічних кодів в часовій області. Приведені процедури синтезу БЧХ кодів, засновані на спектральних методах. Теоретичні дослідження супроводжуються великою кількістю прикладів та задач.

Призначено для студентів технічних ВНЗ, а також може бути корисним для викладачів ВНЗ та інженерно-технічних працівників.

КрыловаВ.А.
К 19 Помехоустойчивое кодирование. Методы и алгоритмы циклических БЧХ кодов: учеб. пособ./ В.А. Крылова – Харьков: НТУ «ХПИ», 2016. – 200 с. – На рус.яз.

ISBN

В пособии изложены основы теории линейных кодов, а также исследования циклических двоичных БЧХ кодов и недвоичных кодов Рида-Соломона. Рассмотрены основные методы и алгоритмы кодирования/декодирования циклических кодов во временной области. Представлены процедуры синтеза БЧХ кодов, основанные на спектральных методах. Теоретические исследования сопровождаются большим числом примеров и задач.

Предназначено для студентов технических вузов, а также может быть полезным для преподавателей вузов и инженерно-технических работников.

Ил. 21.Библиогр. 15 наим.

УДК 681.5 (004)
ББК 32.988-5

ISBNВ 978-617-687-086-9

ВВЕДЕНИЕ

Основным средством обеспечения высокой помехоустойчивости сложной системы является введение избыточности, необходимой для обнаружения и исправления ошибок, возникающих при работе системы связи. Теоретической базой эффективного использования вводимой избыточности является теория кодирования.

Теория кодирования основана на использовании глубокого аппарата современных абстрактных разделов математики и в первую очередь алгебры. Поиск хороших кодов, контролирующих ошибки связан с мощными структурами современной алгебры. Многие найденные коды, в том числе циклические БЧХ коды основаны на структурах колец многочленов и полей Галуа. Кроме того, эти алгебраические понятия и методы являются необходимым рабочим инструментом для конструирования кодеров и декодеров. Поэтому 1-я глава книги посвящена изложению теории поля, арифметики полей Галуа и построению расширенных конечных полей.

С практической точки зрения существенные ограничения на все схемы кодирования и декодирования дискретных данных накладываются сложностью (и стоимостью) декодера. Новый подход к решению данной проблемы был найден в работах Рида и Соломона (1960 г.), Боуза и Чоудхури (1960 г.), Горенстйна, Цирлера и Питерсона (1961). Выбрав в качестве алфавита кода элементы расширенного поля Галуа, удалось свести задачу к решению алгебраического уравнения, корни которого определяют местоположение ошибок. Вычислительная сложность реализации такой процедуры на порядок меньше вычислительной сложности непосредственного декодирования с помощью исчерпывающего перебора. В данной книге представлены методы и алгоритмы декодирования циклических кодов, позволяющие строить алгебраические декодеры.

Значительное место в книге отведено рассмотрению методов помехоустойчивого кодирования, основанных на алгоритмах кодирования

и декодирования важнейших кодовых схем, таких как циклические коды, БЧХ коды, коды Рида-Соломона. Материал книги сосредоточен на основных алгоритмах для анализа и реализации данных кодов. Основные идеи декодирования описываются с помощью простых иллюстративных примеров.

Настоящее учебное пособие состоит из пяти глав, в которых последовательно рассматриваются основополагающие вопросы в области построения и декодирования наиболее перспективных корректирующих кодов – БЧХ и Рида-Соломона кодов. Это предоставляет возможность использовать приведенный материал не только в учебных, но и в научных приложениях.

В первой главе рассматриваются минимальные сведения, из теории конечных полей, необходимые для понимания следующих глав. Представлены структуры и арифметика полей Галуа, которые основаны на кольцах многочленов.

Во второй главе описаны подкласс линейных кодов – циклические коды. С помощью полиномиального представления циклических кодов описаны основные алгоритма кодирования и декодирования.

Главы 3 и 4 посвящены изложению кодов Боуза-Чоудхури-Хоквингема (БЧХ) и Рида-Соломона (РС) во временном представлении. Особенно уделено внимание методам и алгоритмам декодирования данных кодов. Описаны алгоритмы Питерсона-Горенстейна-Цирлера, Еклида и Берлекэмпа-Месси для решения ключевого уравнения, корни которого определяют местоположение ошибок. Также изложены методы для исправления стираний и ошибок в кодовой последовательности РС и БЧХ кодов.

Пятая глава посвящена описанию процедурам кодирования/декодирования БЧХ и РС кодов, основанных на спектральных методах. Показано, что существуют преобразования Фурье на векторном пространстве последовательностей длины n над полем Галуа $GF(p)$, так как циклические коды определяются как коды, в которых некоторые спектральные компоненты слов равны нулю. Описаны основные методы декодирования, в том числе с исправлением ошибок и стираний, позволяющие работать в частотной области.

Данная книга, сопровождается большим числом примеров и задач, поэтому будет полезной и интересной не только для специалистов, работающих в области теории кодирования, но и для инженеров, занимающихся вопросами ее применения.

ГЛАВА 1.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ТЕОРИИ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

1.1 Введение в теорию групп, колец и полей

В основу построения известных избыточных кодов является их алгебраическая структура, позволяющая изучать различные свойства кода, а также обеспечивать возможность практической реализации кодирующих и декодирующих устройств. Арифметические системы, в которых определены одна или несколько операций (сложение или умножение) являются важнейшими объектами в современной алгебре. Примерами арифметических систем являются:

1. Абелева группа – множество математических объектов, которые можно «складывать» и «вычитать».
2. Кольцо – множество математических объектов, которые можно «складывать», «вычитать» и «умножать».
3. Поле – множество математических объектов, которые можно «складывать», «вычитать», «умножать» и «делить».

Прежде чем переходить к формальным понятиям, изложим краткие сведения из теории чисел.

Если a, b и c – целые числа и $a=bc$, то говорят что a делится на b или что b является делителем a. Наибольшим общим делителем (НОД) двух чисел называется наибольшее целое положительное число, являющееся делителем обоих чисел. Два числа взаимно просты, если их НОД равен 1. Для любой пары целых чисел a и b существует единственная пара чисел q (частное) и r (остаток), таких что $a=qb+r$, где $0<r<|b|$.

Если два числа a и b дают при делении на число p один и тот же остаток, то говорят, что числа сравнимы по модулю p. Такое сравнение записывается в виде

$$a = b \pmod{p}.$$

Все числа сравнимы по модулю p, образуют *класс вычетов* по модулю p. Любое число в классе называется *вычетом* по модулю p. Всем числам класса вычетов соответствует один и тот же остаток. Так как всего имеется p остатков: 0, 1, 2, …. $p-1$, то существует p различных классов

вычетов. Вычет, равный самому остатку, называется наименьшим неотрицательным вычетом. Выбрав из каждого класса вычетов по модулю p по одному вычету, получим полную систему вычетов по модулю p. Обычно в качестве полной системы вычетов используют наименьшие неотрицательные вычеты: 0, 1, 2 … $p-1$.

Например. Пусть $p = 4$, тогда числа 1, 5, 9, 13, 17, 21 и т.д. образуют класс вычетов по модулю 4. Наименьший вычет в этом классе равен 1, а полную систему вычетов по модулю 4 образуют числа {0, 1, 2, 3}.

С учетом определенного выше понятия сравнения чисел по модулю p введены операции сложения и умножения чисел по модулю целого числа p. При этом результат равен наименьшему вычету класса, к которому принадлежит число, получаемое в результате применения операции сложения (умножения) чисел. Т.е. результат равен остатку от деления числа, получаемого при обычном сложении (умножении) на модуль p.

Например. При $p = 5$ таблица сложения и умножения чисел по модулю 5 выглядит следующим образом

+	**0**	**1**	**2**	**3**	**4**
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	**0**	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Группы

Алгебраическая система $\{G, *\}$, образованная непустым множеством G и операцией $*$, определенной для любых двух элементов a и b из G называется *группой*, если выполнены следующие аксиомы:

1. Замкнутость – для каждой пары a и b из множества G элемент $c=a*b$ также принадлежит этому множеству.

2. Ассоциативность – для всех a, b и c из множества G

$$a*(b*c) = (a*b)*c.$$

3. Существование единицы – во множестве существует единичный элемент l, т.е. такой, что $a*l=l*a=a$.

4. Существование обратных элементов – для любого элемента a из множества G существует обратный элемент a^{-1}, такой что $a* a^{-1}= a^{-1}*a = l$

Группа называется *коммутативной* или *абелевой*, если кроме аксиом 1-4 выполняется следующая аксиома коммутативности: для двух произвольных элементов a и b из множества G справедливо $a*b = b*a$.

Например, совокупность всех действительных чисел образует группу относительно операции обычного сложения. Единичным элементом группы (нулем) является число 0. Полная система вычетов по модулю 6 (G={0,1,2,3,4,5})является группой с операцией сложения по модулю 6. Единичным элементом этой группы является 0, а обратный элемент находится из равенства$a+(-a) = 0(\text{mod } 6)$. Так, если $a = 2$, то $(-a) = 4$ и т.д.

Порядок любой подгруппы конечной группы является делителем порядка группы. Тогда любую конечную группу можно однозначно разложить на смежные классы. Если M – порядок конечной группы, N – порядок ее подгруппы, $M = N*J$, то элементы группы могут быть разложены в двумерную таблицу размером J строк на N столбцов, где первая строка есть собственно элементы подгруппы, а остальные строчки есть ее смежные классы.

Кольца

Кольцом R называется множество элементов, на котором определены две операции: сложение и умножение, и выполняются следующие аксиомы:

1. Множество R является аддитивной абелевой группой.

2. Для любых двух элементов a и b из Rопределено их произведение: $a*b = c$ принадлежит множеству R (замкнутость операции умножения).

3. Для любых трех элементов a, b и c из множества Rвыполняется ассоциативный закон, т.е. $a*(b*c)=(a*b)*c$и$a+(b+c) = (a+b)+c$.

4. Для любых трех элементов a, b и c из множества Rвыполняется дистрибутивный закон, т.е. справедливы равенства: $a*(b+c) = a*b+a*c$и $(b+c)*a =b*a+c*a$

Если в кольце существует единичный элемент относительно операции умножения, то это кольцо называется кольцом с единицей.

Например, все целые положительные и отрицательные числа и нуль образуют коммутативное кольцо с единицей относительно обычных операций сложения и умножения. Множество всех вещественных чисел образуют коммутативное кольцо с единицей относительно обычных операций сложения и умножения. Каждый ненулевой элемент кольца является единицей.

Поля

Полем называется множество с двумя определенными операциями – сложением и умножением, при этом каждый ненулевой элемент имеет мультипликативный обратный элемент (т.е. обратный по умножению). Другими словам, полем называется множество, которое является аддитивной абелевой группой, ненулевые же элементы этого множества образуют мультипликативную абелевую группу, и выполняется закон дистрибутивности. По аналогии с группами число элементов поля называется порядком поля. Поля, порядки которых конечны, называются *конечными полями*. Конечные поля имеют наибольшее значение в теории кодирования.

Любое поле обладает следующими свойствами:

1. Множество образует абелеву группу по сложению.

2. Поле замкнуто относительно умножения, и множеству ненулевых элементов образует абелеву группу по умножению.

3. Закон дистрибутивности $(a + b)\, c = ac + bc$.

Конечное поле является замкнутым относительно двух операций сложения и умножения. Замкнутость означает, что результаты сложения и умножения также принадлежат этой группе.

Единичный элемент относительно сложения обозначается через 0 и называется нулем, аддитивный обратный элементу a элемент – через «$–a$». Единичный элемент относительно умножения обозначают через 1 и называют единицей, мультипликативный обратный к элементу a элемент – через «a^{-1}». При этом $a+0=a$ и $a \cdot 1=a$для любого элемента поля. Под вычитанием $(a - b)$ понимается $a +(-b)$, под делением (a/b) понимается $a*b^{-1}$.

При этом для аддитивного и мультипликативного элементов выполняются выражения:

$$\alpha +(-\alpha) = 0,\ \alpha * \alpha^{-1} = 1.$$

Множество всех действительных чисел образует поле. Существует также поле комплексных чисел, поле рациональных чисел, но не может быть поле целых чисел, поскольку обратные элементы по умножению, кроме единицы, не являлись бы целыми. Все эти поля содержат бесконечное множество элементов. В теории помехоустойчивого кодирования используются поля, которые содержат конечное число элементов.

Множество чисел (0, 1, 2, …. p −1), где p – простое число, образует конечное поле, в котором сложение и умножение производятся по модулю p. Поле с p элементами, если оно существует, называется *конечным полем* или *полем Галуа* и обозначается $GF(p)$. Любое конечное поле можно сконструировать как множество остатков от деления натуральных чисел на $p\{0, 1, 2, \ldots. p-1\}$ с операциями сложения и умножения по модулю p. Ниже приведены примеры конечных полей $GF(p)$.

При $p = 2$ наименьшее двоичное поле $GF(2)$ содержит нулевой элемент и единичный элемент $\{0, 1\}$. Операции сложения и умножения в конечном поле $GF(2)$

+	**0**	**1**
0	0	1
1	1	0

×	**0**	**1**
0	0	0
1	0	1

Пример операций сложения и умножения в конечном поле $GF(3)$, состоящем из $p = 3$ элементов $\{0, 1, 2\}$.В этом конечном поле элементы «1» и «2» обратные друг другу по сложению, а «2» обратно «2» по умножению.

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

×	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

Пример операций сложения и умножения в конечном поле $GF(5)$, состоящем из $p = 5$ элементов $\{0, 1, 2, 3, 4\}$. В поле из пяти элементов по сложению обратные друг другу «1» и «4», а также «2» и «3», а по умножению обратные друг другу «2» и «3», а также «4» и «4».

+	**0**	**1**	**2**	**3**	**4**
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	**0**	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Конечные поля существуют не при любом числе элементов, а только в том случае, если их количество – простое число *p*или его степень p^m, где m – целое число. Поле $GF(p)$ называется *простым конечным полем*, а поле $GF(p^m)$ называется *расширенным конечным полем* $GF(p)$.

Например, поле $GF(4)$, таблицы сложения и умножения которого построены по модулю 4, не является конечным полем, так как в таблице умножения не каждый ненулевой элемент имеет обратный элемент.

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

×	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Однако, расширенное поле $GF(4)=GF(2^2)$, содержит подполе $GF(2)$, является конечным полем. И умножение в расширенном поле $GF(4)$ не является умножением по модулю 4 и сложение не является по модулю 4, т.к. операции сложения и умножения в расширенных полях сложнее.

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

×	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Таким образом, как показывают примеры арифметику полей Галуа $GF(2)$ и $GF(3)$ можно описать как сложение и умножение по модулю 2 и 3 соответственно, а арифметику в поле $GF(4)$ так описать нельзя. Таким образом, поле $GF(p)$ является полем тогда и только тогда, когда p – равно простому числу. Операции в расширенных конечных полях (порядка $q = p^m$, $m>1$), изучаемых позднее − несколько сложнее, чем сложение и умножение по модулю q.

1.2 Правила арифметики формальных полиномов

В теории помехоустойчивого кодирования для представления кодовых последовательностей используется полиномиальное представление двоичных векторов

Кодовый полином или многочленом над полем $GF(p)$, отвечающий кодовому вектору $f = (f_{n-1}, f_{n-2}, \ldots, f_1, f_0)$ называется математическое выражение

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \ldots + f_1x + f_0, \qquad (1.1)$$

где символ x − формальная переменная, служащая для указания своей степенью позиции того или иного кодового символа как коэффициента полинома, коэффициенты $f_{n-1}, f_{n-2}, \ldots, f_0$ принадлежит полю $GF(p)$, а индексы и показатели степеней являются целыми числами.

Например, полином $f(x) = x^6 + x^4 + x^2 + 1$, коэффициенты которого принадлежат полю $GF(2)$ для двоичного кода, соответствует кодовый вектор $f = 1010101$, где единицы стоят на позициях 6, 4, 2, 0, а все остальные элементы равны 0. Над полиномами в поле двоичных символов $GF(2)$ можно производить различные арифметические операции.

Нулевым многочленом называется многочлен $f(x)=0$. *Приведенным многочленом* называется многочлен, старший коэффициент f_{n-1} которого равен 1. Два многочлена равны, если равны все их коэффициенты. *Степенью* ненулевого многочлена $f(x)$ называется индекс старшего коэффициента f_{n-1}, степень многочлена обозначается через $\deg f(x)$.

Например, степень многочлена $f(x)=x^7+x^5+x^2+1$ равна $\deg f(x)=7$.

Для двух многочленов $a(x)$ и $b(x)$ из поля $GF(p)$ выполняются арифметические операции:

−*сложение* двух многочленов $a(x)$ и $b(x)$из поля $GF(p)$

$$a(x) + b(x) = \sum_{i=0}^{\infty} (a_i + b_i)x^i, \qquad (1.2)$$

где «+» − операция сложения по модулю 2 коэффициентов при одинаковых степенях x.

Таблица 1.1 –Сложение по модулю 2

a	***b***	***a + b***
0	0	0
0	1	1
1	0	1
1	1	0

Например, $a(x)=x^4+x^2+x$(a = 10110) и $b(x)=x^6+x^3+x^2+1$ (b = 1001101) из поля $GF(2)$:

$$\begin{array}{r} 10110 \\ +\ 1001101 \\ \hline 1011011 \end{array}$$

$a(x)+b(x)=(x^4+x^2+x)+(x^6+x^3+x^2+1)=x^6+x^4+x^3+(1+1)x^2+x+1 = x^6+x^4+x^3+x+1$,

где сумма коэффициентов находится по таблице сложения элементов конечного поля $GF(2)$ (табл. 1.1).

–*умножение* двух многочленов $a(x)$ и $b(x)$ из поля $GF(p)$

$$a(x)b(x) = \sum_{i=0} (\sum_{j=0}^{i} a_j b_{i-j}) x^i \ . \qquad (1.3)$$

т.е. произведение получается по обычному правилу перемножения степенных функций, однако получаемые коэффициенты при данной степени x складываются по модулю 2.

Например, $a(x)=x^3+x^2+1$(a = 1101) и $b(x)=x^2+x+1$(b = 111) из поля $GF(2)$, $a(x) \cdot b(x)=(x^3+x^2+1) \cdot (x^2+x+1)$.

$$\begin{array}{r} \text{x}1101 \\ 111 \\ \hline 1101 \\ 1101 \\ 1101 \\ \hline 100011 \end{array}$$

$a(x) \cdot b(x)=(x^3+x^2+1) \cdot (x^2+x+1) = x^5+x^4+x^2+x^4+x^3+x+x^3+x^2+1 = x^5+$

$+(1+1)x^4++(1+1)x^3+(1+1)x^2+x+1 = x^5 + x + 1$,

где степень произведения равна сумме степеней множителей.

− деление двух многочленов $a(x)$ и $b(x)$ из поля $GF(p)$, т.е. многочлен $a(x)$может делиться на многочлен $b(x)$, если существует такой многочлен что $r(x)=a(x)\cdot b(x)$.При этом деление полиномов (многочленов) не всегда возможно даже на ненулевой многочлен. Например, если степень делимого меньше степени делителя. Деление полиномов производится по правилам деления степенных функций, при этом операция вычитания заменяется суммированием по модулю два.

Например, $a(x) = x^4+x^2+x+1 (a = 10111)$ и $b(x) = x+1$ $(b = 11)$ из поля $GF(2)$, найдем частное от деления $a(x)$ на $b(x)$.

x^4+x^2+x+1	$x+1$
x^4+x^3	x^3+x^2+1
x^3+x^2+x+1	
x^3+x^2	
$x+1$	
$x+1$	
0	

10111	11
11	1101
1111	
11	
11	
11	
0	

При сложении по модулю 2 сумма двух единиц (двух элементов полинома с одинаковыми степенями) будет равна 0, а не привычным в десятичной системе счисления двум. Операции вычитания и сложения по модулю 2 совпадают.

Основные свойства делимости полиномов:

• если $a(x)$ одновременно делится на $b(x)$ и делит $b(x)$, то $a(x)=\alpha b(x)$, где α – элемент поля $GF(p)$;

• если каждый из многочленов $a(x)$ и $b(x)$ делится на $r(x)$, то их сумма $a(x)+b(x)$ и разность $a(x) - b(x)$ делится на $r(x)$;

• для любой пары многочленов $a(x)$ и $b(x)$ существует единственная пара многочленов $Q(x)$ (частное) и $r(x)$ (остаток) таких, что

$$a(x) = Q(x)b(x) + r(x)$$

причем степень $r(x)$ меньше степени $b(x)$.

•*наибольший общий делитель* (НОД) двух многочленов $a(x)$ и $b(x)$ определяется как приведенный многочлен наибольшей степени, делящий одновременно оба из них. НОД обозначается $d(x) = НОД[a(x), b(x)]$. Если НОД двух многочленов равен 1, то они называются *взаимно простыми*.

1.3 Структура и арифметика полей Галуа. Расширенные конечные поля

В предыдущем подразделе мы ввели определение поля, но не указали процедур построения полей Галуа, а именно их таблиц сложения и умножения. Так как идеи теории помехоустойчивого кодирования основаны на арифметических системах полей Галуа, то далее мы будем изучать поля Галуа основанные на кольцах многочленов.

Многочлен $f(x)$, делящийся только на многочленα·$f(x)$ или α, где α – произвольный ненулевой элемент поля $GF(p)$, называется *неприводимым многочленом*. Приведенный неприводимый многочлен называется простым многочленом. В конечном поле $GF(2)$ неприводимый многочлен $f(x)$ может делиться только на себя $f(x)$ и на 1 (α=1). Аналогом неприводимого полинома является простое число в поле вещественных чисел.

Если при делении полиномов $a(x)$ и $b(x)$ из $GF(p)$ на $f(x)$ получаются одинаковые остатки, то многочлены $a(x)$ и $b(x)$ сравнимы между собой по модулю многочлена $f(x)$ из $GF(p)$ или $a(x)=b(x)(\mathrm{mod}\ f(x))$.

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

$$f(x)=2x^5 + x^4 + x^2 + 2.$$

Тогда с учетом таблиц сложения и умножения в поле $GF(3)$ получим:

$f(0)=2\cdot0^5 + 0^4 + 0^2 + 2=2,$

$f(1)=2\cdot1^5 + 1^4 + 1^2 + 2=0,$

$f(2)=2\cdot2^5 + 2^4 + 2^2 + 2=2.$

Аналогично можно вычислить значения многочлена $GF(p)$ в расширении поля $GF(p^m)$. Пусть, например, имеется многочлен над полем $GF(2)$ $f(x)=x^3 + x + 1$. Тогда для элементов расширенного поля $GF(4)$ получим:

$f(0)=0^3 + 0 + 1=1$, $f(1)=1^3 + 1 + 1=1,$

$f(2)=2^3 + 2 + 1=2$, $f(3)=3^3 + 3 + 1=3.$

Если $f(\alpha)=0$, то элемент α называется *корнем многочлена* $f(x)$ или корнем уравнения $f(x)=0$. При этом многочлен не обязательно имеет корни в своем собственном поле. Многочлен $f(x)=x^3 + x + 1$ не имеет корней в $GF(2)$, а также в $GF(4)$. Неприводимый многочлен нельзя разложить на множители, используя только многочлены с коэффициентами из поля $GF(p)$. Это означает, что $f(x)$ не имеет корней в поле $GF(p)$.

Например, полином $f(x)=x^2 + 1$ раскладывается на множители (приводим) в поле $GF(2)$ $x^2 + 1=(x + 1)(x + 1)$ и при этом корнем данного многочлена является элемент из поля $GF(2)$ равный 1 так как $f(1)=0$. А многочлен $f(x)=x^2 + x + 1$ неприводим в поле $GF(2)$, т.е. не раскладывается на множители и не имеет корней в поле $GF(2)$.

Конечные поля можно построить из колец многочленов таким же образом, каким были построены поля из кольца целых чисел. Для произвольного приведенного многочлена $f(x)$ ненулевой степени над полем $GF(p)$ *кольцо многочленов* по модулю $f(x)$ называется множество всех многочленов над полем $GF(p)$ степень которых не превосходит степени многочлена $f(x)$, с операциями сложения и умножения многочленов по модулю $f(x)$.

Например, кольцо многочленов для приведенного полинома $f(x)=x^3+1$ с коэффициентами из поля $GF(2)$ состоит из многочленов по модулю $f(x)$ $\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$. Умножение в этом кольце выполняется следующим образом:

$(x^2+1)\cdot(x^2) = x^4 + 1$,приводим по модулю x^3+1, получим $(x^2+1)\cdot(x^2) = x^2+x$.

В предыдущем подразделе было показано, что конечное поле или поле Галуа $GF(p)$ из p элементов существует, если p – простое число.

Кольцо многочленов по модулю приведенного многочлена $f(x)$ является полем тогда и только тогда, когда многочлен $f(x)$ неприводим в поле $GF(p)$, т.е. $f(x)$ – простой многочлен.

Если над полем $GF(p)$ найден неприводимый многочлен $f(x)$ степени m, то можно построить расширенной поле Галуа, содержащее p^m элементов. Элементами α расширенного поля $GF(p^m)$ могут быть все многочлены степени m – 1 или меньше, коэффициенты которых лежат в простом поле $GF(p)$. Число p^m называется порядком расширенного поля и определяет количество различных многочленов. При этом используемый полином $f(x)$ должен быть неприводим в поле $GF(p)$, т.е. его нельзя

разложить на множители, используя только многочлены с коэффициентами из $GF(p)$. Это означает также, что $f(x)$ не имеет корней в поле $GF(p)$. К сожалению, регулярных методов неприводимых многочленов не существует, как правило, они определяются перебором (табл. 1.2).

Правила сложения и умножения полиномов – элементов расширенного конечного поля получаются из обычных правил сложения и умножения полиномов с последующим приведением результата по модулю некоторого специального многочлена $f(x)$ степени m. Такое приведение эквивалентно делению многочлена результата на $f(x)$ и использованию только остатка.

Таблица 1.2 – Неприводимые многочлены

Степень	Простые многочлены
2	x^2+x+1
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^3+x^2+1$
9	x^9+x^4+1
10	$x^{10}+x^3+1$
11	$x^{11}+x^2+1$
12	$x^{12}+x^6+x^4+x+1$
13	$x^{13}+x^4+x^3+x+1$
14	$x^{14}+x^{10}+x^6+x+1$
15	$x^{15}+x+1$
16	$x^{16}+x^{12}+x^3+x+1$
17	$x^{17}+x^3+1$
18	$x^{18}+x^7+1$
19	$x^{19}+x^5+x^2+x+1$
20	$x^{20}+x^3+1$
21	$x^{21}+x^2+1$
22	$x^{22}+x+1$
23	$x^{23}+x^5+1$
24	$x^{24}+x^7+x^2+x+1$
25	$x^{25}+x^3+1$
26	$x^{26}+x^6+x^2+x+1$
27	$x^{27}+x^5+x^2+x+1$
28	$x^{28}+x^3+1$

В качестве примера построим конечное поле $GF(2)$ и его расширенное поле $GF(2^2)$, используя неприводимы многочлен $f(x) = x^2+x+1$. Элементами конечного поля $GF(2)$ являются 0 и 1, а элементами расширенного поля $GF(2^2)$ являются элементы – многочлены с коэффициентами из поля $GF(2)$ {0, 1, x, x+1} (табл. 1.3).

Таблица 1.3 – Элементы расширенного поля $GF(2^2)$

Многочлены	Двоичные обозначения	Целочисленные обозначения	Степенные обозначения
0	00	0	x^0
1	01	1	x^1
X	10	2	x^2
x+1	11	3	x^3

В поле $GF(2)$ обычные операции умножения (на 0 и 1) и деления (на 1) не выводят результат за пределы множества 0, 1. Однако при сложении и вычитании элементов это требование может уже не выполняться 1+1=2, −1+(−1)=−2. Любое расширенное поле $GF(2^m)$ является полем характеристики 2, вследствие чего при вычислении коэффициентов полиномов, представляющих элементы $GF(p^m)$, используется арифметика по модулю два. Таким образом, свойства конечного поля будут соблюдаться, если в качестве операции сложения использовать суммирование по модулю 2 (mod 2)

$$0 + 0 = 0; \quad 0 + 1 = 1; \quad 1 + 0 = 1; \quad 1 + 1 = 0.$$

При построении таблиц сложения и умножения для элементов расширенного поля $GF(2^2)$ необходимо учитывать, что при перемножении полиномов получаются полиномы степени 2, не принадлежащих множеству элементов $GF(2^2)$. Например

$$(x+1)\cdot(x+1) = x^2+1.$$

Поэтому полученный результат необходимо привести по модулю x^2+x+1, т.е. необходимо найти остаток от деления полинома x^2+1 на x^2+x+1, тогда получим: $x^2+x+1=(x^2+1)\cdot 1+x$, где 1 – частное от деления, а x – остаток от деления. Таким образом, результат умножения элементов расширенного поля $GF(2^2)$ равен $(x+1)\cdot(x+1) = x$.

В таблице 1.4 приведены результаты сложения и умножения элементов расширенного поля $GF(2^2)$.

Таблица 1.4 – Сложение и умножение в поле $GF(2^2)$

+	**0**	**1**	***x***	***x*+1**
0	0	1	***x***	***x*+1**
1	1	0	***x*+1**	***x***
x	***x***	***x*+1**	0	1
***x*+1**	***x*+1**	***x***	**1**	0

×	**0**	**1**	***x***	***x*+1**
0	0	0	0	0
1	0	1	***x***	***x*+1**
x	0	***x***	***x*+1**	1
***x*+1**	0	***x*+1**	**1**	***x***

Таким образом, конечные поля существуют только для порядков $q=p^m$ (p – простое, m – натуральное). Простое поле порядка p, $GF(p)$, можно трактовать как множество $\{0, 1, \ldots, p-1\}$остатков от деления целых чисел на pс операциями сложения и умножения по модулю p. Расширенное поле $GF(q)$порядка $q=p^m$ при $m>1$ можно ассоциировать с множеством остатков от деления полиномов над $GF(p)$ на некоторый неприводимый полином $f(x)$ степени mс операциями сложения и умножения по модулю $f(x)$. Другими словами, поле $GF(q)$ можно представить всеми полиномами над простым полем $GF(p)$ степени не выше $m-1$ с обычным полиномиальным сложением. Умножение выполняется в два шага – сначала как обычное умножение полиномов, но с удержанием в качестве конечного итога лишь остатка от деления полученного произведения на неприводимый полином.

1.4 Примитивный элемент. Мультипликативный порядок элементов поля

Особым свойством конечных полей является связь между собой всех ненулевых элементов и возможность выражения каждого из них через один элемент α, как некоторую целую степень этого элемента. Из таблицы 1.2 видно, что элементы поля $GF(4)$ за исключением нуля могут быть представлены в виде степени элемента x:

$0, 1, x, x+1 \rightarrow 0, x^0, x^1, x^2$.

Примитивным элементом расширенного поля $GF(q)$называется такой элемент α, что все элементы поля, за исключением нуля, могут быть представлены в виде степени элемента α.

Например, в поле $GF(5)$ $2^1=2$, $2^2=4$, $2^3=3$, $2^4=1$, таким образом, элемент 2 является примитивным элементом поля $GF(5)$.

Примитивный элемент полезен при построении полей, так как если один из них найден, то перемножая степени этого примитивного элемента

можно построить таблицу умножения в поле $GF(q)$.Множество $q-1$ ненулевых элементов расширенного поля $GF(q)$ образует *циклическую мультипликативную группу*, т.е. элементы находятся между собой в соотношении 1, α, α^2, α^3, … $\alpha^{q-1}=1$. Так как все элементы принадлежат полю $GF(q)$, то в рассматриваемой последовательности появятся повторения, так что для некоторых l и s $(l > s)$ $\alpha^s = \alpha^l$, а значит $\alpha^{l-s} = 1$.

Минимальное натуральное число d_α для которого $\alpha^{d_\alpha} = 1$ называется *мультипликативным порядком* элемента α. Мультипликативный порядок любого ненулевого элемента поля $GF(q)$ делит $q-1$, т.е. число ненулевых элементов.

Например, элемент 2 поля $GF(7)=\{0, 1, 2, 3, 4, 5, 6\}$ имеет мультипликативный порядок $d_2=3$, так как для него $2^1=2$, $2^2=4$, $\mathbf{2^3=1}$подобно этому найдем остальные мультипликативные порядки

$d_3=6 \rightarrow 2^4=2$, $2^5=4$, $\mathbf{2^6=1;}$

$d_4=3 \rightarrow 2^7=2^1=2$, $2^8=2^2=4$, $\mathbf{2^9=2^3=1}$;

$d_5=6, \rightarrow 2^{10}=2^4=2$, $2^{11}=2^5=4$, $\mathbf{2^{12}=2^6=1}$;

$d_6=3, \rightarrow 2^{13}=2^7=2$, $2^{14}=2^8=4$, $2^{15}=2^3=1$.

Все найденные мультипликативные порядки делят число ненулевых элементов поля $q-1=7-1=6$. При этом элементы 3 и 5 поля $GF(7)$ являются примитивными, тогда как остальные ненулевые элементы непримитивны. Действительно, для элемента 3: $3^1=3$, $3^2=2$, $3^3=6$, $3^4=4$, $3^5=5$, $3^6=3^0=1$. Для непримитивного элемента поля 2 подобные вычисления дают $2^1=2$, $2^2=4$, $2^3=1$, $2^4=2$, $2^5=4$, $2^6=1$, т.е. возведением2 в различные степени можно получить лишь некоторые (но не все) ненулевые элементы $GF(7)$.

В поле $GF(8)$ число ненулевых элементов поля – простое: $q-1=7$, а значит его делители − только числа 1 и 7. Так как единственный элемент мультипликативного порядка 1 – единица поля, все остальные ненулевые элементы имеют максимальный мультипликативный порядок. В поле $GF(8)$ все ненулевые элементы поля за исключением единицы примитивны, так как число ненулевых элементов – простое, а значит, возможные мультипликативные порядки исчерпываются значениями 1 и 7. Тем самым, возведение любого из этих элементов в степень от 0 до 6 генерирует все ненулевые элементы поля 1, α^1, α^2, … $\alpha^{q-1}=\alpha^0=1$, т.е. пробегают все ненулевые элементы поля $GF(q)$.

1.5 Построение расширенных конечных полей $GF(p^m)$ в виде множества многочленов

Построение расширенного конечного поля $GF(p^m)$ в виде таблицы многочленов и степеней примитивного элемента начинается с выбора неприводимого полинома степени m над простым полем $GF(p)$

$$f(x) = f_m x^m + f_{m-1}x^{m-1} + \ldots + f_0. \qquad (1.4)$$

Для m-ой степени элемента x по модулю $f(x)$ имеет место равенство

$$x^m = f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \ldots + f_0, \qquad (1.5)$$

где знак «−» заменяется на «+» так как используется сложение по модулю 2.

Примитивность элемента x позволяет обозначить его как α, после чего получим

$$\alpha^m = f_{m-1}\alpha^{m-1} + f_{m-2}\alpha^{m-2} + \ldots + f_0, \qquad (1.6)$$

откуда $\alpha^{m+1} = \alpha \cdot \alpha^m$.

Например, построим расширенное конечное поле $GF(2^3)=GF(8)$ для простого поля $GF(2) = \{0, 1\}$. Элементами βрасширенного поля $GF(2^3)$ будут являться многочлены, построенные с помощью неприводимого многочлена $p(x) = x^3 + x + 1$. Обозначим примитивный элемент $\alpha=x$, тогда имеем $\alpha^3 = \alpha + 1 \rightarrow x^3 = x + 1$. Вычисли степени элемента α:

$\boldsymbol{\alpha^0} \rightarrow \underline{1}$;

$\boldsymbol{\alpha^1} \rightarrow \underline{x^1}$;

$\boldsymbol{\alpha^2} \rightarrow \underline{x^2}$;

$\boldsymbol{\alpha^3} \rightarrow x^3 = \underline{x + 1}$;

$\boldsymbol{\alpha^4} \rightarrow x^4 = x^3 \cdot x = (x + 1) \cdot x = \underline{x^2 + x}$;

$\boldsymbol{\alpha^5} \rightarrow x^5 = x^4 \cdot x = (x^2 + x) \cdot x = x^3 + x^2 = x + 1 + x^2 = \underline{x^2 + x + 1}$;

$\boldsymbol{\alpha^6} \rightarrow x^6 = x^5 \cdot x = (x^2 + x + 1) \cdot x = x^3 + x^2 + x = x + 1 + x^2 + x = \underline{x^2 + 1}$;

$\boldsymbol{\alpha^7} \rightarrow x^7 = x^6 \cdot x = (x^2 + 1) \cdot x = x^3 + x = x + 1 + x = \underline{1}$.

Представления элементов расширенного конечного поля $GF(2^3)$ заданного неприводимым многочленом $p(x) = x^3 + x + 1$ указаны в таблице 1.5.

Таблица 1.5 – Различные представления элементов поля $GF(2^3)$

Элементы поля	Степень α	Полином	Вектор
0	0	0	000
β_0	α^0	1	001
β_1	α^1	x	010
β_2	α^2	x^2	100
β_3	α^3	$x + 1$	011
β_4	α^4	$x^2 + x$	110
β_5	α^5	$x^2 + x + 1$	111
β_6	α^6	$x^2 + 1$	101

$\beta_7 \rightarrow \alpha^7 = \alpha^0 = 1$.

Представление элементов поля по степеням примитивного элемента α удобно, в частности при умножении элементов друг на друга. Для этого достаточно сложить их степени по модулю $p^m - 1$. Например, для элементов расширенного поля $GF(2^3)$:

$\beta_1 \cdot \beta_3 = \alpha^1 \cdot \alpha^3 = \alpha^{1+3} = \alpha^4$.
$\beta_4 \cdot \beta_5 = \alpha^4 \cdot \alpha^5 = \alpha^{4+5} = \alpha^9$ (9 mod 7) $= \alpha^2$.

Прямое вычисление дает тот же результат, но процедура вычисления трудоемкая.

$\beta_1 \cdot \beta_3 = x \cdot (x + 1) = x^2 + x$.
$\beta_4 \cdot \beta_5 = (x^2 + x) \cdot (x^2 + x + 1) = x^4 + x^3 + x^3 + x^2 + x^2 + x = x^4 + x =$
$= x^2 + x + x = x^2$;

Для расширенного поля $GF(2^3)$ построены таблицы сложения и умножения элементов поля (табл. 1.6 и 1.7).

Таблица 1.6 – Сложение элементов поля $GF(2^3)$

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

Таблица 1.7 – Сложение элементов поля $GF(2^3)$

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

Построим расширенное конечное поле $GF(2^4)$ для конечного поля $GF(2)$ на основе заданного неприводимого многочлена $p(x) = x^4 + x + 1$. Примитивный элемент расширенного поля $GF(2^4)$ является $\alpha = x$, тогда $\alpha^4 = \alpha + 1 \rightarrow x^4 = x + 1$. Найдем все степени ненулевые элементов α, а также соответствующие полиномы, степень которых не превышает m=3:

$\boldsymbol{\alpha^0} \rightarrow 1$;

$\boldsymbol{\alpha^1} \rightarrow \underline{x^1}$;

$\boldsymbol{\alpha^2} \rightarrow \underline{x^2}$;

$\boldsymbol{\alpha^3} \rightarrow \underline{x^3}$;

$\boldsymbol{\alpha^4} \rightarrow x^4 = \underline{x + 1}$;

$\boldsymbol{\alpha^5} \rightarrow x^5 = x^4 \cdot x = (x + 1) \cdot x = \underline{x^2 + x}$;

$\boldsymbol{\alpha^6} \rightarrow x^6 = x^5 \cdot x = (x^2 + x) \cdot x = \underline{x^3 + x^2}$;

$\boldsymbol{\alpha^7} \rightarrow x^7 = x^6 \cdot x = (x^3 + x^2) \cdot x = x^4 + x^3 = \underline{x + 1 + x^3}$;

$\boldsymbol{\alpha^8} \rightarrow x^8 = x^7 \cdot x = (x^3 + x + 1) \cdot x = x^4 + x^2 + x = x + 1 + x^2 + x = \underline{x^2 + 1}$;

$\boldsymbol{\alpha^9} \rightarrow x^9 = x^8 \cdot x = (x^2 + 1) \cdot x = \underline{x^3 + x}$;

$\boldsymbol{\alpha^{10}} \rightarrow x^{10} = x^9 \cdot x = (x^3 + x) \cdot x = x^4 + x^2 = \underline{x + 1 + x^2}$;

$\boldsymbol{\alpha^{11}} \rightarrow x^{11} = x^{10} \cdot x = (x^2 + x + 1) \cdot x = \underline{x^3 + x^2 + x}$;

$\boldsymbol{\alpha^{12}} \rightarrow x^{12} = x^{11} \cdot x = (x^3 + x^2 + x) \cdot x = x^4 + x^3 + x^2 = \underline{x + 1 + x^3 + x^2}$;

$\boldsymbol{\alpha^{13}} \rightarrow x^{13} = x^{12} \cdot x = (x^3 + x^2 + x + 1) \cdot x = x^4 + x^3 + x^2 + x = x + 1 + x^3 + x^2 + x = \underline{x^3 + x^2 + 1}$;

$\boldsymbol{\alpha^{14}} \rightarrow x^{14} = x^{13} \cdot x = (x^3 + x^2 + 1) \cdot x = x^4 + x^3 + x = x + 1 + x^3 + x = \underline{x^3 + 1}$;

$\boldsymbol{\alpha^{15}} \rightarrow x^{15} = x^{14} \cdot x = (x^3 + 1) \cdot x = x^4 + x = x + 1 + x = \underline{1}$.

В таблице 1.8даны представления элементов расширенного поля $GF(2^4)$, заданного неприводимого полинома $p(x) = x^4 + x + 1$, а также полинома $p(x) = x^4 + x^3 + 1$.

Таблица 1.8 − Различные представления элементов поля $GF\ (2^4)$

Элементы поля $GF(2^4)$	$p(x) = x^4 + x + 1$			$p(x) = x^4 + x^3 + 1$		
	Представление элементов поля			Представление элементов поля		
	полином	вектор	степень α	полином	вектор	степень γ
0	0	0	0	0	0	0
β_0	1	0001	1	1	0001	1
β_1	x	0010	α^1	x	0010	γ^1
β_2	x^2	0100	α^2	x^2	0100	γ^2
β_3	x^3	1000	α^3	x^3	1000	γ^3
β_4	$x+1$	0110	α^4	$x^3 + 1$	1001	γ^4
β_5	x^2+x	0110	α^5	$x^3+ x +1$	1011	γ^5
β_6	$x^3 + x^2$	1100	α^6	$x^3 + x^2 + x + 1$	1111	γ^6
β_7	$x^3+ x +1$	1011	α^7	$x^2 + x + 1$	0111	γ^7
β_8	$x^2 + 1$	0101	α^8	$x^3 + x^2 + x$	1110	γ^8
β_9	$x^3 + x$	1010	α^9	$x^2 + 1$	0101	γ^9
β_{10}	$x^2 + x + 1$	0111	α^{10}	$x^3 + x$	1010	γ^{10}
β_{11}	$x^3 + x^2 + x$	1110	α^{11}	$x^3 + x^2 + 1$	1101	γ^{11}
β_{12}	$x^3 + x^2 + x + 1$	1111	α^{12}	$x+1$	0011	γ^{12}
β_{13}	$x^3 + x^2 + 1$	1101	α^{13}	x^2+x	0110	γ^{13}
β_{14}	$x^3 + 1$	1001	α^{14}	$x^3 + x^2$	1100	γ^{14}

$\alpha^{15} = \alpha^0 = 1$; $\gamma^{15} = \gamma^0 = 1$

Ненулевые элементы $GF(2^4)$ расположены в порядке нарастания степени примитивного элемента и образуют циклическую группу порядка 15. При этом$\alpha^{15}=1$, $\alpha^{16}=\alpha$, $\alpha^{17}=\alpha^2$,...,$\alpha^{30}=1$ и т.д. Примитивным в поле $GF(2^4)$ является не только один элемент α, но и α^2, α^4, α^8 и ряд других, а элементы α^3 и α^5 таковыми не являются.

Представление элементов поля по степеням примитивного элемента, α используется при умножении элементов друг на друга. Для этого достаточно сложить их степени по модулю p^m-1. Например,

$\beta_{10}\beta_{13}=(x^2+x+1)(x^3+x^2+1)=\alpha^{10}\alpha^{13} \leftrightarrow (10+13) mod 15 = 8 \leftrightarrow \alpha^8 = x^2+1.$

Прямые вычисления дают то же, но более трудоемко:

$\beta_{10}\beta_{13}=(x^2+x+1)(x^3+x^2+1)= x^5+x^4+x^3+x^4+x^3+x^2+x^2+x+1=$

$= x^5+x+1 = x^2+x+x+1 = x^2+1.$

Для того чтобы сложить элементы расширенного поля $GF(2^4)$ можно воспользоваться представлением элементов в двоичной или полиномиальной форме, при этом используются поразрядное сложение по модулю два. Например найдем сумму двух элементов из расширенного поля $GF(2^4)$, заданного полиномом $p(x)= x^4 + x + 1$:

$\alpha^5 + \alpha^4 = 0110 + 0011 = 0101 = \alpha^8;$

$\alpha^{11} + \alpha^{10} = (x^3+x^2+x) + (x^2+x+1) = x^3+x^2+x+x^2+x+1 = x^3+1.$

Для расширенного поля $GF(2^4)$, порождаемое многочленом $p(x)= x^4 + x + 1$, а также $p(x)= x^4 + x^3 + 1$ построены таблицы сложения и умножения элементов поля (Приложение Г).

Установим некоторые вспомогательные результаты, которые в дальнейшем помогут в решении основной задачи, т.е. построении циклических кодов с заданными характеристиками. Основным свойством конечных полей и полиномов является связь между элементами конечного поля.

1. Все ненулевые элементы β конечного поля $GF(2^m)$ являются степенями одного примитивного элемента:

$\beta = \alpha^s$, $s= 0,1,2,...,p^m-2$; при этом $\alpha^{p^m-1}=1.$

2. Порядком βэлемента $\beta^{p^m}=\beta$конечного поля называется наименьшее значение β, для которого$\beta^{p^m}=1$. Пусть $\beta = \alpha^i$. Поскольку ненулевые элементы β образуют циклическую группу, порядок элемента α^i может быть определен из равенства

$$q = \frac{p^m - 1}{\text{НОД}\ [p^m - 1, i]}, \qquad (1.7)$$

где НОД− наибольший общий делитель. Порядки элементов x^q-1 лежат в пределах от 1 (элемент $p(x)$) до n (примитивные элементы), но p^m-1 всегда кратно порядку элемента.

3. Среди всех элементов расширенного поля $GF(2^m)$лишь элементы основного подполя $GF(2)$, т.е. 0 и 1, удовлетворяет равенству

$$\alpha^2 = \alpha.$$

4. Для любых элементов α и β поля $GF(2^m)$

$$(\alpha^2 + \beta^2) = \alpha^2 + \beta^2.$$

5. Для заданного простого числа p существует только одно поле с данным числом элементов, хотя оно может быть представлено разными способами. Два поля, различающиеся только представлениями, называются изоморфными.

При использовании циклических кодов оперируют кодовыми словами, представляемыми часто в виде многочленов (полиномов) над полем $GF(2^m)$, то есть полиномами, коэффициентами при степенях x в которых являются элементы расширенного поля α^i. При сложении, умножении и делении действия с этими полиномами выполняются по законам обычной алгебры с той лишь разницей, что сложение, умножение и деление коэффициентов α^i происходит по законам алгебры поля $GF(2^m)$.

Например, пусть

$$\begin{aligned} a(x) &= \alpha^{12}x^4 + \alpha^{14}x^3 + \alpha^2 x \\ b(x) &= \alpha^{11}x^3 + \alpha^8 x^2 + \alpha^{12} \end{aligned},$$

тогда сложение:

$$\begin{aligned} a(x) + b(x) &= \alpha^{12}x^4 + (\underbrace{\alpha^{14} + \alpha^{11}})x^3 + \alpha^8 x^2 + \alpha^2 x + \alpha^{12} = \\ &= \alpha^{12}x^4 + \alpha^{10}x^3 + \alpha^8 x^2 + \alpha^2 x + \alpha^{12} \end{aligned},$$

умножение:

$$\begin{aligned} a(x)\cdot b(x) &= (\alpha^{12}x^4 + \alpha^{14}x^3 + \alpha^2 x)\cdot(\alpha^{11}x^3 + \alpha^8 x^2 + \alpha^{12}) = \\ &= \alpha^{12}\alpha^{11}x^7 + \alpha^{12}\alpha^8 x^6 + \alpha^{12}\alpha^{12}x^4 + \alpha^{14}\alpha^{11}x^6 + \alpha^{14}\alpha^8 x^5 + \alpha^{14}\alpha^{12}x^3 + \\ &+ \alpha^2\alpha^{11}x^4 + \alpha^2\alpha^8 x^3 + \alpha^2\alpha^{12}x = \\ &= \alpha^8 x^7 + \alpha^5 x^6 + \alpha^9 x^4 + \alpha^{10}x^6 + \alpha^7 x^5 + \alpha^{11}x^3 + \alpha^{13}x^4 + \alpha^{10}x^3 + \alpha^{14}x = \\ &= \alpha^8 x^7 + \alpha^0 x^6 + \alpha^7 x^5 + \alpha^{10}x^4 + \alpha^{14}x^3 + \alpha^{14}x \end{aligned},$$

деление:

$$a(x)/b(x) = \alpha^{12}x^4 + \alpha^{14}x^3 + \alpha^2 x \qquad \left|\underline{\alpha^{11}x^3 + \alpha^8 x^2 + \alpha^{12}}\right.$$

$$\alpha^{12}x^4 + \underline{\alpha^9 x^3 + \alpha^{13}x} \qquad \left|\alpha x + \alpha^{12}\right.$$

$$0 \qquad \alpha^8 x^3 + 0 \cdot x$$

$$\underline{\alpha^8 x^3 + \alpha^5 x^2 + \alpha^9}$$

$$0 \qquad \alpha^5 x^2 + \alpha^9$$

Таким образом, частное от деления полинома $a(x)$ на полином $b(x)$ равно $Q(X) = \alpha x + \alpha^{12}$ и остаток $R(x) = \alpha^5 x^2 + \alpha^9$.

1.6 Корни полиномов и построение минимальных многочленов

Ключевым при построении кодов и их декодировании является вопрос о корнях полиномов, соответствующих кодовым комбинациям. Из теории полиномов над полем вещественных чисел (не конечных) известно, что полином степени m всегда имеет m корней, только не все они обязательно лежат в поле вещественных чисел (на вещественной оси). Часть корней может находиться в поле комплексных чисел как некотором расширении поля вещественных чисел.

Ааналогия этому имеется и в конечных полях. Любой многочлен степени m, в том числе и неприводимый над полем $GF(p)$ (не имеющий корней среди элементов этого поля), всегда имеет m корней в расширении $GF(p^m)$, и этими корнями является часть элементов поля $GF(p^m)$. Таким образом, при подстановке элемента β_i, который принадлежит расширенному полю $GF(p^m)$, в двоичный неприводимый полином (не имеющий корней в поле $GF(2)$) $f(x) = f(\beta) = 0$, говорят, что полином $f(x)$ имеет корень β, принадлежащий расширенному полю $GF(p^m)$.

Например, неприводимый полином $p(x) = x^3 + x + 1$ не имеет корней в поле $GF(2) = \{0, 1\}$, так как при подстановке 0 и 1 в многочлен $p(x)$ получим:

$p(0) = 0^3 + 0 + 1 = 1$; $p(1) = 1^3 + 1 + 1 = 1$.

Однако, если использовать в качестве корня элемент α^2 расширенного поля $GF(2^3)$ (табл. 1.4), то получим:

$p(\alpha^2) = (\alpha^2)^3 + \alpha^2 + 1 = \alpha^6 + \alpha^2 + 1 = 1 + 1 = 0$.

Как элементы конечного поля, корни находятся между собой в определенном соотношении, т.е. если $f(x)$ – неприводимый полином с коэффициентами из поля $GF(p)$ и β_i – его корень, то β_i^p, $\beta_i^{p^2}$, $\beta_i^{p^3}$, ...также являются его корнями.

Если $f(x)$ – двоичный полином с коэффициентами из поля $GF(2)$ и α^i – его корень из поля $GF(2^m)$ то элементы

$$\alpha^i, \alpha^{2i}, \alpha^{4i}, \ldots, \alpha^{\tau}$$

где $\tau = 2^{k-1}$ (k делит m), также являются его корнями.

Элементы из расширенного поля $GF(2^m)\alpha^i, \alpha^{2i}, \alpha^{4i}, \ldots, \alpha^{\tau}$, которые являются корнями одного и того же многочлена над $GF(2)$ называются *сопряженными* относительно $GF(2)$.

В общем случае у элемента может быть больше, чем один, сопряженный элемент – на самом деле m.

Степени сопряженных элементов поля образуют *циклотомический смежный класс*

$$C_i = \{i, 2i, 4i, ..., 2^{k-1}i\},$$

где i – степень α элемента расширенного поля $GF(2^m)$.

Циклотомические смежные классы, называемые также циклическими множествами обладают свойством расщепления множества вычетов целых чисел Z по модулю $n = 2^m - 1$. Весь набор показателей степеней примитивного элемента в поле $GF(2^m)$ распадается на не перекрывающиеся циклотомические классы, т.е. циклотомические смежные классы не пересекаются. При этом каждый из циклотомических классов содержит набор показателей степеней примитивного элемента, соответствующих корням одного из полиномов $f(x)$.

Например, найдем циклотомические смежные классы по модулю 7 для расширенного поля $GF(2^3)$, где $m = 3$, а степени элементов $\alpha = \{0, 1, 2, 3, 4, 5, 6\}$:

– $i = 0 \rightarrow C_0 = \{0\}$;

– $i = 1 \rightarrow C_1 = \{1, 2, 4\}$;

– $i = 3 \rightarrow C_3 = \{3, 6, 5\}$ (при$3 \cdot 4 = 12$, но $12 \bmod 7 = 5$).

Возвращаясь к предыдущему примеру, можно убедится, что полином $p(x) = x^3 + x + 1$ наряду с α^2 имеет в $GF(2^3)$ следующие корни:

– $\alpha \rightarrow p(\alpha) = (\alpha^1)^3 + \alpha + 1 = \alpha^3 + \alpha + 1 = 1 + 1 = 0$.

– $\alpha^4 \rightarrow p(\alpha^4) = (\alpha^4)^3 + \alpha^4 + 1 = \alpha^{12} + \alpha^4 + 1 = \alpha^5 + \alpha^4 + 1 = 1 + 1 = 0$.

Таким образом, элементы α, α^2, α^4 являются сопряженными.

Каждый из p^m-1 ненулевых элементов поля $GF(2^m)$ является одним из корней полинома $x^{p^m-1}-1$, который в свою очередь, раскладывается на произведение неприводимых полиномов $\varphi_i(x)$ меньшей степени. Каждый из циклотомических классов содержит набор показателей степеней примитивного элемента, соответствующих корням одного из полиномов $\varphi_i(x)$.

Двоичный полином наименьшей степени, для которого элемент α^i из расширенного конечного поля $GF(2^m)$ является корнем, называется *минимальным многочленом* $\varphi_i(x)$. При этом степень минимального многочлена равна числу элементов (мощности) соответствующего циклотомического класса. Тогда многочлен $\varphi_s(x)$ с помощью его корней α может быть представлен

$$\varphi_s(x) = \prod_{i_S \in C_S} (x + \alpha^{i_s}), \tag{1.8}$$

(«−» заменен на «+», т.к. вычисления осуществляются по модулю 2 и вычитание и сложение неразличимы).

Примитивный элемент α расширенного поля $GF(2^m)$ удовлетворяет неравенству $\alpha^{2^m-1} = \alpha^{n-1} = 1$ ($n = 2^m-1$) и все ненулевые элементы поля являются некоторой степенью примитивного элемента. Таким образом, бином $x^{n-1}-1 = x^{n-1}+1$ степени $n = 2^m-1$ имеет разложение на сомножители первой степени в поле $GF(2^m)$:

$$x^{n-1}+1 = (x-\beta_1)(x-\beta_2)\ldots = \prod_{i=0}^{2^m-2}(x+\alpha^i). \tag{1.9}$$

С учетом формулы (1.8) бином $x^{n-1}+1$ имеет следующее разложение на неприводимые двоичные сомножители в двоичном поле

$$(x^{n-1}+1) = \prod_{i=0}^{M} \varphi_{l_i}(x), \tag{1.10}$$

где M – число циклотомических классов.

Алгоритм нахождения всех неприводимых делителей бинома $(x^{n-1}+1)$:

1. Найти все циклотомические классы C_s по модулю 2^m-1.

2. Найти сопряженные элементы.

3. Для каждого циклотомического класса C_sвычислить минимальный многочлен по формуле (1.8).

Например, построим распределение элементов поля $GF(2^3)$ по циклотомическим классам и для каждого из них вычислим минимальный многочлен.

1. Циклотомические множества по модулю 7

   $C_0 = \{0\}$;

   $C_1 = \{1, 2, 4\}$;

   $C_3 = \{3, 6, 5\}$.

2. Определим сопряженные элементы поля $GF(2^3)$

   $C_0 = 1$;

   $C_1 = \alpha, \alpha^2, \alpha^4$;

   $C_3 = \alpha^3, \alpha^6, \alpha^5$.

3. Для каждого циклотомического класса C_s вычислим минимальный многочлен

− класс C_0 содержит один элемент, таким образом неприводимый над полем $GF(2)$ полином имеющий в качестве корня элемент $\alpha^0=1$ поля $GF(2^3)$ должен быть полиномом первой степени

$$\varphi_0(x) = \underline{x + 1}.$$

− класс C_1 содержит три элемента, т.е. минимальный многочлен корни которого $\alpha, \alpha^2, \alpha^4$ из поля $GF(2^3)$ должен быть третьей степени, используя (1.8) найдем многочлен

$$\varphi_1(x) = (x + \text{корень1})(x + \text{корень2})(x + \text{корень3}),$$

$$\varphi_1(x) = (x + \alpha)\cdot(x + \alpha^2)\cdot(x + \alpha^4) = (x^2 + x\alpha + x\alpha^2 + \alpha^3)(x + \alpha^4) =$$

$$= x^3 + x^2\alpha + x^2\alpha^2 + x\alpha^3 + x^2\alpha^4 + x\alpha^5 + x\alpha^6 + \alpha^7 = x^3 + x^2\, x + x^2\, x^2 + x\,(1+x) +$$

$$+ x^2\,(x+x^2) + x\,(1+x+x^2) + x\,(1+x^2)+1 = x^3+x^3+x^4+x+x^2+x^3+x^4+x+x^2+x^3+x+x^3+1=$$

$$= \underline{x^3 + x + 1}.$$

−класс C_3 содержит три элемента, т.е. минимальный многочлен корни которого $\alpha^3, \alpha^6, \alpha^5$ из поля $GF(2^3)$ должен быть третьей степени, используя (1.8) найдем многочлен (умножение степеней по модулю 7)

$$\varphi_1(x) = (x + \alpha^3)\cdot(x + \alpha^6)\cdot(x + \alpha^5) = (x^2 + x\alpha^3 + x\alpha^6 + \alpha^2)(x + \alpha^5)$$

$$= x^3 + x^2\alpha^3 + x^2\alpha^6 + x\alpha^2 + x^2\alpha^5 + x\alpha^1 + x\alpha^4 + 1 = x^3 + x^2\,(x+1) + x^2\,(x^2+1) + x\,x^2 +$$

$$+x^2\,(x^2+x+1) + x\,x + x\,(x^2+x) + 1 = x^3+x^3+x^2+x^4+x^2+x^3+x^4+x^3+x^2+x^2+x^3+x^2+1=$$

$$= \underline{x^3 + x^2 + 1}.$$

В таблице 1.11 представлены минимальные многочлены для каждого циклотомичнского класса элементов расширенного поля $GF(2^3)$.

Таблица 1.11 – Распределение элементов по циклатомическим классам поля $GF(2^3)$, минимальные многочлены

Циклотомические классы C_s	Сопряженные элементы. Корни полинома $\varphi_i(x)$	Минимальный многочлен $\varphi_i(x)$
$C_0 = \{0\}$	1	$\varphi_0(x) = x + 1$
$C_1 = \{1, 2, 4\}$;	$\alpha, \alpha^2, \alpha^4$	$\varphi_1(x) = x^3 + x + 1$
$C_3 = \{3, 6, 5\}$	$\alpha^3, \alpha^6, \alpha^5$	$\varphi_3(x) = x^3 + x^2 + 1$

Распределение элементов расширенного конечного поля $GF(2^4)$ порождаемое неприводимым многочленом $p(x) = x^4 + x + 1$ по циклотомическим классам с указаниям соответствующих минимальных многочленов представлено в таблице 1.12.

Таблица 1.12 – Распределение элементов по циклатомическим классам поля $GF(2^4)$, минимальные многочлены

Циклотомические классы	Сопряженные элементы. Корни многочлена $\varphi_i(x)$	Минимальные многочлены $\varphi_i(x)$
$C_0=\{0\}$	α^0	$\varphi_0(x) = x + 1$
$C_1=\{1, 2, 4, 8\}$	$\alpha^1, \alpha^2, \alpha^4, \alpha^8$	$\varphi_1(x) = x^4 + x + 1$
$C_3=\{3, 6, 9, 12\}$	$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$\varphi_3(x) = x^4 + x^3 + x^2 + x + 1$
$C_5=\{5, 10\}$	α^5, α^{10}	$\varphi_5(x) = x^2 + x + 1$
$C_7=\{7, 11, 13, 14\}$	$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	$\varphi_7(x) = x^4 + x^3 + 1$

Распределение элементов расширенного конечного поля $GF(2^4)$ порождаемое неприводимым многочленом $p(x) = x^4 + x^3 + 1$ по циклотомическим классам с указаниям соответствующих минимальных многочленов представлено в приложении Д (табл. Д.1).

Используя найденные минимальные многочлены в поле $GF(2)$, корни которых принадлежать расширенному полю $GF(2^3)$разложим полином $x^{p^m-1}-1 = x^7-1 = x^7 + 1$ коэффициенты которого принадлежат полю $GF(2)$ на простые сомножители (неприводимые полиномы) $\varphi_i(x)$. Поскольку сложение и вычитание по модулю 2 неразличимы, то записи x^7-1 и $x^7 + 1$ эквивалентны

$$x^7 + 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1).$$

Минимальный многочлен, корнем которого является примитивный элемент расширенного поля $GF(2^m)$, называется *примитивным многочленом*. Его степень всегда равна m. Когда неприводимый многочлен $p(x)$, задающий операции в поле, является также и примитивным многочленом, примитивным элементом поля будет элемент $\alpha = x$. В таблице 1.13 представлены примитивные многочлены для m от 1 до 20.

Таблица 1.13 – Примитивные многочлены до степени $m = 20$

$x+1$	$x^6 + x + 1$	$x^{11} + x^2 + 1$	$x^{16} + x^{12} + x^3 + x + 1$
$x^2 + x + 1$	$x^7 + x^3 + 1$	$x^{12} + x^6 + x^4 + x + 1$	$x^{17} + x^3 + 1$
$x^3 + x + 1$	$x^8 + x^4 + x^3 + x^2 + 1$	$x^{13} + x^4 + x^3 + x + 1$	$x^{18} + x^7 + 1$
$x^4 + x + 1$	$x^9 + x^4 + 1$	$x^{14} + x^9 + x^5 + x + 1$	$x^{19} + x^5 + x^2 + x + 1$
$x^5 + x^2 + 1$	$x^{10} + x^3 + 1$	$x^{15} + x + 1$	$x^{20} + x^3 + 1$

Использовавшийся ранее при построениирасширенного конечного поля$GF(2^4)$ неприводимый полином $p(x) = x^4 + x + 1$ примитивен, на основании чего в качестве примитивного элемента поля был взят $\alpha = x$.

Контрольные вопросы

1. Дайте определения конечного поля. Приведите пример конечных полей.

2. Операция сложения и умножения в конечном поле.

3. Дайте определения расширенного поля $GF(p^m)$. Приведите пример.

4. Построить расширенное поле $GF(5)$ и написать таблицы сложения и умножения.

5. Единичный элемент относительно сложения и относительно умножения в конечном поле.

6. Построение расширенных конечных полей на основе многочленов.

7. Построить расширенной поле $GF(2^4)$ на основе неприводимого многочлена $p(x) = x^3+x^2+1$.

8. Найти значение выражения для элементов поля $GF(2^4)((\alpha^{12}+\alpha^5)*\alpha^6+\alpha^{14})/(\alpha^8+\alpha^3)$.

9. Раскрыть понятие неприводимого многочлена.

10. Доказать, что многочлен $p(x) = x^3+x^2+2$ неприводим над полем $GF(3)$.

11. Найти произведение $(\alpha^2x^8 +\alpha^3x^6+\alpha^6x^5+\alpha x^3+\alpha^4x+1)\cdot(\alpha^6x^4+x^3+\alpha x^2+1)$ для элементов поля $GF(2^3)$.

12. Корни многочленов в поле $GF(p)$ и в полях $GF(p^m)$.

13. Определение примитивного элемента расширенного поля $GF(p^m)$.

14. Раскрыть понятие сопряженных элементов расширенного поля $GF(p^m)$.

15. Распределение сопряженных элементов поля по циклотомическим классам.

16. Найти остаток от деления $(\alpha^3x^{10} +\alpha^3x^8+\alpha^6x^7+\alpha^5x^6+\alpha x^3+\alpha^2x^2+\alpha^4x+1)/(\alpha^6x^5+\alpha^2x^4+\alpha x^2+x+1)$.

17. Циклотомические классы и сопряженные элементы для полей $GF(2^3)$, $GF(2^4)$.

18. Минимальные многочлены для расширенных полей $GF(p^m)$.

19. Найти все минимальные многочлены для поля $GF(2^4)$.

20. Раскрыть понятие примитивного многочлена.

ГЛАВА 2.

ЦИКЛИЧЕСКИЕ КОДЫ

2.1 Полиномиальные циклические коды

Циклические коды составляют класс кодов, исправляющих ошибки, кодирование и декодирование которых основано на полиномиальном представлении. Простая реализация этих кодов использует регистры сдвига и логические схемы.

Удобным инструментом изучения и построения циклических кодов служит полиномиальное представление (повторяющее *z*-преобразование в теории дискретных систем). Кодовый полином, отвечающий кодовому вектору $v=(v_0, v_1, \ldots, v_{n-2}, v_{n-1})$, определяется равенством

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + ... + v_1x^1 + v_0, \qquad (2.1)$$

где $v_0, v_1, \ldots, v_{n-2}, v_{n-1}$ − коэффициенты кодового полинома, которые являются элементами кодового слова (для q-ичного кода – элементы поля $GF(q)$); x – формальная переменная, служащая для указания своей степенью позиции кодового символа как коэффициента полинома.

Пример2.1.Полином $v(x) = x^6 + x^4 + x^2 + 1$, коэффициенты которого принадлежат полю $GF(2)$ для двоичного кода, соответствует кодовый вектор $v = 1010101$, где единицы стоят на позициях 6, 4, 2, 0, а все остальные элементы равны 0. Над полиномами в поле двоичных символов $GF(2)$ можно производить различные арифметические операции.

Суммой двух полиномов $v(x)$ и $f(x)$ $GF(2)$ называется полином из $GF(2)$, определяемый следующим образом:

$$v(x) + f(x) = \sum_{i=0}^{n-1} (v_i + f_i) \cdot x^i, \qquad (2.2)$$

где «+» − операция сложения по модулю 2 коэффициентов при одинаковых степенях x.

Произведением двух полиномов из $GF(2)$ называется полином из $GF(2)$ определяемый следующим образом:

$$v(x)\cdot f(x)=\sum_{i=0}^{n-1}\left(\sum_{j=0}^{i}v_i\cdot f_{i-j}\right)\cdot x^i, \qquad (2.3)$$

т.е. произведение получается по обычному правилу перемножения степенных функций, однако получаемые коэффициенты при данной степени x складываются по модулю 2.

Циклические коды – коды, у которых кодовая комбинация может быть получена путем циклической перестановки символов комбинации принадлежащей к этому же коду. Т.е. если кодовая комбинация $v_0, v_1, \ldots, v_{n-2}, v_{n-1}$ является разрешенной комбинацией циклического кода, то комбинация $v_{n-1}, v_0, v_1, \ldots, v_{n-2}, v_{n-1}$ также принадлежит этому коду. Основные параметры циклического кода: n – длина кодовой последовательности, k – длина информационной последовательности, r– количество проверочных символов.

В полиномиальном представлении циклический сдвиг на одну позицию, обозначенный $v^{(1)}(x)$, соответствует умножению на x по модулю x^n-1

$$v(x)\in C \Leftrightarrow v^{(1)}(x)=xv(x) \bmod (x^n+1)\in C. \qquad (2.4)$$

Таким образом, полином $v^{(1)}(x)$ получается путем умножения исходного полинома $v(x)$ на x и удержанием остатка от деления результата на бином x^n-1. Повторяя сдвиг t раз, можно убедиться, что кодовый полином $v^{(t)}(x)$ t-кратно сдвинутого слова v

$$v^{(t)}(x)=x^t v(x) \bmod (x^n+1). \qquad (2.5)$$

В двоичной арифметике операции «+» и «−» дают одинаковый результат (по модулю 2), поэтому в дальнейшем не будем различать эти операции.

Операция циклического сдвига реализуется на регистре сдвига показанном на рис 2.1.

Пример2.2. Для $n = 7$ кодовый вектор равен $v = 1101010$ $(x^6 + x^5 + x^3 + x)$, после сдвига на одну позицию получим вектор $v^{(1)}(x) = 1010101$ $(x^6 + x^4 + x^2 + 1)$. В полиномиальном представлении

$v(x) = x^6 + x^5 + x^3 + x$,

$v^{(1)}(x) = (x^6 + x^5 + x^3 + x)\cdot x \bmod (x^7+1) = (x^7 + x^6 + x^4 + x^2) \bmod (x^7+1)$,

$$
\begin{array}{l|l}
x^7 + x^6 + x^4 + x^2 & x^7+1 \\
\cline{2-2}
x^7+1 & 1 \\
\cline{1-1}
x^6 + x^4 + x^2 + 1 &
\end{array}
$$

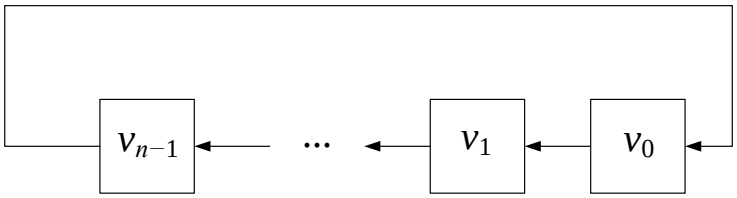

Рисунок 2.1 – Циклический регистр сдвига

При построении циклических кодов используются *неприводимые* в поле двоичных числе $GF(2)$ многочленов. Неприводимые многочлены не могут быть представлены в виде произведения многочленов низших степеней с коэффициентами из того же поля, т.е. такой многочлен делится только на самого себя или на единицу и не делится ни на какой другой многочлен. Неприводимые многочлены в циклических кодах играют роль *образующих или порождающих полиномов*

$$g(x) = g_r x^r + g_{r-1} x^{r-1} + \quad ... \quad + g_0, \tag{2.6}$$

где $r = \deg [g(x)] = n-k$ – количество проверочных символов, коэффициенты $g_r = 1$, $g_0 \neq 0$.

Важным свойством циклических кодов является то, что все кодовые слова-полиномы кратны порождающему полиному $g(x)$, который задается своими *корнями*, называемые *нулями кода*.

Порождающий полином циклического кода длины n является делителем бинома x^n-1 (x^n+1). Таким образом для того чтобы найти некоторый порождающий многочлен, надо знать разложение бинома x^n+1 на неприводимые множители $\varphi_j(x)$, $j = 1, 2, \ldots, l$

$$(x^n + 1) = \varphi_1(x)\varphi_2(x) \ldots \varphi_l(x). \tag{2.7}$$

Таким образом, порождающий полином равен

$$g(x) = \prod_{j=1,2,..l} \varphi_j(x). \tag{2.8}$$

Например, на множестве двоичных многочленов, т.е. полиномов с коэффициентами из поля $GF(2) = \{0, 1\}$, биномx^7-1 имеет следующее разложение

$$x^7 + 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1).$$

Связь между структурой $g(x)$ и минимальным кодовым расстоянием $d_{\min}$порождаемого им кода, заключается в следующем: для того чтобы задаваемый полином $g(x)$ код имел минимальное кодовое расстояние $d_{\min}$, количество отличных от нуля коэффициентов $g(x)$ должно быть не менее $d_{\min}$.

Тогда можно построить следующие циклические коды:

− двоичный циклический код (7, 4, 3)с n=7, k=4, r=3, $d_{\min} = 3$и с порождающим полиномом $g(x) = x^3 + x + 1$;

− двоичный циклический код (7, 6, 2)с n=7, k=6, r=1, $d_{\min} = 2$и с порождающим полиномом $g(x) = x + 1$;

− двоичный циклический код (7, 3, 4)с n=7, k=3, r=4, $d_{\min} = 4$и с порождающим полиномом $g(x) = (x^3 + x + 1)(x + 1)$.

2.2 Кодирование двоичных циклических кодов

Предположим, что необходимо закодировать информационную последовательность $u = (u_{k-1}, \ldots u_1, u_0)$или соответствующий ей полином

$$u(x) = u_{k-1}x^{k-1} + \quad ... \quad + u_1x^1 + u_0. \qquad (2.9)$$

Существует несколько различных способов кодирования. Принципиально наиболее просто комбинации циклического кода можно получить, умножая многочлен $u(x)$ на порождающий полином кода $g(x)$. Такой способ легко реализуется, однако он имеет недостаток, что получающиеся в результате умножения комбинации кода не содержат информационных символов в явном виде. После исправления ошибок такие комбинации для выделения информационных символов приходится делить на порождающий полином. Ситуацию можно упростить, если контрольные символы переписать в конец кода, т.е. после информационных символов. Таким образом, циклический код можно образовать двумя способами.

1. *Несистематическое кодирование* – умножение информационной комбинации двоичного циклического кода $u(x)$ на порождающий полином $g(x)$

$$v(x) = u(x)g(x). \qquad (2.10)$$

Например, устройство, производящее умножение информационного многочлена $u(x) = u_2x^2+u_1x^1,+u_0$ на порождающий полином $g(x) = x^3 + x + 1$ показано на рис. 2.2.

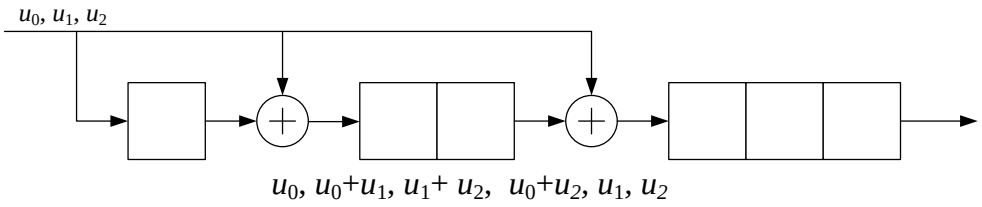

Рисунок 2.2 – Схема умножения полинома $u_2x^2+u_1x^1,+u_0$ на $g(x) = x^3 + x + 1$

Недостаток такого способа кодирования, заключается в том, что получаемый код не имеет четкого разделения разрядов на информационные и избыточные, что затрудняет процесс декодирования.

2. *Систематическое кодирование* – умножение информационной комбинации $u(x)$ на одночлен x^r, имеющий ту же степень, что и порождающий полином $g(x)$, с добавлением к этому произведению остатка $R(x)$, полученного после деления произведения $u(x)\cdot x^r$ на порождающий полином $g(x)$

$$v(x) = u(x)x^r + R(x), \qquad (2.11)$$

где $R(x)$ − остаток от деления $u(x)\cdot x^r$ на $g(x)$.

В случае общего (n, k) кода предварительное умножение всегда производится на x^{n-k}. Степень остатка всегда меньше степени делителя, которая равна $n-k$. Поскольку кодируемая последовательность умножалась на x^{n-k}, в делимое входят лишь члены степени, не меньшей $n-k$. Поэтому добавление остатка к частотному не изменяет ни одного из членов в обоих многочленах. Выбирая в качестве делителя порождающий многочлен,

можно добиться того, что результат всегда будет кодовое слово. Схема, реализующая этот алгоритм для (6,3)-кода, показана на рис. 2.3. При использовании этой схемы информационная последовательность u_2, u_1, u_0 подается на вход по одному символу за такт. Содержимое различных ячеек после каждого сдвига показано на рис. 2.3. После трех сдвигов остаток (три проверочных символа) запоминается в трех ячейках регистра сдвига. Произведя длинные операции деления, читатель может удостоверится в том, что согласно этой схеме операции сдвига и вычитания повторяются на каждом шаге. При необходимости частное может быть получено в виде последовательных членов в цепи обратной связи.

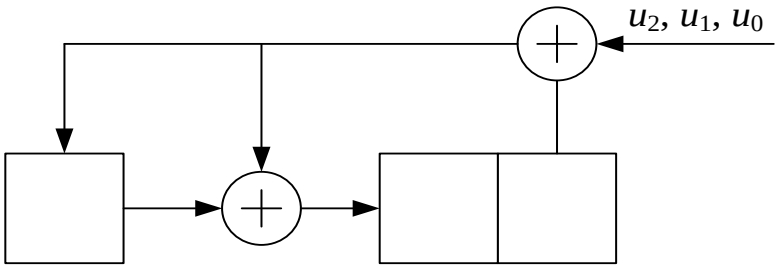

Рисунок 2.3 – Схеме умножения полинома на x^3
и деления на $g(x) = x^3 + x + 1$

Таким образом, кодовое слово состоит из неизменной информационной части u, после которого расположено $(n-k)$ проверочных символов. Проверочные символы являются коэффициентами полинома $g(x)$, то есть остатка от деления $u(x) \cdot x^{n-k}$ на порождающий полином $g(x)$.

Схема, показанная на рис. 2.3 является простой, так как все арифметические операции выполняются в поле $GF(2)$. В общем случае кода над полем $GF(q)$ следует выполнять также операции умножения. Предположим, что порождающий многочлен имеет вид $g(x) = g_{n-k}x^{n-k} + g_{n-k-1}x^{n-k-1} + \ \ldots \ + g_0$, где коэффициенты лежат в поле $GF(q)$. Тогда схема на рис. 2.3 должна быть изменена следующим образом. Прежде всего, все операции сложения должны быть изменены на сложение в $GF(q)$ и каждая ячейка регистра сдвига должна содержать элемент из $GF(q)$. Далее, результат на выходе сумматора, формирующего член обратной связи, нужно умножить на g_{n-k}^{-1}. Для каждого ненулевого g_i, $i<n-k$, член обратной связи должен быть умножен на $-g_i$. Сумма

полученного произведения с результатом выхода предыдущей ячейки является входным значением i-й ячейки регистра (i меняется от 0 до $n-k-1$). Обобщенная схема кодирования показана на рис. 2.4. Произведя деление, можно показать, что эта схема дает требуемый остаток.

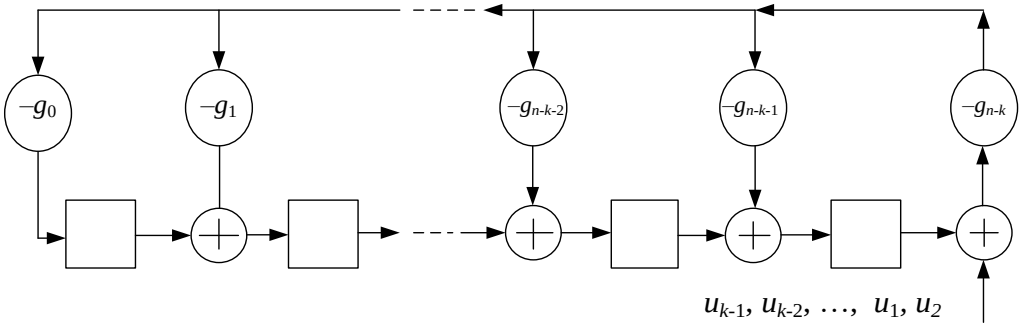

Рисунок 2.4 – Схема умножения на x^{n-k} и деления на $g(x)$

Пример 2.3. Закодируем информационную последовательность $u(x) = 1110 = x^3 + x^2 + x$, с использованием кода, задаваемого порождающего полиномом $g(x) = 1101 = x^3 + x^2 + 1$.

1. Несистематический циклический код

$v(x) = u(x)g(x) = (x^3 + x^2 + x)\,(x^3 + x^2 + 1) = x^6+x^5+x^4+x^5+x^4+x^3+x^3+x^2+x =$
$= x^6 + x^2 + x$,

$$
\begin{array}{r}
\times 1110 \\
1101 \\
\hline
1110 \\
1110 \\
1110 \\
\hline
1000110
\end{array}
$$

2. Систематический циклический код

− умножим информационную последовательность $u(x)$ на одночлен x^r, где $r = 3 - \deg g(x)$

$u(x)\cdot x^r = (x^3 + x^2 + x)\cdot x^3 = x^6 + x^5 + x^4$,

$$
\begin{array}{r}
\times 1110 \\
1000 \\
\hline
1110000
\end{array}
$$

− разделим произведение $u(x)\cdot x^r$ на порождающий полином

$x^6+x^5+x^4$	x^3+x^2+1
$x^6+x^5+x^3$	x^3+x
x^4+x^3	
x^4+x^3+x	
x	

1110000	1101
1101	1010
1100	
1101	
10	

Остаток от деления равен $R(x) = x = 10$.

− прибавим полученный остаток $R(x)$ к произведению $u(x)\cdot x^r$

$v(x) = (x^6 + x^5 + x^4) + x = x^6 + x^5 + x^4 + x$.

$$\begin{array}{r} 1110000 \\ +\quad\ 10 \\ \hline 1110010 \end{array}$$

Таким образом, четырёх-символьную комбинацию $u(x) = 1110$ можно представить систематическим циклическим кодом $v(x) = 1110010$.

Еще один способ задания циклического кода основан на использовании порождающей или проверочной матриц. Порождающая матрица G имеющая k строк и n столбцов, содержит k базисных линейно независимых кодовых комбинаций. Наиболее удобна для использования каноническая форма порождающей матрицы, строки которой в информационной части образуют квадратную k x k единичную матрицу. Построение порождающей матрицы сводится к составлению единичной транспонированной и дополнительной матрицы, элементы которой представляют собой остатки от деления единицы с нулями на порождающий полином и полученные элементы приписываются справа от единичной матрицы.

$$G = \begin{vmatrix} 1 & 0 & \ldots & 0 & b_{11} & b_{12} & \ldots & b_{1n-k} \\ 0 & 1 & \ldots & 0 & b_{21} & b_{22} & \ldots & b_{2n-k} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots. \\ 0 & 0 & \ldots & 1 & b_{k1} & b_{k2} & \ldots & b_{kn-k} \end{vmatrix}. \qquad (2.12)$$

Проверочная часть порождающей матрицы циклического кода может быть построены с помощью следующих полиномов

$$\begin{aligned} & x^{n-1} \bmod g(x), \\ & \quad \vdots \\ & x^{n-k+1} \bmod g(x), \\ & x^{n-k} \bmod g(x). \end{aligned} \qquad (2.13)$$

Однако не все остатки от деления могут быть использованы в качестве элементов дополнительной матрицы, а лишь те из них, вес которых $w \geq d_{\min} - 1$. Длина остатков должна быть не менее количества контрольных разрядов, а число остатков должно равняться числу информационных разрядов.

Например, необходимо используя порождающую матрицу, построить код, исправляющий одинарные ошибки ($t = 1$) при передаче комбинации из 4-х ($k = 4$) символов. Код, исправляющий одинарные ошибки должен обеспечивать минимальное кодовое расстояние $d_{\min} = 3$ ($d_{\min} \geq 2t+1$). Таким образом, число разрядов дополнительной порождающей подматрицы равно 3 ($r = 3$), число столбцов равно 4 ($k = 4$) и каждый остаток от деления должен содержать три разряда. Выбираем из приложения А порождающий полином $g(x) = x^3 + x + 1$. Число единиц в каждом остатке от деления должно быть $w \geq d_{\min} - 1 \geq 3-1 \geq 2$. Для построения проверочной подматрицы найдем остатки от деления

$x^6 \bmod (x^3 + x + 1) = x^2 + 1$,

$x^5 \bmod (x^3 + x + 1) = x^2 + x + 1$,

$x^4 \bmod (x^3 + x + 1) = x^2 + x$,

$x^3 \bmod (x^3 + x + 1) = x + 1$.

Таким образом, систематическая порождающая матрица циклического код (7, 4, 3) $g(x) = x^3 + x + 1$ имеет вид

$$G = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{vmatrix}.$$

Проверочный полином $h(x)$, который может быть ассоциирован с проверочной матрицей H связан с порождающим полиномом следующим соотношением

$$g(x)h(x) = x^n + 1. \tag{2.14}$$

Если известен порождающий полином, то проверочный полином легко вычисляется

$$h(x) = \frac{x^n + 1}{g(x)} = h_k x^k + ... + h_1 x^1 + h_0 x^0. \tag{2.15}$$

Тогда проверочную матрицу H, которая имеет $r = n-k$ строк и n столбцов, связана с порождающей матрицей уравнением

$$H \cdot G^T = 0, \tag{2.16}$$

где G^T – транспонированная порождающая матрица.

При этом эквивалентное уравнение имеет вид

$$G \cdot H^T = 0. \tag{2.17}$$

Пример 2.4. Циклический код (7, 4, 3) с порождающим многочленом $g(x) = x^3 + x + 1$ имеет проверочный полином

$$h(x) = \frac{x^7 + 1}{x^3 + x + 1} = x^4 + x^2 + x + 1.$$

Проверочная матрица этого кода имеет следующий вид

$$H = \begin{vmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{vmatrix}. \tag{2.18}$$

Вектор v над полем $GF(p)$ является кодовым словом тогда и только тогда, когда

$$vH^T = 0. \tag{2.19}$$

Поскольку многочлен $v(x)$ каждого кодового слова делится на порождающий полином $g(x)$, то корни $g(x)$, при подстановке в $g(x)$ обращающие его в ноль, являются также и корнями многочлена $v(x)$. Число таких корней равно степени порождающего полинома $r = n-k$.

Таким образом, многочлен $v(x)$ с коэффициентами из поля $GF(p)$ будет кодовым словом в том и только в том случае, если элементы из расширенного поля $GF(p^m)$ $\{\beta_1, \beta_2 \ldots \beta_r\}$ являются его корнями

$$v(\beta_j) = v_0\beta_j^0 + v_1\beta_j^1 + \ldots + v_{n-2}\beta_j^{n-2} + v_{n-1}\beta^{n-1} = \sum_{i=1}^{n-1} v_i\beta_j^i = 0. \qquad (2.20)$$

Такая запись эквивалентна матричной форме $vH^T = 0$, где

$$H = \begin{vmatrix} \beta_1^0 & \beta_1^1 & \beta_1^2 \ldots & \beta_1^{n-1} \\ \beta_2^0 & \beta_2^1 & \beta_2^2 \ldots & \beta_2^{n-1} \\ \ldots & \ldots & \ldots\ \ldots\ \ldots & \\ \beta_{n-k}^0 & \beta_{n-k}^1 & \beta_{n-k}^2 \ldots & \beta_{n-k}^{n-1} \end{vmatrix}. \qquad (2.21)$$

Строки проверочной матрицы H содержат степени корней порождающего полинома $g(x)$, а следовательно, и корней кодовых многочленов. Каждый элемент матрицы H есть столбец из m элементов p-ичного представления корня – элемента поля $GF(p^m)$. Так как все корни неприводимого полинома разбиваются на непересекающиеся циклотомические классы, то проверочная матрица содержит ряд строк, функционально связанных между собой, т.е. выражаемых друг через друга. Это строки соответствующие множеству $\beta, \beta^p, \beta^{p^2}, \beta^{p^3}, \ldots$ корней одного циклотомического класса. Из каждого такого множества корней $\beta, \beta^p, \beta^{p^2}, \beta^{p^3}, \ldots$ следует выбрать по одному (любому), и соответствующие им строки оставить в матрице, а остальные удалить как функционально зависимые от удерживаемых элементов.

Рассмотрим построение проверочной матрицы циклического кода с минимальным расстоянием $d = 3$ для кодовых полиномов из поля $GF(2)$, при этом порождающий полином $g(x)$ неприводим и примитивен, а длина кода составляет $n = 2^m-1$. Корнем порождающего полинома $g(x)$ является примитивный элемент поля α, а показатели всех корней принадлежат

одному циклотомическому классу. Тогда проверочная матрица такого кода имеет вид

$$H = \lfloor \alpha^0 \ \alpha^1 \ \alpha^2 \ ... \ \alpha^{n-1} \rfloor,$$

и сам код называется код Хэмминга.

Тогда с учетом формулы (1.28) получим

$$c \cdot H^T = (c_0, c_1, ..., c_{n-1}) \cdot \begin{bmatrix} 1 \\ \alpha \\ \dots \\ \alpha^{n-1} \end{bmatrix} = c_0 + c_1\alpha + ... + c_{n-1}\alpha^{n-1} = c(\alpha) = 0.$$

Пример 2.5. Рассмотрим код (7, 4, 3) с порождающим полиномом $g(x) = x^3 + x + 1$, который является примитивным. Тогда проверочная матрица кода имеет вид

$$H = \lfloor 1 \ \alpha \ \alpha^2 ... \alpha^6 \rfloor = \begin{vmatrix} 0\,0\,1\,0\,1\,1\,1 \\ 0\,1\,0\,1\,1\,1\,0 \\ 1\,0\,0\,1\,0\,1\,1 \end{vmatrix}.$$

Полученная матрица совпадает с полученной ранее проверочной матрицей. Смена порядка на обратный объясняется изменением в (1.29) порядка записи степеней переменной x.

2.3 Обнаружение и исправление ошибок в циклических кодах

Одним из методов декодирования циклических кодов является *алгоритм Меггита*, пригодного для исправления как одиночных, так и l-кратных ошибок ($l \leq 3$). Идея исправления ошибок базируется на том, что ошибочная комбинация после определенного числа циклических сдвигов «подгоняется» под остаток таким образом, что в сумме с остатком она дает исправленную комбинацию. Остаток при этом представляет собой не что иное, как разницу между искаженными и правильными символами, единицы в остатке стоят на местах искаженных разрядов в «подогнанной» циклическим сдвигами комбинации. Подгоняют искаженную комбинацию до тех пор, пока число единиц в остатке не будет равно числу ошибок, которое еще способен исправить данный код. При этом, число исправляемых данным кодом или меньше t единиц может быть равно

числу ошибок t. Место ошибки в кодовой комбинации не имеет значения. Достаточно получить один остаток, вес которого $w \le t$ и этого достаточно для исправления искаженной комбинации.

Обнаружение и исправление ошибок происходит по остаткам от деления принятой комбинации $f(x)$ на порождающий полином $g(x)$. Если принятая комбинация делится на порождающий полином без остатка, то код принят безошибочно. Остаток от деления свидетельствует об ошибке, но не указывает какой именно. Чтобы найти ошибочный разряд и исправить его необходимо осуществить следующие процедуры:

1. Принятая комбинация делится на порождающий полином и находится остаток от деления

$$R(x) = f(x) \bmod g(x). \qquad (2.22)$$

2. Подсчитывается количество единиц в остатке от деления (вес остатка w), если $w \le t$, где t– допустимое число исправляемых данным кодом ошибок, то принятая комбинация складывается по модулю два с полученным остатком. Сумма даст исправленную комбинацию. Если $w > t$, то

3.Делится на порождающий полином $g(x)$, комбинация, полученная в результате циклического сдвига. Если в остатке $w \le t$, то складываем делимое с остатком. Затем производим циклический сдвиг вправо комбинации полученной в результате суммирования последнего делимого с остатком. Полученная комбинация уже не содержит ошибок. Если после первого циклического сдвига и последующего деления остаток получается таким, что его вес $w > t$, то

4. Повторяется процедура 3 до тех пор, пока не будет $w \le t$. В этом случае комбинация, полученная в результате последнего циклического сдвига, суммируется с остатком от деления этой комбинации на порождающий полином, а затем

5. Производится циклический сдвиг вправо на столько разрядов, на сколько была сдвинута суммируемая с последним остатком комбинация относительной принятой. В результате получим исправленную комбинацию.

Пример 2.6. Показать процесс исправления одиночной ошибки в принятой комбинации $f(x)$=1000110, если передавалась кодовая комбинация $v(x)$=1001110 циклического кода (7, 4, 3), способного исправлять однократную ошибку (t = 1) и с порождающим полиномом $g(x) = x^3 + x + 1$ (1011): $f(x) = v(x) + e(x)$ = 1001110 + 0001000 = 1000110.

1. Делим принятую ошибкой комбинацию на образующий полином

```
1000110 | 1011
1011    | 1011
--------|
1111
 1011
-----
1000
  1011
-----
   11
```

Остаток от деления R = 11 с весом w=2.

2. Сравниваем вес полученного остатка w с возможным для данного кода числом исправляемых ошибок t, если $w \le t$ то принятую комбинацию складывают с полученным остатком и полученная сумма дает исправленную комбинацию, если $w > t$, то производят циклический сдвиг принятой комбинации влево на один разряд. Полученную комбинацию делят на порождающий полином: $w > t \rightarrow 2 > 1$.

3. Так как $w > t$ производим циклический сдвиг влево принятой комбинации и делим на $g(x)$:

```
0001101 | 1011
 1011   | 1
--------|
  110
```

Остаток от деления R = 111 с весом w=2, $w > t \rightarrow 2 > 1$.

4. Повторяем процедуру п. 3 до тех пор, пока не будет $w \le t$

```
0011010 | 1011
 1011   | 11
--------|
  1100
  1011
-------
   111
```

Остаток от деления R = 111 с весом w=3, $w > t \rightarrow 3 > 1$.

```
0110100 | 1011
 1011   | 111
-------
  1100
  1011
-------
   1110
   1011
-------
    101
```

Остаток от деления $R = 101$ с весом $w=2$, $w>t \rightarrow 2>1$.

```
1101000 | 1011
1011    | 1111
--------
1100
1011
--------
  1110
  1011
--------
  1010
  1011
--------
     1
```

Остаток от деления $R = 1$ с весом $w=1$, $w = t \rightarrow 1 = 1$.

5. Так как последний полученный остаток равен 1 и выполняется условие $w \leq t$, то складываем последнее делимое с последним остатком

```
  1101000
+       1
---------
  1101001
```

6. Производим циклический сдвиг комбинации, полученной в результате суммирования последнего делимого с последним остатком, вправо на 4 разряда (т.к. перед этим 4 раза сдвигали принятую комбинацию влево):

1) 1101001 →1110100,

2) 1110100 → 0111010,

3) 0111010 → 0011101,

4) 0011101 → 10011101.

Последняя комбинация 1001110 соответствует переданной и не содержит ошибки.

Схема декодирования принятой комбинации показана на рис. 2.5. Принимаемые символы накапливаются в регистре сдвига и одновременно вводятся в схему деления на $g(x)$. После приема седьмого бита содержимое этого регистра сдвигается на один разряд в каждом такте, а схема деления

модифицирует синдромный полином (остаток от деления) и проверяет совпадение с полиномом

$x^6 \bmod (1+x+x^3) = 1+x^2 \rightarrow 101$.

Как только на выходе схемы проверки появится 1, будет исправлена ошибка в позиции x^6. В тот же самый момент исправление вводится по обратной связи в схему деления и, тем самым, обнуляет остаток от деления. Нулевой остаток может рассматриваться как сигнал об успешном завершении декодирования. Проверка на нулевой остаток схемы деления позволяет обнаруживать некоторые аномалии по окончании процедуры декодирования.

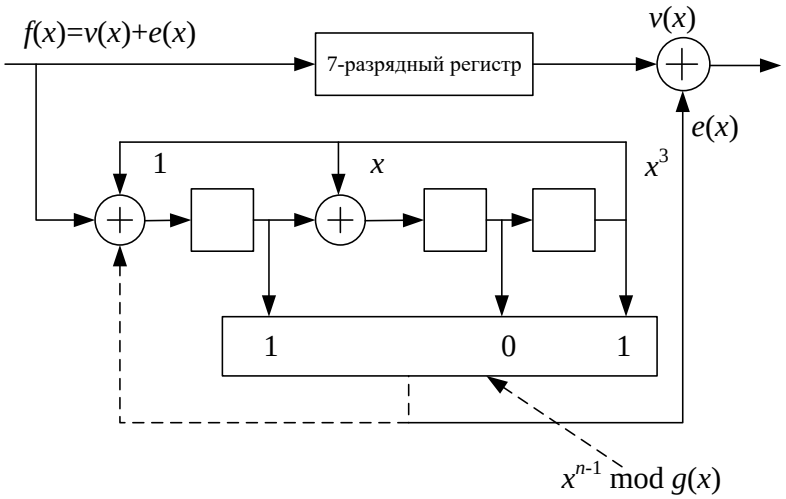

Рисунок 2.5 – Декодер двоичного циклического (7, 4, 3) кода

Общий алгоритм построения и декодирования циклических кодов сводится к следующим стандартным процедурам:

1. Выбор числа контрольных символов r производят как и для кода Хэмминга с исправлением одиночной ошибки по выражению

– если задана длина кода n, то число контрольных разрядов

$$r_{d=3} \geq \log_2(n+1),$$

– если задано число информационных разрядов k, то число контрольных разрядов

$$r_{d=3} \geq \log_2((k+1)+\log_2(k+1)).$$

2. Выбор порождающего полинома $g(x)$ производится по таблицам неприводимых двоичных многочленов (приложение Б). Степень

порождающего полинома не может меньше числа контрольных символов r. Если в приложении имеется ряд многочленов с данной степенью, то из них следует выбирать самый короткий, однако число ненулевых элементов многочлена $g(x)$ не должно быть меньше кодового расстояния d.

3. Построение и выбор параметров единичной порождающей подматрицы происходит из условия, что число столбцов и строк определяется числом информационных разрядов, т.е. ранг единичной матрицы равен k. Дополнительную проверочную подматрицу составляют из остатков, полученных от деления единицы с нулями на образующий полином $g(x)$, при этом должны соблюдаться следующие условия:

− число остатков должно быть равно числу информационных символов k;

− число разрядов в каждом остатке должно быть равно числу контрольных символов r, следовательно, число разрядов проверочной подматрицы должно быть равно степени порождающего многочлена;

− число единиц каждого остатка, т.е. его вес, должно быть не менее величины $w = d_{min}-1$;

− количество нулей, приписываемых к единицы при делении ее на многочлен $g(x)$ таким, чтобы соблюдались предыдущие условия.

4. Порождающая матрица составляется путем дописывания элементов дополнительной матрицы справа от единичной матрицы или путем умножения элементов единичной матрицы на порождающий полином.

5. Нахождение всех комбинаций циклического кода осуществляется путем суммирования по модулю 2 всевозможных сочетаний строк образующей матрицы.

6. Обнаружение и исправление ошибок происходит по остаткам от деления принятой комбинации на порождающий полином, если остаток равен нулю, то код принят безошибочно, иначе выполняется процедура декодирования.

Рассмотренные выше циклические коды имеют минимальное кодовое расстояние $d_{min}= 3$, однако с увеличением корректирующей способности кода $t = \lfloor (d_{\min} - 1)/2 \rfloor$ сложность декодера, основанного на комбинаторном обнаружении ошибок, становится большой.

2.4 Синдромный многочлен

Ошибки в циклических кодах обнаруживаются и исправляются при помощи остатков от деления полученной комбинации на порождающий полином. Остатки от деления являются опознавателями ошибок – *синдромами*, но не указывают непосредственно на место ошибки в циклическом коде

$$s(x) = f(x) \bmod g(x) = e(x) \bmod g(x). \qquad (2.23)$$

Если $s(x) \neq 0$ то это является условием наличия ошибки в принятой последовательности. При наличии в принятой последовательности $f(x)$ хотя бы одной ошибки вектор синдрома $s(x)$ будет иметь, по крайней мере, один нулевой элемент, при этом факт наличия ошибки легко обнаружить, объединив по ИЛИ выходы всех ячеек регистра синдрома.

Поскольку все кодовые слова делятся на $g(x)$, то $S(x)$, очевидно, не зависит от переданного кодового слова, а зависит лишь от комбинации ошибок. Остаток от деления вычисляется с помощью регистра сдвига, обратные связи которого определяются делителем. Если на вход подавать делимое, то частное будет появляться в виде последовательности символов в цепи обратной связи, а остаток будет содержаться в регистре. Например, на рис. 2.3 показано приведение любого многочлена по модулю x^3+x+1. Эта же схема порождает последовательные столбцы проверочной матрицы. Если подавать на вход произвольную последовательность, то схема будет сдвигать ее и суммировать столбцы проверочной матрицы, отвечающие ненулевым элементам входной последовательности. Тогда, по этой схеме может выполняться требуемое матричное умножение.

Еще один вид синдрома используется, если порождающий полином является произведением двух или большего числа неприводимых сомножителей

$g(x) = g_1(x) \cdot g_2(x) \cdot \ldots \cdot g_j(x)$.

Тогда *синдромные компоненты* вектора S равны

$S_1(x) = f(x) \bmod g_1(x)$,

$S_2(x) = f(x) \bmod g_2(x)$,

… … … … …

$S_j(x) = f(x) \bmod g_j(x)$.

Согласно китайской теоремы об остатках для многочленов существует взаимно однозначное соответствие между многочленами $S(x)$ и наборами синдромных компонент $S_1(x)$, $S_2(x)$ и т.д. Эти два представления синдрома содержат одну и ту же информацию. Рассмотрим это на примере. Пусть порождающий полином вычисляется с помощью произведения двух неприводимых многочленов степени 4

$$g(x) = (x^3 + x + 1)(x^3 + x^2 + 1) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.$$

Такой полином порождает код длиной равной 7. Проверочная матрица, соответствующая $S(x)$, может быть найдена последовательными сдвигами регистра с обратными связями, определяемыми многочленом $g(x)$. Она имеет вид

$$H^T = \begin{bmatrix} 1\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,1 \end{bmatrix}.$$

Если задавать синдром как

$S_1(x) = f(x) \bmod (x^3 + x + 1)$,

$S_2(x) = f(x) \bmod (x^3 + x^2 + 1)$,

то соответствующая матрица имеет вид

$$[H']^T = \begin{bmatrix} 1\,0\,0 & 1\,0\,0 \\ 0\,1\,0 & 0\,1\,0 \\ 0\,0\,1 & 0\,0\,1 \\ 1\,1\,0 & 1\,0\,1 \\ 0\,1\,1 & 1\,1\,1 \\ 1\,1\,1 & 1\,1\,0 \\ 1\,0\,1 & 0\,1\,1 \end{bmatrix}.$$

При этом $S_1(x)$ и $S_2(x)$ можно вычислять независимо, используя два отдельных регистра сдвига с обратными связями, задаваемыми $g_1(x)$ и $g_2(x)$. Если синдром был вычислен в виде $S(x)$ мы хотим представить его в виде $S_1(x)$ и $S_2(x)$, то необходимое преобразование можно получить, находя

линейные комбинации строк H, порождающие H'. Если обозначить через h_i i-ю строку H,а через $h_i^{'}$, i-ю строку H', то можно видеть, что

$$h_1^{'} = h_1 + h_4 + h_6,$$
$$h_2^{'} = h_2 + h_4 + h_5 + h_6,$$
$$h_3^{'} = h_3 + h_5 + h_6,$$
$$h_4^{'} = h_1 + h_4 + h_5 + h_6,$$
$$h_5^{'} = h_2 + h_5 + h_6,$$
$$h_6^{'} = h_3 + h_4 + h_5.$$

Обратное преобразование можно получить, решив эту систему уравнений. Оно имеет вид

$$h_1 = h_1' + h_3' + h_6',$$
$$h_2 = h_1' + h_2' + h_3' + h_4' + h_6',$$
$$h_3 = h_1' + h_2' + h_4' + h_5' + h_6',$$
$$h_4 = h_2' + h_5',$$
$$h_5 = h_1' + h_4',$$
$$h_6 = h_2' + h_3' + h_5' + h_6'.$$

Контрольные вопросы

1. Представление циклических кодов в полиномиальном виде.
2. Порождающий полином циклического кода.
3. Раскройте понятие нули циклического кода.
4. Построение циклического кода с минимальным кодовым расстоянием d_{min}и длиной n.
5. Алгоритм кодирование двоичных циклических кодов.
6. Закодировать систематическим кодом (7, 4, 3) информационную последовательность $u(x) = x^3 + x^2 + 1$.
7. Закодировать несистематическим кодом (7, 4, 3) информационную последовательность $u(x) = x^3 + x^2 + 1$.
8. Построить проверочную матрицу для кода (7, 4, 3) с порождающим многочленом $g(x) = x^3+x+1$.
9. Построить проверочную матрицу кода длины 15 с порождающим многочленом $g(x) = x^8+x^7+x^6+x^4+1$.
10. Алгоритм декодирование циклических кодов.
11. Раскройте понятие синдрома ошибки.
12. Нахождения синдрома ошибок.
13. Алгоритм Меггита для исправления ошибок.
14. Раскройте понятие синдромного многочлена.
15. Двоичный циклический код, задаваемый порождающим полиномом $g(x) = x^4+x^3+x^2+1$, позволяет исправлять 2 ошибки. Чему равна длина и минимальное расстояние этого кода.
16. Основные характеристики циклических кодов.

ГЛАВА 3.

КОДЫ БОУЗА-ЧОУДХУРИ-ХОКВИНГЕМА

3.1 Определение циклических кодов Боуза-Чоудхури-Хоквингема

Коды Боуза-Чоудхури-Хоквингема (БЧХ) представляют обширный класс кодов, способных исправлять несколько ошибок. Среди кодов БЧХ при небольших длинах существуют хорошие коды с простыми методами кодирования и декодирования.

Коды БЧХ представляют собой класс линейных циклических кодов, исправляющих кратные ошибки, и являются обобщением известных кодов Хэмминга. Коды БЧХ обычно задаются через корни порождающего многочлена степени $n-k$. Данные коды определяются представленным ниже образом.

БЧХ код длины n, исправляющий t-кратные ошибки, это циклический блочный код над полем $GF(p)$ корнями порождающего многочлена которого являются α^b, α^{b+1}, … α^{b+2t-1}, где α – элемент расширенного конечного поля $GF(p^m)$, b – целое число.

Таким образом, порождающий многочлен (n, k, d_{min}) кода БЧХ может быть представлен наименьшим общим кратным

$$g(x) = \hat{I}\hat{I}\hat{E} \quad \left[M_b(x), M_{b+1}(x), ..., M_{b+2t-1}(x)\right], \tag{3.1}$$

где $M_j(x)$ – минимальные многочлены элементов для элементовα^j.

БЧХ коды имеющие длину $n = p^m-1$ и задаваемые порождающим полиномом $g(x)$ (3.1) называют *примитивными кодами БЧХ*. Наибольшее распространение получили двоичные примитивные БЧХ коды длины $n = 2^m-1$, где m – любое целое число:n = 3, 7, 15, 31, 63, 127, …. , способные исправлять t ошибок. В приложении В даны параметры некоторых двоичных примитивных БЧХ кодов, порождающие многочлены представлены в восьмеричной системе счисления. Однако иногда построенные таким образом коды БЧХ могут исправлять более t ошибок. Поэтому величина $d = 2t + 1$ называется *конструктивным кодовым расстоянием* кода. Истинное минимальное расстояние кода d может быть

больше конструктивного, т.е. ряд кодов БЧХ может исправлять ошибки кратности большей, чем та, которую задают при построении этого кода.

Как правило, для двоичных БЧХ кодов значение степени b первого элемента расширенного поля выбирают $b = 1$, что приводит к порождающему полиному наименьшей степени, т.е. к наименьшему числу избыточных символов в кодовом слове. Тогда корнями порождающего многочлена являются элементы расширенного поля $GF(p^m)$: $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$ и порождающий многочлен (n, k, d_{min}) двоичного БЧХ кода

$$g(x) = \text{НОК}\left[M_1(x), M_2(x), ..., M_{2t}(x)\right]. \qquad (3.2)$$

Набор элементов$\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$называется *нулями кода БЧХ.*

Таким образом, для построения двоичного примитивного БЧХ кода заданной длины $n = p^m - 1$ для некоторого числа m и числа t– количество ошибок, которое необходимо исправить, следует выполнить следующее:

− выбрать примитивный многочлен степени m и построить расширенное поле $GF(p^m)$;

− найти минимальные многочлены $\varphi_j(x)$ для α^j, где $j = 1, \ldots, 2t$;

− найти порождающий полином $g(x) = \text{НОК}\left[\varphi_1(x), \varphi_2(x), ..., \varphi_{2t}(x)\right]$.

Пример3.1.Над полем $GF(2^3)$ порождаемым примитивным многочленом $p(x) = x^3 + x + 1$ строится двоичный (7, 4, 3) БЧХ код с длиной $n = 2^3 - 1 = 7$ и $m = 3, t = 1$. Для нахождения порождающего многочлена $g(x)$ воспользуемся минимальными многочленами $\varphi_i(x)$, найденными для элементов расширенного поля $GF(2^3)$ $\{\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$ в таблице 1.11. Так как количество исправляемых ошибок заданным кодом $t = 1$, то необходимо использовать $2t = 2$ корней: α, α^2. Эти элементы входят в один смежный класс $(\alpha, \alpha^2, \alpha^4)$.Поэтому минимальный многочлен для корней α, α^2 из табл.1.1 является полином третьей степени

$$\varphi_1(x) = \varphi_2(x) = x^3 + x + 1.$$

Таким образом, порождающий полином (7, 4, 3) БЧХ код определяется выражением

$$g(x) = \text{НОК}\left[M_1(x), M_2(x)\right] = \text{НОК}\left[\varphi_1(x), \varphi_2(x)\right] = x^3 + x + 1.$$

Пример 3.2. Построим двоичный БЧХ код длины $n = 2^4 - 1 = 15$ $(m = 4)$, способного исправлять двукратные ошибки $t = 2$. Элементы расширенного поля $GF(2^4)$ и минимальные многочлены представлены для этих элементов в табл. 2.1. Так как количество исправляемых ошибок

заданным кодом $t = 2$, то необходимо использовать $2t = 4$ корней: $\alpha, \alpha^2, \alpha^3, \alpha^4$.

$$g(x) = \text{НОК}\left[x^4 + x^3 + x^2 + x + 1,\ x^4 + x + 1\right] = x^8 + x^7 + x^6 + x^4 + 1.$$

Так как степень многочлена $g(x)$ равна 8, то количество проверочных разрядов в кодовом слове $r = 8$, а количество информационных символов $k = n - r$ и $k = 15 - 8 = 7$. Мы получили код с параметрами (15, 7, 5), способного исправлять 2 ошибки.

Таблица 3.1 – Представление расширенного поля $GF(2^4)$, порождаемое неприводимым многочленом $p(x) = x^4 + x + 1$

Элементы	Представление элементов поля			Минимальные многочлены
	полином	вектор	степень α	
0	0	0	0	
β_0	1	0001	1	$x+1$
β_1	z	0010	α^1	x^4+x+1
β_2	z^2	0100	α^2	x^4+x+1
β_3	z^3	1000	α^3	$x^4 + x^3 + x^2 + x + 1$
β_4	$z+1$	0110	α^4	x^4+x+1
β_5	z^2+x	0110	α^5	x^2+x+1
β_6	$z^3 + z^2$	1100	α^6	$x^4 + x^3 + x^2 + x + 1$
β_7	$z^3+ z+1$	1011	α^7	$x^4 + x^3 + 1$
β_8	$z^2 + 1$	0101	α^8	$x^4 + x + 1$
β_9	$z^3 + x$	1010	α^9	$x^4 + x^3 + x^2 + x + 1$
β_{10}	$z^2 + z + 1$	0111	α^{10}	$x^2+ x + 1$
β_{11}	$z^3 + z^2 + z$	1110	α^{11}	$x^4 + x^3 + 1$
β_{12}	$z^3 + z^2 + z + 1$	1111	α^{12}	$x^4 + x^3 + x^2 + x + 1$
β_{13}	$z^3 + z^2 + 1$	1101	α^{13}	$x^4 + x^3 + 1$
β_{14}	$z^3 + 1$	1001	α^{14}	$x^4 + x^3 + 1$

Как было отмечено ранее элементы расширенного поля α^b, α^{b+1}, … α^{b+2t-1}, являются корнями порождающего многочлена $g(x)$, тогда и для всех кодовых слов $v(x)$, кратных $g(x)$, они также являются корнями

$$v(\alpha^i) = 0, b \le i \le b+2t-1. \qquad (3.3)$$

Таким образом, все кодовые слова удовлетворяют следующей системе $2t$ уравнений в матричной форме

$$(v_0, v_1, ..., v_{n-1}) \cdot \begin{vmatrix} 1 & 1 & ... & 1 \\ \alpha^b & \alpha^{b+1} & \cdots & \alpha^{b+2t-1} \\ \alpha^{2b} & \alpha^{2(b+1)} & \cdots & \alpha^{2(b+2t-1)} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha^{(n-1)b} & \alpha^{(n-1)(b+1)} & \cdots & \alpha^{(n-1)(b+2t-1)} \end{vmatrix} = 0. \qquad (3.4)$$

Для двоичных БЧХ кодов со значением $b = 1$ получим

$$(v_{0,} v_1, ..., v_{n-1}) \cdot \begin{vmatrix} 1 & 1 & ... & 1 \\ \alpha^1 & \alpha^2 & ... & \alpha^{2t} \\ \alpha^2 & \alpha^4 & ... & \alpha^{4t} \\ \vdots & \vdots & \cdots & \vdots \\ \alpha^{n-1} & \alpha^{2(n-1)} & \cdots & \alpha^{2t(n-1)} \end{vmatrix} = 0. \qquad (3.5)$$

Такая запись эквивалентна $vH^T = 0$, где проверочная матрица двоичного БЧХ кода с $b = 1$ имеет вид

$$H = \begin{vmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \alpha^{2t} & \alpha^{4t} & \cdots & \alpha^{2t(n-1)} \end{vmatrix}. \qquad (3.6)$$

Нижней границей минимального расстояния БЧХ кодов является: если порождающий многочлен $g(x)$ циклического (n, k, d) кода имеет l последовательных корней, например α^b, α^{b+1}, … α^{b+l}, то минимальное расстояние

$$d \geq 2l + 1. \qquad (3.7)$$

Если длина кода nявляется делителем $2^m - 1$, но не совпадает с этим числом, то код называется непримитивным. В таблице 3.2 приведены параметры некоторых *непримитивных двоичных циклических кодов* и даны порождающие полиномы в восьмеричной форме.

Таблица 3.2 – Непримитивные двоичные БЧХ коды

m	***n***	***k***	***t***	Порождающий полином в восьмеричной форме
8	17	9	2	727
6	21	12	2	1663
11	23	12	3	5343
10	33	22	2	5145
10	33	12	4	3777
20	41	21	4	6647133
23	47	24	5	42073357
12	65	63	2	10761
12	65	40	4	354303067
9	73	46	4	1717773537

Одним из представителей непримитивных циклических кодов является код Голея, открытый в 1949 г. еще до появления теории кодирования. Это код с длиной $n = 23$, $k = 12$ способен исправлять тройные ошибки. Порождающий полином $g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$ является делителем бинома $x^{23}-1$, поэтому его корень β принадлежит показателю 23, т.е. $\beta^{23}=1$. Циклотомический класс C_β состоит из 11 элементов $\{\alpha^1, \alpha^2,\alpha^4, \alpha^8,\alpha^{16}, \alpha^9,\alpha^{18}, \alpha^{13}, \alpha^3, \alpha^6,\alpha^{12}\}$. В ряду корней имеются только 4 последовательные степени $\alpha^1, \alpha^2,\alpha^3, \alpha^4$, поэтому конструктивное расстояние этого кода равно 5, а действительное значение минимального расстояния равно 7. Код Голея (23, 12, 7) занимает уникальное место в теории кодирования, в силу хорошей упаковки сфер, однако на практике из-за малой длины кода не нашел конкретных приложений.

3.2 Кодирование БЧХ кодов

Линейность циклических БЧХ кодов означает применимость к ним всех методов кодирования и декодирования, рассмотренных в предыдущей главе.

Поскольку любой кодовый полином $v(x)$ *несистематического циклического БЧХ кода* есть произведение информационного $u(x)$ и порождающего$g(x)$ полиномов (2.10), компоненты соответствующего вектора $v(x)$ являются сверткой информационной последовательности $u_0, u_1, \ldots, u_{k-1}$ с последовательностью коэффициентов порождающего

полинома g_0, g_1, …, $g_{r\text{-}1}$. Говорят, что коэффициенты многочленов $u(x)$ и $g(x)$ свертываются регистром сдвига, так как

$$v_j = \sum_{i=0}^{r} g_i u_{j-i} . \qquad (3.8)$$

Подобная операция реализуется фильтром с конечной импульсной характеристикой (КИХ-фильтром), представленным на рис. 3.1.

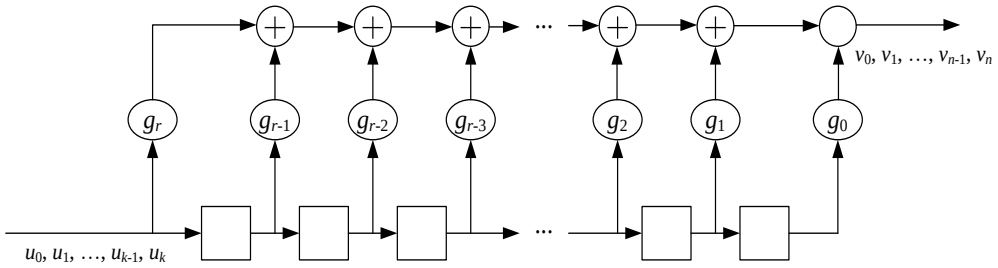

Рисунок 3.1 – Линейный регистр сдвига без обратной связи

На рис. 3.2 приведен пример умножения информационной последовательности на полином $g(x)$ для $g(x) = x^8+x^7+x^4+x^2+x+1$.

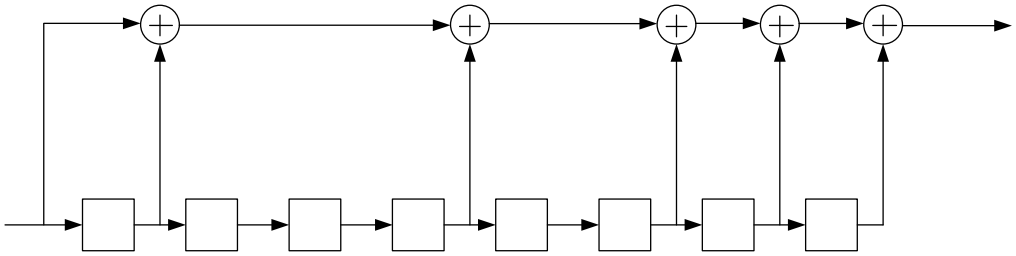

Рисунок 3.2 – Устройство умножения на многочлен $x^8+x^7+x^4+x^2+x+1$

Для построения циклических *систематических БЧХ кодеров* целесообразно использовать регистры сдвига с линейной обратной связью, что открывает дополнительные возможности упрощения кодера в аппаратной реализации. Можно интерпретировать n символов, содержащихся в регистре длины n как коэффициенты многочлена степени $n-1$. Для циклического сдвига многочлена используется замкнутый в кольцо регистр сдвига. На рис. 2.1 изображен n-разрядный регистр сдвига

с линейной обратной связью, используемый для циклического сдвига многочлена степени $n-1$, который вычисляет $x \cdot v(x) \bmod (x^n-1)$.

Согласно п. 2.2 ключевой операцией в построении систематического кодового полинома циклического БЧХ кода является нахождение остатка $R(x)$от деления произведения $u(x) \cdot x^r$ на порождающий полином $g(x)$ (2.11). Регистр сдвига можно также использовать для деления произвольного многочлена на фиксированный многочлен. Выполняющая эту операцию цепь почти повторяет обычную процедуру деления многочленов. Чтобы понять, каким образом структура, представленная далее, вычисляет $R(x)$ разделим $u(x) \cdot x^r = u_{k-1}x^{n-1}+u_{k-2}x^{n-2}+\ldots+u_0x^{n-k}$ на $g(x) = x^r+g_{r-1}x^{r-1}+\ldots+g_0$ используя правило длинного деления.

Первая итерация

$$\begin{array}{l|l} u_{k-1}x^{n-1} & x^r+g_{r-1}+\ldots+g_0 \\ \underline{u_{k-1}x^{n-1}+u_{k-1}g_{r-1}x^{n-2}+\ldots+u_{k-1}g_0x^{n-r-1}} & \overline{u_{k-1}x^{n-r-1}} \\ r_{n-2}x^{n-2}+r_{n-3}x^{n-3}+\ldots+r_{n-r-1}x^{n-r-1}=r_1(x) & \end{array}$$

Вторая итерация

$$\begin{array}{l|l} (r_{n-2}+u_{k-2})x^{n-2}+r_{n-3}x^{n-3}+\ldots+r_{n-r-1}x^{n-r-1} & x^r+g_{r-1}+\ \ldots\ +g_0 \\ \underline{(r_{n-2}+u_{k-2})x^{n-2}+\ldots+(r_{n-2}+u_{k-2})g_0x^{n-r-2}} & \overline{(r_{n-2}+u_{k-2})x^{n-r-2}} \\ r_{n-3}x^{n-3}+r_{n-4}x^{n-4}+\ldots+r_{n-r-2}x^{n-r-2}=r_2(x) & \end{array}$$

На первой итерации $g(x)$ умножается на $u_{k-1}x^{n-r-1}$ и результат вычитается из делимого. В итоге первый остаток $r_1(x)=r_{n-2}x^{n-2}+r_{n-3}x^{n-3}+\ldots+r_{n-r-1}x^{n-r-1}$ имеет степень не более $n-1$ и после сложения с $u_{k-2}x^{n-2}$используется на второй итерации как новое делимое. При выполнении второй итерации из него вычитается $(r_{1,n-2}+u_{k-2})x^{n-r-2}\ g(x)$, давая второй остаток $r_2(x)=r_{n-3}x^{n-3}+r_{n-4}x^{n-4}+\ldots+r_{n-r-2}x^{n-r-2}$, степень которого не превышает $n-3$. Остаток $r_2(x)$ вновь суммируется с $u_{k-3}x^{n-3}$ и используется как делимое на третьей итерации и т.д. Легко видеть, что на i-й итерации предыдущий остаток, сложенный с $u_{k-i}x^{n-i}$ служит делимым, из которого вычитается произведение полинома $g(x)$ на x в степени, уравнивающей степени вычитаемого и делимого, а также на коэффициент, равный старшему коэффициенту делимого. Результатом этого оказывается очередной остаток, с которым производится те же действия на следующей итерации. После k итерации k-й остаток имеет степень, меньшую r, являясь требуемым остатком $r(x)$.

Таким образом, на k-м шаге вычисляются рекуррентные равенства для частного $q(x)$ и остатка от деления $R(x)$

$$\begin{aligned} q^{(r)}(x) &= q^{(r-1)}(x) + R_{n-r}^{(r-1)} x^{k-r}, \\ R^{(r)}(x) &= R^{(r-1)} + R_{n-r}^{(r-1)} x^{k-r} g(x). \end{aligned} \tag{3.9}$$

Подобное деление реализуется систематическим циклическим кодером, показанным на рис. 3.3. и основанным на регистре сдвига с линейной обратной связью.

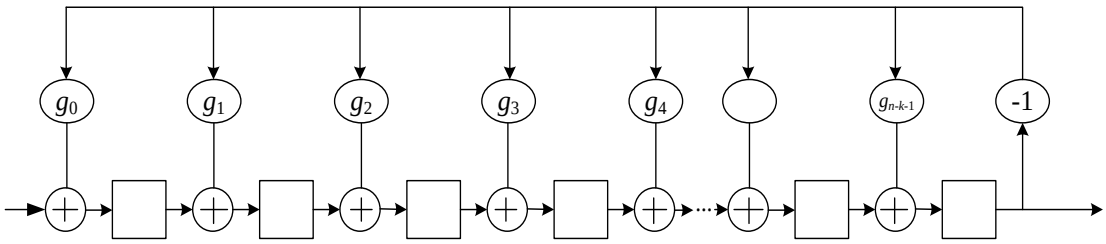

Рисунок 3.3 –Устройства деления на многочлен $g(x)$

Изображенная на рис. 3.3 цепь является цепью деления произвольного многочлена на фиксированный многочлен $g(x)$. Единственной не отраженной на цепи операцией является вычитание члена $R_{n-r}x^{n-r}$ из самого себя, выполнять которую не надо, так как результат всегда равен нулю. После n сдвигов на выходе регистра буде вычислено частное, а в регистре окажется записанным остатком от деления. На рис. 3.4 приводится пример, реализующий деление произвольного многочлена на многочлен $g(x)=x^8+x^7+x^4+x^2+x+1$.

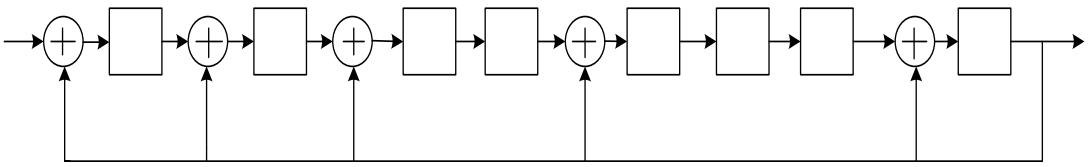

Рисунок 3.4 – Устройство деления на многочлен $g(x)=x^8+x^7+x^4+x^2+x+1$

Алгоритм кодирования двоичным БЧХ кодом заключается в следующем:

1. Задание параметров кодера

   t – количество исправляемых ошибок кодом;

   n – длина кода, которая равна 2^m-1;

   b–значение степени элемента α^b расширенного поля $GF(2^m)$, с которого начинается перечисление $2t$ корней α^b, α^{b+1}, … α^{b+2t-1}.

2. Нахождение минимального кодового расстояния $d_{min} = 2t-1$.

3. Построение расширенного поля$GF(2^m)$ с заданным примитивным многочленом $p(x)$.

4. Нахождение всех минимальных многочленов для каждого элемента расширенного поля $GF(2^m)$: $\varphi_j(x)$ для α^j.

5. Нахождение минимальных многочленов $[\varphi_1(x), \varphi_2(x),..., \varphi_{2t}(x)]$ для корней α^b, α^{b+1}, … α^{b+2t-1}.

6. Вычисление порождающего полинома $g(x)$как наименьшего общего кратного минимальных многочленов $g(x) = \acute{I}\hat{I}\hat{E}\ [\varphi_1(x), \varphi_2(x),..., \varphi_{2t}(x)]$.

7. Нахождение количества проверочных символов $r = \deg[g(x)]$.

8. Нахождение количества информационных символов $k = n - r$.

9. Задание информационной последовательность $u(x)$ длины k.

10. Кодирование информационной последовательности $u(x)$ построенным БЧХ кодом:

− несистематический БЧХ код $v(x) = u(x)g(x)$;

− систематический БЧХ код $v(x) = u(x)x^r + R(x)$, $R(x) = u(x)\cdot x^r \bmod g(x)$.

Пример 3.3. Построить двоичный БЧХ код длины $n = 2^m-1 = 15$ ($m = 4$), способного исправлять трёхкратные ошибки ($t = 3$). Расширенное поле $GF(2^4)$ порождаемое неприводимым и примитивным многочленом $p(x) = x^4 + x + 1$ представлено в табл. 2.1. Закодировать несистематическими систематическим двоичным БЧХ кодом информационную последовательность $u(x) = x^4+x^3+x+1$.

Используя алгоритм кодирования двоичным БЧХ кодом вычислим:

1. Параметры кода: $t = 3$, $n = 15$, $b = 1$.

2. Минимальное кодовое расстояние $d_{min} = 2t+1 = 2\cdot3+1 = 7$.

3. Расширенное поле $GF(2^4)$ порождаемое неприводимым и примитивным многочленом $p(x) = x^4 + x + 1$ представлено в табл. 2.1.

4. Минимальные многочлены для элемента расширенного поля

$\alpha^0 - x+1$;
$\alpha^1 - x^4+x+1$;
$\alpha^2 - x^4+x+1$;
$\alpha^3 - x^4+x^3+x^2+x+1$;
$\alpha^4 - x^4+x+1$;
$\alpha^5 - x^2+x+1$;
$\alpha^6 - x^4 + x^3 + x^2 + x + 1$;
$\alpha^7 - x^4 + x^3 + 1$;
$\alpha^8 - x^4 + x + 1$;
$\alpha^9 - x^4 + x^3 + x^2 + x + 1$;
$\alpha^{10} - x^2 + x + 1$;
$\alpha^{11} - x^4 + x^3 + 1$;
$\alpha^{12} - x^4 + x^3 + x^2 + x + 1$;
$\alpha^{13} - x^4 + x^3 + 1$;
$\alpha^{14} - x^4 + x^3 + 1$

5. Так как количество исправляемых ошибок заданным кодом $t = 3$ и $b = 1$, то необходимо использовать элементы поля α^b, α^{b+1}, …,$\alpha^{b+2t-1} = \alpha^1$, α^2, α^3,α^4, α^5, α^6. Минимальные многочлены для соответствующих корней:

– α, α^2, $\alpha^4 \rightarrow \varphi_1(x) = x^4 + x + 1$;
– α^3,$\alpha^6 \rightarrow \varphi_3(x) = x^4 + x^3 + x^2 + x + 1$;
– $\alpha^5 \rightarrow \varphi_5(x) = x^2 + x + 1$.

6. Порождающий полином двоичного БЧХ кода длины $n = 15$ определяется выражением

$$g(x) = НОК\left[M_1(x), M_2(x), M_3(x)\right] = НОК\left[\varphi_1(x), \varphi_3(x), \varphi_5(x)\right],$$

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

7. Количества проверочных символов $r = \deg[g(x)] \rightarrow r = 10$.

8. Количества информационных символов $k = n - r = 15 - 10 = 5$.

9. Кодирование информационной последовательности $u(x)=x^4+x^3+x+1$ несистематическим БЧХ кодом

$$v(x) = u(x)g(x).$$

$$v(x) = (x^4 + x^3 + x + 1)(x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1) = x^{14} + x^{13} + x^{11} + x^{10} +$$
$$+ x^{12} + x^{11} + + x^9 + x^8 + x^9 + x^8 + x^6 + x^5 + x^8 + x^7 + x^5 + x^4 + x^6 + x^5 +$$
$$+ x^3 + x^2 + x^5 + x^4 + x^2 + + x + x^4 + x^3 + x + 1 =$$
$$= x^{14} + x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^4 + 1.$$

10. Кодирование информационной последовательности $u(x)=x^4+x^3+x+1$ систематическим БЧХ кодом

$$v(x) = u(x)x^r + R(x),\ R(x) = u(x)\cdot x^r \bmod g(x).$$

$u(x)\cdot x^r = (x^4+x^3+x+1)\cdot x^{10} = x^{14}+x^{13}+x^{11}+x^{10}$;

$R(x) = u(x)\cdot x^r \bmod g(x) = (x^{14}+x^{13}+x^{11}+x^{10}) \bmod (x^{10}+x^8+x^5+x^4+x^2+x+1)$

$x^{14}+x^{13}+x^{11}+x^{10}$	$x^{10}+x^8+x^5+x^4+x^2+x+1$
$x^{14}+x^{12}+x^9+x^8+x^6+x^5+x^4$	x^4+x^3+1
$x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^6+x^5+x^4$	
$x^{13}+x^{11}+x^8+x^7+x^5+x^4+x^3$	
$x^{10}+x^9+x^7+x^6+x^3$	
$x^{10}+x^8+x^5+x^4+x^2+x+1$	
$x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$	

$R(x) = x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$,

$v(x) = u(x)x^r + R(x) = x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$.

3.3 Декодирование БЧХ кодов

Коды БЧХ являются циклическими, и, следовательно, к ним применимы любые методы декодирования циклических кодов. Однако существуют существенно лучшие алгоритмы, которые специально разработаны для декодирования БЧХ кодов. Исторически первый метод декодирования был найден Питерсоном в 1960 г. для двоичных кодов, а также Горенстейном и Цирлером в 1961 г. для недвоичных кодов. Существенное продвижение в сторону упрощения алгоритма было найдено Берлекэмпом в 1968 г. и в дальнейшем усовершенствован Месси в 1969 г. Более ясный метод декодирования БЧХ кодов основанный на использовании алгоритма Евклида предложил Сугияма, Касахара, Хирасава и Намекава в 1975 г.

Для описания принципов декодирования БЧХ кодов введем основные обозначения, используемые при работе алгоритмов исправления ошибок в принятой кодовой последовательности $f(x)$, если передано кодовое слово $c(x)$.

Пусть при передаче кодовой последовательности $c(x)$ произошло v ошибок и был принят полином

$$f(x) = c(x) + e(x), \tag{3.10}$$

где $e(x)$ – многочлен ошибок, представляющий

$$e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + ... + e_{j_v} x^{j_v}, \qquad (3.11)$$

где e_{j_l} – величина *l*-ой ошибки (если двоичный БЧХ код $e_{j_l} = 1$), *v* – число ошибок в принятом слове ($0 \le v \le t$).

Декодеру неизвестно ни позиции ошибок – j_1, j_2, … j_v, ни значения ошибок $-e_{j_1}, e_{j_2}, ..., e_{j_v}$, ни количество ошибок *v*, которые произошли при передаче кодовой последовательности. Для исправления ошибок необходимо найти значения всех этих чисел.

Главной идей в декодировании БЧХ кодов является использование элементов конечного поля для нумерации позиций кодового слова. Например, для принятого кодового вектора $f(x) = (f_0, f_1, \ldots, f_{n-1})$ соответствует нумерация позиций кодового слова элементам поля $GF(p^m)$. Нумерация этих позиций называются *локаторами позиций*.

Значения кодового вектора	$\boldsymbol{f_0}$	$\boldsymbol{f_1}$	$\boldsymbol{f_2}$	…	$\boldsymbol{f_{n-1}}$
Локаторы позиций	$\mathbf{1}$	$\boldsymbol{\alpha^1}$	$\boldsymbol{\alpha^2}$	…	$\boldsymbol{\alpha^{n-1}}$

Поскольку α^i – локатор *i*-ой позиции α^i различны для всех *i*, то локаторы и позиции однозначно определяют друг друга. Таким образом в многочлене ошибок $e(x)$ позиции ошибок j_1, j_2, … j_v, могут быть обозначены через *локаторы ошибок*: α^{j_k} – локатор *k*-ой ошибки, $k = 1, 2, \ldots t$.

Тогда, необходимо найти следующие множества:

$\{e_{j_1}, e_{j_2}, ..., e_{j_v}\}$ – значения ошибок, где e_j – значения ошибок, которые принадлежат $\{0, 1\}$для двоичных БЧХ кодов.

$\{\alpha^{j_1}, \alpha^{j_2}, ..., \alpha^{j_v}\}$ – локаторы ошибок, где элементы α^iпринадлежат расширенному полю $GF(p^m)$.

Как было указано ранее элементы поля $GF(p^m)$ являются нулями кода в точках $f(\alpha^i) = 0$, если они принадлежат корням порождающего полинома $g(x)$, т.е. для любых α^i, являющихся корнями $g(x)$, выполняется равенство

$$f(\alpha^j) = v(\alpha^j) + e(\alpha^j) = e(\alpha^j). \qquad (3.12)$$

Таким образом

$$f(\alpha^j) = \sum_{j=0}^{n-1} e_j \alpha_i^j,$$

где $i = 1, 2, \ldots r$, для всех α^j являющихся корнями $g(x)$.

Значения принятого кодового многочлена $f(x)$ в нулях кода называются *синдромами* и обозначаются S_j. *Синдром* принятого кодового слова $f(x)$ состоит из $2t$ компонент: $S_j = f(\alpha^i)$ для $j = 1, 2, \ldots 2t$, $i = b, b+1, \ldots, b+2t-1$, при этом не зависит от переданного слова и определяется только полиномом ошибок

$$S_j = f(\alpha^i) = v(\alpha^i) + e(\alpha^i) = e(\alpha^i),$$

где $e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + \ldots + e_{j_v} x^{j_v}$.

Тогда, с учетом корней порождающего полинома – нулей кода (3.3), получим следующую систему из $2t$ уравнений относительно v неизвестных локаторов ошибок α^j и v неизвестных величин ошибок e_j

$$\begin{aligned}
S_1 &= f(\alpha^b) = e_{j_1}\alpha^{bj_1} + e_{j_2}\alpha^{bj_2} + \ldots + e_{j_v}\alpha^{bj_v} \\
S_2 &= f(\alpha^{b+1}) = e_{j_1}\alpha^{(b+1)j_1} + e_{j_2}\alpha^{(b+1)j_2} + \ldots + e_{j_v}\alpha^{(b+1)j_v} \\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
S_{2t} &= f(\alpha^{b+2t-1}) = e_{j_1}\alpha^{(b+2t-1)j_1} + e_{j_2}\alpha^{(b+2t-1)j_2} + \ldots + e_{j_v}\alpha^{(b+2t-1)j_v}
\end{aligned} . \qquad (3.13)$$

Для двоичного БЧХ кода со значением $b = 1$ получим компоненты синдрома

$$\begin{aligned}
S_1 &= e_{j_1}\alpha^{j_1} + e_{j_2}\alpha^{j_2} + \ldots + e_{j_v}\alpha^{j_v} \\
S_2 &= e_{j_1}\alpha^{2j_1} + e_{j_2}\alpha^{2j_2} + \ldots + e_{j_v}\alpha^{2j_v} \\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
S_{2t} &= e_{j_v}\alpha^{2tj_1} + e_{j_2}\alpha^{2tj_2} + \ldots + e_{j_v}\alpha^{2tj_v}
\end{aligned} . \qquad (3.14)$$

С помощью полученной системы уравнений зная значения компонентов синдрома $S = \{S_1, S_2, \ldots, S_{2t}\}$, можно определить число ошибок v, их локаторы α^j и их величины e_j. При этом не трудно отыскать компоненты синдромного вектора по принятой последовательности. Вычисление синдромов БЧХ кода S_j по принятой последовательности $f(x)$ осуществляется путем деления многочлена $f(x)$ на минимальные многочлены $\varphi_i(x)$. В результате деления находятся остатки от деления

$$R_j(x) = f(x) \bmod \varphi_j(x), j = 1, 2, \ldots 2t. \qquad (3.15)$$

Далее компоненты синдрома находятся как значения остатков в нулях кода $\{\alpha^j\}$

$$S_j = R_j(\alpha^j). \tag{3.16}$$

При этом компоненты S_j с нечетными номерами (j = 1, 3, … $2t-1$) вычисляются по формуле (3.16), а компоненты S_j с четными номерами (j = 2, 4, … $2t$) вычисляются

$$S_{2j} = S_j^2. \tag{3.17}$$

Однако систему уравнений (3.14) трудно решать непосредственно, поэтому воспользуемся искусственным приемом, определив некоторые промежуточные переменные, которые могут быть вычислены по компонентам синдрома и по которым можно вычислить затем локаторы ошибок. Для этого введем *многочлен локаторов ошибок* от переменной x:

$$\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + ... + \sigma_v x^v. \tag{3.18}$$

Многочлен локаторов ошибок определяется как многочлен, корнями которого являются величины *обратные к локатором ошибок* α^j

$$\sigma(x) = (1 - x\alpha^{j_1})(1 - x\alpha^{j_2}) ... (1 - x\alpha^{j_v}) = \prod_{l=1}^{v} (1 + \alpha^{j_l}). \tag{3.19}$$

Если известны коэффициенты многочлена $\sigma(x)$ то для вычисления локаторов ошибок нужно найти его корни. Поэтому по заданным компонентам синдрома необходимо вычислить коэффициенты $\sigma_1, \sigma_2, \ldots \sigma_v$.

Уравнения, связывающие синдромные компоненты и коэффициенты полинома локаторов ошибок

$$S_{j+v} + \sigma_1 S_{j+v-1} + \sigma_2 S_{j+v-2} + ... + \sigma_v S_j = 0, j = 1,\ldots v. \tag{3.20}$$

Таким образом, получаем систему уравнений

$$\sigma_1 S_{j+v-1} + \sigma_2 S_{j+v-2} + ... + \sigma_v S_j = -S_{j+v}, j = 1,\ldots v,$$

т.е. систему линейных уравнений, связывающих компоненты синдрома с коэффициентами многочлена $\sigma(x)$. Эту систему можно записать в матричном виде

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_{v-1} & S_v \\ S_2 & S_3 & S_4 & \cdots & S_v & S_{v+1} \\ S_3 & S_4 & S_5 & \cdots & S_{v+1} & S_{v+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ S_v & S_{v+1} & S_{v+2} & \cdots & S_{2v-2} & S_{2v-1} \end{bmatrix} \begin{bmatrix} \sigma_v \\ \sigma_{v-1} \\ \sigma_{v-2} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ -S_{v+3} \\ \vdots \\ -S_{2v} \end{bmatrix}. \qquad (3.21)$$

Полученная система уравнений (3.14) получила название – *ключевое уравнение декодирования БЧХ кодов*. Решая ключевое уравнение (3.21) мы получим коэффициенты полинома локаторов ошибок $\sigma(x)$. Позиции ошибок $j_1, j_2, \ldots j_v$, находятся как элементы поля $GF(p^m)$, обратные корням полинома локаторов ошибок $\sigma(x)$. По найденным локаторам определяются величины ошибок из системы уравнений (3.14).

Решение ключевого уравнения требует интенсивных вычислений в процедуре декодирования БЧХ кодов. В настоящее время известны следующие методы решения ключевого уравнения:

1. *Алгоритм Берлекемпа-Мэсси*.

2. *Евклидов алгоритм*.

3. Прямое решение – декодер *Питерсона-Горенстейна-Цирлера*.

Далее в подразделах будут рассмотрены все алгоритмы решения ключевого уравнения.

Таким образом, общий алгоритм декодирования циклических БЧХ кодов состоит из реализации следующих задач:

1. Вычислить значения компонентов синдрома S_j принятого полинома $f(x)$ в нулях кода $S_j = f(\alpha^i)$ ($j = 1, 2, \ldots, 2t$, $i = b, b+1, \ldots, b+2t-1$).

2. Найти коэффициенты $\sigma_1, \sigma_2, \ldots \sigma_v$ многочлена локаторов ошибок и построить полином локаторов ошибок $\sigma(x)$ используя один из алгоритмов решения ключевого уравнения (BMA, EA, PGZ).

3. Найти корни полинома локаторов ошибок, которыми являются элементыα^i расширенного поля $GF(p^m)$. Для поиска корней $\sigma(x)$ используется *метод Ченя* – последовательная подстановка всех ненулевых элементов α^i ($i = 0, 1, \ldots 2^m-2$) и проверка условия $\sigma(\alpha^i)=0$.

4. Найти значения позиций ошибок j_1, j_2 … j_vкак обратные величины найденным корням $\alpha^i=\alpha^{-j}$. Обратный элемент $\beta^{-1}= \alpha^b$ элемента $\beta= \alpha^k$ определяется как $b = 2^m – 1 – k$. Также обратные элементы могут быть найдены с помощью таблицы умножения элементов поля $GF(p^m)$.

5. Найти значения ошибок (этап ненужный для двоичных кодов).

6. Исправить принятое слово на вычисленных позициях для вычисления значений ошибок.

3.4 Алгоритм Питерсона-Горенстейна-Цирлера (PGZ)

Решение ключевого уравнения (3.21) и нахождение коэффициентов σ_iполинома локаторов ошибок $\sigma(x)$ может быть реализовано с помощью стандартной техники решения системы линейных уравнений. Однако заранее неизвестно число ошибок в принятом слове. Поэтому необходимо проверять гипотезу о том, что действительное количество ошибок в принятом слове равно v.

Декодер Питерсона начинает с предположения о том, что произошло максимальное число ошибок $v_{\max} = t$. Далее вычисляется определитель матрицы $\det M_i$, для $i = v_{\max} = t$

$$M_i = \det\begin{bmatrix} S_1 & S_2 & \cdots & S_i \\ S_2 & S_3 & \cdots & S_{i+1} \\ \vdots & \vdots & \vdots & \vdots \\ S_i & S_{i+1} & \cdots & S_{2i-1} \end{bmatrix}. \qquad (3.22)$$

Полученной значение определителя матрицы сравнивается с нулем. Если определитель равен нулю, то число ошибок меньше, чем предполагалось. Значение i уменьшается на единицу и снова проверяется определитель матрицы $\det M_i$. Процедура повторяется до тех пор пока $i>1$. Как только окажется, что определитель не равен нулю, то мы получили правильное значение для v . Затем вычисляется обратная матрица для матрицы синдромов и находятся значения коэффициентов полинома $\sigma_1, \sigma_2, \ldots \sigma_v$, где $v = i$. Если определитель равен нулю для $i = 1, 2, \ldots t$, то декодирование считается безуспешным и регистрируется обнаружение неисправляемой ошибки.

Таким образом, алгоритм PGZ для декодирования БЧХ кода и решения ключевого уравнения состоит из следующих этапов (рис. 3.1):

1. На вход алгоритма поступает принятое кодовое слово $f(x)$.

2. Вычисляем компоненты синдрома

$S_j = f(\alpha^j)$, $j = 1, \ldots 2t$, α^j – нули кода и $i = b, b+1, \ldots, b+2t-1$.

3. Полагаем $i = v_{\max} = t$.

4. Строим матрицу из компонентов синдрома S_i для нахождения коэффициентов многочлена локаторов ошибок

$$M_i = \det \begin{bmatrix} S_1 & S_2 & \cdots & S_i \\ S_2 & S_3 & \cdots & S_{i+1} \\ \vdots & \vdots & \vdots & \vdots \\ S_i & S_{i+1} & \cdots & S_{2i-1} \end{bmatrix}$$

5. Вычисляем определитель матрицы M_i. Если он равен нулю, уменьшаем i на единицу и возвращаемся к шагу 4, иначе пункт 6.

6. Устанавливаем количество ошибок $v = i$.

7. Обращаем матрицу M и вычисляем коэффициенты многочлена $\sigma(x)=\sigma_1, \sigma_2, \ldots \sigma_v$

$$\begin{bmatrix} \sigma_v \\ \sigma_{v-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = M^{-1} \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ \vdots \\ -S_{2v} \end{bmatrix}.$$

8. Вычисляем корни многочлена $\sigma(x)$. Эта процедура заключается в последовательном вычислении $\sigma(\alpha^i)$ для каждого i и проверки полученных значений на нуль $\sigma(\alpha^i)=0$.

9. Находим локаторы ошибок с помощью полученных значений α^i, как обратные корням многочлена локаторов ошибок $\alpha^j = \alpha^{-i}$.Степени локаторов ошибокα^j являются значениями позиций ошибок $j_1, j_2 \ldots j_v$.

10. Если БЧХ код – двоичный, то ошибки известны и исправление заключается в инвертировании разрядов на позициях $j_1, j_2 \ldots j_v$. В противном случае вычисляются значения ошибок с помощью алгоритма Форни.

11. Исправляем ошибки и получаем кодовое слово $c(x)$.

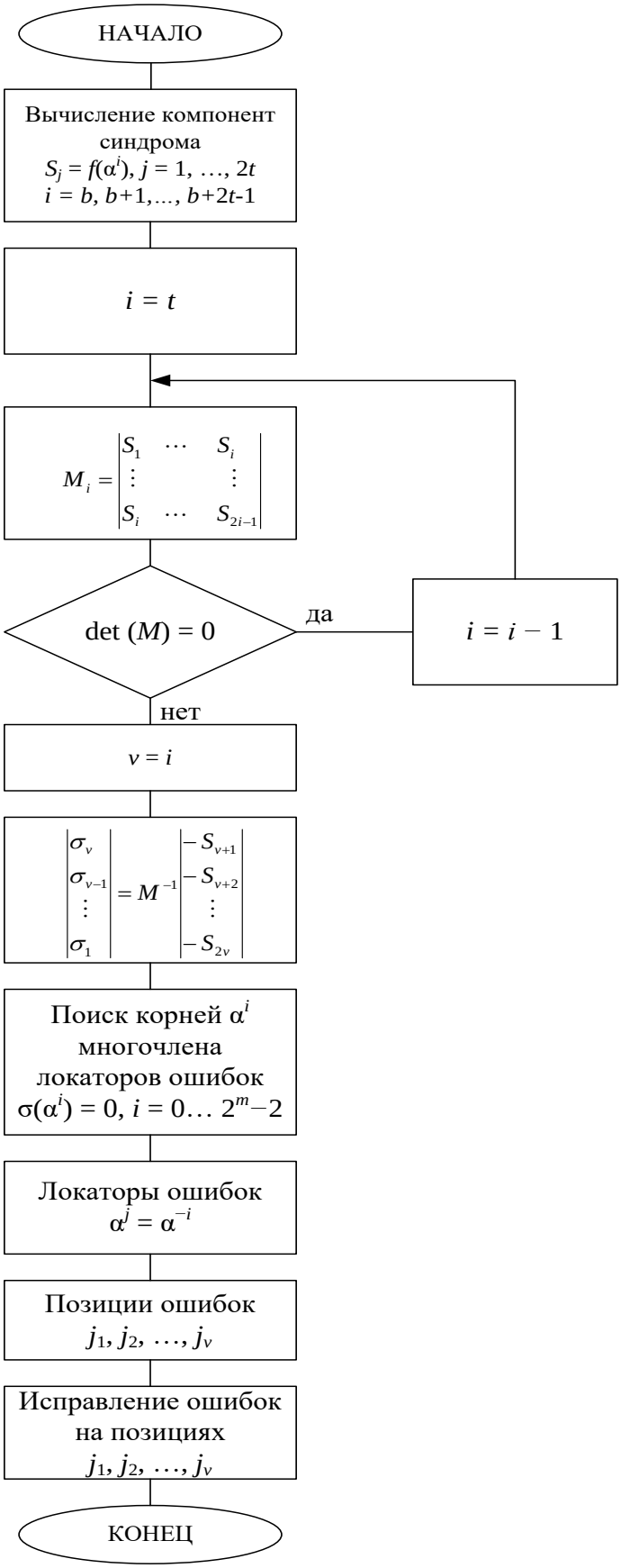

Рисунок 3.1 –Алгоритм Питерсона-Горенстейна-Цирлера

Пример 3.4. Рассмотрим процедуру декодирования БЧХ кода с параметрами (15, 5, 7) исправляющего $t = 3$ ошибки и имеющего порождающий полином $GF(2^4)$

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Пусть информационная последовательность имеет вид $u(x) = x^4 + x^2 + x$. При систематическом кодировании кодовая последовательность $c(x) = u(x)x^r + R(x)$ $= x^{14}+x^{12}+x^{11}+x^8+x^4+x^3+x^2+x$. Для примера будем считать что произошло две ошибки и вектор ошибки равен $e(x) = x^7 + x^2$. Следовательно принятое кодовое слово $f(x) = x^{14}++x^{12}+x^{11}+x^8+x^7+x^4+x^3+x$. Декодером были получены компоненты синдрома с помощью принятого кодового слова и табл. 1.Г (приложение Г). сложения элементов в расширенном поле $GF(2^4)$:

$S_1=f(\alpha)=(\alpha)^{14}+(\alpha)^{12}+(\alpha)^{11}+(\alpha)^8+(\alpha)^7+(\alpha)^4+(\alpha)^3+(\alpha)=\alpha^{12}$.

$S_2=f(\alpha^2)=(\alpha^2)^{14}+(\alpha^2)^{12}+(\alpha^2)^{11}+(\alpha^2)^8+(\alpha^2)^7+(\alpha^2)^4+(\alpha^2)^3+(\alpha^2)=\alpha^{28}+\alpha^{24}+\alpha^{22}+\alpha^{16}+$
$+\alpha^{14}+\alpha^8+\alpha^6+\alpha^2=\alpha^9$.

$S_3 = f(\alpha^3)=(\alpha^3)^{14}+(\alpha^3)^{12}+(\alpha^3)^{11}+(\alpha^3)^8+(\alpha^3)^7+(\alpha^3)^4+(\alpha^3)^3+(\alpha^3)=\alpha^{42}+\alpha^{36}+\alpha^{33}+\alpha^{24}+$
$+\alpha^{21}+\alpha^{12}++\alpha^9+\alpha^3 = 0$.

$S_4 = f(\alpha^4)=(\alpha^4)^{14}+(\alpha^4)^{12}+(\alpha^4)^{11}+(\alpha^4)^8+(\alpha^4)^7+(\alpha^4)^4+(\alpha^4)^3+(\alpha^4)=\alpha^{56}+\alpha^{56}+\alpha^{44}+\alpha^{32}+$
$+\alpha^{28}+\alpha^{16}+\alpha^{12}+\alpha^4 = \alpha^3$.

$S_5 = f(\alpha^5)=(\alpha^5)^{14}+(\alpha^5)^{12}+(\alpha^5)^{11}+(\alpha^5)^8+(\alpha^5)^7+(\alpha^5)^4+(\alpha^5)^3+(\alpha^5)=\alpha^{70}+\alpha^{60}+\alpha^{55}+\alpha^{40}+$
$+\alpha^{35}+\alpha^{20}++\alpha^{15}+\alpha^5 = \alpha^0$.

$S_6 = f(\alpha^6)=(\alpha^6)^{14}+(\alpha^6)^{12}+(\alpha^6)^{11}+(\alpha^6)^8+(\alpha^6)^7+(\alpha^6)^4+(\alpha^6)^3+(\alpha^6)=\alpha^{84}+\alpha^{72}+\alpha^{66}+\alpha^{48}+$
$+\alpha^{42}+\alpha^{24}+\alpha^{18}+\alpha^6 = 0$.

$$\boldsymbol{S_1 = \alpha^{12}, S_2 = \alpha^9, S_3 = 0, S_4 = \alpha^3, S_5 = \alpha^0, S_6 = 0.}$$

Рассмотрим работу алгоритма Питерсона по поиску коэффициентов полинома локаторов ошибок $\sigma_1, \sigma_2, \ldots \sigma_v$.

Предположим, что произошло максимальное количество ошибок $v = 3$ и построим матрицу для $i = 3$

$$M_3 = \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^9 & 0 \\ \alpha^9 & 0 & \alpha^3 \\ 0 & \alpha^3 & 1 \end{bmatrix}.$$

Найдем определитель, полученный матрицы

$$M_3 = \det\begin{bmatrix} \alpha^{12} & \alpha^9 & 0 \\ \alpha^9 & 0 & \alpha^3 \\ 0 & \alpha^3 & 1 \end{bmatrix} = \alpha^{12}(\alpha^6 + 0) + \alpha^9(\alpha^9 + 0) + 0(\alpha^{12} + 0) = \alpha^{18} + \alpha^{18} = 0.$$

Определитель равен нулю, следовательно, предполагаем, что произошло $v = 2$ ошибок и строим матрицу для $i = 2$

$$M_2 = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^9 \\ \alpha^9 & 0 \end{bmatrix}.$$

Находим определитель матрицы M_2

$$M_2 = \det\begin{bmatrix} \alpha^{12} & \alpha^9 \\ \alpha^9 & 0 \end{bmatrix} = \alpha^{18}$$

Определитель не равен нулю, соответственно произошло две ошибки.

Таким образом, решив систему линейных уравнений (1.52) необходимо найти коэффициенты σ_1, σ_2 для полинома $\sigma(x)$ локаторов ошибки

$$M_2 \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix},$$

или

$$\begin{bmatrix} \alpha^{12} & \alpha^9 \\ \alpha^9 & 0 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha^3 \end{bmatrix}.$$

Тогда коэффициенты σ_1, σ_2 находятся как

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = M_2^{-1} \begin{bmatrix} 0 \\ \alpha^3 \end{bmatrix},$$

где M_2^{-1} – обратная матрица M_2.

Для нахождения обратной матрицы воспользуемся известной формулой

$$M^{-1} = \frac{1}{|M|} \left| A_{ij} \right|^T,$$

где $|M|$ – определитель матрицы, A_{ij} – транспонированная матрица, элементами которой являются алгебраические дополнения a_{ij}.

Алгебраические дополнения a_{ij} получаются из матрицы M путем нахождения минора матрицы M (вычеркивания i-ой строки и j-го столбца) умноженного на $(-1)^{i+j}$.

Найдем алгебраические дополнения для матрицы $M_2 = \begin{bmatrix} \alpha^{12} & \alpha^{9} \\ \alpha^{9} & 0 \end{bmatrix}$:

$a_{11} = (-1)^2 \cdot 0 = 0$; $a_{12} = (-1)^3 \cdot \alpha^9 = \alpha^9$;

$a_{21} = (-1)^3 \cdot \alpha^9 = \alpha^9$; $a_{22} = (-1)^4 \cdot \alpha^{12} = \alpha^{12}$.

Построим обратную матрицу M_2^{-1}

$$M_2^{-1} = \frac{1}{|M|}\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \frac{1}{\alpha^3}\begin{bmatrix} 0 & \alpha^{9} \\ \alpha^{9} & \alpha^{12} \end{bmatrix} = \alpha^{12}\begin{bmatrix} 0 & \alpha^{9} \\ \alpha^{9} & \alpha^{12} \end{bmatrix} = \begin{bmatrix} 0 & \alpha^{6} \\ \alpha^{6} & \alpha^{9} \end{bmatrix}.$$

Находим коэффициенты полинома локаторов ошибок

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 0 & \alpha^{6} \\ \alpha^{6} & \alpha^{9} \end{bmatrix}\begin{bmatrix} 0 \\ \alpha^{3} \end{bmatrix} = \begin{bmatrix} 0 \cdot 0 + \alpha^{6}\alpha^{3} \\ \alpha^{6} \cdot 0 + \alpha^{9}\alpha^{3} \end{bmatrix} = \begin{bmatrix} \alpha^{9} \\ \alpha^{12} \end{bmatrix}.$$

Следовательно, коэффициенты равны $\sigma_1 = \alpha^{12}$, $\sigma_2 = \alpha^9$ и искомый полином локаторов ошибок

$$\sigma(x) = 1 + \alpha^{12}x + \alpha^{9}x^2.$$

Для нахождения корней полученного полинома $\sigma(x)$ используется метод Ченя – проверка условия $\sigma(\alpha^j) = 0$ для всех ненулевых элементов расширенного поля $GF(2^4)$.

Многочлен $\sigma(x)$ имеет корни α^8 и α^{13}:

$$\sigma(\alpha^8) = 1 + \alpha^{12}\alpha^8 + \alpha^9\alpha^{16} = 1 + \alpha^{20} + \alpha^{25} = 1 + \alpha^5 + \alpha^{10} = 1 + 1 = 0.$$

$$\sigma(\alpha^{13}) = 1 + \alpha^{12}\alpha^{13} + \alpha^9\alpha^{26} = 1 + \alpha^{25} + \alpha^{35} = 1 + \alpha^{10} + \alpha^5 = 1 + 1 = 0.$$

Локаторы ошибок равны элементам, обратные корням:

– для элемента α^8 обратным является элемент α^7, так как из табл. 1.10 находим $\alpha^8 \cdot \alpha^7 = 1$;

– для элемента α^{13} обратным является элемент α^2, так как из табл. 1.10 находим $\alpha^{13} \cdot \alpha^2 = 1$.

Таким образом, ошибки произошли во втором и седьмом разряде передаваемой кодовой комбинации. И так как мы рассматриваем двоичный БЧХ код, то исправление ошибок, сводится к инвертированию символов принятых на 2-ой и 7-ой позициях.

3.5 Алгоритм Евклида

Декодирование с помощью алгоритма Евклида представляет собой один из способов декодирования, отличный от рассмотренных ранее. Принцип его работы считается простым для понимания, однако он менее эффективен в практическом использовании. В основе алгоритма Евклида лежит хорошо известная процедура нахождения наибольшего общего делителя (НОД) двух полиномов (или целых чисел).

Алгоритм Евклида представляет собой рекуррентный метод нахождения НОД двух чисел или двух многочленов. В основе этого алгоритма лежит следующее утверждение. Пусть a и b два целых числа или многочлена, причем $a \geq b$ или $\deg[(a)] \geq \deg[(b)]$. Разделим a на b. Если остаток от деления r равен 0, то $d = b$ является наибольшим общим делителем. Если остаток не равен 0, заменим a на b, b на r и повторим деление.

Пример 3.5. Найти НОД целых чисел $a = 186$ и $b = 66$. Следуя алгоритму Евклида получаем

Шаг 1. $r = 186 \bmod 66 = 54$, $r \neq 0 \rightarrow a = 66$, $b = 54$.

Шаг 2. $r = 66 \bmod 54 = 12$, $r \neq 0 \rightarrow a = 54$, $b = 12$.

Шаг 3. $r = 54 \bmod 12 = 6$, $r \neq 0 \rightarrow a = 12$, $b = 6$.

Шаг 4. $r = 12 \bmod 6 = 0$, $r = 0$.

Таким образом, НОД (186, 66) = 6.

В процессе нахождения НОД согласно алгоритму Евклида вычисляются два числа f и g (или многочлена $f(x)$ и $g(x)$), для которых $f\,a + g\,b = d$. На каждом шаге алгоритма вычисляются промежуточные числа (многочлены) f_i и g_i такие, что $f_i\,a + g_i\,b = r_i$. При этом f_i и g_i получаются с учетом двух предыдущих значений по формуле

$$f_i = f_{i-2} - q_i \cdot f_{i-1},\ g_i = g_{i-2} - q_i g_{i-1}, \tag{3.23}$$

где q_i – отношение двух предыдущих остатков

$$q_i = \left[\frac{r_{i-2}}{r_{i-1}} \right]. \tag{3.24}$$

Знак []обозначает целую часть дроби.

Используя рекуррентные соотношения между f_i, g_i и r_i выполним следующие преобразования в алгоритме нахождения НОД двух чисел $a = 186$ и $b = 66$.

Шаг 1. Задаем начальные установки: $\boldsymbol{i = -1}$, $f_{-1} = 1$, $g_{-1} = 0$.

Приводим уравнение к требуемой форме

$f_{-1}a + g_{-1}b = r_{-1} \rightarrow r_{-1} = 1 \cdot 186 + 0 \cdot 66 = \underline{186}$.

Шаг 2. Задаем начальные установки: $\boldsymbol{i = 0}$, $f_0 = 0$, $g_0 = 1$.

Приводим уравнение к требуемой форме

$f_0 a + g_0 b = r_0 \rightarrow r_0 = 0 \cdot 186 + 1 \cdot 66 = \underline{66.}$

Шаг 3. $\boldsymbol{i = 1}$.

$f_1 = f_{-1} - q_1 \cdot f_0, \quad g_1 = g_{-1} - q_1 \cdot g_0,$

$q_1 = [r_{-1}/r_0] = [186/66] = 2,$

$f_1 = 1 - 2 \cdot 0 = 1, \; g_1 = 0 - 2 \cdot 1 = -2,$

$f_1 a + g_1 b = r_1 \rightarrow r_1 = 1 \cdot 186 - 2 \cdot 66 = \underline{54}$.

Шаг4. $\boldsymbol{i = 2}$.

$f_2 = f_0 - q_2 \cdot f_1, \quad g_2 = g_0 - q_2 \cdot g_1,$

$q_2 = [r_0/r_1] = [66/54] = 1,$

$f_2 = 0 - 1 \cdot 1 = -1, \; g_2 = 1 - 1 \cdot (-2) = 3,$

$f_2 a + g_2 b = r_2 \rightarrow r_2 = -1 \cdot 186 + 3 \cdot 66 = \underline{12}$.

Шаг5. $\boldsymbol{i = 3}$.

$f_3 = f_1 - q_3 \cdot f_2, \quad g_3 = g_1 - q_3 \cdot g_2,$

$q_3 = [r_1/r_2] = [54/12] = 4,$

$f_3 = 1 - 4 \cdot (-1) = 5, \; g_3 = -2 - 4 \cdot 3 = -14,$

$f_3 a + g_3 b = r_3 \rightarrow r_3 = 5 \cdot 186 + (-14) \cdot 66 = \underline{6}$.

Шаг6. $\boldsymbol{i = 4}$.

$f_4 = f_2 - q_4 \cdot f_3, \quad g_4 = g_2 - q_4 \cdot g_3,$

$q_4 = [r_2/r_3] = [12/6] = 2,$

$f_4 = -1 - 2 \cdot 5 = -11, \quad g_4 = 3 - 2 \cdot (-14) = 31,$

$f_4 a + g_4 b = r_4 \rightarrow r_4 = (-11) \cdot 186 + 31 \cdot 66 = \underline{0}$.

Процесс нахождения НОД (186, 66) можно также представить в табличном виде (табл. 3.3).

Таблица 3.3 – Процесс нахождения НОД (186, 66)

Шаг *i*	f_i	g_i	$d_i(r_i)$	q_i	$r_i = f_i \cdot a + g_i \cdot b$
−1	1	0	186	-	1·186+0·66=186
0	0	1	66	-	0·186+1·66=66
1	1	−2	54	[186/66] = 2	1·186−2·66=54
2	−1	3	12	[66/54] = 1	−1·186+3·66=12
3	5	−14	6	[54/12] = 4	5·186−14·66=6
4	−11	31	0	[12/6] = 2	−11·186+31·66=0

Выполненные преобразования используются при быстром декодировании кодов БЧХ для решения ключевого уравнения с помощью алгоритма Евклида. Так как декодирование БЧХ кодов осуществляется для многочленов, повторим вышеизложенные рассуждения применительно к многочленам.

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 $a(x)\cdot f(x) + b(x)\cdot g(x) = d(x)$.

Пример 3.6. Найти наибольший общий делитель $d(x)$ двух многочленов $a(x) = x^3 + 1$ и $b(x) = x^2 + 1$, а также многочлены $f(x)$ и $g(x)$ для которых выполняется $a(x)\cdot f(x) + b(x)\cdot g(x) = d(x)$ в двоичном поле.

Шаг 1. Задаем начальные установки: $\boldsymbol{i = -1}$, $f_{-1}(x)= 1$, $g_{-1}(x)= 0$.

Приводим уравнение к требуемой форме

$f_{-1}(x)a(x) + g_{-1}(x)b(x) = r_{-1}(x) \rightarrow r_{-1}(x) = 1\cdot(x^3+1) + 0\cdot(x^2+1) = \underline{x^3+1}.$

Шаг 2. Задаем начальные установки: $\boldsymbol{i = 0}$, $f_0(x)= 0$, $g_0(x)= 1$.

Приводим уравнение к требуемой форме

$f_0(x)a(x) + g_0(x)b(x) = r_0(x) \rightarrow r_0(x) = 0\cdot(x^3+1) + 1\cdot(x^2+1) = \underline{x^2+1}.$

Шаг 3. $\boldsymbol{i = 1}$.

$f_1(x)= f_{-1}(x) - q_1(x)\cdot f_0(x)$,

$g_1(x)= g_{-1}(x) - q_1(x)\cdot g_0(x)$,

$q_1(x) = [r_{-1}(x)/r_0(x)] = [x^3+1/x^2+1] = \underline{x}$,

$f_1(x) = 1 + x\ 0 = 1$, $g_1(x)= 0 + x\cdot 1 = x$,

$f_1(x)a(x) + g_1(x)b(x) = r_1(x) \rightarrow r_1(x) = 1\cdot(x^3+1) + x\cdot(x^2+1) = x+1.$

Шаг4. $\boldsymbol{i = 2}$.

$f_2(x)= f_0(x) - q_2(x)\cdot f_1(x)$,

$g_2(x)= g_0(x) - q_2(x)\cdot g_1(x)$,

$q_2(x) = [r_0(x)/r_1(x)] = [x^2+1/x+1] = \underline{x+1}$,

$f_2(x) = 0 + (x+1)\,1 = x+1,\ \ g_2(x) = 1 + (x+1)\,x = x^2+x+1,$

$f_1(x)a(x) + g_1(x)b(x) = r_1(x) \rightarrow r_1 = (x+1)\cdot(x^3+1) + (x^2+x+1)\cdot(x^2+1) = 0.$

Искомое значение НОД $d(x)$ определяется последним ненулевым значением $r_i(x)$, т.е. $d(x) = r_1(x) = x+1$. Процесс нахождения НОД (x^3+1, x^2+1) можно также представить в табличном виде (табл. 3.4).

Таким образом, НОД (x^3+1, x^2+1) = $x+1$.

Таблица 3.4 – Процесс нахождения НОД (x^3+1, x^2+1)

Шаг *i*	f_i	g_i	$d_i(r_i)$	q_i	$r_i = f_i \cdot a + g_i \cdot b$
−1	1	0	x^3+1	-	$1\cdot(x^3+1)+0\cdot(x^2+1)= x^3+1$
0	0	1	x^2+1	-	$0\cdot(x^3+1)+1(x^2+1)= x^2+1$
1	1	x	$x+1$	$[x^3+1/x^2+1]=x$	$1\cdot(x^3+1)+x\cdot(x^2+1)=x+1$
2	$x+1$	x^2+x+1	0	$[x^2+1/x+1]=x+1$	$(x+1)(x^3+1)+(x^2+x+1)($ $x^2+1)=0$

При декодировании БЧХ кодов интересуются не конечным результатом алгоритма Евклида, а промежуточными результатами, которые можно представить в виде: $a(x)\cdot f_i(x) + b(x)\cdot g_i(x) = r_i(x)$.

Для дальнейшего описания алгоритма Евклида необходимо ввести обозначение *полином значений ошибок*, который связан с многочленом локаторов ошибок как

$$\Omega(x) = \sigma(x)S(x), \tag{3.25}$$

где синдромный полином имеет вид

$$S(x) = S_1 + S_2 x + \ldots + S_{2t} x^{2t-1}. \tag{3.26}$$

Из уравнения (3.21) следует, что многочлен значений ошибок удовлетворяет условию

$$\Omega(x) = \sigma(x) \cdot S(x) \bmod x^{2t}. \tag{3.27}$$

Алгоритм декодирования БЧХ кода заключается в определении многочлена $\Omega(x)$, удовлетворяющего уравнению (3.21). Это решение может быть найдено с помощью расширенного алгоритма Евклида следующем образом: $b(x)\cdot g_i(x) = r_i(x) \bmod a(x)$, полгая

$a(x) = x^{2t}$, $b(x) = S(x)$, $g_i(x) = \sigma_i(x)$, $r_i(x) = \Omega_i(x)$.

При этом используется свойство алгоритма Евклида:

$$\deg[g_i(x)] + \deg[r_{i-1}(x)] = \deg[a(x)].$$

Если $a(x) = x^{2t}$, то

$$\deg[\sigma_i(x)] + \deg[\Omega_{i-1}(x)] = 2t,\ \deg[\sigma_i(x)] + \deg[\Omega_i(x)] < 2t.$$

При появлении $v \le t$ошибок имеем: $\deg[\Omega_i(x)] < \deg[\sigma_i(x)] \le t$.

Существует единственный с точностью до постоянного множителя (элемента поля) многочлен $\sigma_i(x)$ степени $\le t$, удовлетворяющий ключевому уравнению $\Omega_i(x) = S(x)\sigma_i(x) \bmod x^{2t}$.

Кроме того, если $\deg[\Omega_{i-1}(x)] \ge t$ при $\deg[\sigma_i(x)] \le t$, то $\deg[\sigma_{i+1}(x)] > t$.

Поэтому промежуточные результаты на i-ом шаге дают единственное решение ключевого уравнения и алгоритм Евклида следует применять до тех пор, пока не будет выполнено условие $\deg[\Omega_{i-1}(x)] < t$.

Для нахождения многочлена локаторов ошибок $\sigma(x)$ необходимо применить расширенный алгоритм Евклида к многочленам $a(x) = x^{2t}$, $b(x) = S(x)$, т.е. вычислить НОД (x^{2t}, $S(x)$). Если на i –ом шаге алгоритма получено решение

$$r_i(x) = f_i(x) \cdot x^{2t} + g_i(x) \cdot S(x), \qquad (3.28)$$

такое, что $\deg[r_i(x)] < t$, то $\Omega_i(x) = r_i(x)$ и $\sigma_i(x) = g_i(x)$, тогда

$$x^{2t} f(x) + S(x)\sigma(x) = \Omega(x). \qquad (3.29)$$

При этом для задачи декодирования БЧХ кодов полином $f_i(x)$ не представляет интереса, так как решение (3.27) ищется по модулю x^{2t}.

Предварительно на начальном этапе для формирования начальных условий вычисляются значения многочленов локаторов и значений ошибок на $i = -1$, $i = 0$ шагах. Предполагается, что

$$\sigma_{-1}(x) = 0,\ \sigma_0(x) = 1,\ \Omega_{-1}(x) = x^{2t},\ \Omega_0(x) = S(x).$$

Вид синдромного многочлена $S(x)$ зависит от принятой кодовой комбинации $f(x)$, поступившей на вход декодера, и определяется при

помощи подстановки значений корней порождающего многочлена в $f(x)$ по формуле(3.26).

Для инициализации алгоритма Евклида на шагах «−1», «0» записываются исходные данные процедуры декодирования, а каждый последующий шаг начинается с вычисления $q_i(x)$

$$q_i(x) = \left[\frac{\Omega_{i-2}(x)}{\Omega_{i-1}(x)} \right]. \qquad (3.30)$$

Значения $\sigma_i(x)$ и $\Omega_i(x)$ вычисляются в соответствии с выражениями

$$\sigma_i(x) = \sigma_{i-2}(x) + q_i \cdot \sigma_{i-1}(x),\ \Omega_i(x) = \Omega_{i-2}(x) + q_i \cdot \Omega_{i-1}(x). \qquad (3.31)$$

При реализации процедуры декодирования после завершения очередного шага проверяется условие

$$\deg[\Omega_i(x)] < t. \qquad (3.32)$$

Если степень $\Omega_i(x)$ меньше t, то процедура вычисления значений $\sigma_i(x)$ и $\Omega_i(x)$ и дальнейшее значение $q_i(x)$ не вычисляются.

В табл. 3.5 представлен процесс декодирования БЧХ кода алгоритмом Евклида. На рисунке 3.2 представлена схема декодирования по алгоритму Евклида.

Таблица 3.5 – Алгоритм Евклида

Шаг	$q_i(x)$	$\Omega_i(x)$	$\sigma_i(x)$
−1	–	x^{2t}	0
0	–	$S(x)$	1
1	$q_1(x) = \left[\frac{\Omega_{-1}(x)}{\Omega_0(x)} \right]$	$\Omega_1(x) = \Omega_{-1}(x) + q_1 \cdot \Omega_0(x)$	$\sigma_1(x) = \sigma_{-1}(x) + q_1 \cdot \sigma_0(x)$
…	…	…	…
i	$q_i(x) = \left[\frac{\Omega_{i-2}(x)}{\Omega_{i-1}(x)} \right]$	$\Omega_i(x) = \Omega_{i-2}(x) + q_i \cdot \Omega_{i-1}(x)$	$\sigma_i(x) = \sigma_{i-2}(x) + q_i \cdot \sigma_{i-1}(x)$

Алгоритм Евклида решения ключевого уравнения заключается в следующем.

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

2. Использовать начальные условия

$$\sigma_{-1}(x) = 0, \sigma_0(x) = 1, \Omega_{-1}(x) = x^{2t}, \Omega_0(x) = S(x).$$

3. Остановится, если $\deg[\Omega_i(x)] < t$ и положить $\sigma(x) = \sigma_i(x), \Omega(x) = \Omega_i(x)$, иначе пункт 4.

4. Выполнить деление $q_i(x) = \left[\frac{\Omega_{i-2}(x)}{\Omega_{i-1}(x)}\right]$.

5. Вычислить $\Omega_i(x) = \Omega_{i-2}(x) + q_i \cdot \Omega_{i-1}(x)$, $\sigma_i(x) = \sigma_{i-2}(x) + q_i \cdot \sigma_{i-1}(x)$.

6. Выполнить пункт 3.

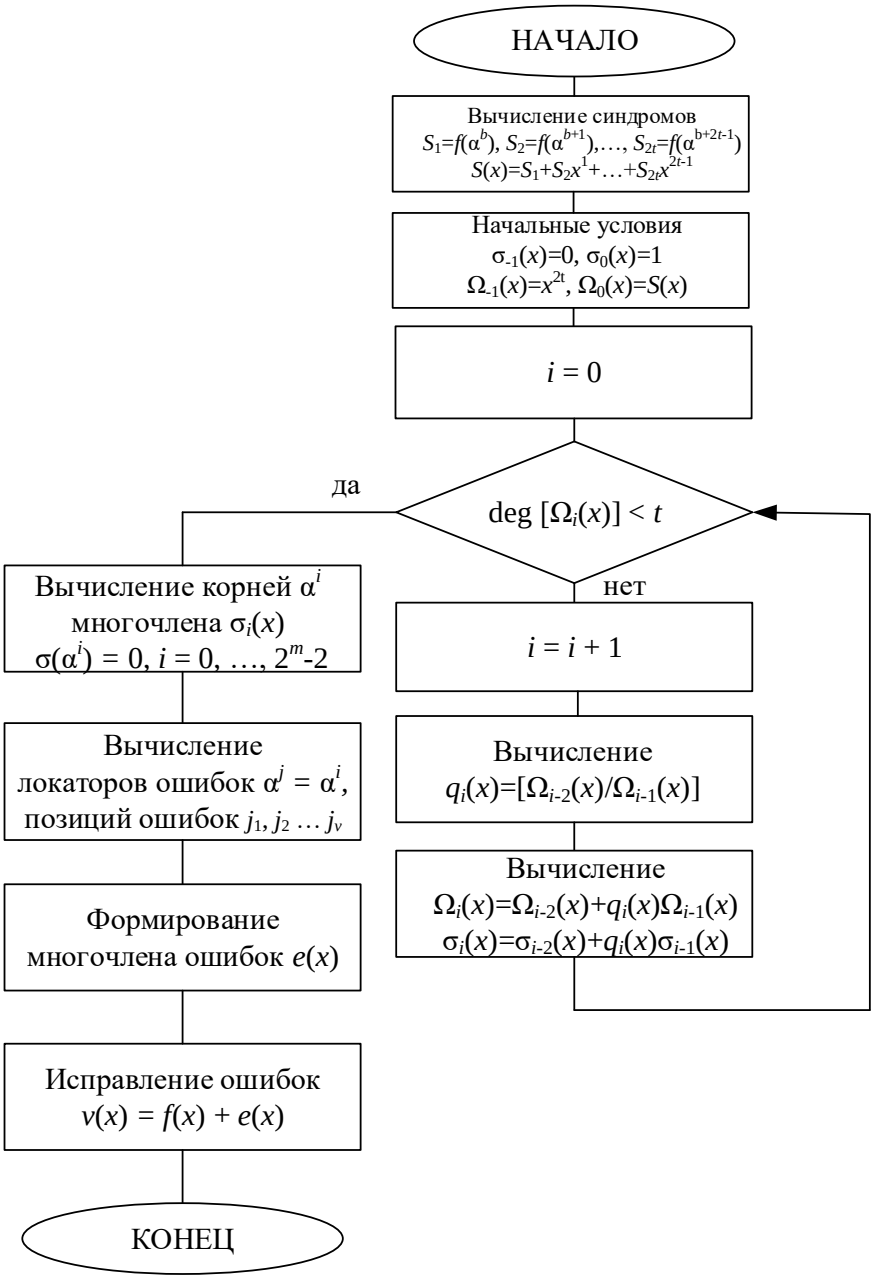

Рисунок 3.2 – Алгоритм Евклида

Пример 3.7. Рассмотрим (15, 5, 7) БЧХ код, исправляющий три ошибки. Он имен порождающий полином $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$. Пусть информационная последовательность имеет вид $u(x) = x^4 + x^2 + x$. При систематическом кодировании кодовая последовательность $c(x) = x^{14}+x^{12}+x^{11}+x^8+x^4+x^3+x^2+x$. Для примера будем считать что произошло три ошибки и вектор ошибки равен $e(x) = x^{12} + x^6 + 1$. Следовательно

принятое кодовое слово $f(x) = x^{14}+x^{11}+x^8+x^6+x^4+x^3+x^2+x+1$.Декодером были получены компоненты синдрома с помощью принятого кодового слова и табл. 3.1 сложения элементов в расширенном поле $GF(2^4)$:

$\boldsymbol{S_1 = \alpha, S_2 = \alpha^2, S_3 = \alpha^8, S_4 = \alpha^4, S_5 = 1^0, S_6 = \alpha.}$

Таким образом, получаем синдромный полином

$$S(x) = \alpha x^5 + x^4 + \alpha^4 x^3 + \alpha^8 x^2 + \alpha^2 x^1 + \alpha .$$

С помощью алгоритма Евклида найдем многочлен локаторов ошибок, т.е. решим ключевое уравнение $x^6 f(x) + s(x)\sigma(x) = \Omega(x)$.

Шаг 0 и −1. $\sigma_{-1}(x) = 0,\ \sigma_0(x) = 1,$

$\Omega_0(x)=S(x)=\alpha x^5+x^4+\alpha^4x^3+\alpha^8x^2+\alpha^2x^1+\alpha,\ \Omega_{-1}(x) =x^6.$

Шаг 1.

$q_1(x)=[\Omega_{-1}/\Omega_0] = [x^6/\alpha x^5+x^4+\alpha^4x^3+\alpha^8x^2+\alpha^2x^1+\alpha] = \alpha^{14}x+\alpha^{13};$

$\Omega_1(x)=\Omega_{-1}(x)+q_1(x)\ \Omega_0(x) =x^6 + (\alpha^{14}x+\alpha^{13})(\alpha x^5+x^4+\alpha^4x^3+\alpha^8x^2+\alpha^2x^1+\alpha) =$

$= x^6+\alpha^{15}x^6+\alpha^{14}x^5+\alpha^{18}x^4+\alpha^{22}x^3+\alpha^{16}x^2+\alpha^{15}x+\alpha^{14}x^5+\alpha^{13}x^4+\alpha^{17}x^3+\alpha^{21}x^2+$

$+\alpha^{15}x+\alpha^{14}= (1+\alpha^{15})x^6+(\alpha^{14}+\alpha^{14})x^5+(\alpha^{18}+\alpha^{13})x^4+(\alpha^{22}+\alpha^{17})x^3+(\alpha^{16}+\alpha^{21})x^2+$

$+(\alpha^{15}+\alpha^{15})x+\alpha^{14}=0+0+(\alpha^3+\alpha^{13})x^4+(\alpha^7+\alpha^2)x^3+(\alpha^1+\alpha^6)x^2+0+\alpha^{14}=$

$= \alpha^8x^4+\alpha^{12}x^3+\alpha^{11}x^2+\alpha^{14};$

$\sigma_1(x) = \sigma_{-1}(x)+q_1(x)\sigma_0(x) = 0+(\alpha^{14}x+\alpha^{13}) = \alpha^{14}x+\alpha^{13}.$

Проверка: $\deg[\Omega_1(x)] < 3;\ 5 < 3$ − нет, поэтому выполнить шаг 2.

Шаг 2.

$q_2(x)=[\Omega_0/\Omega_1]=[(\alpha x^5+x^4+\alpha^4x^3+\alpha^8x^2+\alpha^2x^1+\alpha)/(\alpha^8x^4+\alpha^{12}x^3+\alpha^{11}x^2+\alpha^{14})]=\alpha^8x+\alpha^2;$

$\Omega_2(x)=\Omega_0(x)+q_2(x)\ \Omega_1(x) =(\alpha x^5+x^4+\alpha^4x^3+\alpha^8x^2+\alpha^2x^1+\alpha)+(\alpha^8x+\alpha^2)\ (\alpha^8x^4+\alpha^{12}x^3+$

$+\alpha^{11}x^2+\alpha^{14})=\alpha^{14}x^3+\alpha^3x^2+\alpha^{12}x;$

$\sigma_2(x) = \sigma_0(x)+q_2(x)\ \sigma_1(x) = 1+(\alpha^8x+\alpha^2)\ (\alpha^{14}x+\alpha^{13})=1+\alpha^7x^2+\alpha^6x+\alpha x+1=$

$=\alpha^7x^2+\alpha^{11}x.$

Проверка: $\deg[\Omega_2(x)] < 3;\ 3 < 3$ − нет, поэтому выполнить шаг 3.

Шаг 3.

$q_3(x)=[\Omega_1/\Omega_2]=[(\alpha^8x^4+\alpha^{12}x^3+\alpha^{11}x^2+\alpha^{14})/(\alpha^{14}x^3+\alpha^3x^2+\alpha^{12}x)] = \alpha^9x;$

$\Omega_3(x)=\Omega_1(x)+q_3(x)\ \Omega_2(x) = (\alpha^8x^4+\alpha^{12}x^3+\alpha^{11}x^2+\alpha^{14})+\alpha^9x\ (\alpha^{14}x^3+\alpha^3x^2+\alpha^{12}x)=$

$=\alpha x^2+\alpha^{14};$

$\sigma_3(x) = \sigma_1(x)+q_3(x)\ \sigma_2(x) = (\alpha^{14}x+\alpha^{13})+\alpha^9x\ (\alpha^7x^2+\alpha^{11}x)=\alpha^{14}x+\alpha^{13}+\alpha x^3+\alpha^5x^2=$

$= \alpha x^3+\alpha^5x^2+\alpha^{14}x+\alpha^{13}.$

Проверка: $\deg[\Omega_3(x)] < 3,\ 2 < 3$ − да, поэтому остановка.

Таким образом, полином локаторов ошибок

$\sigma(x) = \alpha x^3+\alpha^5x^2+\alpha^{14}x+\alpha^{13} = \alpha^{13}(1+\alpha x+\alpha^7x^2+\alpha^3x^3).$

В таблице 3.6 представлены выполненные вычисления для нахождения полинома локаторов ошибок с помощью алгоритма Евклида.

Таблица 3.6 – Вычисление полинома локаторов ошибок с помощью алгоритма Евклида

Шаг	$q_i(x)$	$\Omega_i(x)$	$\sigma_i(x)$
−1	–	x^6	0
0	–	$\alpha x^5+x^4+\alpha^4x^3+\alpha^8x^2+\alpha^2x^1+\alpha$	1
1	$\alpha^{14}x+\alpha^{13}$	$\alpha^8x^4+\alpha^{12}x^3+\alpha^{11}x^2+\alpha^{14}$	$\alpha^{14}x+\alpha^{13}$
2	$\alpha^8x+\alpha^2$	$\alpha^{14}x^3+\alpha^3x^2+\alpha^{12}x$	$\alpha^7x^2+\alpha^{11}x$
3	α^9x	$\alpha x^2+\alpha^{14}$	$\alpha x^3+\alpha^5x^2+\alpha^{14}x+\alpha^{13}$

Степень полинома $\deg[\sigma(x)] = 3$, т.е. произошло три ошибки в принятой последовательности $f(x)$. Для поиска позиций ошибок необходимо найти корни полинома $\sigma(x)$, используя метод Ченя – полный перебор всех элементов расширенного поля $GF(2^4)$.

$\sigma(\alpha^0) = 1 + \alpha + \alpha^7 + \alpha^3 = \alpha^3+\alpha^3=0;$

$\sigma(\alpha^1) = 1 +\alpha\cdot\alpha+\alpha^7\alpha^2+\alpha^3\alpha^3 = 1+\alpha^2+\alpha^9+\alpha^6=\alpha^4;$

$\sigma(\alpha^2) = 1+\alpha^1\alpha^2+\alpha^7\alpha^4+\alpha^3\alpha^6 = 1+\alpha^3+\alpha^{11}+\alpha^9 = \alpha^{14}+\alpha^2 = \alpha^{13};$

$\sigma(\alpha^3) = 1+\alpha^1\alpha^3+\alpha^7\alpha^6+\alpha^3\alpha^9 = 1+\alpha^4+\alpha^{13}+\alpha^{12} = \alpha^1+\alpha^1 = 0;$

$\sigma(\alpha^4) = 1+\alpha^1\alpha^4+\alpha^7\alpha^8+\alpha^3\alpha^{12} = 1+\alpha^5+\alpha^{15}+\alpha^{15} = \alpha^{10}+0 = \alpha^{10};$

$\sigma(\alpha^5) = 1+\alpha^1\alpha^5+\alpha^7\alpha^{10}+\alpha^3\alpha^{15} = 1+\alpha^6+\alpha^2+\alpha^3 = \alpha^{13}+\alpha^6 = 1;$

$\sigma(\alpha^6) = 1+\alpha^1\alpha^6+\alpha^7\alpha^{12}+\alpha^3\alpha^{18} = 1+\alpha^7+\alpha^4+\alpha^6 = \alpha^9+\alpha^{12} = \alpha^8;$

$\sigma(\alpha^7) = 1+\alpha^1\alpha^7+\alpha^7\alpha^{14}+\alpha^3\alpha^{21} = 1+\alpha^8+\alpha^{11}+\alpha^9 = \alpha^{14}+\alpha^2 = \alpha^{13};$

$\sigma(\alpha^8) = 1+\alpha^1\alpha^8+\alpha^7\alpha^{16}+\alpha^3\alpha^{24} = 1+\alpha^9+\alpha^8+\alpha^{12} = \alpha^7+\alpha^9 = 1;$

$\sigma(\alpha^9) = 1+\alpha^1\alpha^9+\alpha^7\alpha^{18}+\alpha^3\alpha^{27} = 1+\alpha^{10}+\alpha^{10}+1 = \alpha^5+\alpha^5 = 0;$

$\sigma(\alpha^{10}) = 1+\alpha^1\alpha^{10}+\alpha^7\alpha^{20}+\alpha^3\alpha^{30} = 1+\alpha^{11}+\alpha^{12}+\alpha^3 = \alpha^{12}+\alpha^{10} = \alpha^3;$

$\sigma(\alpha^{11}) = 1+\alpha^1\alpha^{11}+\alpha^7\alpha^{22}+\alpha^3\alpha^{33} = 1+\alpha^{12}+\alpha^{14}+\alpha^6 = \alpha^{11}+\alpha^8 = \alpha^7;$

$\sigma(\alpha^{12}) = 1+\alpha^1\alpha^{12}+\alpha^7\alpha^{24}+\alpha^3\alpha^{36} = 1+\alpha^{13}+\alpha^1+\alpha^9 = \alpha^6+\alpha^3 = \alpha^2;$

$\sigma(\alpha^{13}) = 1+\alpha^1\alpha^{13}+\alpha^7\alpha^{26}+\alpha^3\alpha^{39} = 1+\alpha^{14}+\alpha^3+\alpha^{12} = \alpha^3+\alpha^{10} = \alpha^{12};$

$\sigma(\alpha^{14}) = 1+\alpha^1\alpha^{14}+\alpha^7\alpha^{28}+\alpha^3\alpha^{42} = 1+\alpha^{15}+\alpha^5+\alpha^0 = 0+\alpha^{10} = \alpha^{10};$

По найденным корням $\boldsymbol{\alpha^9}$, $\boldsymbol{\alpha^3}$, $\boldsymbol{\alpha^0}$ вычисляем позиции ошибок как обратные элементы полученных значений:

− для элемента α^3 обратным является элемент $\boldsymbol{\alpha^{12}}$, так как из табл. 1.10 находим $\alpha^3 \cdot \alpha^{12} = 1$;

− для элемента α^9 обратным является элемент $\boldsymbol{\alpha^6}$, так как из табл. 1.10 находим $\alpha^9 \cdot \alpha^6 = 1$;

− для элемента α^0 обратным является элемент $\boldsymbol{\alpha^0}$, так как из табл. 1.10 находим $\alpha^0 \cdot \alpha^0 = 1$.

Таким образом, позиции ошибок$j_1 = 12$, $j_2 = 6$, $j_3 = 0$.

Вектор ошибок $e(x) = x^{12} + x^6 + 1$ и ошибки произошли во двенадцатом, шестом и нулевом разрядах передаваемой кодовой комбинации. И так как мы рассматриваем двоичный БЧХ код, то исправление ошибок, сводится к инвертированию символов принятых на 12-ой, 6-ой и 0-ой позициях.

3.6 Алгоритм Берлекэмпа-Месси

Основными вычислениями при декодировании БЧХ кодов является решение системы уравнений

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_{v-1} & S_v \\ S_2 & S_3 & S_4 & \cdots & S_v & S_{v+1} \\ S_3 & S_4 & S_5 & \cdots & S_{v+1} & S_{v+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ S_v & S_{v+1} & S_{v+2} & \cdots & S_{2v-2} & S_{2v-1} \end{bmatrix} \begin{bmatrix} \sigma_v \\ \sigma_{v-1} \\ \sigma_{v-2} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ -S_{v+3} \\ \vdots \\ -S_{2v} \end{bmatrix}. \qquad (3.33)$$

Так как строки матрицы S линейно независимы, то определитель матрицы S не равен нулю и матрица обратима, что дает единственное решение (σ_1, σ_2, …, σ_v) системы (3.33). Если гипотеза о линейной сложности vверна, то существует единственное решение этой системы уравнений, и оно является решением этой задачи. При небольших значениях vэту систему можно решить с помощью обращения матрицы и для этого необходимо произвести порядка v^3 действий. Однако на практике часто требуется использовать коды, исправляющие большое число ошибок и тогда требуется более эффективный метод решения ключевого

уравнения. Такой метод был разработан Берлекэмпом-Месси. В основе этого метода лежит тот факт, что матричное уравнение не произвольно и матрица обладает специальной структурой.

Предположим, что нам известен вектор σ, тогда первая строка системы уравнений определяет значение S_{v+1} через значения S_1, …, S_v, вторая строка определяет S_{v+2} через S_2, …, S_{v+1} и т.д.

$$\begin{aligned}
S_{v+1} &= \sigma_1 S_v + \sigma_2 S_{v-1} + \sigma_3 S_{v-2} + \quad ... \quad + \sigma_{v-1} S_2 + \sigma_v S_1;\\
S_{v+2} &= \sigma_1 S_{v+1} + \sigma_2 S_v + \sigma_3 S_{v-1} + \quad ... \quad + \sigma_{v-1} S_3 + \sigma_v S_2;\\
S_{v+3} &= \sigma_1 S_{v+2} + \sigma_2 S_{v+1} + \sigma_3 S_v + \quad ... \quad + \sigma_{v-1} S_4 + \sigma_v S_3;\\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots\\
S_{2v-1} &= \sigma_1 S_{2v} + \sigma_2 S_{2v-3} + \sigma_3 S_{2v-4} + ... + \sigma_{v-1} S_v + \sigma_v S_{v-1}\\
S_{2v} &= \sigma_1 S_{2v-1} + \sigma_2 S_{2v} + \sigma_3 S_{2v-3} + \quad ... \quad + \sigma_{v-1} S_{v+1} + \sigma_v S_v.
\end{aligned} \tag{3.34}$$

Для фиксированного σ это уравнение определяет регистр сдвига с линейной обратной связью, множители в отводах которого задаются вектором σ.На рисунке 3.3 представлен регистр сдвига с обратной связью построенный для каждой строки системы уравнений (3.34). При этом в начальном состоянием линейный регистр сдвига содержит последовательность S_v,$S_{v\text{-}1}$, …S_1. Для каждого такта на выходе регистра сдвига образуется элемент синдрома ошибки S_1, S_2, …, S_{2v}.

В общем виде этот последовательный процесс описывается уравнением

$$S_j = -\sum_{i=1}^{v} \sigma_i S_{j-i}\ ,\ j = v+1, \ldots, 2v. \tag{3.35}$$

Переформулированная таким образом задача сводится к построению регистра сдвига с линейной обратно связью, генерирующего известную последовательность компонент синдрома. И задача состоит в том, чтобы среди большого числа таких регистров найти регистр сдвига с наименьшей длиной. Это позволит определить вектор ошибки минимального веса в принятом слове, т.е. определить многочлен локаторов ошибок σ(x) наименьшей степени.

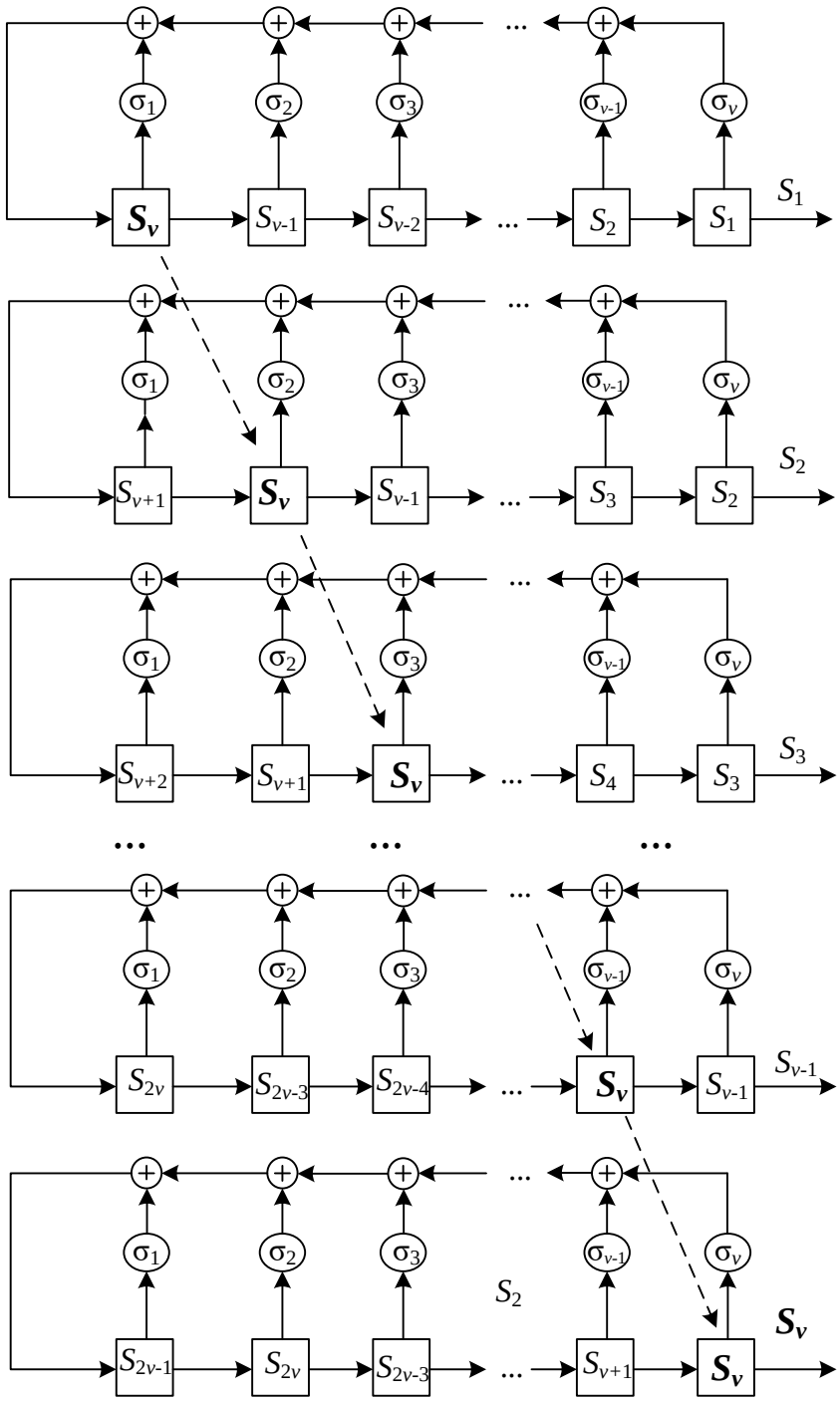

Рисунок 3.3 – Многочлен локаторов ошибок в цепи регистра сдвига

Таким образом, алгоритм Берлекэмпа-Месси лучше всего рассматривать как итеративный процесс построения минимального линейного регистра (сдвига) с обратной связью (ЛРОС), в общем виде показанному на рис. 3.4., который генерирует известную последовательность компонент синдрома $S_1, S_2, \ldots, S_{2t}$.

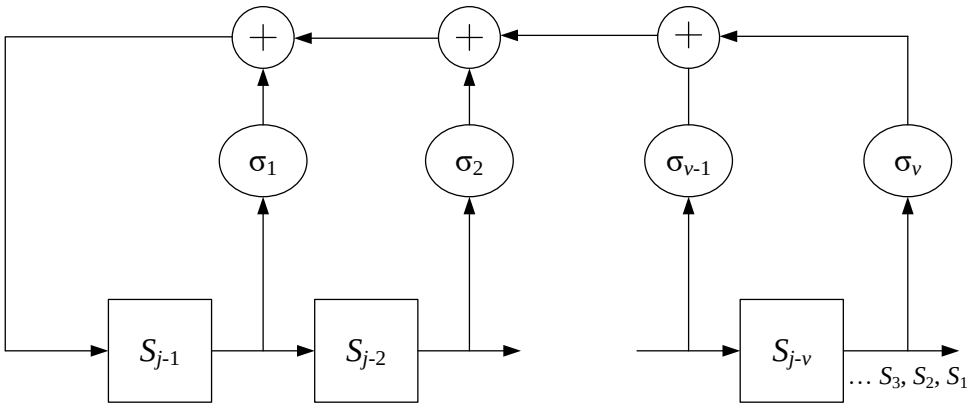

Рисунок 3.4 – ЛРОС с отводами $\sigma_1, \sigma_2, \ldots \sigma_v$и выходами$S_1, S_2, \ldots, S_{2t}$

Для построения требуемого регистра сдвига необходимо найти две величины:

−длину регистра сдвига L;

−многочлен обратной связи $\sigma(x)$

$$\sigma(x) = \sigma_v x^v + \sigma_{v-1} x^{v-1} + \ \ldots \ + \sigma x + 1,$$

где $\deg[\sigma(x)] \le L$.

Таким образом, необходимо найти регистр сдвига с обратной связью наименьшей длины, который генерирует последовательность $S_1, S_2, \ldots, S_{2t}$при соответствующем начальном состоянии. Для каждой итерации i, начиная с $k = 1$, строится регистр сдвига, генерирующий первые k компонент синдрома и целью алгоритма Берлекэмпа-Месси является построение многочлена (обратной связи) $\sigma^{(k)}(x)$ наименьшей степени, удовлетворяющего следующему уравнению

$$\sum_{j=0}^{L_k} S_{k-j} \sigma_j^{(k)} = 0 . \qquad (3.36)$$

Решение этой задачи эквивалентно условию, что многочлен

$$\sigma^{(k)}(x) = 1 + \sigma_1^{(k)} x + \ldots + \sigma_{L_k}^{(k)} x^{L_k} \qquad (3.37)$$

является многочленом обратной связи ЛРОС, который генерирует ограниченную последовательность синдромов.

Регистр сдвига с длиной L_k и коэффициентами $\sigma^{(k)}(x)$ является регистром сдвига минимальной длины, генерирующим $S_1, S_2, \ldots, S_k$.и к

началу k–ой итерации уже будет задан список регистров сдвига: L_1, $\sigma^{(1)}(x)$; L_2, $\sigma^{(2)}(x)$; …, L_{k-1}, $\sigma^{(k-1)}(x)$. Необходимо найти способ построения нового регистра минимальной длины L_k, $\sigma^{(k)}(x)$, генерирующего последовательность S_1, S_2, …, S_{k-1}, S_k. Это можно выполнить использовав предыдущий регистр сдвига, в котором при необходимости будут надлежащим образом изменены длина и весовые множители. Тогда на k–ой итерации вычислим следующий выход (k–1)-го регистра сдвига

$$S_k^{'} = -\sum_{j=1}^{n-1} \sigma_j^{(r-1)} S_{k-j} \,. \tag{3.38}$$

Вычитая $S_k^{'}$ из требуемого выхода S_k получаем величину d_k, называемую *несовместность* (невязкой, расхождение, различие)

$$d_k = S_k - S_k^{'} = S_k + \sum_{j=1}^{n-1} \sigma_j^{(k-1)} S_{k-j} \,, \tag{3.39}$$

которая является мерой соответствия синдромной последовательности и генерируемой линейным регистром сдвига с обратной связью, а также содержит корректирующий множитель для вычисления $\sigma^k(x)$ на следующей итерации. При этом возможны два случая:

− если $d_k = 0$, то уравнение (1.66) удовлетворяется с равенством

$$\sigma^{(k)}(x) = \sigma^{(k-1)}(x), \;\; L_k = L_{k-1} \tag{3.40}$$

и k–ая итерация будет закончена;

− если $d_k \neq 0$, то необходимо изменить весовые множители в цепи обратной связи регистра сдвига

$$\sigma^{(k)}(x) = \sigma^{(k-1)}(x) + d_k d_m^{-1} x^{k-m} \sigma^{(m-1)}(x) \,, \tag{3.41}$$

где $\sigma^{(m-1)}(x)$ – один из многочленов регистра сдвига, встречавшихся в одной из предыдущих итераций, при этом выбирается $m<k$ и такое, что $d_m \neq 0$, т.е. m равно номеру последнего шага, предшествующего шагу k, на котором $d_m \neq 0$.

Итеративное вычисление $\sigma^{(k)}(x)$ продолжается, пока не удовлетворятся одно либо оба условия:

$$k-1 \geq L_k + t - 1 \quad \text{èëè} \quad k = 2t\,.$$

Начальными условиями алгоритма являются

$$\sigma(x) = 1,\ L = 0,\ k = 0,\ B(x) = 1.$$

Например, рассмотрим последовательность $s_1, s_2, \ldots = 010111100\ldots$.

Найдем длину регистра сдвига L. Если бы последовательность состояла из одних нулей, то мы бы заключили, что L=0. В случае последовательности из всех единиц $L = 1$ $\sigma_1 = 1$, а уравнение (3.33) имеет вид $s_i+\sigma_1 s_{i-1} = 0$. Так как это не соответствует условию задания, то проверяем гипотезу о том, что $L = 2$. Система (3.33) тогда имеет вид

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Единственное решение уравнения $\sigma_1 = 0$, $\sigma_2 = 1$. Проверим, порождает ли соответствующий регистр сдвига всю последовательность

$\sigma_1 s_2+\sigma_2 s_1=\sigma_1 1+\sigma_2 0 = 0 = s_3$;

$\sigma_1 s_3+\sigma_2 s_2=\sigma_1 0+\sigma_2 1 = 1 = s_4$;

$\sigma_1 s_4+\sigma_2 s_3=\sigma_1 1+\sigma_2 0 = 0 \neq s_5 = 1$.

На пятом символе получили несовпадение, следовательно, гипотеза не верна. Проверим гипотезу $L = 3$.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Решение данной системы $\sigma_1 = 0$, $\sigma_2 = 1$, $\sigma_3 = 1$. Тогда получим

$\sigma_1 s_3+\sigma_2 s_2+\sigma_3 s_1=\sigma_1 1+\sigma_2 0+\sigma_3 0 = 1 = s_4$;

$\sigma_1 s_4+\sigma_2 s_3+\sigma_3 s_2=\sigma_1 1+\sigma_2 0+\sigma_3 1 = 1 = s_5$;

$\sigma_1 s_5+\sigma_2 s_4+\sigma_3 s_3=\sigma_1 1+\sigma_2 1+\sigma_3 0 = 1 = s_6$;

$\sigma_1 s_6+\sigma_2 s_5+\sigma_3 s_4=\sigma_1 1+\sigma_2 1+\sigma_3 1 = 0 \neq s_7 = 1$.

Опять получили несовпадение, но уже на седьмом символе.

Проверим гипотезу $L = 4$.

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \sigma_4 \\ \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Решение данной системы $\sigma_1 = 0$, $\sigma_2 = 0$, $\sigma_3 = 1$, $\sigma_4 = 1$. Тогда получим

$\sigma_1 s_4+\sigma_2 s_3+\sigma_3 s_2+\sigma_4 s_1 =\sigma_1 1+\sigma_2 0+\sigma_3 1+\sigma_4 0 = 1 = s_5$;

$\sigma_1 s_5+\sigma_2 s_4+\sigma_3 s_3+\sigma_4 s_2 = \sigma_1 1+\sigma_2 1+\sigma_3 0+\sigma_4 1 = 1 = s_6$;

$\sigma_1 s_6+\sigma_2 s_5+\sigma_3 s_4+\sigma_4 s_3 = \sigma_1 1+\sigma_2 1+\sigma_3 1+\sigma_4 0 = 1 = s_7$;

$\sigma_1 s_7+\sigma_2 s_6+\sigma_3 s_5+\sigma_4 s_4 = \sigma_1 1+\sigma_2 1+\sigma_3 1+\sigma_4 1 = 0 = s_8$;

$\sigma_1 s_8+\sigma_2 s_7+\sigma_3 s_6+\sigma_4 s_5 = \sigma_1 0+\sigma_2 1+\sigma_3 1+\sigma_4 1 = 0 = s_9$.

Таким образом, задача решена.

На рисунке 3.5 представлена схема алгоритма Берлекэмпа-Месси для нахождения полинома локаторов ошибок. В алгоритме используются следующие обозначения:

S_j– компоненты синдрома ошибки;

$\sigma(x)$ – рассчитываемый полином локаторов ошибок;

k – номер итерации алгоритма;

d_k– значение невязки, т.е. сумма левой и правой части k+1-го уравнения, которая обращается в нуль, если полином локаторов удовлетворяет уравнению;

m – номер шага, на котором полином локаторов ошибок, последний раз модифицировался;

L – количество членов в левой части уравнений, которое отражает количество искаженных символов, предполагаемое во время итераций алгоритма;

$B(x)$ – полином модификации, который используется как вспомогательный для модификации полинома локаторов для «подстройки» его под уравнения системы.

Начиная с полинома локаторов ошибок $\sigma(x)$=1, вычисляется, так называемая, невязка (расхождение, несоответствие) этого полинома первому уравнению. Далее учитывая эту невязку, делается коррекция полинома локаторов, при необходимости наращивается его степень, а дальше все повторяется так, чтобы при условии удовлетворения всех предыдущих уравнений, он также стал удовлетворять и текущему уравнению, и так до тех пор, пока не будет удовлетворены все уравнения. Поскольку в алгоритме степень полинома наращивается только при необходимости, то решением и будет полином минимальной степени.

В результате работы алгоритма, в случае, если истинное количество искаженных символов больше, чем кратность исправляемых ошибок, возможно (но необязательно), что степень полинома локаторов не будет совпадать с предполагаемым количеством искаженных символов, т.е. $L \neq \deg[\sigma(x)]$, в этом случае считается, что решение системы уравнений не

найдено. Этот случай следует трактовать, как невозможность исправления ошибок. Однако, если даже L= degx[σ(x)], это все равно не является стопроцентной гарантией того, что найден корректный полином локаторов. Если полином локаторов σ(x) успешно найден, то тогда находят все корни полинома локаторов ошибок $\sigma(\alpha^j) = 0$ путем перебора всех ненулевых элементов поля Галуа.

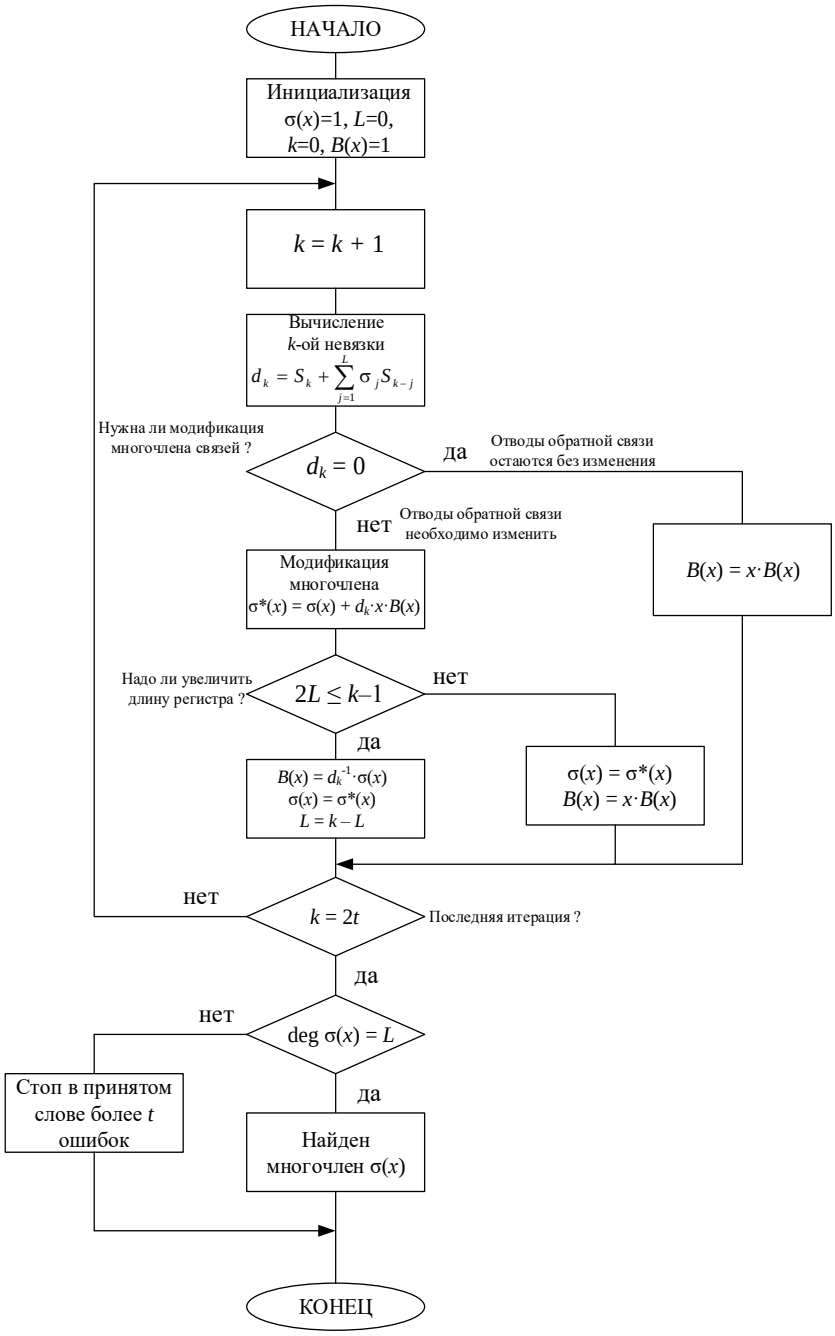

Рисунок 3.5 – Алгоритм Берлекэмпа-Месси для вычисления многочлена локаторов ошибок

Схема декодера для кодов БЧХ, работающего на основе алгоритма Берлекэмпа-Месси, представлена на рис. 3.6.

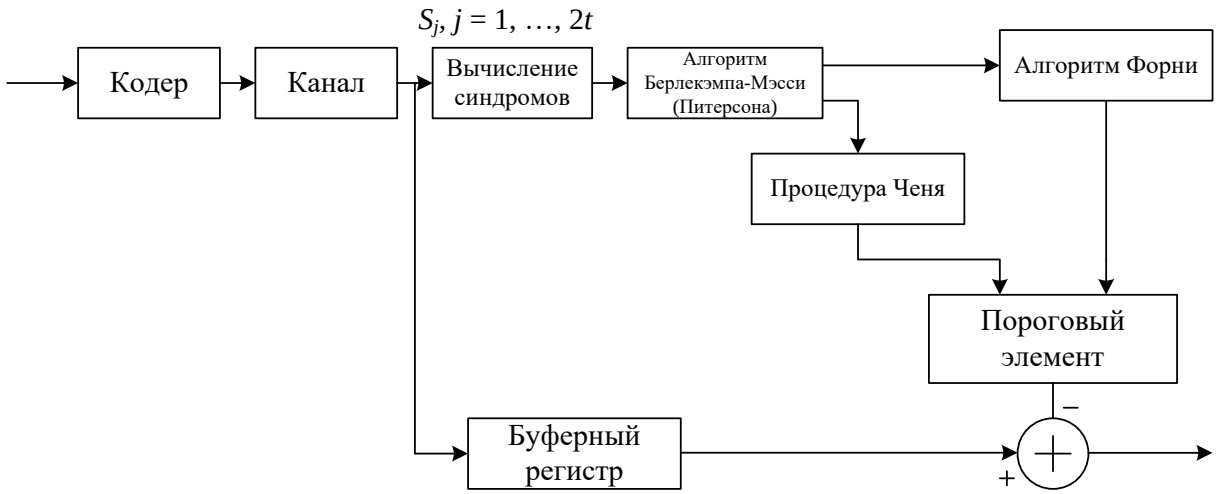

Рисунок 3.6 – Схема БЧХ кодера

Пример 3.8. В этом примере многочлен локаторов ошибок для декодирования принятой кодовой последовательности $f(x) = x^{14}+x^{11}+x^8+x^6+x^4+x^3+x^2+x+1$ (БЧХ код(15, 5, 7) с $t = 3$) из примера 3.7 находится с помощью алгоритма Берлекэмпа-Месси.

Запишем компоненты синдрома из примера 13

$\boldsymbol{S_1 = \alpha, S_2 = \alpha^2, S_3 = \alpha^8, S_4 = \alpha^4, S_5 = 1^0, S_6 = \alpha.}$

Выполним алгоритм Берлекэмпа-Месси и найдем многочлен локаторов ошибок $\sigma(x)$.

Инициализация.

$\sigma(x) = 1$, $L = 0$, $k = 0$, $B(x) = 1$.

Шаг 1. $k = 1$.

$$d_1 = S_1 + \sum_{j=1}^{L=0} \sigma_{j-1} S_{1-j} = S_1 = \alpha;$$

d_1=0, α=0 ? – нет → модификация многочлена обратной связи;

$\sigma^*(x) = \sigma(x)+d_1 \cdot x \cdot B(x) = 1+\alpha \cdot x$;

$2L \le k-1$, $0 \le 0$? – да → длина регистра увеличивается;

$B(x) = d_1^{-1} \cdot \sigma(x) = \alpha^{14} \cdot 1 = \alpha^{14}$; {для $d_1=\alpha$ обратный элемент $d_1^{-1} = \alpha^{14}$}

$L = k - L = 1 - 0 = 1$;

$\sigma(x) = \sigma^*(x) = 1+\alpha \cdot x$;

$k = 2t$, 1 = 6 ? – нет → увеличиваем шаг итерации;

$k = k+1 = 1 + 1 = 2.$

Шаг 2. $k = 2$.

$$d_2 = S_2 + \sum_{j=1}^{L=1} \sigma_{j-1} S_{2-j} = S_2 + \sigma_1 S_1 = \alpha^2 + \alpha \cdot \alpha = 0;$$

$d_2=0$, $0 = 0$? – да → без модификации многочлена обратной связи;

$B(x) = x \cdot B(x) = x \cdot \alpha^{14} = \alpha^{14} x$;

$k = 2t$, $2 = 6$? – нет → увеличиваем шаг итерации;

$k = k+1 = 2 + 1 = 3.$

Шаг 3. $k = 3$.

$$d_3 = S_3 + \sum_{j=1}^{L=1} \sigma_{j-1} S_{3-j} = S_3 + \sigma_1 S_2 = \alpha^8 + \alpha \cdot \alpha^2 = \alpha^8 + \alpha^3 = \alpha^{13};$$

$d_3=0$, $\alpha^{13} = 0$? – нет → модификация многочлена обратной связи;

$\sigma^*(x) = \sigma(x) + d_3 \cdot x \cdot B(x) = 1 + \alpha \cdot x + \alpha^{13} \cdot x \cdot \alpha^{14} x = 1 + \alpha \cdot x + \alpha^{27} x^2 = 1 + \alpha \cdot x + \alpha^{12} x^2$;

$2L \le k-1$, $2 \le 2$? – да → длина регистра увеличивается;

$B(x) = d_3^{-1} \cdot \sigma(x) = \alpha^2 \cdot (1 + \alpha x) = \alpha^2 + \alpha^3 x$; {для $d_3 = \alpha^{13}$ обратный элемент $d_3^{-1} = \alpha^2$}

$L = k - L = 3 - 1 = 2$;

$\sigma(x) = \sigma^*(x) = 1 + \alpha \cdot x + \alpha^{12} x^2$;

$k = 2t$, $3 = 6$? – нет → увеличиваем шаг итерации;

$k = k+1 = 3 + 1 = 4.$

Шаг 4. $k = 4$.

$$d_4 = S_4 + \sum_{j=1}^{L=2} \sigma_{j-1} S_{4-j} = S_4 + \sigma_1 S_3 + \sigma_2 S_2 = \alpha^4 + \alpha \cdot \alpha^8 + \alpha^{12} \cdot \alpha^2 =$$

$$= \alpha^4 + \alpha^9 + \alpha^{14} = \alpha^{14} + \alpha^{14} = 0;$$

$d_4=0$, $0 = 0$? – да → без модификации многочлена обратной связи;

$B(x) = x \cdot B(x) = x\,(\alpha^2 + \alpha^3 x) = \alpha^2 x + \alpha^3 x^2$;

$k = 2t$, $4 = 6$? – нет → увеличиваем шаг итерации;

$k = k+1 = 4 + 1 = 5.$

Шаг 5. $k = 5$.

$$d_5 = S_5 + \sum_{j=1}^{L=2} \sigma_{j-1} S_{5-j} = S_5 + \sigma_1 S_4 + \sigma_2 S_3 = 1 + \alpha \cdot \alpha^4 + \alpha^{12} \cdot \alpha^8 =$$

$$= 1 + \alpha^5 + \alpha^{20} = 1 + \alpha^5 + \alpha^5 = 1 + 0 = 1;$$

$d_5=0$, $1 = 0$? – нет → модификация многочлена обратной связи;

$\sigma^*(x) = \sigma(x) + d_5 \cdot x \cdot B(x) = (1 + \alpha x + \alpha^{12} x^2) + 1 \cdot x \cdot (\alpha^2 x + \alpha^3 x^2) = 1 + \alpha x + \alpha^{12} x^2 +$

$+ \alpha^2 x^2 + \alpha^3 x^3 = 1 + \alpha x + \alpha^7 x^2 + \alpha^3 x^3$;

$2L \le k-1$, $4 \le 4$? – да → длина регистра увеличивается;

$B(x) = d_5^{-1} \cdot \sigma(x) = 1 \cdot (1+\alpha \cdot x+\alpha^{12}x^2) = 1+\alpha \cdot x+\alpha^{12}x^2$ {для $d_5=\alpha^0$ обратный элемент $d_5^{-1}=1$};

$L = k - L = 5 - 2 = 3$;

$\sigma(x) = \sigma^*(x) = 1+\alpha x+\alpha^7 x^2+ \alpha^3 x^3$;

$k = 2t$, $5 = 6$? – нет → увеличиваем шаг итерации;

$k = k+1 = 5 + 1 = 6$.

Шаг 6. $k = 6$.

$$d_6 = S_5 + \sum_{j=1}^{L=3} \sigma_{j-1} S_{5-j} = S_6 + \sigma_1 S_5 + \sigma_2 S_4 + \sigma_3 S_3 = \alpha + \alpha \cdot 1 + \alpha^7 \cdot \alpha^4 + \alpha^3 \cdot \alpha^8 =$$

$$= \alpha + \alpha + \alpha^{11} + \alpha^{11} = 0 + 0 = 0;$$

$d_6=0$, $0 = 0$? – да → без модификации многочлена обратной связи;

$B(x) = x \cdot B(x) = x \cdot (1+\alpha \cdot x+\alpha^{12}x^2) = x+\alpha x^2+\alpha^{12}x^3$;

$k = 2t$, $6 = 6$? – да → последняя итерация, конец.

В таблице 3.7приведены необходимые вычисления для нахождения многочлена локаторов ошибок с помощью алгоритма Берлекэмпа-Месси.

Таблица 3.7 –Вычисление полинома локаторов ошибок с помощью алгоритма Берлекэмпа-Месси

k	d_k	$\sigma^*(x)$	$B(x)$	$\sigma(x)$	L
0	–	–	1	1	0
1	α	$1+\alpha \cdot x$	α^{14}	$1+\alpha \cdot x$	1
2	0	$1+\alpha \cdot x$	$\alpha^{14}x$	$1+\alpha \cdot x$	1
3	α^{13}	$1+\alpha \cdot x+\alpha^{12}x^2$	$\alpha^2+\alpha^3 x$	$1+\alpha \cdot x+\alpha^{12}x^2$	2
4	0	$1+\alpha \cdot x+\alpha^{12}x^2$	$\alpha^2 x+\alpha^3 x^2$	$1+\alpha \cdot x+\alpha^{12}x^2$	2
5	1	$1+\alpha x+\alpha^7 x^2+ \alpha^3 x^3$	$1+\alpha \cdot x+\alpha^{12}x^2$	$1+\alpha x+\alpha^7 x^2+ \alpha^3 x^3$	3
6	0	$1+\alpha x+\alpha^7 x^2+ \alpha^3 x^3$	$x+\alpha x^2+\alpha^{12}x^3$	$1+\alpha x+\alpha^7 x^2+ \alpha^3 x^3$	3

Таким образом, многочлен локаторов ошибок равен

$$\sigma(x) = 1+\alpha x+\alpha^7 x^2+ \alpha^3 x^3.$$

Полученный многочлен $\sigma(x)$ совпадает с тем, который был найден с помощью процедур Евклида. При этом, очевидно, что многочлен локаторов ошибок $\sigma(x)$, найденный алгоритмом Берлекэмпа-Месси имеет

те же корни, что и многочлен, построенный с помощью алгоритма Евклида: α^9, α^3, α^0.

Таким образом, позиции ошибок $j_1 = 12$, $j_2 = 6$, $j_3 = 0$. Вектор ошибок $e(x) = x^{12} + x^6 + 1$ и ошибки произошли во двенадцатом, шестом и нулевом разрядах передаваемой кодовой комбинации.

Описанный алгоритм может быть реализован в виде программы для универсальных или специализированных ЭВМ, предназначенных для вычислений в полях Галуа. По числу операций в конечном поле этот алгоритм обладает высокой эффективностью.

3.7 Характеристики БЧХ кодов

Стандартный алгоритм декодирования кодов БЧХ является алгоритмом с ограниченным расстоянием. Это означает, что ни одна комбинация, содержащая больше t ошибок, не исправляется. Вероятность ошибки последовательности для кода БЧХ длиной n, исправляющего t ошибок, задается вероятностью появления в кодовом слове более t ошибок, т.е.

$$P_s = \sum_{i=t+1}^{n} C_n^i p^i (1-p)^{n-i}, \qquad (3.42)$$

где p – вероятность ошибки символа в рассматриваемом канале.

Чтобы получить верхнюю границу вероятности ошибки символа при использовании двоичных БЧХ кодов, необходимо предположить, что набор $i>t$ ошибок в канале приведет к тому, что декодированное слово будет отличаться от переданного в $i+t$ позициях, так что $(i+t)/n$-я часть из k информационных символов будет декодирована неправильно. Таким образом, получим

$$P_b \le \sum_{i=t+1}^{n} \frac{i+t}{n} C_n^i p^i (1-p)^{n-i}. \qquad (3.43)$$

Для некоторых БЧХ кодов на рис. 3.7 представлены графики зависимости вероятности ошибки бита при использовании двоичной ФМ []. При этом используемые алгоритмы декодирования БЧХ кодов дают

значительный выигрыш от кодирования. Например, при длине блока $n = 511$ выигрыш составляет примерно 4 дБ (при $P_0 = 10^{-5}$)

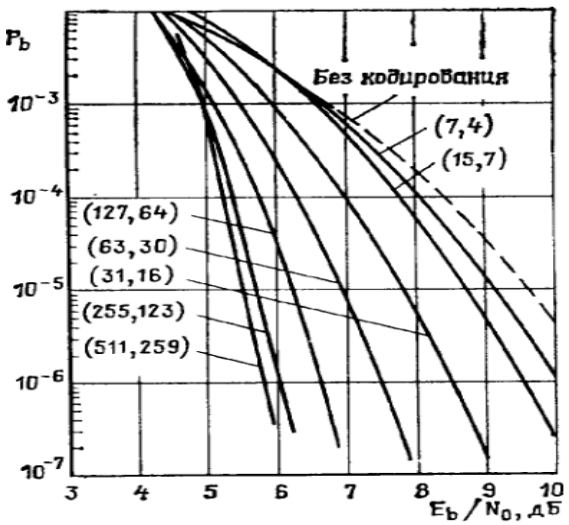

Рисунок 3.7 – Характеристики БЧХ кодов

Как следует из сопоставления представленных графиков, энергетическая эффективность кодов БЧХ с уменьшением кодовой скорости понижается. Более длинные коды, приводящие к несколько большему выигрышу, также допускают эффективное использование. Кроме того, допускается значительная гибкость при выборе скорости кода.

Также можно оценить параметры алгоритмов декодирования с исправлением ошибок и стираний, приведенных в главе 5. Описанные там алгоритмы гарантируют исправление всех комбинаций из t ошибок и s стираний, для которых выполняется условие $d \geq 2t + s + 1$. Любая комбинация, не удовлетворяющая этому условию, приведет к ошибке декодирования либо к отказу декодирования. Таким образом, предполагая, что вероятность ошибки p, а вероятность стирания s, получаем вероятность ошибки последовательности

$$P_s = \sum_{i=0}^{n} \sum_{\substack{j=2 \\ (j \geq 0)}}^{\substack{n-i \\ (t-i)+1}} C_n^{i,j} p^i s^i (1-p-s)^{n-i-j}, \qquad (3.44)$$

где $C_n^{i,j} = \dfrac{n!}{i!\, j!(n-i-j)!}$.

К сожалению, эти алгоритмы декодирования не очень эффективны в гауссовском канале. При фазовой модуляции с оптимальным

трехуровневым квантованием сигнала на выходе демодулятора выигрыш по сравнению с жестким решением составляет всего 0,2 дБ. Однако в некоторых других каналах алгебраический алгоритм исправления ошибок и стираний может оказаться весьма полезным. Например, при передаче по каналу с сильной интерференцией, которая может быть обнаружена и представлена как стирание.

БЧХ коды обладают существенно лучшей эффективностью по сравнению с кодами Хэмминга, однако эти характеристики все еще очень далеки от предельных значений. Кроме того, сложность декодирования данных кодов при большом значении n очень велика. Поэтому коды БЧХ небольшой длины, так же как и коды Хэмминга, в основном применяются в качестве составляющих элементов более эффективных каскадных кодов.

Контрольные вопросы

1. Дайте определения БЧХ кодов. Приведите пример примитивных БЧХ кодов.

2. Алгоритм построения двоичных БЧХ кодов.

3. Построить двоичный БЧХ код длины $n = 15$, который способен исправлять 3-х кратные ошибки.

4. Найти порождающий полином двоичного БЧХ кода длины $n = 15$, способного исправлять двойные ошибки.

5. Закодировать двоичным БЧХ кодом (15, 5, 7) информационную последовательность $u(x) = x^6+x^5+x^3+x+1$.

6. Алгоритм декодирования двоичных БЧХ кодов.

7. Решение ключевого уравнения для двоичных БЧХ кодов.

8. Нахождения компонент вектора ошибок двоичного БЧХ кода.

9. Нахождение многочлена локаторов ошибок и позиции ошибок.

10. Нахождение корней многочлена локаторов ошибок.

11. Алгоритм Берлекэмпа-Месси для решения ключевого уравнения.

12. Алгоритм Евклида для решения ключевого уравнения.

13. Алгоритм Питерсона-Горенстейна-Цирлера для решения ключевого уравнения.

14. Используя алгоритм декодирования Питерсона-Горенстейна-Цирлера найти многочлен локаторов ошибок, локаторы и значения ошибок, если значения синдромных компонент равны $S_1=\alpha$, $S_2=\alpha^2$, $S_3=\alpha^8$, $S_4=\alpha^4$, $S_5=1$, $S_6=\alpha$ для $GF(2^4)$.

15. Используя алгоритм декодирования Евклида найти многочлен локаторов ошибок, локаторы и значения ошибок, если значения синдромных компонент равны $S_1=\alpha$, $S_2=\alpha^2$, $S_3=\alpha^8$, $S_4=\alpha^4$, $S_5=1$, $S_6=\alpha$ для $GF(2^4)$.

16. Используя алгоритм декодирования Берлекэмпа-Месси найти многочлен локаторов ошибок, локаторы и значения ошибок, если значения синдромных компонент равны $S_1=\alpha$, $S_2=\alpha^2$, $S_3=\alpha^8$, $S_4=\alpha^4$, $S_5=1$, $S_6=\alpha$ для $GF(2^4)$.

17. Метод Ченя для нахождения корней многочлена ошибок.

18. Декодировать принятую кодовую последовательность $f(x) =x^{14}+x^{11}+x^8+x^6+x^4+x^3+x^2+x+1$ для БЧХ кода (15, 5, 7) с порождающим многочленом $g(x)=x^{10}+x^8+x^5+x^4+x^2+x+1$.

19. Рассмотрите двоичный код БЧХ длиной 15 в поле $GF(2^4)$, исправляющий три ошибки. Пусть принятое слово $f(x) = x^8+x^3+x$. Найдите многочлен локаторов ошибок с помощью алгоритма Евклида.

20. Рассмотрите двоичный код БЧХ длиной 15 в поле $GF(2^4)$, исправляющий три ошибки. Пусть принятое слово $f(x) = x^{12}+x^{10}+x^5$. Найдите многочлен локаторов ошибок с помощью алгоритма Берлекэмпа-Месси.

21. Рассмотрите двоичный код БЧХ длиной 15, исправляющий двойные ошибки, порождающий многочлен которого имеет корни α, α^2, α^3, α^4. Предполагая, что арифметические операции в $GF(2^4)$ задаются с помощью неприводимого многочлена $p(x) = x^3+x+1$, постройте блок-схему (с точностью, показывающей положение отдельных сумматоров по модулю 2) регистров для вычисления синдромов $S_1=f(\alpha)$ и $S_2=f(\alpha^3)$.

22. Для условий предыдущей задачи начертите структурную схему устройства, осуществляющего процедуру Ченя.

ГЛАВА 4.

КОДЫ РИДА-СОЛОМОНА – НЕДВОИЧНЫЕ БЧХ КОДЫ

4.1 Определение кодов Рида-Соломона

Особое место среди недвоичных БЧХ кодов занимают коды Рида-Соломона (РС коды), открытые еще до изобретения БЧХ кодов. Эти коды обладают уникальными свойствами, которые будут рассматриваться ниже. Они лежат в основе многих кодовых конструкций (каскадных кодов) и используются во многих прикладных задачах (контроллерах магнитных дисков, оптических дисков, накопителей на магнитных лентах). Также коды РС используются во многих системах передачи данных. Например, знаменитый РС код (255, 223, 33) используется в системах космической связи НАСА (NASA). На аудио компакт-дисках стандарта CD цифровая информация записывается с помощью комбинации укороченных (32, 28) и (28, 24) кодов РС. Стандарт цифрового телевизионного вещания DVB-T(Digital Video Broadcasting-Terrestrial)включает (204, 188) код РС, а его дальнейшая модификация DVB-H предусматривает в дополнение (255, 191) код РС.

Коды РС определяются над полем $GF(p^m)$ как множество кодовых слов, компоненты которых равны значениям некоторых определенных многочленов. Далее рассматриваются коды над конечными полями характеристики $p^m=2^m$, как наиболее распространенные. Таким образом, символы кодовых слов являются элементами поля $GF(2^m)$, а длина кода равна $n = 2^m-1$. Любое информационное сообщение может быть рассмотрено как последовательность из блоков (символов) по m бит. Тогда обозначим такое сообщение в виде информационного полинома степени $k-1$

$$u(x) = u_0 + u_1 x + u_2 x^2 + \quad ... \quad + u_{k-1} x^{k-1} \tag{4.1}$$

с коэффициентами u_i, которые представляют собой символы принадлежащие полю $GF(2^m)$, $i = 0, 1, \ldots, k-1$. Всего имеется 2^{mk} таких многочленов.

Например, информационная последовательность, состоящая из 12 бит равна u = 101011100110. Для представления в полиномиальной форме $u(x)$ в расширенном поле $GF(2^3)$ необходимо разбить информационную последовательность на блоки по m = 3 бита и каждому блоку поставить в соответствии с табл. 1.5 элемент поля

$101 = \alpha^6$, $011 = \alpha^3$, $100 = \alpha^2$, $110 = \alpha^4$.

Таким образом, информационная последовательность представляет полином вида

$u(x) = \alpha^6 x^3 + \alpha^3 x^2 + \alpha^2 x^1 + \alpha^4$.

Для кодов РС справедливо, что символы кодовых комбинаций (коэффициенты кодовых многочленов) и корни этих многочленов (нули кода) – это элементы одного и того же поля $GF(2^m)$. Вычисляя значения многочлена (4.1) для ненулевых элементов поля $GF(2^m)$ получаем кодовое слово $v(x)$ кода РС с параметрами $(2^m-1, k, d)$

$$v = (u(\alpha^0), u(\alpha^1), u(\alpha^2), ..., u(\alpha^{2^m-2})). \quad (4.2)$$

Нулями РС кода исправляющего t ошибок являются $2t$ последовательных степеней примитивного элемента поля Галуа α^b, α^{b+1}, ..., α^{b+2t-1}. Если при построении порождающего полинома $g(x)$ согласно формуле (3.1)игнорировать сопряженные по степени 2 циклы элементов α, α^2, …, α^{2t}и составлять его как произведение биномов вида $x + \alpha^i$, то получившийся полином окажется недвоичным. Вследствие этого код, порождаемый им, также окажется недвоичным, поскольку символы кода будут принадлежать расширенному полю $GF(2^m)$. Так как над полем $GF(2^m)$ минимальный многочлен равен $\varphi_i(x) = (x + \alpha^i)$ и поле символов и поле локаторов ошибок совпадают, то все минимальные многочлены и все делители порождающего код многочлена являются линейными (т.е. имеют степень 1). Таким образом, порождающий многочлен кода РС определяется

$$g(x) = \prod_{i=b}^{b+2t-1} (x + \alpha^i), \quad (4.3)$$

где b – любое целое число, которое принимает значение степени первого корня.

Как правило, значение bпринимают равным 0 или 1. Например, при b=1 порождающий полином записывается в виде

$$g(x) = (x + \alpha^1)(x + \alpha^2)\ldots(x + \alpha^{2t}). \quad (4.4)$$

Степень многочлена равна $2t$, откуда следует, что параметры кода РС связаны соотношением $n-k = 2t$. Тогда следуя границы БЧХ кодов минимальное кодовое расстояние (n, k, d) РС кода над расширенном поле $GF(2^m)$ удовлетворяет неравенству $d \geq n-k+1$.

Согласно границы Синглтона коды РС принадлежат классу кодов с максимальным достижимым расстоянием (классу МДР-кодов).

Граница Силтона. Для любого q-ичного кода с длиной n и минимальным расстоянием d количество кодовых слов удовлетворяет неравенству

$$M \leq q^{n-d+1}. \quad (4.5)$$

Коды, у которых число кодовых слов равно правой части неравенства (4.5) называются кодами с максимальным достижимым расстоянием. Для кодов РС $M = q^k$, тогда с учетом $d = n-k+1$ получим

$$q^k = q^{n-n+k-1+1} = q^k. \quad (4.6)$$

Таким образом, РС коды является МДР кодами и они оптимальны по критерию расстояния среди всех 2^m-ичных кодов той же длины и скорости.

Очевидно, что любой 2^m-ичный символ можно записать как двоичный
m-битовый блок и тогда мы получаем двоичный линейный код длины $n_b = mn = m(2^m-1)$ с числом информационных битов $k_b = mk$ и скоростью $R = k_b/n_b = k/n$. Минимальное кодовое расстояние такого кода не меньше $2t$и в классе двоичных он не обладает выдающимися свойствами в части исправления случайных ошибок. С другой стороны, такой код эффективен в борьбе с пакетами ошибок: пакет, накрывающий до $t = (d-1)/2$ последовательных 2^m-ичных символов искажает примерно в m раз больше

последовательных двоичных символов. Но такая конфигурация ошибок корректируется в силу исправления кодом РС вплоть до t ошибок. Поэтому коды РС способны исправлять пакеты ошибок вплоть до длины, близкой к mt. В этом заключается огромная популярность кодов РС в практических системах.

Пример 4.1. Построим РС код с минимальным кодовым расстоянием $d = 5$ над расширенным полем $GF(2^3)$, порождаемого неприводимым многочленом $p(x) = x^3+x+1$. Для этого кода, способного справлять двукратные ошибки, длина равна $n = 2^3-1=7$. Так как код способен исправлять $t = 2$ ошибки, то для построения порождающего полинома необходимо использовать $2t = 4$ последовательных элементов расширенного поля $GF(2^3)$. Для степени первого корня $b = 1$ и для элементов α^1, α^2, α^3, α^4 поля $GF(2^3)$, используя (4.3) найдем порождающий полином кода

$$g(x) = (x+\alpha^1)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4) = (x^3 + x\alpha^2 + x\alpha + \alpha^3)(x+\alpha^3)(x+\alpha^4) =$$
$$= (x^3 + x^2\alpha^2 + x^2\alpha + x\alpha^3 + x^2\alpha^3 + x\alpha^5 + x\alpha^4 + \alpha^6)(x+\alpha^4) = x^4 + x^3\alpha^2 + x^3\alpha + x^2\alpha^3 +$$
$$+ x^3\alpha^3 + x^2\alpha^5 + x^2\alpha^4 + x\alpha^6 + x^3\alpha^4 + x^2\alpha^6 + x^2\alpha^5 + x\alpha^7 + x^2\alpha^7 + x\alpha^9 + x\alpha^8 + \alpha^{10} =$$
$$= x^4 + (\alpha^2 + \alpha + \alpha^3 + \alpha^4)x^3 + (\alpha^3 + \alpha^5 + \alpha^4 + \alpha^6 + \alpha^5 + \alpha^7)x^2 + (\alpha^6 + \alpha^7 + \alpha^9 + \alpha^8)x + \alpha^{10}.$$

Используя таблицу сложения элементов расширенного поля $GF(2^3)$ (1.6), а также с учетом, что $\alpha^7=1$, $\alpha^9=\alpha^2$, $\alpha^8=\alpha^1$, $\alpha^{10}=\alpha^3$ получим

$$g(x) = x^4 + (\alpha^4 + \alpha^6)x^3 + (\alpha^5 + \alpha^4)x^2 + (\alpha^2 + \alpha^4)x + \alpha^3 = x^4 + \alpha^3x^3 + x^2 + \alpha x + \alpha^3.$$

Таким образом, найден код РС с параметрами (7, 3, 5) для которого информационный многочлен представляет собой последовательность трех восьмеричных символов (что эквивалентно 9 битам). При отображении символов $GF(2^3)$ в вектора длины 3 получаем двоичный (21, 9, 5) код, способный исправлять до двух случайных ошибок и любой пакет до 4-х ошибок.

Если выбрать значение степени первого корня $b = 0$ тогда для формирования порождающего полинома используются элементы α^0, α^1, α^2, α^3 и получим

$$g(x) = (x+\alpha^0)(x+\alpha^1)(x+\alpha^2)(x+\alpha^3) = x^4 + \alpha^2x^3 + \alpha^5x^2 + \alpha^5x + \alpha^6.$$

4.2 Принципы кодирования кодов Рида-Соломона

Для кодирования информационной последовательности $u(x)$ кодом РС (n, k, d) используются алгоритмы кодирования циклических кодов, которые были описаны в п. 2.2. Пусть задан полином

$$u(x) = u_{k-1}x^{k-1} + ... + u_1x^1 + u_0 \tag{4.7}$$

исходного информационного сообщения, а также найден полином $R(x)$вычисленный как остаток от деления полинома $u(x)$, сдвинутого на r позиций влево (умноженного на x^r), на порождающий полином $g(x)$. Иными словами пусть найден полином

$$R(x) = u(x) \cdot x^r \bmod (g(x)). \tag{4.8}$$

Тогда систематический код Рида Соломона представляет коэффициенты некоторого полинома $v(x)$ вычисляемого как сумма сдвинутого на r позиций влево информационного полинома и полинома остатка $R(x)$

$$v(x) = u(x) \cdot x^r + R(x). \tag{4.9}$$

При этом такое кодирование является систематическим в силу того, что информационные коэффициенты можно легко отделить от коэффициентов остатка (контрольных коэффициентов) без какого специального кодирования.

$\boldsymbol{u_{k-1}}$	**…**	$\boldsymbol{u_1}$	$\boldsymbol{u_0}$	$\boldsymbol{R_{r-1}}$	**…**	$\boldsymbol{R_0}$

Для получения несистематического кода РС необходимо информационный полином $u(x)$ на порождающий полином $g(x)$

$$v(x) = u(x) \cdot g(x). \tag{4.10}$$

При несистематическом кодировании информационные символы не отделяются от проверочных в кодовой последовательности РС.

Кроме того, относительно полинома $v(x)$ делается два достаточно простых, но важных утверждения:

- Полином $v(x)$ делится без остатка на порождающий полином $g(x)$.
- Уравнение $v(x)=0$ имеет те же корни, что и уравнение $g(x)=0$.

Корнями являются элементы поля Галуа $GF(2^m)$.

Как первое, так и второе утверждение является важным тем, что именно на нем и базируется вся логика обнаружения ошибок. Если коэффициенты полинома $v(x)$ не искажены, то его деление на порождающий полином даст нулевой остаток, так же как и подстановка в него любого из корней α^b, α^{b+1}, ..., α^{b+2t-1} обратит его в ноль. Если же коэффициенты полинома $v(x)$ были искажены (при передаче по сети или при хранении не более $r = 2t$ символов – всегда), деление на порождающий полином даст ненулевой остаток, также как и подстановка корней α^b, α^{b+1}, ..., α^{b+2t-1} в полином $v(x)$ даст ненулевые значения.

Тогда с учетом вышесказанного можно представить схему алгоритма кодирования (рис. 4.1) информационных сообщений с использованием кодов РС при заданной длине сообщения k и кратности исправляемой ошибки t.

Пример 4.2. Построить код Рида-Соломонанад расширенным полем $GF(2^3)$, порождаемого неприводимым многочленом $p(x) = x^3+x+1$. Код РС должен исправлять двукратные ошибки $t = 2$.Длина кода $n = 2^3–1=7$. Выбрать значение степени первого корня $b = 4$. Полученным несистематическим кодом закодировать информационную последовательность, состоящую из трех восьмеричных символов: $u(x) = \alpha^4x^2+x+\alpha^3$.

Для построения порождающего полинома заданного кода необходимо использовать $2t = 4$ последовательных элементов расширенного поля $GF(2^3)$: α^4, α^5, α^6, α^7. С учетом представления элементов поля $GF(2^3)$ и таблицы сложения элементов поля (табл. 1.6), а также $\alpha^7 = 1$ построим порождающий полином

$$g(x) = (x+\alpha^4)(x+\alpha^5)(x+\alpha^6)(x+1) = x^4 + (\alpha^2+1)x^3 + (\alpha^2+1)x^2 +$$
$$+(\alpha+1)x + \alpha = x^4 + \alpha^6x^3 + \alpha^6x^2 + \alpha^3x + \alpha.$$

Таким образом, построен (7, 3, 5) код РС с порождающим многочленом $g(x) = x^4+\alpha^6x^3+\alpha^6x^2+\alpha^3x+\alpha$.

Закодируем информационную последовательность $u(x)$ полученным кодом

$$v(x) = u(x) \cdot g(x) = (\alpha^4 x^2 + x + \alpha^3)(x^4 + \alpha^6 x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha) =$$

$$= \alpha^4 x^6 + \alpha^{10} x^5 + \alpha^{10} x^4 + \alpha^7 x^3 + \alpha^5 x^2 + x^5 + \alpha^6 x^4 + \alpha^6 x^3 + \alpha^3 x^2 + \alpha x + \alpha^3 x^4$$

$$+ \alpha^9 x^3 + \alpha^9 x^2 + \alpha^6 x + \alpha^4 = \alpha^4 x^6 + (\alpha^{10} + 1) x^5 + (\alpha^{10} + \alpha^6 + \alpha^3) x^4 +$$

$$(\alpha^7 + \alpha^6 + \alpha^9) x^3 + (\alpha^5 + \alpha^3 + \alpha^9) x^2 + (\alpha + \alpha^6) x + \alpha^4 =$$

$$= \alpha^4 x^6 + \alpha x^5 + \alpha^6 x^4 + 0 x^3 + 0 x^2 + \alpha^5 x + \alpha^4 = \alpha^4 x^6 + \alpha x^5 + \alpha^6 x^4 + \alpha^5 x + \alpha^4.$$

При передаче кодового слова по каналу связи или при хранении данных на носители, информация может быть искажена в силу действия шумов в канале передаче данных ли повреждения носителя данных. В таком случае можно говорить, что на кодовую последовательность буде наложено некоторое искажение $e(x)$ и полином $v(x)$ будет складываться вместе с некоторым полиномом искажений $e(x)$ и в результате будем иметь полином искаженного кадра $f(x) = v(x) + e(x)$. При этом могут искажаться любые коэффициенты кодового полинома, как информационные, так и избыточные.

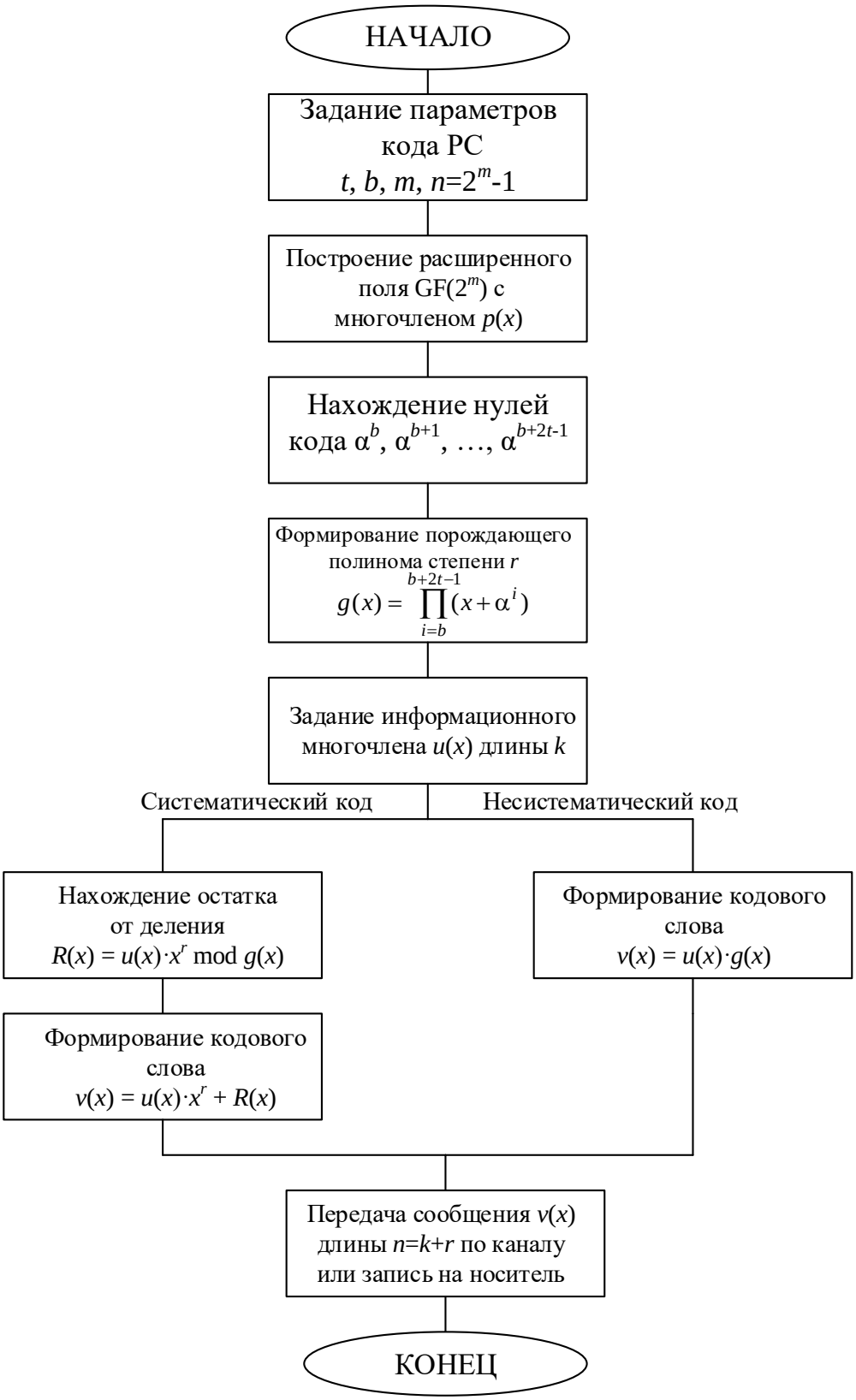

Рисунок 4.1 – Алгоритм кодирования кодом Рида-Соломона

4.3 Декодирование кодов Рида-Соломона

Пусть имеется принятый кодовый полином $f(x)$, состоящий из k информационных и r избыточных символов, принятый по сети или считанный с носителя и представляющий сумму $f(x) = v(x) + e(x)$.

Тогда *синдром ошибки* (синдром – совокупность симптомов, характеризующее заболевание) совокупность компонентов S_j, $j = 1, \ldots, 2t$, вычисляемых путем подстановки в полином $f(x)$ корней α^b, α^{b+1}, ..., α^{b+2t-1} порождающего полинома $g(x)$. Учитывая то, что мы ранее установили, что полином $f(x)$ неискаженной последовательности обращается в ноль при подстановке в него корней α^b, α^{b+1}, ..., α^{b+2t-1}, то справедливо равенство

$$S_j = f(\alpha^i) = e(\alpha^i). \qquad (4.11)$$

где j – номер компоненты синдрома ($j = 1, 2, \ldots, 2t$), i – степень корня элемента поля ($i = b, b+1, \ldots, b+2t-1$).

Таким образом, синдром не зависит от самого исходного неискаженной последовательности $v(x)$ и полностью характеризуется только ошибкой $e(x)$. Именно на этом и базируется вся технология расшифровки синдрома с целью нахождения локаторов ошибок и их величин

Кроме того, вводится понятие *полинома синдрома ошибки* $S(x)$, который может быть представлен как

$$S(x) = S_1 + S_2 x + \quad \ldots \quad + S_{2t} x^{2t-1}. \qquad (4.12)$$

Отметим, что в некоторых источник *синдромный полином* обозначается

$$S(x) = 1 + S_1 x + S_2 x^2 + \quad \ldots \quad + S_{2t} x^{2t}. \qquad (4.13)$$

Очевидно, что если все компоненты синдрома равны нулю, то можно говорить, что искажений не было, в противном случае, если синдром ненулевой, то можно декодировать и исправить ошибки, произошедшие в передаваемой кодовой последовательности.

Алгоритмы декодирования кодов РС близки к алгоритмам для декодирования двоичных БЧХ кодов. Как было указано в п. 3.3 для декодирования двоичных БЧХ кодов вычисляется многочлен локаторов ошибок

$$\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + ... + \sigma_v x^v, \qquad (4.14)$$

где v – число ошибок в принятом слове ($0 \le v \le t$).

С помощью корней α^jполинома локаторов ошибок $\sigma(x)$ находятся локаторы ошибок $\alpha^k=\alpha^{-j}$ и значения позиций ошибок $j_1, j_2, \ldots j_v$ (степени локаторов ошибок) для многочлена ошибок

$$e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + ... + e_{j_v} x^{j_v}. \qquad (4.15)$$

Так как для двоичных БЧХ кодов коэффициентами (значениями ошибок $e_{j_1}, e_{j_2}, ..., e_{j_v}$) полинома $e(x)$ являются биты 0 или 1, то для исправления ошибок необходимо инвертировать искаженные биты на найденных позициях.

Однако для декодирования кодов РС, которые имеют дело не с битами, а символами, необходимо найти не только позиции ошибок $j_1, j_2, \ldots j_v$, но и значения ошибок $e_{j_1}, e_{j_2}, ..., e_{j_v}$. При этом декодеру также неизвестно и количество ошибок v, которые произошли при передаче кодовой последовательности.

Для нахождения значений ошибок e_{j_l} введем понятие полинома значений ошибок, связав его с синдромным полиномом $S(x)$ и полиномом локаторов ошибок$\sigma(x)$ следующим образом

$$\Omega(x) = (S(x) \cdot \sigma(x)) \bmod x^{2t}. \qquad (4.16)$$

При этом полином локаторов ошибок имеет степень $\deg[\sigma(x)] = v$, а полином значений ошибок имеет степень $\deg[\Omega(x)] = v-1$.

Нахождение значений ошибок осуществляется с помощью *алгоритма Форни*, основанного на процедуре деления. Ниже представлено выражение, используемое для кодов РС с произвольным множеством $2t$ последовательных корней α^j для нахождения значения ошибки e_{j_l} на позициях $j_l (1 \le l \le v, b \le j \le b+2t-1)$

$$e_{j_l} = (\alpha^{j_l})^{1-b} \frac{\Omega(\alpha^{-j_l})}{\sigma'(\alpha^{-j_l})}, \tag{4.17}$$

где $\sigma'(x)$ − формальная производная по x многочлена локаторов ошибок $\sigma(x)$, $1 \le l \le v; b \le j \le b+2t-1$, b – степень первого корня; α^{-j} – корень$\sigma(x)$; α^{j} – локатор ошибки.

На этапе вычисления ошибок полином локаторов $\sigma(x)$ уже вычислен, поэтому формальную производную можно вычислить

$$\sigma'(x) = \sum_{j=1}^{v} (j\sigma_j) x^{j-1}. \tag{4.18}$$

В алгоритме декодирования кодов РС для решения ключевого уравнения

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_{v-1} & S_v \\ S_2 & S_3 & S_4 & \cdots & S_v & S_{v+1} \\ S_3 & S_4 & S_5 & \cdots & S_{v+1} & S_{v+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ S_v & S_{v+1} & S_{v+2} & \cdots & S_{2v-2} & S_{2v-1} \end{bmatrix} \begin{bmatrix} \sigma_v \\ \sigma_{v-1} \\ \sigma_{v-2} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ -S_{v+3} \\ \vdots \\ -S_{2v} \end{bmatrix}, \tag{4.19}$$

и нахождения коэффициентов σ_1, σ_2, …, σ_v полинома локаторов ошибок $\sigma(x)$ используются уже рассмотренные в разделе 3 методы:

− алгоритм Питерсона-Горенстейна-Цирлера (PGZ);

− алгоритм Берлекэмпа-Месси (BMA);

− алгоритм Евклида (EA).

Таким образом, в общем виде алгоритм декодирования кодов РС представлен на рис. 4.2. При этом если ошибки в принятой кодовой последовательности отсутствуют, то декодер получит нулевой синдром и примет верное решение об отсутствии искажений в последовательности $f(x)$. В случае если искажено не более t символов и получена искаженная комбинация, то синдром будет ненулевым, и декодер примет верное решение о наличии искажений в кадре, и о необходимости его исправления. Соответственно, декодер найдет истинный полином локаторов, найдет его корни, с помощью которых вычислит соответствующие локаторы и величины ошибок, и произведет восстановление исходной последовательности. Если в принятой

последовательности искажено более t символов, то декодеру не удастся найти полином локаторов или число найденных корней для полинома локаторов не совпадет с его степенью. В таких случаях декодер принимает решение о неисправимых искажениях.

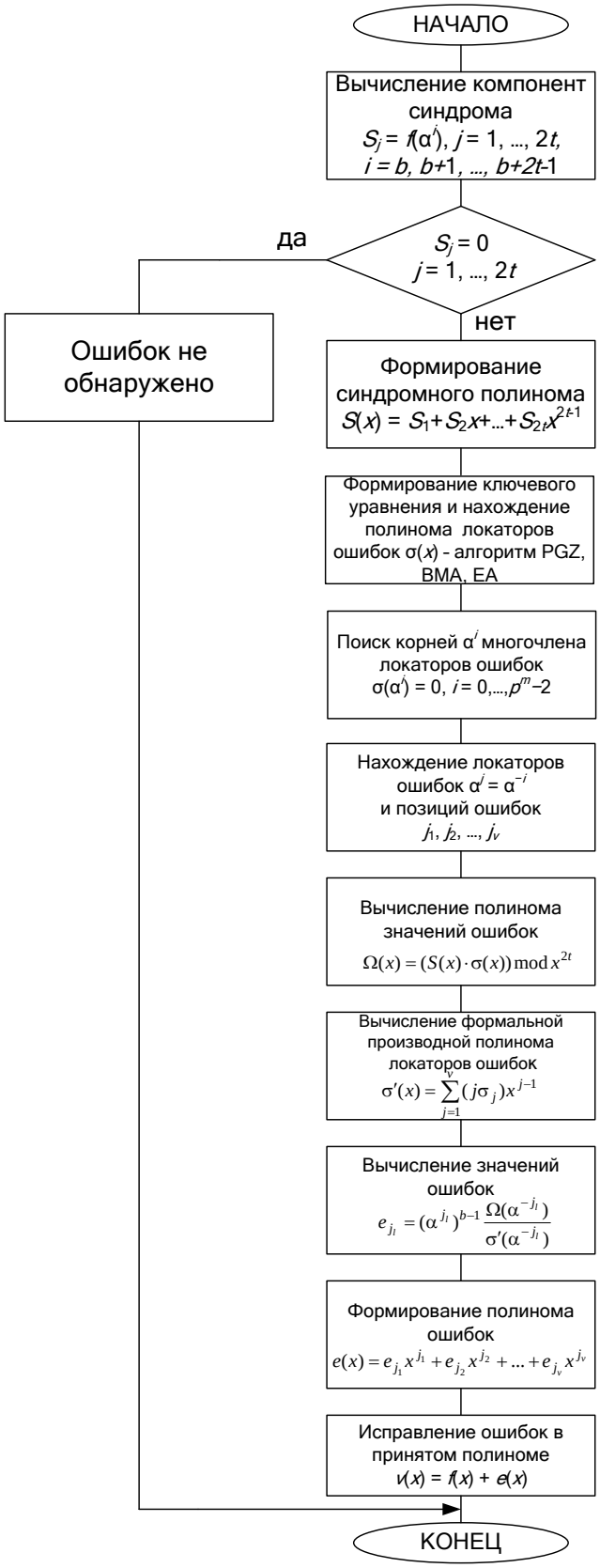

Рисунок 4.2 – Алгоритм декодирование кодов РС

4.4 Решение ключевого уравнения для декодера РС с помощью алгоритма Питерсона-Горенстейна-Цирлера

Для решения системы уравнения (4.18) предварительно необходимо выяснить, сколько ошибок в действительности произошло и система может быть решена только в том случае, когда порядок системы (размерность матрицы и число уравнений) равно действительному числу ошибок v. Число ошибок может быть определено с помощью следующей процедуры. Положить v равным максимальному значению t и вычислить определить матрицы $\det M_v$. Если он отличен от нуля, то принять $v=t$, в противном случае уменьшить t на единицу и повторить проверку. Остановиться на том значении v для которого впервые $\det M_v \neq 0$. Алгоритм для декодирования кодов РС по Питерсона-Горенстейна-Цирлера описан в п. 3.4 и представлен на рис. 3.1.

Пример 4.3. Пусть имеется код Рида-Соломона (7, 3, 5), построенного в примере 4.1 и способного исправлять двукратные ошибки $t=2$, с порождающим полиномом. Порождающий полином задается корнями $\alpha^0, \alpha^1, \alpha^2, \alpha^3$ и равен $g(x) = x^4+\alpha^2x^3+\alpha^5x^2+\alpha^5x+\alpha^6$. Расширенное поле $GF(2^3)$порождается неприводимым многочленом $p(x) = x^3 + x + 1$ (табл. 1.5). Кодовый многочлен, полученный с помощью несистематического кодирования $v(x) = u(x)g(x) = x^4 + \alpha^2x^3 + \alpha^5x^2 + \alpha^5x + \alpha^6$.При передаче кодовой последовательности $v(x)$ по каналу связи произошло две ошибки, и принятая последовательность имеет вид $f(x) = \alpha x^6 + x^4 + \alpha^2x^3 + \alpha^3x^2 + \alpha^5x + \alpha^6$. Таким образом, неизвестный для декодера вектор ошибок равен $e(x) = \alpha x^6 + \alpha^2x^2$. Необходимо восстановить исправленную кодовою последовательность и исправить ошибки.

Исходными данными для декодирования кодовой комбинации $f(x)$ являются: принятая кодовая последовательность $f(x) = \alpha x^6 + x^4 + \alpha^2x^3 + \alpha^3x^2 + \alpha^5x + \alpha^6$ и корни порождающего полинома $\alpha^0, \alpha^1, \alpha^2, \alpha^3$ и способность кода исправлять ошибки кратности $t=2$ включительно. При этом, локаторы ошибок, значения ошибок и корни порождающего полинома это элементы поля $GF(2^3)$.

Процесс декодирования начинается с вычисления компонентов синдрома S_j ($j = 1, 2, 3, 4$) с помощью формулы (4.11) и построения синдромного многочлена (4.12).

1. Вычисление компонент синдрома S_1,S_2, S_3, S_4.

$S_1 = f(x=\alpha^0) = \alpha+1+\alpha^2+\alpha^3+\alpha^5+\alpha^6 = \alpha^4$;

$S_2 = f(x=\alpha^1) = 1+\alpha^4+\alpha^5+\alpha^5+\alpha^6+\alpha^6 = \alpha^5$;

$S_3 = f(x=\alpha^2) = \alpha^6+\alpha+\alpha+1+1+\alpha^6 = 0$;

$S_4 = f(x=\alpha^3) = \alpha^5+\alpha^5+\alpha^4+\alpha^2+\alpha+\alpha^6 = \alpha^6$;

Синдромный многочлен $S(x) = \alpha^4+\alpha^5x+\alpha^6x^3$.

2. Определение числа ошибок.

Предположим, что произошло максимальное количество ошибок $i = t = 2$, тогда найдем определитель матрицы (3.22)

$$M_2 = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} \alpha^4 & \alpha^5 \\ \alpha^5 & 0 \end{bmatrix},$$

$\det M_2 = \alpha^4 \cdot 0+\alpha^5 \cdot \alpha^5 = \alpha^{10} = \alpha^3$. $\det M_2 \neq 0$.

Так как определитель матрицы не равен нулю число ошибок в принятом слове равно $v=2$.

3. Вычисление полинома локаторов ошибок.

Для нахождения коэффициентов σ_1, …, $\sigma_v = \sigma_1$, σ_2 полинома локаторов ошибок $\sigma(x)$ необходимо решить систему уравнений (4.18)

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \cdot \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} \alpha^4 & \alpha^5 \\ \alpha^5 & 0 \end{bmatrix} \cdot \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha^6 \end{bmatrix}.$$

Тогда коэффициенты σ_1, σ_2 находятся как

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = M_2^{-1} \begin{bmatrix} 0 \\ \alpha^6 \end{bmatrix},$$

где M_2^{-1} − обратная матрица M_2.

Для нахождения обратной матрицы используется формула

$$M^{-1} = \frac{1}{|M|} \left|A_{ij}\right|^T,$$

где $|M|$ − определитель матрицы, A_{ij} – транспонированная матрица, элементами которой являются алгебраические дополнения a_{ij}.

Алгебраические дополнения a_{ij} получаются из матрицы M путем нахождения минора матрицы M (вычеркивания i-ой строки и j-го столбца)умноженного на $(-1)^{i+j}$.

Найдем алгебраические дополнения для матрицы $M_2 = \begin{bmatrix} \alpha^4 & \alpha^5 \\ \alpha^5 & 0 \end{bmatrix}$:

$a_{11} = (-1)^2 \cdot 0 = 0$; $a_{12} = (-1)^3 \cdot \alpha^5 = \alpha^5$;

$a_{21} = (-1)^3 \cdot \alpha^5 = \alpha^5$; $a_{22} = (-1)^4 \cdot \alpha^4 = \alpha^4$.

Построим обратную матрицу M_2^{-1}

$$M_2^{-1} = \frac{1}{|M|}\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \frac{1}{\alpha^3}\begin{bmatrix} 0 & \alpha^5 \\ \alpha^5 & \alpha^4 \end{bmatrix} = \alpha^4\begin{bmatrix} 0 & \alpha^5 \\ \alpha^5 & \alpha^4 \end{bmatrix} = \begin{bmatrix} 0 & \alpha^2 \\ \alpha^2 & \alpha^1 \end{bmatrix}.$$

Находим коэффициенты полинома локаторов ошибок

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 0 & \alpha^2 \\ \alpha^2 & \alpha^1 \end{bmatrix}\begin{bmatrix} 0 \\ \alpha^6 \end{bmatrix} = \begin{bmatrix} 0 \cdot 0 + \alpha^6\alpha^2 \\ \alpha^2 \cdot 0 + \alpha^1\alpha^6 \end{bmatrix} = \begin{bmatrix} \alpha^1 \\ 1 \end{bmatrix}.$$

Следовательно, коэффициенты равны $\sigma_1 = 1$, $\sigma_2 = \alpha^1$ и искомый полином локаторов ошибок $\sigma(x) = 1 + x + \alpha x^2$.

4. Нахождение локаторов ошибок.

Поиск корней α^i полинома локаторов ошибок осуществляется с помощью метода Ченя – проверка условия $\sigma(\alpha^j) = 0$ для всех ненулевых элементов поля $GF(2^3)$ $\{1, \alpha, \alpha^2, \ldots, \alpha^7\}$.

$\sigma(1) = 1+1+\alpha \cdot 1 = \alpha$;

$\sigma(\alpha) = 1+\alpha+\alpha \cdot \alpha^2 = \alpha^3+\alpha^3 = 0$;

$\sigma(\alpha^2) = 1+\alpha^2+\alpha \cdot \alpha^4 = \alpha^6+\alpha^5 = \alpha^4$;

$\sigma(\alpha^3) = 1+\alpha^3+\alpha \cdot \alpha^6 = \alpha+1 = \alpha^3$;

$\sigma(\alpha^4) = 1+\alpha^4+\alpha \cdot \alpha = \alpha^5+\alpha^2 = 1$;

$\sigma(\alpha^5) = 1+\alpha^5+\alpha \cdot \alpha^3 = \alpha^4+\alpha^4 = 0$;

$\sigma(\alpha^6) = 1+\alpha^6+\alpha \cdot \alpha^5 = \alpha^2+\alpha^6 = 1$;

$\sigma(\alpha^7) = 1+\alpha^7+\alpha \cdot 1 = 0+\alpha = \alpha$.

Корнями полинома $\sigma(x)$ являются элементы α, α^5.

Локаторы ошибок равны элементам, которые являются обратными корням многочлена ошибок:

– для элемента α^1 обратным является элемент α^6, так как из табл. 1.7 находим $\alpha^1 \cdot \alpha^6 = 1$;

– для элемента α^5 обратным является элемент α^2, так как из табл. 1.7 находим $\alpha^5 \cdot \alpha^2 = 1$.

Локаторами ошибок являются элементы α^6, α^2.

5. Нахождение позиций ошибок

Позиции ошибок находятся как значения степеней локаторов ошибок α^6, α^2.

Ошибки находятся на позициях $j_1 = 6$, $j_2 = 2$.

6. Определение полинома значений ошибок.

Полином значений ошибок определяется $\Omega(x) = (S(x) \cdot \sigma(x)) \bmod x^{2t}$.

$S(x)\cdot\sigma(x) = (\alpha^4+\alpha^5x+\alpha^6x^3)\cdot(1+x+\alpha x^2) = \alpha^4+\alpha^4x+\alpha^5x^2+\alpha^5x+\alpha^5x^2+\alpha^6x^3+\alpha^6x^3+$
$\alpha^6x^4+\alpha^7x^5 = \alpha^4+(\alpha^4+\alpha^5)x+(\alpha^5+\alpha^5)x^2+(\alpha^6+\alpha^6)x^3+\alpha^6x^4+\alpha^7x^5=\alpha^4+x+0\cdot x^2+$
$+0\cdot x^3+\alpha^6x^4+x^5$.

$\alpha^4+x+\alpha^6x^4+x^5 \bmod x^4 = x+\alpha^4$, $\Omega(x) = x+\alpha^4$.

7. Вычисление значений ошибок и формирование многочлена ошибок.

Нахождение величин ошибок (коэффициентов)e_1, e_2для формирования полинома ошибок $e(x)$ находятся с помощью метода Форни

$$e_{j_l} = (\alpha^{j_l})^{1-b}\frac{\Omega(\alpha^{-j_l})}{\sigma'(\alpha^{-j_l})}.$$

Для вычисления значений ошибок найдем формальную производную $\sigma'(x)$ полинома локаторов ошибок $\sigma(x) = 1 + x + \alpha x^2$

$\sigma'(x) = 0 + 1 + 2\alpha x = 1.$

Вычислим значение коэффициента e_1 для локатора ошибки $\alpha^j = \alpha^6$ и для корня $\alpha^{-j} = \alpha^1$, $\Omega(x) = x+\alpha^4$.

$$e_1 = (\alpha^6)^{1-0}\frac{\Omega(x=\alpha^1)}{\sigma'(x=\alpha^1)} = \alpha^6\frac{\alpha^1+\alpha^4}{1} = \alpha^6\cdot\alpha^2 = \alpha^8 = \alpha.$$

Вычислим значение коэффициента e_1 для локатора ошибки $\alpha^j = \alpha^2$ и для корня $\alpha^{-j} = \alpha^5$, $\Omega(x) = x+\alpha^4$.

$$e_2 = (\alpha^2)^{1-0}\frac{\Omega(x=\alpha^5)}{\sigma'(x=\alpha^5)} = \alpha^2\frac{\alpha^5+\alpha^4}{1} = \alpha^2\cdot 1 = \alpha^2.$$

На позиции $j_1 = 6$ значение ошибки равно – $e_1 = \alpha$.

На позиции $j_2 = 2$ значение ошибки равно – $e_2 = \alpha^2$.

Таким образом, многочлен ошибок $e(x) = e_1x^{j_1} + e_2x^{j_2} = \alpha x^6 + \alpha^2x^2$.

8. Исправление ошибок.

Исправление ошибок в принятой последовательности $f(x)$ осуществляется путем сложения комбинации $f(x)$ и полинома ошибок $e(x)$

$v(x) = f(x) + e(x) = (\alpha x^6 + x^4 + \alpha^2x^3 + \alpha^3x^2 + \alpha^5x + \alpha^6) + (\alpha x^6 + \alpha^2x^2) =$
$= x^4 + \alpha^2x^3 + \alpha^5x^2 + \alpha^5x + \alpha^6.$

Таким образом, исправлены две ошибки и восстановленный кодовый полином равен $v(x) = x^4 + \alpha^2x^3 + \alpha^5x^2 + \alpha^5x + \alpha^6$.

4.5 Решение ключевого уравнения для декодера РС с помощью алгоритма Евклида

В алгоритме декодирования кодов Рида-Соломона для решения ключевого уравнения и нахождения полинома локаторов ошибок может использоваться алгоритм Евклида, который описан в п. 3.5 для двоичных кодов БЧХ. При этом алгоритм Евклида вычисляет не только полином локаторов ошибок $\sigma(x)$, но и одновременно полином значений ошибок $\Omega(x)$.

В основе алгоритма лежит уже рассмотренная в п. 3.5 процедура нахождения НОД двух полиномов $d(x) = x^2$ и $c(x) = S(x)$ и при декодировании интересуются не конечным результатом алгоритма Эвклида, а промежуточными результатами. Если на i-ом шаге алгоритма для некоторых значений $\Omega_i(x) = r_i(x)$, $\sigma_i(x) = b_i(x)$ и $\deg[r_i(x)] < t$ получено выражение

$$r_i(x) = a_i(x)x^{2t} + b_i(x)S(x), \tag{4.19}$$

тогда полином значений ошибок равен

$$\Omega(x) = S(x)\sigma(x) + x^{2t}a(x). \tag{4.20}$$

При этом используется свойство алгоритма Евклида:

$\deg[b_i(x)]+\deg[r_{i-1}(x)] = \deg[d(x)]$.

Если $d(x) = x^{2t}$, то $\deg[\sigma_i(x)]+\deg[\Omega_{i-1}(x)] = 2t$,

$\deg[\sigma_i(x)]+\deg[\Omega_i(x)] < 2t$.

При появлении $v \le t$ошибок имеем$\deg[\Omega_i(x)] < \deg[\sigma_i(x)] \le t$.

Существует единственный с точностью до постоянного множителя (элемента поля) многочлен $\sigma_i(x)$ степени меньше или равно t, удовлетворяющий уравнению (4.16). Кроме того, если $\deg[\Omega_{i-1}(x)] \ge t$ при $\deg[\sigma_i(x)] \le t$и $\deg[\Omega_i(x)] < t$, то $\deg[\sigma_{i+1}(x)] > t$. Поэтому промежуточные результаты на i-ом шаге дают единственное интересующее нас решение ключевого уравнения.

Таким образом, алгоритм Евклида для решения ключевого уравнения следует применять до тех пор, пока не будет выполнено условие $\deg[\Omega_i(x)] < t$, если синдромный многочлен вычислялся по формуле (4.12).Важно отметить, если синдромный многочлен формируется по

формуле (4.13), то условием окончания работы алгоритма Евклида является $\deg[\Omega_i(x)] \le t$,

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

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

После того, как многочлены локаторов и значений ошибок найдены, многочлен ошибок вычисляется по алгоритму Форни.

Алгоритм Евклида для декодирования кодового полинома $f(x)$ закодированного кодом РС, способного исправлять ошибки кратности t, длины $n = 2^m-1$ с порождающим многочленом $g(x)$, корнями которого являются элементы α^b, α^{b+1}, ..., α^{b+2t-1} расширенного поля $GF(2^m)$, заключается в следующем:

1. Вычислить компоненты синдрома

$$S_j = f(\alpha^i),\ j = 1, \ldots, 2t,\ i = b, \ldots, b+2t-1.$$

2. Построить синдромный многочлен $S(x)$

$$S(x) = S_1 + S_2 x + \quad \ldots \quad + S_{2t} x^{2t-1}.$$

3. Установить начальные значения

$\sigma_{-1}(x) = 0,\ \sigma_0(x) = 1,\ \Omega_{-1}(x) = x^{2t},\ \Omega_0(x) = S(x)$.

4. Проверить степень полинома $\deg[\Omega_i(x)] < t$, если степень меньше значения t, тогда выполнить пункт 5, иначе выполнить пункт 7.

5. Вычислить значение $q_i(x) = \left[\frac{\Omega_{i-2}(x)}{\Omega_{i-1}(x)}\right]$.

6. Установить новые значения и выполнить пункт 4

$\Omega_i(x) = \Omega_{i-2}(x) + q_i(x) \cdot \Omega_{i-1}(x)$, $\sigma_i(x) = \sigma_{i-2}(x) + q_i(x) \cdot \sigma_{i-1}(x)$.

7. Положить $\sigma(x) = \sigma_i(x), \Omega(x) = \Omega_i(x)$.

8. Найти значение корней α^i полинома локаторов ошибок $\sigma(x)$ с помощью метода Ченя $\sigma(\alpha^i) = 0$, для $i = 0, 1, \ldots, 2^m-2$.

9. Найти локаторы ошибок α^j, как элементы обратные корням α^i многочлена локаторов ошибок α^j,= α^{-i}. Степени локаторов ошибок являются значениями позиций ошибок $j_1, j_2 \ldots j_v$.

10. Вычислить формальную производную многочлена локаторов ошибок $\sigma'(x)$

$$\sigma'(x) = \sum_{j=1}^{v} (j\sigma_j) x^{j-1}.$$

11. С помощью алгоритма Форни найти значения ошибок ($l = 1, \ldots, v$)

$$e_{j_l} = (\alpha^{j_l})^{1-b} \frac{\Omega(\alpha^{-j_l})}{\sigma'(\alpha^{-j_l})}.$$

12. Формируется многочлен ошибок $e(x)$ по формуле (4.15), коэффициентами которого являются значения ошибок e_l,а степенями слагаемых являются локаторы ошибок.

13. Исправить ошибки в принятой кодовой последовательности, посредством сложения кодовой комбинации и многочлена ошибок$v(x) = f(x) + e(x)$.

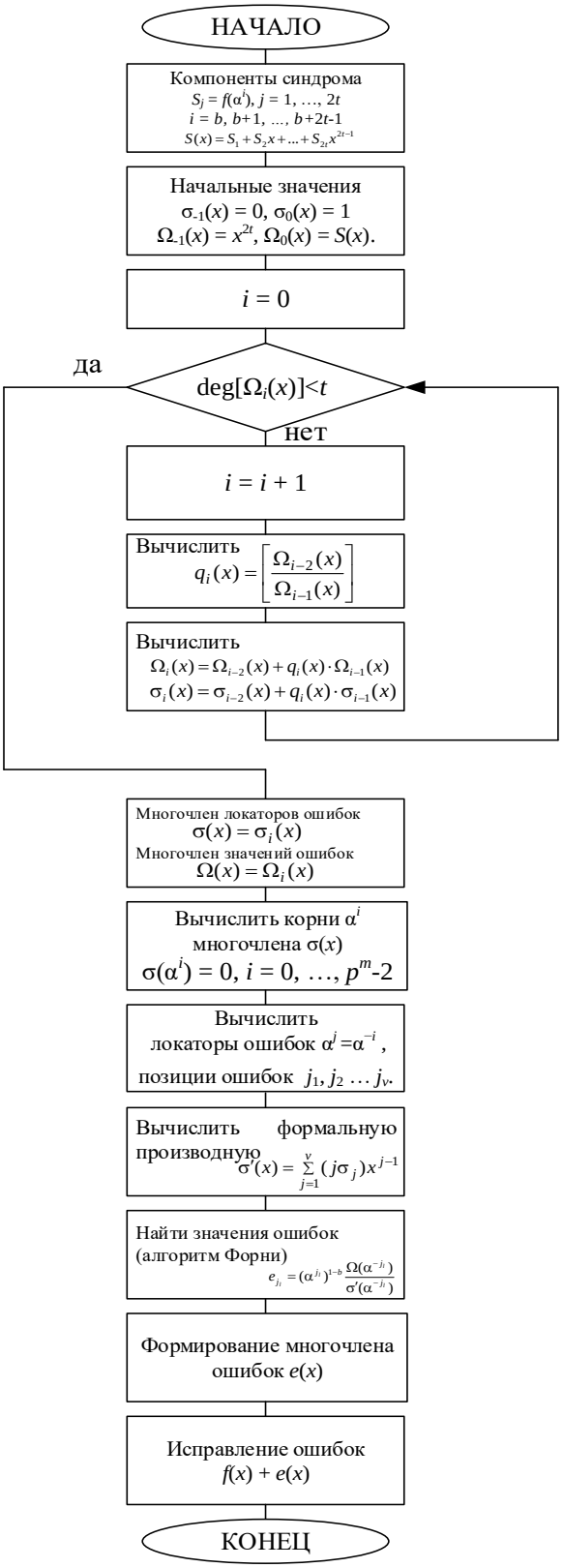

Рисунок 4.3 – Декодирование кодов РС при помощи алгоритма Евклида

Пример 4.4. Для условий задания указанных в примере 4.3 необходимо осуществить декодирования принятой кодовой последовательности $f(x) = \alpha x^6 + x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha^5 x + \alpha^6$ с помощью алгоритма Евклида.

1. Вычислить компоненты синдрома S_1,S_2, S_3, S_4**.**

Из примера 4.3 $S_1 = \alpha^4$;$S_2 = \alpha^5$;$S_3 = 0$;$S_4 = \alpha^6$.

Синдромный многочлен$S(x) = \alpha^4+\alpha^5 x+\alpha^6 x^3$.

2. Установить начальные значения.

$\sigma_{-1}(x) = 0, \sigma_0(x) = 1, \Omega_{-1}(x) = x^4, \Omega_0(x) = S(x) = \alpha^4 + \alpha^5 x + \alpha^6 x^3$.

3. Так как условие $\deg[\Omega_i(x)] < t$ не выполняется и степень полинома$\Omega_i(x)$больше чем количество исправляемых ошибок кодом:$\Omega_0(x)<2\rightarrow(3<2)$, то выполним деление

$$q_1(x) = \left[\frac{\Omega_{-1}(x)}{\Omega_0(x)}\right] = \left[\frac{x^4}{\alpha^4 + \alpha^5 x + \alpha^6 x^3}\right] = \alpha x.$$

4. Вычислить значения $\sigma_1(x)$, $\Omega_1(x)$.

$\sigma_1(x) = \sigma_{-1}(x) + q_1 \cdot \sigma_0(x) = 0 + \alpha x \cdot 1 = \alpha x$,

$\Omega_1(x) = \Omega_{-1}(x) + q_1 \cdot \Omega_0(x) = x^4 + \alpha x \cdot (\alpha^4 + \alpha^5 x + \alpha^6 x^3) =$

$x^4 + \alpha^5 x + \alpha^6 x^2 + x^4 = \alpha^5 x + \alpha^6 x^2$.

5. Так как условие $\deg[\Omega_i(x)] < t$ не выполняется и степень полинома $\Omega_i(x)$ больше чем количество исправляемых ошибок кодом $\Omega_1(x)<2 \rightarrow (2<2)$, то выполним деление

$$q_2(x) = \left[\frac{\Omega_0(x)}{\Omega_1(x)}\right] = \left[\frac{\alpha^4 + \alpha^5 x + \alpha^6 x^3}{\alpha^5 x + \alpha^6 x^2}\right] = x + \alpha^6.$$

6. Вычислить значения $\sigma_2(x)$, $\Omega_2(x)$.

$\sigma_2(x) = \sigma_0(x) + q_2 \cdot \sigma_1(x) = 1 + (x + \alpha^6) \cdot \alpha x = 1 + \alpha^1 x^2 + \alpha^7 x = 1 + x + \alpha x^2$,

$\Omega_2(x) = \Omega_0(x) + q_2 \cdot \Omega_1(x) = (\alpha^4 + \alpha^5 x + \alpha^6 x^3) + (x + \alpha^6) \cdot (\alpha^5 x + \alpha^6 x^2) =$

$= \alpha^4 + \alpha^5 x + \alpha^6 x^3 + \alpha^5 x^2 + \alpha^6 x^3 + \alpha^4 x + \alpha^5 x^2 = \alpha^4 + x.$

7. Условие $\deg[\Omega_i(x)] < t$ выполняется, т.к степень полинома $\Omega_i(x)$ меньше количества исправляемых ошибок $\Omega_2(x)<2\rightarrow(1<2)$, таким образом процедура декодирования завершается и получены:

− многочлен локаторов ошибок $\sigma(x) = \sigma_2(x) = 1 + x + \alpha x^2$,

− многочлен значение ошибок $\Omega(x) = \Omega_2(x) = x + \alpha^4$.

8. Корнями многочлена локаторов ошибок $\sigma(x)$ являются элементы α, α^5, соответственно локаторами ошибок – элементы α^6, α^2.(пример 4.3).

9. Формальная производная $\sigma'(x)$ для полинома локаторов ошибок

$\sigma'(x) = 0 + 1 + 2\alpha x = 1.$

10. Значения ошибок и многочлен ошибок $e(x)$

$$e_1 = (\alpha^6)^{1-0} \frac{\Omega(x = \alpha^1)}{\sigma'(x = \alpha^1)} = \alpha^6 \frac{\alpha^1 + \alpha^4}{1} = \alpha^6 \cdot \alpha^2 = \alpha^8 = \alpha^1.$$

$$e_2 = (\alpha^2)^{1-0} \frac{\Omega(x = \alpha^5)}{\sigma'(x = \alpha^5)} = \alpha^2 \frac{\alpha^5 + \alpha^4}{1} = \alpha^2 \cdot 1 = \alpha^2.$$

Значения ошибок:

для локатора ошибки α^6 – α;

для локатора ошибки α^2 – α^2.

Таким образом, многочлен ошибок $e(x) = \alpha x^6 + \alpha^2 x^2$.

11. Исправление ошибок в принятой последовательности $f(x)$ осуществляется путем сложения комбинации $f(x)$ и полинома ошибок $e(x)v(x) = f(x) + e(x) = (\alpha x^6 + x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha^5 x + \alpha^6) + (\alpha x^6 + \alpha^2 x^2) =$
$x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 x + \alpha^6$.

Результат декодирования полностью совпадает с полученным результатом в примере 4.3.

4.6 Решение ключевого уравнения для декодера РС с помощью алгоритма Берлекэмпа-Месси

Для декодирования кодов РС и нахождения коэффициентов σ_1, σ_2, …, σ_v полинома локаторов ошибок $\sigma(x)$ используется алгоритм Берлекэмпа-Месси, который описан для БЧХ кодов в п. 3.6. Описываемый алгоритм решает совместно две задачи:

− определение длины регистра сдвига L;

− определение коэффициентов полинома $\sigma(x)$.

Это упрощение возможно, благодаря тому, что матрица коэффициентов имеет специфический вид: строки коэффициентов являются сдвигами предыдущих строк.

Для решения ключевого уравнения (4.18) необходимо найти регистр сдвига с обратной связью наименьшей длины L, генерирующий заданную последовательность S_1, S_2, …, S_{2v} при начальном состоянии S_v, S_{v-1}, …S_1.

Процедура является рекуррентной. Для каждого k, начиная с k=1, строится регистр, генерирующий первые k элементов последовательности (S_1, S_2, …, S_{2v}). На k-ой итерации вычисляется k-й выход предыдущего ($k-1$)-го регистра

$$S_k^{'} = -\sum_{j=1}^{L} \sigma_j^{(r-1)} S_{k-j} \,. \tag{4.21}$$

При этом истинный элемент последовательности S_k может не совпадать с $S_k^{'}$. Тогда вычисляется невязка (несовместность)

$$d_k = S_k - S_k^{'} = S_k + \sum_{j=1}^{L} \sigma_j^{(k-1)} S_{k-j} \,, \tag{4.22}$$

или

$$d_k = \sum_{j=0}^{L} \sigma_j^{k-1} S_{k-j} \,. \tag{4.23}$$

Если невязка нулевая, то итерация закончена. В противном случае, необходимо модифицировать вектор, задающий регистр, чтобы сделать ее нулевой. Новый полином вычисляется

$$\sigma^{(k)}(x) = \sigma^{(k-1)}(x) + d_k d_m^{-1} x^{k-m} \sigma^{(m-1)}(x) \,, \tag{4.24}$$

где $\sigma^{(m-1)}(x)$ – один из многочленов регистра сдвига, встречавшихся в одной из предыдущих итераций, при этом выбирается$m<k$и такое, что $d_m \neq 0$, т.е. mравно номеру последнего шага, предшествующего шагу k, на котором $d_m \neq 0$.

Начальными значениями для алгоритма являются

$\sigma(x) = 1$, $L = 0$, $k = 0$, $B(x) = 1$.

Алгоритм Берлекэмпа-Месси для вычисления многочлена локаторов ошибок.

1. Инициализация $\sigma(x) = 1$, $L = 0$, $k = 0$, $B(x) = 1$.
2. Полагаем $k = k + 1$. Вычисляем невязку $d_k = S_k - S_k^{'} = S_k + \sum_{j=1}^{n-1} \sigma_{j-1}^{(k-1)} S_{k-j}$.
3. Если $d_k = 0$, выполняем сдвиг $B(x) = x \cdot B(x)$ и переходим к пункту 5, в противном случае выполняем пункт 4.

4. Формируем новый полином $\sigma'(x) = \sigma(x) + d_k\, x \cdot B(x)$. Сохраняем предыдущий полином $B(x) = d_k^{-1}\sigma(x)$. Меняем связи регистра сдвига $\sigma(x) = \sigma'(x)$, а также длину регистра сдвига $L = k - L$.

5. Если $k \neq 2t$, возвращаемся к пункту 2. В противном случае работа закончена и результаты работы алгоритма – полином локаторов ошибок $\sigma(x)$ и длина регистра сдвига $L = \deg[\sigma(x)]$.

Используя найденный полином локаторов ошибок $\sigma(x)$ находят локаторы ошибок α^j и соответственно позиции ошибок j_1, j_2 … j_v. Для нахождения значений ошибок e_{j_l} вычисляют полином значений ошибок, следующим образом $\Omega(x) = (S(x) \cdot \sigma(x)) \bmod x^{2t}$. С помощью алгоритма Форни находят значения ошибок $e_l = (\alpha^j)^{1-b} \dfrac{\Omega(\alpha^{-j})}{\sigma'(\alpha^{-j})}$.

Пример 4.5. Для условий задания указанных в примере 4.3 необходимо осуществить декодирования принятой кодовой последовательности $f(x) = \alpha x^6 + x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha^5 x + \alpha^6$ с помощью алгоритма Берлекэмпа-Месси.

1. Вычислить компоненты синдрома S_1, S_2, S_3, S_4.

Из примера 4.3 $S_1 = \alpha^4$; $S_2 = \alpha^5$; $S_3 = 0$; $S_4 = \alpha^6$.

Синдромный многочлен $S(x) = \alpha^4 + \alpha^5 x + \alpha^6 x^3$.

2. Установить начальные значения.

$\sigma(x) = 1$, $L = 0$, $k = 0$, $B(x) = 1$.

3. Поиск полинома $\sigma(x)$.

Шаг 1. $k = 1$.

$$d_1 = S_1 + \sum_{j=1}^{L=0} \sigma_{j-1} S_{1-j} = S_1 = \alpha^4$$

Проверка условия $d_1 = 0$:

$\alpha^4 = 0$? – нет → модификация многочлена обратной связи;

$\sigma^*(x) = \sigma(x) + d_1 \cdot x \cdot B(x) = 1 + \alpha^4 \cdot x$;

Проверка условия $2L \leq k - 1$:

$0 \leq 0$? – да → длина регистра увеличивается;

$B(x) = d_1^{-1} \cdot \sigma(x) = \alpha^3 \cdot 1 = \alpha^3$; {для $d_1 = \alpha^4$ обратный элемент $d_1^{-1} = \alpha^3$}

$L = k - L = 1 - 0 = 1$;

$\sigma(x) = \sigma^*(x) = 1 + \alpha^4 \cdot x$;

Проверка условия $k = 2t$:

1 = 4 ? – нет → увеличиваем шаг итерации;

$k = k+1 = 1 + 1 = 2$.

Шаг 2. $k = 2$.

$$d_2 = S_2 + \sum_{j=1}^{L=1} \sigma_{j-1} S_{2-j} = S_2 + \sigma_1 S_1 = \alpha^5 + \alpha^4 \cdot \alpha^4 = \alpha^5 + \alpha^8 = \alpha^5 + \alpha = \alpha^6 .$$

Проверка условия $d_2=0$:

$\alpha^6 = 0$? – нет → модификация многочлена обратной связи;

$\sigma^*(x) = \sigma(x)+d_2 \cdot x \cdot B(x) = 1+\alpha^4 \cdot x + \alpha^6 \cdot x \cdot \alpha^3 = 1+\alpha^4 x +\alpha^2 x = 1+\alpha x$.

Проверка условия $2L \le k-1$:

$2 \le 1$? – нет → длина регистра сдвига не увеличивается;

$\sigma(x) = \sigma^*(x) = 1+\alpha x$.

$B(x) = x \cdot B(x) = x \cdot \alpha^3 = \alpha^3 x$;

Проверка условия $k = 2t$:

2 = 4 ? – нет → увеличиваем шаг итерации;

$k = k+1 = 2 + 1 = 3$.

Шаг 3. $k = 3$.

$$d_3 = S_3 + \sum_{j=1}^{L=1} \sigma_{j-1} S_{3-j} = S_3 + \sigma_1 S_2 = 0 + \alpha \cdot \alpha^5 = \alpha^6 .$$

Проверка условия $d_3=0$:

$\alpha^6 = 0$? – нет → модификация многочлена обратной связи;

$\sigma^*(x) = \sigma(x)+d_3 \cdot x \cdot B(x) = (1+\alpha x)+\alpha^6 \cdot x \cdot \alpha^3 \cdot x = 1+\alpha x+\alpha^2 x^2$.

Проверка условия $2L \le k-1$:

$2 \le 2$? – да → длина регистра увеличивается;

$B(x) = d_3^{-1} \cdot \sigma(x) = \alpha \cdot (1+\alpha x) = \alpha + \alpha^2 x$.{для $d_3=\alpha^6$ обратный элемент $d_3^{-1} = \alpha^1$};

$L = k - L = 3 - 1 = 2$;

$\sigma(x) = \sigma^*(x) = 1+\alpha x+\alpha^2 x^2$;

Проверка условия $k = 2t$:

3 = 4 ? – нет → увеличиваем шаг итерации;

$k = k+1 = 3 + 1 = 4$.

Шаг 4. $k = 4$.

$$d_4 = S_4 + \sum_{j=1}^{L=2} \sigma_{j-1} S_{4-j} = S_4 + \sigma_1 S_3 + \sigma_2 S_2 = \alpha^6 + \alpha \cdot 0 + \alpha^2 \alpha^5 = \alpha^6 + 1 = \alpha^2 .$$

Проверка условия $d_4=0$:

$\alpha^4 = 0$? – нет → модификация многочлена обратной связи;

$\sigma^*(x) = \sigma(x)+d_4 \cdot x \cdot B(x)=(1+\alpha x+\alpha^2 x^2)+\alpha^2 x(\alpha+\alpha^2 x) = 1+\alpha x+\alpha^2 x^2+\alpha^3 x+\alpha^4 x^2=$

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

Проверка условия $2L \le k-1$:

$4 \le 3$? – нет → длина регистра сдвига не увеличивается;

$\sigma(x) = \sigma^*(x) = 1+x+\alpha x^2$;

$B(x) = x \cdot B(x) = x \cdot (\alpha+\alpha^2 x) = \alpha x+\alpha x^2$;

Проверка условия $k = 2t$:

$4 = 4$? – – да → последняя итерация, конец.

Таким образом, искомый полином локаторов ошибок $\sigma(x) = 1 + x + \alpha x^2$.

4. Нахождение позиций ошибок.

Корнями полинома $\sigma(x)$ являются элементы α, α^5 (пример 4.3).

Локаторы ошибок равны элементам, которые являются обратными корням многочлена ошибок:

– для элемента α^1 обратным является элемент α^6, так как из табл. 1.7 находим $\alpha^1 \cdot \alpha^6 = 1$;

– для элемента α^5 обратным является элемент α^2, так как из табл. 1.7 находим $\alpha^5 \cdot \alpha^2 = 1$.

Локаторами ошибок являются элементы $\underline{\alpha^6, \alpha^2.}$

Позиции ошибок – $j_1 = 6$, $j_2 = 2$.

5. Определение полинома значений ошибок $\Omega(x)$.

$S(x) \cdot \sigma(x) = (\alpha^4+\alpha^5 x+\alpha^6 x^3) \cdot (1+x+\alpha x^2) = \alpha^4+\alpha^4 x+\alpha^5 x^2+\alpha^5 x+\alpha^5 x^2+\alpha^6 x^3+\alpha^6 x^3+$

$+\alpha^6 x^4+\alpha^7 x^5 = \alpha^4+(\alpha^4+\alpha^5)x+(\alpha^5+\alpha^5)x^2+(\alpha^6+\alpha^6)x^3+\alpha^6 x^4+\alpha^7 x^5 =$

$=\alpha^4+x+0 \cdot x^2+0 \cdot x^3+ \alpha^6 x^4+x^5$.

$\alpha^4+x+\alpha^6 x^4+x^5 \bmod x^4 = x+\alpha^4$.

$\Omega(x) = x+\alpha^4$.

6. Вычисление значений ошибок e_1, e_2 и формирование многочлена ошибок $e(x)$.

Для вычисления значений ошибок найдем формальную производную $\sigma'(x)$ полинома локаторов ошибок $\sigma(x) = 1 + x + \alpha x^2$

$\sigma'(x) = 0 + 1 + 2\alpha x = 1$.

Вычислим значение коэффициента e_1 для локатора ошибки $\alpha^j = \alpha^6$ и для корня $\alpha^{-j} = \alpha^1$.

$$e_1 = (\alpha^6)^{1-0} \frac{\Omega(x = \alpha^1)}{\sigma'(x = \alpha^1)} = \alpha^6 \frac{\alpha^1 + \alpha^4}{1} = \alpha^6 \cdot \alpha^2 = \alpha^8 = \alpha.$$

Вычислим значение коэффициента e_1 для локатора ошибки $\alpha^j = \alpha^2$ и для корня $\alpha^{-j} = \alpha^5$.

$$e_2 = (\alpha^2)^{1-0} \frac{\Omega(x = \alpha^5)}{\sigma'(x = \alpha^5)} = \alpha^2 \frac{\alpha^5 + \alpha^4}{1} = \alpha^2 \cdot 1 = \alpha^2.$$

Многочлен ошибок $e(x) = \alpha x^6 + \alpha^2 x^2$.

7. Исправление ошибок.

Исправление ошибок в принятой последовательности $f(x)$ осуществляется путем сложения комбинации $f(x)$ и полинома ошибок $e(x)$

$$v(x) = f(x) + e(x) = (\alpha x^6 + x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha^5 x + \alpha^6) + (\alpha x^6 + \alpha^2 x^2) =$$

$$= x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 x + \alpha^6.$$

Таким образом, исправлены две ошибки и восстановленный кодовый полином равен $v(x) = x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 x + \alpha^6$

Все рассмотренные алгоритмы декодирования кодов РС дали одинаковые результаты, с точностью до константного множителя. При этом важно отметить, что алгоритм Евклида вычисляет одновременно и полином локаторов ошибок $\sigma(x)$ и многочлен значений ошибок $\Omega(x)$. При этом оба полинома имеют те же самые корни, что и полиномы, вычисленные алгоритмами Берлекэмпа-Месси и Питерсона. В результате значения ошибок совпадают.

Алгоритм Евклида в отличие от алгоритма Берлекэмпа-Месси использует все синдромы уже на первом шаге процедуры. Однако по числу операций в конечном поле алгоритм Берлекэмпа-Месси обычно эффективнее Евклида. Кроме того, три рассмотренных метода декодирования для (двоичных и недвоичных) кодов БЧХ являются неполными, так как исправляют ограниченное число ошибок – декодирование с ограниченным расстоянием. Многие из комбинаций ошибок большего веса не могут быть исправлены этими алгоритмами, но обнаруживаются и приводят к отказу от декодирования. Обнаружение неисправимых комбинаций ошибок происходит, либо в процедуре Ченя, когда хотя бы один из корней многочлена локаторов ошибок не принадлежит множеству локаторов позиций кодового слова, либо на этапе вычисления синдромов, когда количество нулевых синдромов оказывается больше или равным половине кодового расстояния.

4.7 Декодирование РС кодов с исправлением стираний и ошибок

Коды РС могут быть использованы в каналах, в которых кроме ошибок происходят и стирания. Существует много ситуаций, когда решение о принятом символе не может считаться надежным. Если значение принятого сигнала слишком близко к нулю, то декодер отказывается от решения о принятом символе и в таких случаях принятый символ «стирается», а такое решение называют стиранием. Тогда принятое по каналу слово состоит из входных символов канала (часть из них может быть с ошибками) и пробелов (или эквивалентных символов), обозначающих стирания. Декодер РС должен исправить ошибки и восстановить стертые символы.

Введение стираний, по сравнению с исправлением только ошибок, обладает преимуществом – декодеру известны позиции с ненадежными символами. Пусть d минимальное кодовое расстояние, t число ошибок и s число стираний в принятом слове. Тогда код РС с минимальным расстоянием d позволяет декодировать любую конфигурацию, содержащую t ошибок и s стираний

$$d \geq 2t + s + 1. \tag{4.25}$$

Последнее неравенство означает, что исправление ошибок требует вдвое больше усилий (и избыточности), чем исправление стираний, так как позиции стертых символов известны. Порождающий полином для РС кода, способного исправлять ошибки и стирания определяется

$$g(x) = \prod_{i=b}^{b+d-2} (x + \alpha^i), \tag{4.26}$$

где α^i – нули РС кода, b – первая степень элементов расширенного поля, нулей кода, d – минимальное кодовое расстояние кода.

Далее мы покажем, как модифицируется процедура декодирования для исправления ошибок и стираний в РС кодах. Мы будем предполагать, что стертый символ заменяется на известный заранее символ, например, на нулевой. Поэтому в случае ошибок ни положение, ни величины ошибок не известны, но в случае стираний их положение известно, но величины искажений, появившихся при замене стертых символов на нулевые значения, не известны.

Пусть $v(x)$ переданное слово и $e(x)$ полином искажений. Так как декодирование необходимо осуществить для кодов РС, то коэффициенты полиномов $v(x)$ и $e(x)$ – это элементы поля $GF(2^m)$. В случае t ошибок и s стираний полином $e(x)$ имеет вид

$$e(x) = e_1 x^{i_1} + ... + e_t x^{i_t} + \nu_1 x^{j_1} + ... + \nu_s x^{j_s}, \qquad (4.27)$$

где $i_1, i_2, \ldots, i_t$ – позиции ошибок, $j_1, j_2, \ldots, j_s$ – известные позиции стираний, $e_1, e_2, \ldots, e_t$ – величины ошибок на позициях $i_1, i_2, \ldots, i_t$, $\nu_1, \nu_2, \ldots, \nu_s$ – величины искажений на стертых позициях $j_1, j_2, \ldots, j_s$.

Для исправления ошибок и стираний в принятом кодовом слове в процессе декодирования необходимо вычислить:

– локаторы ошибок (α^i), степени которых являются значениями ошибочных позиций $i_1, i_2, \ldots, i_v$(v – количество ошибок в принятом кодовом слове);

– значения ошибок $e_1, e_2, \ldots, e_v$;

– локаторы стираний (α^j), степени которых являются значениями стертых позиций$j_1, j_2, \ldots, j_s$ (s – количество стертых позиций);

– значения стертых символов $\nu_1, \nu_2, \ldots, \nu_s$.

Для дальнейшего описания алгоритма декодирования кодов РС с исправлением ошибок и стираний введем понятия *многочлена локаторов стираний*

$$\upsilon(x) = \prod_{l=1}^{s} (1 + \alpha^{j_l} x), \qquad (4.28)$$

где α^{j_l} – локаторы стираний, $1 \le l \le s$.

Поскольку локаторы стираний известны до декодирования, то коэффициенты этого полинома $\upsilon_1, \upsilon_2, \ldots, \upsilon_s$могут быть вычислены к началу декодирования

$$\upsilon(x) = 1 + \upsilon_1 x + \upsilon_2 x^2 + ... + \upsilon_s x^s. \qquad (4.29)$$

Так как многочлен локаторов ошибок известен, можно скомбинировать его с многочленом локаторов стираний и дальнейшее декодирование проводить так же как и в случае наличия только ошибок.

Если $\sigma(x)$ – полином локаторов ошибок, а $\upsilon(x)$ – полином локаторов стираний, тогда *полином локаторов искажений* (ошибок и стираний)

$$\tau(x) = \sigma(x) \cdot \upsilon(x). \tag{4.30}$$

Корнями полинома $\tau(x)$ являются величины, обратные локаторам искажений, т.е. если k есть позиция ошибки или стирания, то $\tau(\alpha^k) = 0$.

Также в процессе декодирования необходимо вычислить значения стертых позиций $\nu_1, \nu_2, \ldots, \nu_s$, которые по аналогии с поиском значений ошибок находятся с помощью алгоритма Форни. Для этого введем понятие *полинома значений искажений* (ошибок и стираний), который определяется следующим образом

$$\omega(x) = S(x)\tau(x) \bmod x^{d-1} \tag{4.31}$$

или

$$\omega(x) = S(x)\sigma(x)\upsilon(x) \bmod x^{d-1}. \tag{4.32}$$

При этом полином локаторов искажений имеет степень $\deg[\tau(x)] = t + s$, а полином величин искажений имеет степень $\deg[\omega(x)] = t + s - 1$.

Для декодирования кодового слова с ошибками и стиранием необходимо сформировать *модифицированный синдромный многочлен*

$$T(x) = S(x)\upsilon(x) \bmod x^{d-1}. \tag{4.33}$$

Модифицированный синдромный многочлен состоит из $2t$ компонент синдрома T_j

$$T(x) = T_1 + T_2 x + T_3 x^2 + \ldots + T_{d-1} x^{d-1}. \tag{4.34}$$

Тогда с помощью *модифицированного алгоритма Форни* получаем значения ошибок e_{i_l}, которые в принятом кодовом слове находятся на позициях i_l $(1 \le l \le \nu)$ и значения стираний ν_{j_l}, которые в принятом кодовом слове находятся на позициях $j_1, j_2, \ldots, j_s$

$$e_{i_l} = (\alpha^{i_l})^{1-b} \frac{\omega(\alpha^{-i_l})}{\tau'(\alpha^{-i_l})}, \tag{4.35}$$

где $\tau'(x)$ − формальная производная по x полином локаторов искажений $\tau(x)$, $1 \le l \le v$, α^{-i} – корень полинома локаторов ошибок $\sigma(x)$, α^{i} – локатор ошибки.

Значения стираний находятся следующим образом

$$\nu_{j_l} = (\alpha^{j_l})^{1-b} \frac{\omega(\alpha^{-j_l})}{\tau'(\alpha^{-j_l})}, \tag{4.36}$$

где $\tau'(x)$ − формальная производная по x полином локаторов искажений $\tau(x)$, $1 \le l \le s$, α^{-j} – корень полинома локаторов стираний $\upsilon(x)$, α^{j} – локатор стирания.

Таким образом, в случае ошибок и стираний синдромный полином $S(x)$и полином локаторов стираний $\upsilon(x)$ известны, а полиномы локаторов ошибок $\sigma(x)$ и полином локаторов искажений $\omega(x)$ должны быть определены в процессе декодирования. Эту задачу можно решить с помощью любого метода декодирования РС кодов: алгоритм Питерсона-Горенстейна-Цирлера, алгоритма Берлекэмпа-Месси, алгоритма Евклида которые будут рассмотрены ниже.

4.7.1. Итеративный алгоритм Берлекэмпа-Месси

Вычисление многочлена локаторов ошибок по модифицированному значению синдрома можно выполнить с помощью алгоритма Берлекэмпа-Месси с учетом некоторых изменений. Идея модифицированного алгоритма состоит в рекуррентной процедуре вычисления многочлена $\sigma(x)$ за $2t$ итераций, начиная с оценок $\sigma(x)=1$, $B(x)=1$. В случае наличия стираний в уравнении(4.23) для невязки d_k компоненты синдрома S_j заменяются компонентами модифицированного синдрома T_j

$$d_k = \sum_{j=0}^{L} \sigma_j T_{k-j} \,. \tag{4.37}$$

Тогда многочлен локаторов ошибок вычисляется с помощью n итераций, при начальных условиях $\sigma(x) = 1$, $B(x) = 1$.

Однако процесс декодирования можно улучшить, если скомбинировать несколько шагов, тогда получим итеративный алгоритм

Берлекэмпа-Месси. Заменим начальные условия для декодирования и инициализацию осуществим с помощью полинома локаторов стираний $\upsilon(x)$

$$\sigma(x) = B(x) = \upsilon(x). \tag{4.38}$$

Невязку d_k будем вычислять по формуле (4.23)с помощью многочлена локаторов искажений $\tau(x)$ и синдромных компонент S_j

$$d_k = \sum_{i=0}^{L} \tau_i S_{k-i}\,. \tag{4.39}$$

При начальных условиях, даваемых многочленом $\upsilon(x)$, алгоритм Берлекэмпа-Месси рекуррентно порождает многочлен локаторов ошибок и стираний $\tau(x)$. Алгоритм вычислений для отыскания полинома локаторов искажений схематически показан на рис. 4.4.Процедура декодирования начинается с инициализации полинома локаторов искажений $\tau(x)$ начальным значением, равного известному полиному локаторов стираний $\upsilon(x)$ $\tau(x) = B(x) = \upsilon(x)$. А также заданием для длины фильтра и начального шага итерации значения, равного известному числу стираний s, т.е. $L = s$, $k = s+1$. На каждом шаге модификации оба полинома, исходный и модифицирующий делятся на $\upsilon(x)$. Поэтому найденный в результате полином $\tau(x)$ также будет делиться на $\upsilon(x)$. Алгоритм работает пока счетчик итераций k не превзойдет значения $2t$. При этом необходимо учитывать, что $\upsilon(x) = 1$ в отсутствии стираний, т.е. $s=0$. В конце декодирования степень найденного полинома и длина фильтра равна $s + t$, если $t \le (d-s-1)/2$. В этом случае процедура успешно завершается. Однако при большем числе ошибок может случится, что степень полинома $\tau(x)$ будет отличаться от длины последнего построенного фильтра. Это будет свидетельствовать о нарушении условия $t \le (d-s-1)/2$ и сопровождаться аварийным завершением алгоритма с сообщением о том, что фильтр не найден.

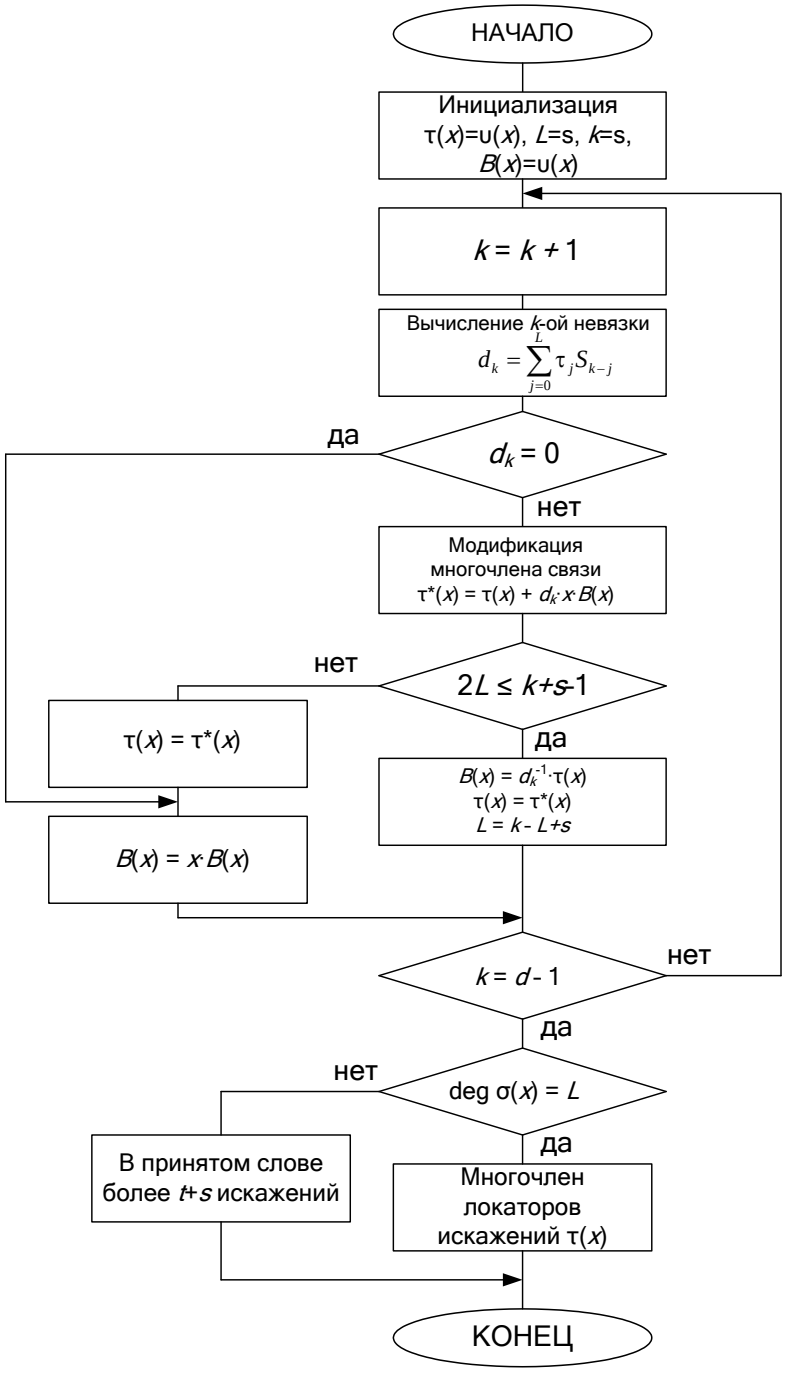

Рисунок 4.4 – Итеративный алгоритм Берлекэмпа-Месси для кодов РС, исправляющий ошибки и стирания

Пример 4.6. Пусть задан код РС с параметрами (7, 3, 5) над полем $GF(2^3)$ с нулями в точках (α^0, α^1, α^2, α^3), где примитивный элемент поля удовлетворяет неравенству $p(\alpha) = \alpha^3+\alpha+1$ (табл. 1.5). Код способен исправлять однократные ошибки $t = 1$ и двукратные стирания $s = 2$, так как минимальное кодовое расстояние кода равно $d = 5$ и соответственно

d=2t+s+1. Первая степень элементов расширенного поля $GF(2^3)$, которые используются для построения порождающего полинома РС кода равен нулю (b = 0). Таким образом, порождающий полином кода РС (7, 3, 5) в соответствии с формулой (4.26) равен

$$g(x) = \prod_{i=b}^{b+d-2}(x+\alpha^i) = \prod_{i=0}^{3}(x+\alpha^i) = (x+\alpha^0)(x+\alpha^1)(x+\alpha^2)(x+\alpha^3) =$$

$$= x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 x + \alpha^6.$$

Информационный полином, который необходимо закодировать с помощью заданного РС кода равен$u(x) = \alpha^4 x^2 + \alpha^2 x + \alpha$.

Кодовый полином$v(x)$ длины n = 7 равен

$$v(x) = g(x)\cdot u(x) = (x^4+\alpha^2x^3+\alpha^5x^2+\alpha^5x+\alpha^6)\cdot(\alpha^4x^2 + \alpha^2x + \alpha) = \alpha^4x^6+\alpha^2x^5+$$
$$+\alpha x^4+\alpha^6x^5+\alpha^4x^4+\alpha^3x^3+\alpha^9x^4+\alpha^7x^3+\alpha^6x^2+\alpha^9x^3+\alpha^7x^2+\alpha^6x+\alpha^{10}x^2+\alpha^8x+\alpha^7 =$$
$$= \alpha^4x^6+(\alpha^2+\alpha^6)x^5+(\alpha+\alpha^4+\alpha^2)x^4+(\alpha^3+1+\alpha^2)x^3+(\alpha^6+1+\alpha^3)x^2+(\alpha^6+\alpha)x+\alpha^7 =$$
$$= \alpha^4x^6+x^5+0\cdot x^4+\alpha^4x^3+\alpha^5x^2+\alpha^5x+\alpha^7 = \alpha^4x^6+x^5+\alpha^4x^3+\alpha^5x^2+\alpha^5x+1.$$

При передаче кодового слова $v(x)$ по каналу связи произошла ошибка на позиции 2 и ее значение равно α^3, а стирания – на позициях 0 и 5 со значениями α^0 и α^0. В результате замены стертых символов нулями, получим:

$$f(x) = \alpha^4x^6+\alpha^4x^3+\alpha^2x^2+\alpha^5x.$$

Многочлен искажений имеет вид $e(x) = \alpha^0 + \alpha^3x^2+x^5$. При этом данный полином декодеру неизвестен за исключением только позиций стираний. Этот полином используется в примере только для проверки правильности результата декодирования.

Так как стирания произошли на позициях 0 и 5, то соответственно локаторами стираний являются элементы α^0 и α^5. Полином локаторов стираний согласно формуле (4.28) равен

$$\upsilon(x) = (1+\alpha^0x)\cdot(1+\alpha^5x) = \alpha^5x^2+\alpha^4x+1.$$

Для декодирования принятого полинома найдем синдромные компонентыS_j для нулей кода α^0, α^1, α^2, α^3

$S_1 = f(\alpha^0) = \alpha^4\alpha^0+\alpha^4\alpha^0+\alpha^2\alpha^0+\alpha^5\alpha^0 = \alpha^4+\alpha^4+\alpha^2+\alpha^5 = \underline{\alpha^3}$;

$S_2 = f(\alpha^1) = \alpha^4\alpha^6+\alpha^4\alpha^3+\alpha^2\alpha^2+\alpha^5\alpha^1 = \alpha^{10}+\alpha^7+\alpha^4+\alpha^6 = \alpha^3+1+\alpha^4+\alpha^6 = \underline{1}$;

$S_3 = f(\alpha^2) = \alpha^4\alpha^{12}+\alpha^4\alpha^6+\alpha^2\alpha^4+\alpha^5\alpha^2 = \alpha^{16}+\alpha^{10}+\alpha^6+\alpha^7 = \alpha^2+\alpha^3+\alpha^6+1 = \underline{\alpha^3}$;

$S_4 = f(\alpha^3) = \alpha^4\alpha^{18}+\alpha^4\alpha^9+\alpha^2\alpha^6+\alpha^5\alpha^3 = \alpha^{22}+\alpha^{13}+\alpha^8+\alpha^8 = \alpha+\alpha^6+\alpha+\alpha = \underline{\alpha^5}.$

Синдромный полином равен $S(x) = \alpha^5x^3+\alpha^3x^2+x+\alpha^3$.

Для нахождения полинома искажений используем итеративный алгоритм Берлекэмпа-Месси.

Начальные установки.

$\tau(x) = \upsilon(x) = \alpha^5 x^2 + \alpha^4 x + 1.$

$L = s = 2,\ k = s = 2,$

$B(x) = \upsilon(x) = \alpha^5 x^2 + \alpha^4 x^2 + 1.$

$S_1 = \alpha^3,\ S_2 = 1,\ S_3 = \alpha^3,\ S_4 = \alpha^5.$

Шаг 1. $k = k+1 = 3$.

Вычисление невязки

$$d_3 = \sum_{i=0}^{2} \tau_j S_{3-j} = \tau_0 S_3 + \tau_1 S_2 + \tau_2 S_1 = \alpha^3 + \alpha^4 \cdot 1 + \alpha^5 \cdot \alpha^3 = \alpha^3 + \alpha^4 + \alpha = \alpha^5.$$

Проверка условия $d_3 = 0 \rightarrow \alpha^5 = 0$? нет.

Модификация многочлена $\tau(x)$.

$\tau^*(x) = \tau(x) + d_3 x\, B(x) = (\alpha^5 x^2 + \alpha^4 x^2 + 1) + \alpha^5 x(\alpha^5 x^2 + \alpha^4 x^2 + 1) = \alpha^5 x^2 + \alpha^4 x^2 + 1 +$
$+ \alpha^3 x^3 + \alpha^2 x^2 + \alpha^5 x = \alpha^3 x^3 + \alpha^3 x^2 + x + 1.$

Проверка условия $2L \le k+s-1 \rightarrow 4 \le 4$? да.

$B(x) = d_3^{-1} \tau(x) = \alpha^2(\alpha^5 x^2 + \alpha^4 x^2 + 1) = x^2 + \alpha^6 x + \alpha^2.$

$\tau(x) = \tau^*(x) = \alpha^3 x^3 + \alpha^3 x^2 + x + 1.$

$L = k - L + s = 3 - 2 + 2 = 3.$

Проверка условия $k = d - 1 \rightarrow 3 = 4$? нет.

Шаг 2. $k = k+1 = 4$.

Вычисление невязки

$$d_4 = \sum_{i=0}^{3} \tau_j S_{4-j} = \tau_0 S_4 + \tau_1 S_3 + \tau_2 S_2 + \tau_3 S_1 = \alpha^5 + 1 \cdot \alpha^3 + \alpha^3 \cdot 1 + \alpha^3 \cdot \alpha^3 =$$

$$= \alpha^5 + \alpha^6 = \alpha.$$

Проверка условия $d_4 = 0 \rightarrow \alpha = 0$? нет.

Модификация многочлена $\tau(x)$.

$\tau^*(x) = \tau(x) + d_4 x\, B(x) = (\alpha^3 x^3 + \alpha^3 x^2 + x + 1) + \alpha \cdot x\,(x^2 + \alpha^6 x + \alpha^2) = \alpha^3 x^3 + \alpha^3 x^2 + x +$
$+ 1 + \alpha x^3 + \alpha^7 x^2 + \alpha^3 x = x^3 + \alpha x^2 + \alpha x + 1.$

Проверка условия $2L \ge k+s-1 \rightarrow 6 \le 5$? нет.

$\tau(x) = \tau^*(x) = x^3 + \alpha x^2 + \alpha x + 1,$

$B(x) = x \cdot B(x) = x(x^2 + \alpha^6 x + \alpha^2) = x^3 + \alpha^6 x^2 + \alpha^2 x.$

Проверка условия $k = d - 1 \rightarrow 4 = 4$? да.

Алгоритм остановлен.

Таким образом, получен полином искажений (ошибок и стираний)

$$\tau(x) = x^3 + \alpha x^2 + \alpha x + 1.$$

Из определения (4.30) находим полином локаторов ошибок

$\sigma(x) = \tau(x)/\upsilon(x) = (x^3 + \alpha x^2 + \alpha x + 1)/(\alpha^5 x^2 + \alpha^4 x + 1) = \alpha^2 x + 1.$

С помощью процедуры Ченя найдем корень полинома локаторов ошибок $\sigma(x) - \alpha^5$

$\sigma(\alpha^5) = \alpha^2\alpha^5+1 = \alpha^7 + 1 = 0$.

Локатором ошибки является элемент обратный корню полинома $\sigma(x)$. Следовательно, локатором ошибки является элемент $\alpha^{-5} = \alpha^2$.

Корнями полинома локаторов стираний $\upsilon(x)$ являются элементы α^0 и α^5.

Для нахождения значений ошибок и стираний находим модифицированный многочлен значений искажений (4.31)

$\omega(x) = S(x)\tau(x) \bmod x^{d-1} = (\alpha^5x^3+\alpha^3x^2+x+\alpha^3)\cdot(x^3+\alpha x^2+\alpha x+1) \bmod x^4 =$

$= \alpha^5x^6+\alpha^4x^5+\alpha x^4+\alpha^5x^2+\alpha^5x+\alpha^3 \bmod x^4 = \alpha^5x^2+\alpha^5x+\alpha^3$.

Найдем формальную производную многочлена локаторов искажений $\tau'(x) = x^2 + \alpha$.

После выполнения алгоритма Берлекэмпа-Месси и процедуры Ченя декодеру известны локаторы искажений и многочлен ошибок в виде

$$e(x) = \nu_0+e_2x^2+\nu_5x^5.$$

Значения ошибки(e_2) и значения стираний (ν_0, ν_5) вычисляем с помощью модифицированного алгоритма Форни (4.35) и (4.36) соответственно, где

– $\alpha^i = \alpha^2$ – локатор ошибки,

– $\alpha^{-i} = \alpha^5$ – корень полинома ошибок $\sigma(x)$,

– $\alpha^{j_1} = \alpha^0$ – локатор стираний,

– $\alpha^{-j_1} = \alpha^0$ – корень полинома стираний $\upsilon(x)$,

– $\alpha^{j_2} = \alpha^5$ – локатор стираний,

– $\alpha^{-j_2} = \alpha^2$ – корень полинома стираний $\upsilon(x)$.

Значение ошибки

$$e_2 = (\alpha^2)\frac{\omega(\alpha^5)}{\tau'(\alpha^5)} = \alpha^2\frac{\alpha^5(\alpha^5)^2 + \alpha^5\alpha^5 + \alpha^3}{(\alpha^5)^2 + \alpha} = \alpha^2\frac{\alpha^{15} + \alpha^{10} + \alpha^3}{\alpha^3 + \alpha} =$$

$$\alpha^2\frac{\alpha + \alpha^3 + \alpha^3}{1} = \alpha^2 \cdot \alpha = \alpha^3.$$

Значения стираний

$$\nu_0 = (\alpha^0)\frac{\omega(\alpha^0)}{\tau'(\alpha^0)} = 1\frac{\alpha^5+\alpha^5+\alpha^3}{1+\alpha} = \frac{\alpha^3}{\alpha^3} = 1.$$

$$\nu_5 = (\alpha^5)\frac{\omega(\alpha^2)}{\tau'(\alpha^2)} = \alpha^5\frac{\alpha^5(\alpha^2)^2+\alpha^5\alpha^2+\alpha^3}{(\alpha^2)^2+\alpha} = \alpha^5\frac{\alpha^5\alpha^4+\alpha^7+\alpha^3}{\alpha^4+\alpha} =$$

$$\alpha^5\frac{\alpha^9+1+\alpha^3}{\alpha^2} = \alpha^5\cdot\alpha^{-2}\cdot(\alpha^2+\alpha) = \alpha^5\cdot\alpha^5\cdot\alpha^4 = \alpha^{14} = 1.$$

Таким образом, многочлен ошибок и стираний равен $e(x) = \alpha^0 + \alpha^3x^2+x^5$.

Результат декодирования совпадает с переданным кодовым словом

$v(x) = f(x) + e(x) = (\alpha^4x^6+\alpha^4x^3+\alpha^2x^2+\alpha^5x)+(\alpha^0 + \alpha^3x^2+x^5) =$

$= \alpha^4x^6+x^5+\alpha^4x^3+\alpha^5x^2+\alpha^5x+1.$

Исправлены одна ошибка и два стирания.

4.7.2. Алгоритм Евклида для вычисления значений ошибок и стираний

Алгоритм Евклида также может быть применен для исправления ошибок и стирания в кодовой последовательности Рида-Соломона, путем вычисления полинома локаторов ошибок $\sigma(x)$ и полинома значений искажений $\omega(x)$. Для этого в алгоритме Евклида, который используется для исправления ошибок, при инициализации и задания начальных значений синдромный многочлен $S(x)$ заменяется на модифицированный синдромный многочлен $T(x)$.

При использовании модифицированного синдромного полинома $T(x)$ и при выборе $\Omega_{-1}(x) = x^{d-1}$ на одном из шагов алгоритма Евклида будет найдено решение уравнения

$$\omega(x) = \sigma(x)\cdot T(x) \bmod x^{d-1}. \qquad (4.40)$$

и выполнены условия $\deg[\sigma(x)] = t$, $\deg[\omega(x)] = t+s-1$ и следовательно, будут найдены полином локаторов ошибок $\sigma(x)$ и полином величин искажений $\omega(x)$.

Начальными условиями алгоритма Евклида для вычисления полинома локаторов ошибок и искажений являются

$$\Omega_{-1}(x) = x^{d-1},\ \Omega_0(x) = T(x).$$

Далее работа алгоритма осуществляется по стандартной блок-схеме, представленной на рис. 4.3. до блока нахождения значений ошибок

(алгоритм Форни). При этом алгоритм останавливается на i-ом шаге, если выполняется условие

$$\deg[\Omega_i(x)] < \frac{d+s-1}{2}. \qquad (4.41)$$

Тогда на на i-ом шаге алгоритма выдается решение

$$\omega(x) = \Omega_i(x) \text{ и } \sigma(x) = \sigma_i(x).$$

Далее используя найденные полином локаторов ошибок $\sigma(x)$ и полином величин искажений $\omega(x)$ с помощью модифицированного алгоритма Форни находятся значения ошибок (4.35) и значения стираний (4.36).

Сложность решения ключевого уравнения с помощью алгоритма Евклида на превышает ct^2, где c – некоторая константа, t – число исправляемых ошибок.

Пример 4.7. Для условий примера 4.6 необходимо исправить ошибки и стирания в принятом кодовом слове $f(x) = \alpha^4x^6+\alpha^4x^3+\alpha^2x^2+\alpha^5x$ используя алгоритм Евклида.

Из условий пример 4.6 известно

− синдромный полином $S(x) = \alpha^5x^3+\alpha^3x^2+x+\alpha^3$,

− полином локаторов стираний $\upsilon(x) = \alpha^5x^2+\alpha^4x+1$.

Найдем модифицированный синдромный многочлен $T(x)$используя формулу (4.33)

$S(x)\cdot\upsilon(x) = (\alpha^5x^3+\alpha^3x^2+x+\alpha^3)\cdot(\alpha^5x^2+\alpha^4x+1) = \alpha^3x^5+\alpha x^4+\alpha^5x^3+\alpha x^2+\alpha^2x^4+x^3+$
$+\alpha^4x^2+x+\alpha^5x^3+\alpha x^2+x+\alpha^3 = \alpha^3x^5+\alpha^4x^4+x^3+\alpha^5x^2+\alpha^3$.

$S(x)\cdot\upsilon(x) \bmod x^4 = (\alpha^3x^5+\alpha^4x^4+x^3+\alpha^5x^2+\alpha^3) \bmod x^4 = x^3+\alpha^5x^2+\alpha^3$.

Модифицированный синдромный многочлен $T(x) = x^3+\alpha^5x^2+\alpha^3$.

Для нахождения полинома искажений и локаторов ошибок используем алгоритм Евклида.

Начальные установки.

$\sigma_{-1}(x) = 0$, $\sigma_0(x) = 1$.

$\Omega_{-1}(x) = x^4$, $\Omega_0(x) = \mathrm{T(x)} = x^3+\alpha^5x^2+\alpha^3$.

Проверка условия $\deg[\Omega_0(x)]<(d+s-1)/2$,

$\deg[\Omega_0(x)]=3$ и $(d+s-1)/2 = 3 \rightarrow 3 < 3$? нет.

Итерация 1.

$$q_1(x) = \left[\frac{\Omega_{-1}(x)}{\Omega_0(x)}\right] = \left[\frac{x^4}{x^3 + \alpha^5 x^2 + \alpha^3}\right] = x + \alpha^5.$$

Вычислить значения $\sigma_1(x)$, $\Omega_1(x)$.

$$\sigma_1(x) = \sigma_{-1}(x) + q_1 \cdot \sigma_0(x) = 0 + (x + \alpha^5) \cdot 1 = x + \alpha^5,$$

$$\Omega_1(x) = \Omega_{-1}(x) + q_1 \cdot \Omega_0(x) = x^4 + (x + \alpha^5) \cdot (x^3 + \alpha^5 x^2 + \alpha^3) =$$

$$x^4 + x^4 + \alpha^5 x^3 + \alpha^3 x + \alpha^5 x^3 + \alpha^3 x^2 + \alpha = \alpha^3 x^2 + \alpha^3 x + \alpha.$$

Проверка условия $\deg[\Omega_1(x)] < (d+s-1)/2 \rightarrow 2 < 3$? да.

Так как $\deg[\Omega_1(x)] < 3$, алгоритм останавливается.

Таким образом, найдены:

− полином локаторов ошибок $\sigma(x) = \sigma_1(x) = x + \alpha^5$,

− полином искажений $\omega(x) = \Omega_1(x) = \alpha^3 x^2 + \alpha^3 x + \alpha$.

Полученные полиномы $\sigma(x)$ и $\omega(x)$ отличаются от полиномов, найденных алгоритмом Берлекэмпа-Месси так как имеют константные множители

$\sigma(x) = x + \alpha^5 = \alpha^5(\alpha^2 x + 1)$, $\omega(x) = \alpha^3 x^2 + \alpha^3 x + \alpha = \alpha^5(\alpha^5 x^2 + \alpha^5 x + \alpha^3)$.

При этом очевидно, что многочлены $\sigma(x)$ и $\omega(x)$ имеют те же самые корни, что и многочлены, найденные алгоритмом Берлекэмпа-Месси.

Заметим, что оба полинома, найденные алгоритмом Евклида, имеют одинаковый константный множитель α^5, т.е. алгоритм находит $\beta\sigma(x)$ и $\beta\omega(x)$ с некоторым множителем $\beta \in GF(2^m)$. Тем не менее, оба полинома имеют те же самые корни, что и полиномы, вычисленные алгоритмом Берлекэмпа-Месси или Питерсона-Горенштейна-Цирлера.

Находим полином локаторов искажений $\tau(x)$ с помощью формулы (4.30)

$\tau(x) = \sigma(x) \cdot \upsilon(x) = (x + \alpha^5)(\alpha^5 x^2 + \alpha^4 x + 1) = \alpha^5 x^3 + \alpha^3 x^2 + \alpha^4 x^2 + \alpha^2 x + x + \alpha^5 =$

$= \alpha^5 x^3 + \alpha^6 x^2 + \alpha^6 x + \alpha^5$.

Корень полинома локаторов ошибок $\sigma(x) - \alpha^5 \rightarrow \sigma(\alpha^5) = \alpha^5 + \alpha^5 = 0$.

Локатор ошибки $\alpha^{-5} = \alpha^2$.

Корни полинома локаторов стираний $\upsilon(x)$ являются элементы α^0 и α^5.

Найдем формальную производную многочлена локаторов искажений

$\tau'(x) = \alpha^5 x^2 + \alpha^6$.

Значение ошибки

$$e_2 = (\alpha^2)\frac{\omega(\alpha^5)}{\tau'(\alpha^5)} = \alpha^2\frac{\alpha^3(\alpha^5)^2 + \alpha^3\alpha^5 + \alpha}{\alpha^5(\alpha^5)^2 + \alpha^6} = \alpha^2\frac{\alpha^{13} + \alpha^8 + \alpha}{\alpha^{15} + \alpha^6} = \alpha^2\frac{\alpha^6 + \alpha + \alpha}{\alpha + \alpha^6} =$$

$$= \alpha^2 \cdot \alpha^{-5} \cdot \alpha^6 = \alpha^2 \cdot \alpha^2 \cdot \alpha^6 = \alpha^{10} = \alpha^3.$$

Значения стираний

$$\nu_0 = (\alpha^0)\frac{\omega(\alpha^0)}{\tau'(\alpha^0)} = 1\frac{\alpha^3 + \alpha^3 + \alpha}{\alpha^5 + \alpha^6} = \frac{\alpha}{\alpha} = 1.$$

$$\nu_5 = (\alpha^5)\frac{\omega(\alpha^2)}{\tau'(\alpha^2)} = \alpha^5\frac{\alpha^3(\alpha^2)^2 + \alpha^3\alpha^2 + \alpha}{\alpha^5(\alpha^2)^2 + \alpha^6} = \alpha^5\frac{\alpha^3\alpha^4 + \alpha^5 + \alpha}{\alpha^9 + \alpha^6} =$$

$$\alpha^5\frac{\alpha^7 + \alpha^6}{\alpha^2 + \alpha^6} = \alpha^5\frac{\alpha^2}{1} = \alpha^7 = 1.$$

Полученные значения ошибок и стираний совпадают с теми значениями, которые вычислены в примере 4.6. с помощью алгоритма Берлекэмпа-Месси.

Таким образом, искомый многочлен искажений равен $e(x) = \alpha^0 + \alpha^3 x^2 + x^5$.

4.7.3. Алгоритм Питерсона-Горенштейна-Цирлерадля вычисления значений ошибок и стираний

Также для кодов РС значения ошибок и стираний можно получить с помощью алгоритма Питерсона-Горенштейна-Цирлера (ПГЦ). Пусть $e(x)$ полином ошибок, ассоциированный с комбинацией t ошибок и s стираний

$$e(x) = \sum_{l=1}^{t} e_{j_l} x^{j_l} + \sum_{m=1}^{s} \nu_{j_m} y^{j_m} . \qquad (4.42)$$

Тогда справедлива следующая система линейных уравнений, связывающая синдромы и значения искажений

$$S_j = e(\alpha^{b+i}) = \sum_{l=1}^{t} e_{j_l} \alpha^{(b+i)_{j_l}} + \sum_{m=1}^{s} \nu_{j_m} \alpha^{(b+1)_{j_m}} , \qquad (4.43)$$

где $j = 1, 2, \ldots d-1$.

Обозначим локатор стирания как α^j и введем в рассмотрение модифицированный полином локаторов стираний

$$P(x) = \prod_{j=1}^{s} (x - \alpha^j) = \sum_{i=0}^{s} P_i x^i . \qquad (4.44)$$

Так как локаторы стираний α^j известны декодеру до декодирования, то полином $P(x)$ может быть вычислен к началу декодирования.

Предположим, что принятое слово содержит ошибки и стирания и обозначим через $T'(x)$модифицированный синдромный полином

$$T'(x) = \sum_{j=1}^{\text{d-s-1}} T_j x^{j-1} . \tag{4.45}$$

Тогда полином значений искажений $\omega(x)$ равен

$$\omega(x) = T'(x)\sigma(x) \bmod x^{d-s-1}. \tag{4.47}$$

Модифицированные синдромные компоненты T_j (j = 1, 2, …, d−s−1), которые используются в декодере ПГЦ, могут быть вычислены следующим образом

$$T_j = \sum_{i=0}^{s} P_i S_{i+j} . \tag{4.48}$$

При этом модифицированные синдромные компоненты связаны с модифицированными значениями ошибок E_1, E_2, …, E_v и локаторами ошибок следующим образом

$$T_j = \sum_{l=1}^{v} E_l \alpha^{i}, \quad E_l = e_i P(\alpha^{i}) . \tag{4.49}$$

Теперь задача декодирования заключается в том, чтобы, зная модифицированный синдром $T = (T_1, T_2, \ldots, T_{d\text{-}s\text{-}1})$, определить число ошибок v, локаторы ошибок α^i и их модифицированные величины $E_1, E_2, \ldots, E_v$. Эта задача решается с помощью алгоритма ПГЦ, в результате работы которого будут получены локаторы ошибок α^i.

Модифицированные значения ошибок можно вычислить с помощью формальной производной полинома ошибок $\sigma'(x)$ и полинома значений искажений $\omega(x)$

$$E_l = (\alpha^{i})^{1-b} \frac{\omega(\alpha^{-i})}{\sigma'(\alpha^{-i})} . \tag{4.50}$$

Для нахождения действительных значений ошибок e_i на позициях $i_1, i_2, \ldots, i_v$ можно воспользоваться соотношением (4.49)

$$e_i = \frac{E_l}{P(\alpha^i)},\ l = 1, 2, ..., v. \qquad (4.51)$$

После определения позиций и значений ошибок необходимо вычислить значения стертых символов $\nu_1, \nu_2, \ldots, \nu_s$ на позициях $j_1, j_2, \ldots, j_s$. Для этого определим следующие полиномы:

– *синдромный полином* $R(x) = \sum_{j=1}^{d-s-1} R_j x^{j-1}$, компоненты которого можно найти из равенства

$$R_j = S_j + \sum_{i=1}^{t} e_i \alpha^{b+j-1}. \qquad (4.52)$$

– *полином локаторов стираний* $\upsilon(x) = \prod_{l=1}^{s} (1 + x\alpha^{j_l})$.

– *полином значений стираний* $\Theta(x)$, который может быть найден с помощью полинома локаторов стираний

$$\Theta(x) = R(x)\upsilon(x) \bmod x^{d-s-1}. \qquad (4.53)$$

Тогда значения стертых символов $\nu_1, \nu_2, \ldots, \nu_s$ на позициях $j_1, j_2, \ldots, j_s$ определяются следующим образом

$$\nu_{j_l} = (\alpha^{j_l})^{1-b} \frac{\Theta(\alpha^{-j_l})}{\upsilon'(\alpha^{-j_l})}. \qquad (4.54)$$

Как было показано выше, вычисление ошибок и стираний происходит после того, как определен полином локаторов ошибок на основе знания модифицированного синдрома. Если стираний нет, т.е. $s = 0$, то модифицированные и обычные величины ошибок совпадают и на этом декодирование заканчивается. Если же стирания присутствуют, то вычисления нужно продолжить с тем, чтобы определить величины стираний $\nu_1, \nu_2, \ldots, \nu_s$.

Алгоритм исправления ошибок и стираний в принятом кодовом слове с помощью процедуры Питерсона-Горенштейна-Цирлера состоит в следующем.

1. Вычислить синдромные компоненты

$$S_j = f(\alpha^j),\ j = 1, \ldots, d-1;\ i = b, \ldots, b+d-1.$$

2. Для локаторов стираний α^j вычислить в соответствии с (4.44) полином локаторов стираний $P(x)$.

3. Найти модифицированные компоненты синдрома T_j ($j = 1, 2, \ldots, d-s-1$) для синдромного полинома $T'(x)$ по формуле (4.48).

4. Предположить $i = \nu_{\max} = t$ и построить матрицу для синдромных компонент T_j для нахождения коэффициентов многочлена локаторов ошибок (3.22).

5. Вычислить определитель матрицы M_i. Если он равен нулю, уменьшаем i на единицу и возвращаемся к шагу 4, иначе пункт 6.

6. Установить количество ошибок $\nu = i$ и вычисляем коэффициенты многочлена $\sigma_1, \sigma_2, \ldots \sigma_\nu$ с помощью обращения матрицы M_i.

7. Вычислить корни многочлена $\sigma(x)$, локаторы ошибок α^i и соответственно позиции ошибок $i_1, i_2, \ldots, i_\nu$.

8. Найти полином значений искажений $\omega(x)$ по формуле (4.47).

9. Вычислить значения модифицированных ошибок E_l (4.50).

10. Найти действительные значения ошибок e_i (4.51).

11. Для вычисления искажения на стертых позициях определяются

− синдромный полином $R(x)$, компоненты которого можно найти из равенства (4.52);

− полином локаторов стираний $\upsilon(x)$ по формуле (4.28);

− полином значений стираний $\Theta(x)$, который может быть найден с помощью полинома локаторов стираний (4.53).

12. Вычислить значения стертых символов ν_{j_l} на позициях $j_1, j_2, \ldots, j_s$ используя формулу (4.54).

13. Построить многочлен искажений $e(x)$(4.27) используя найденные позиции ошибок $i_1, i_2, \ldots, i_\nu$ и позиции стираний $j_1, j_2, \ldots, j_s$, а также значения ошибок e_i и значения стираний ν_j.

Пример 4.8. Для условий примера 4.6 необходимо исправить ошибки и стирания в принятом кодовом слове $f(x) = \alpha^4x^6+\alpha^4x^3+\alpha^2x^2+\alpha^5x$ используя алгоритм Питерсона-Горенштейна-Цирлера.

Из условий пример 4.6 известно

− синдромный полином $S(x) = \alpha^5x^3+\alpha^3x^2+x+\alpha^3$,

− полином локаторов стираний $\upsilon(x) = \alpha^5x^2+\alpha^4x+1$,

– локаторы стираний α^0, α^5.

1. Вычислим компоненты синдрома

$S_1 = \alpha^3$, $S_2 = 1$, $S_3 = \alpha^3$, $S_4 = \alpha^5$.

2. Вычислим полином локаторов стираний

$P(x) = (x+\alpha^0)(x+\alpha^5) = x^2+\alpha^5 x+\alpha^0 x+\alpha^5 = x^2+\alpha^4 x+\alpha^5$.

3. Найдем модифицированные компоненты синдрома

$$T_1 = \sum_{i=0}^{s=2} P_i S_{i+1} = P_0 S_1 + P_1 S_2 + P_2 S_3 = \alpha^5\alpha^3 + \alpha^4\alpha^0 + \alpha^0\alpha^3 = \alpha + \alpha^4 + \alpha^3 = \alpha^5.$$

$$T_2 = \sum_{i=0}^{s=2} P_i S_{i+2} = P_0 S_2 + P_1 S_3 + P_2 S_4 = \alpha^5\alpha^0 + \alpha^4\alpha^3 + \alpha^0\alpha^5 = \alpha^5 + \alpha^0 + \alpha^5 = \alpha^0 = 1.$$

Модифицированный синдромный полином равен

$$T'(x) = T_1 + T_2 x = \alpha^5 + x.$$

4. Полагаем $i = v_{\max} = 1$, тогда $M_1=[T_1]$.

5. Определитель матрицы $\det M_1 = T_1 \neq 0$. Следовательно, число ошибок равно $v=1$.

6. Находим коэффициент σ_1 полинома локаторов ошибок. Для этого решается уравнение

$T_1 \sigma_1 = T_2$, откуда $\sigma_1 = T_2/T_1 = 1/\alpha^5 = 1\cdot\alpha^{-5} = 1\cdot\alpha^2 = \alpha^2$.

Многочлен локаторов ошибок равен $\sigma(x) = 1+\alpha^2 x$.

7. Корнем полинома является элемент α^5

$\sigma(\alpha^5) = 1+\alpha^2\alpha^5 = 1+\alpha^7 = 1+1 = 0$.

Локатор ошибки это величина, обратная корню $\alpha^i = \alpha^{-5} = \alpha^2$.

Таким образом, определена позиция ошибки – $i = 2$.

8. Вычислим полином значений искажений $\omega(x)$

$\omega(x) = T'(x)\sigma(x) \bmod x^{d-s-1} = (\alpha^5+x)(1+\alpha^2 x) \bmod x^2 =$

$= (\alpha^5+\alpha^7 x+x+\alpha^2 x^2) \bmod x^2 = (\alpha^2 x^2 + \alpha^5) \bmod x^2 = \alpha^5$.

$$\omega(x) = \alpha^5.$$

9. Вычислим значение модифицированной ошибки E_1, где

$\alpha^i = \alpha^2$, $\alpha^{-i} = \alpha^5$, $b = 0$, $\sigma'(x) = \alpha^2$;

$$E_1 = (\alpha^2)^{1-0}\frac{\omega(\alpha^5)}{\sigma'(\alpha^5)} = \alpha^2\frac{\alpha^5}{\alpha^2} = \alpha^5.$$

10. Вычислим значение действительной ошибки e_1

$$e_1 = \frac{E_1}{P(\alpha^2)} = \frac{\alpha^5}{(\alpha^2)^2+\alpha^4\alpha^2+\alpha^5} = \frac{\alpha^5}{\alpha^4+\alpha^6+\alpha^5} = \frac{\alpha^5}{\alpha^2} = \alpha^5\cdot\alpha^5 = \alpha^3.$$

11. Для вычисления искажения на стертых позициях определим

– синдромный полином $R(x)$, для этого вычислим компоненты синдрома R_1и R_2

$$R_1 = S_1 + \sum_{i=1}^{1} e_i \alpha^{b+j-1} = S_1 + e_1 \cdot \alpha^0 = \alpha^3 + \alpha^3 \cdot 1 = 0,$$

$$R_2 = S_2 + \sum_{i=1}^{1} e_i \alpha^{b+j-1} = S_2 + e_1 \cdot \alpha^1 = 1 + \alpha^3 \cdot \alpha^1 = 1 + \alpha^2 = \alpha^4,$$

$$R(x) = R_2 x + R_1 = \alpha^6 x + 0 = \alpha^6 x,$$

– полином локаторов стираний $\upsilon(x) = \alpha^5 x^2 + \alpha^4 x + 1$,

– полином значений стираний $\Theta(x) = R(x)\upsilon(x) \bmod x^{d-s-1}$

$\Theta(x) = \alpha^4 x\,(\alpha^5 x^2 + \alpha^4 x + 1) \bmod x^2 = \alpha^9 x^3 + \alpha^8 x^2 + \alpha^4 x \bmod x^2 =$

$= \alpha^2 x^3 + \alpha^1 x^2 + \alpha^4 x \bmod x^2 = \alpha^4 x.$

12. Вычислим значения стертых символов ν_1 и ν_2 на позициях $j_1 = 0$ и $j_2 = 5$, т.к. локаторы ошибок α^0 и α^5.

$\upsilon'(x) = \alpha^4$, $\alpha^{-0} = \alpha^0$, $\alpha^{-5} = \alpha^2$,

$$\nu_1 = (\alpha^0)^{1-0} \frac{\Theta(\alpha^0)}{\upsilon'(\alpha^0)} = 1 \frac{\alpha^4 \cdot \alpha^0}{\alpha^4} = \alpha^4 \cdot \alpha^{-4} = \alpha^4 \cdot \alpha^3 = \alpha^7 = 1,$$

$$\nu_2 = (\alpha^5)^{1-0} \frac{\Theta(\alpha^2)}{\upsilon'(\alpha^2)} = \alpha^5 \frac{\alpha^4 \cdot \alpha^2}{\alpha^4} = \alpha^5 \cdot \alpha^6 \cdot \alpha^{-4} = \alpha^{11} \cdot \alpha^{-4} = \alpha^7 = 1.$$

Таким образом, мы получили многочлен искажений

$e(x) = \nu_0 + e_2 x^2 + \nu_5 x^5 = \alpha^0 + \alpha^3 x^2 + x^5$.

Результат декодирования совпадает с исходным кодовым словом

$v(x) = f(x) + e(x) = (\alpha^4 x^6 + \alpha^4 x^3 + \alpha^2 x^2 + \alpha^5 x) + (\alpha^0 + \alpha^3 x^2 + x^5) =$

$= \alpha^4 x^6 + x^5 + \alpha^4 x^3 + \alpha^5 x^2 + \alpha^5 x + 1.$

4.8 Характеристики кодов Рида-Соломона

Любой линейный код дает возможность исправлять n-k комбинаций символьных стираний, если все $n-k$ стертых символов приходятся на контрольные символы. Однако коды РС имеют замечательное свойств, выражающееся в том, что они могут исправить любой набор $n-k$ символов стираний в блоке. Можно сконструировать коды с любой избыточностью. Но с увеличением избыточности растет сложность ее высокоскоростной реализации. Поэтому наиболее привлекательные коды Рида-Соломона обладают высокой степенью кодирования (низкой избыточностью).

Коды РС чрезвычайно эффективны для исправления пакетов ошибок, т.е. они оказываются эффективными в каналах с памятью. Также они хорошо зарекомендовали себя в каналах с большим набором входных символов. Особенностью кода РС является то, что к коду длины n можно добавить два информационных символа, не уменьшая при этом минимального расстояния. Такой расширенный код имеет длину n+2 и то же количество символов контроля четности, что и исходный код. Вероятность появления ошибки в декодированном символе, можно записать через вероятность появления ошибки в канале

$$P_E = \frac{1}{2^m - 1} \sum j \binom{2^m - 1}{j} p^j (1-p)^{2^m - 1 - j}, \qquad (4.55)$$

где t – количество ошибочных битов в символе, которые может исправить код, а символы содержат m битов каждый.

Для кодов РС вероятность появления ошибок является убывающей функцией длины блока n, а сложность декодирования пропорциональна небольшой степени длины блока. Иногда коды Рида-Соломона применяются в каскадных схемах. В таких системах внутренний сверточный декодер сначала осуществляет некоторую защиту от ошибок за счет мягкой схемы решений на выходе демодулятора, затем сверточный декодер передает данные, оформленные согласно жесткой схеме.

Для того чтобы код успешно противостоял шумовым эффектам, длительность помех должна составлять относительно небольшой процент от количества кодовых слов. Чтобы быть уверенным, что так будет большую часть времени, принятый шум необходимо усреднить за большой промежуток времени, что снизит эффект от неожиданной или необычной полосы плохого приема. Таким образом, код с коррекцией ошибок будет более эффективнее при увеличении размера передаваемого блока, что делает код РС более привлекательным, если желательна большая длина блока. Это можно оценить по семейству кривых, показанному на рис. 4.5 [], где степень кодирования взята равной 7/8, при этом длина блока возрастает с n = 32 символов до n = 256 символов. Таким образом, размер блока возрастает с 160 бит до 2048 бит.

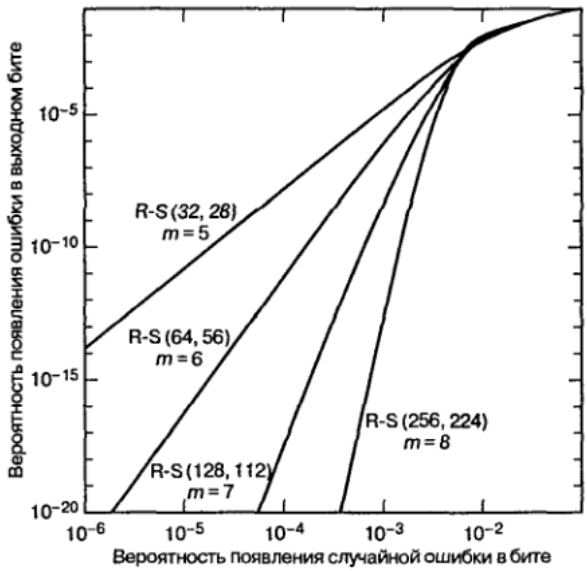

Рисунок 4.5 – Характеристики РС кода как функция размера символов

По мере увеличения избыточности кода, сложность реализации этого кода повышается и при этом для систем связи реального времени должна увеличиваться ширина полосы пропускания. Увеличение избыточности, например увеличение размера символа, приводит к уменьшению вероятности появления битовых ошибок, как можно видеть на рис. 4.6, где кодовая длина равна постоянному значению 64 при снижении числа символов данных с $k = 60$ до $k = 4$ (избыточность возрастает с 4 до 60 символов).

На рис. 4.6 показана передаточная функция - выходная вероятность появления битовой ошибки, зависящей от входной вероятности символьной ошибки. Надежность передачи является монотонной функцией избыточности и будет неуклонно возрастать с приближением степени кодирования к нулю. Однако это не так, если отношение E_b/N_0 фиксировано. Для гауссова канала оптимальное значение степени кодирования находится между 0,6 и 0,7 для канала с райсовским замиранием – около 0,5 и 0, 3 – для канала с релеевским замиранием.

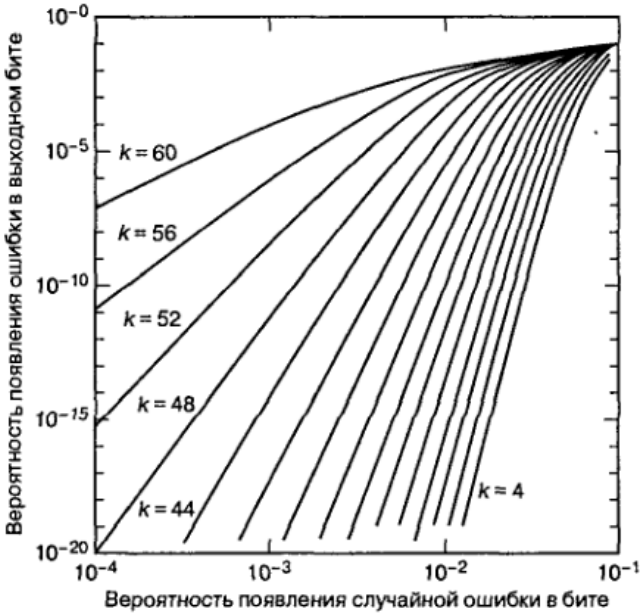

Рисунок 4.6 – Характеристики кода РС как функции избыточности

Любой код в целом обеспечивает все преимущества кодирования, следовательно, как только степень кодирования приближается к единицы (нет кодирования), система проигрывает в надежности передачи. Ухудшение характеристик при низких степенях кодирования является более тонким вопросом, т.к. когда E_b/N_0 фиксировано работает два механизма. Один механизм направлен на снижение вероятности появления ошибок, другой повышает ее. Механизм, снижающий вероятность появления ошибки – это кодирование, чем больше избыточность, тем больше возможности кода в коррекции ошибок. Механизм, повышающий эту вероятность – это снижение энергии, приходящейся на канальный символ, уменьшенная энергия канального символа демодулятор совершать ошибки. В конечном счете, второй механизм подавляет первый, поэтому очень низкие степени кодирования вызывает ухудшение характеристики кода.

Контрольные вопросы

1. Дайте определение кода Рида-Соломона.
2. Почему коды Рида-Соломона широко используются для защиты от ошибок?
3. Какие методы кодирования для кодов Рида-Соломона вы знаете.
4. Как построить порождающий полином кода Рида-Соломона.
5. Определите вид порождающего полинома кода Рида-Соломона (7, 5) над полем $GF(2^3)$.
6. Найдите порождающий многочлен кода Рида-Соломона длиной 15, исправляющего двойные ошибки.
7. Алгоритм кодирования кодов Рида-Соломона.
8. Закодировать информационную последовательность $u(x) = \alpha^5 x^2 + \alpha x + \alpha^2$ кодом Рида-Соломона (7, 3, 5).
9. Найти кодовое слово, соответствующее информационному многочлену $\alpha^5 x + \alpha^3$ для кода Рида-Соломона длины 15, исправляющего одиночные ошибки и построенного по примитивному элементу α поля $GF(2^4)$.
10. Основные этапы декодирования кодов Рида-Соломона.
11. Алгоритмы решения ключевого уравнения в процедуре декодирования кодов Рида-Соломона.
12. Нахождение полинома значений ошибок.
13. Алгоритм Форни для вычисления значений ошибок кодов Рида-Соломона.
14. Найдите полином локаторов ошибок и полином значений ошибок для кода РС длиной 15, исправляющего двойные ошибки, если принятое кодовое слово $f(x) = \alpha x^{10} + \alpha^{10} x^2$.
15. Исправление стираний и ошибок кодов Рида-Соломона.
16. Нахождение многочлена локаторов искажений с помощью алгоритма Питерсона-Горенштейна-Цирлера.
17. Итеративный алгоритм Берлекэмпа-Месси для исправления стираний и ошибок.
18. Алгоритм Евклида для решения ключевого уравнения кодов Рида-Соломона.

19. Пусть для кода Рида-Соломона длиной 15, исправляющего тройные ошибки, принятый вектор равен $f(x) = x^{10}+\alpha^{11}x^{12}$. Стертые символы на позициях x^2 иx^5. Осуществить декодирование ошибок и стираний, решив ключевое уравнение с помощью алгоритма Берлекэмпа-Месси.

20. На вход декодера поступила комбинация $f(x) = \alpha + x^5 + x^{10}$. Вычислить синдромный многочлен и значения ошибок, используя алгоритм Евклида и Форни.

ГЛАВА 5.

ОПИСАНИЕ БЧХ КОДОВ В СПЕКТРАЛЬНОЙ ОБЛАСТИ

5.1 Преобразования Фурье в конечных полях

Рассмотрим еще один подход к описанию полиномиальных кодов, который основан на использовании дискретного преобразования Фурье (ДПФ) кодовых последовательностей, заданных над конечным полем $GF(p)$. Данный подход позволяет в ряде случаев упростить процедуры кодирования и декодирования.

Дискретное преобразование Фурье в поле комплексных чисел вектора $s = \{s_0, s_1, \ldots, s_{n-1}\}$ с комплексными компонентами определяется как вектор $S = \{S_0, S_1, \ldots, S_{n-1}\}$ задаваемый равенством

$$S_k = \sum_{i=0}^{n-1} s_i \exp\left(j\frac{2\pi}{n} ik \right) = \sum_{i=0}^{n-1} s_i \zeta^{ik}\ ,\ k = 0, \ldots, n-1. \qquad (5.1)$$

где $j = \sqrt{-1}$.

Ядром преобразования Фурье является $\zeta = \exp(-j2\pi/n)$, которое является примитивным корнем n-й степени из единиц в поле комплексных чисел: $\zeta^n = 1$, но $\zeta^t = 1$ для любого $t < n$. Исходный образ сигнала во временной области восстанавливается по его спектру с помощью обратного преобразованья Фурье

$$s_i = \frac{1}{n}\sum_{k=0}^{n-1} S_k \exp\left(j\frac{2\pi ik}{n} \right) = \frac{1}{n}\sum_{i=0}^{n-1} S_k \zeta^{-ik}\ ,\ \ i = 0, \ldots, n-1. \qquad (5.2)$$

В конечном поле $GF(p^m)$ примитивный элемент α, обладающий мультипликативным порядком $n = p^m - 1$, также является корнем n-й степени из единиц $\alpha^n = 1$.

Таким образом, проводя аналогию между $\exp(-j2\pi/n)$ и α, можно ввести следующее определение.

Рассмотрим последовательность, которая представлена вектором $u = \{u_0, u_1, \ldots, u_{n-1}\}$ длины $n = p^m - 1$ компоненты которого принадлежат полю $GF(p)$. Записав его в полиномиальной форме

$$u(x) = u_{n-1}x^{n-1} + u_{n-2}x^{n-2} + ... + u_0, \quad (5.3)$$

и подставив вместо x некоторую степень α^k примитивного элемента α поля $GF(p^m)$, получим

$$u'_k = u(\alpha^k) = \sum_{i=0}^{n-1} u_k \alpha^{ik}, \; k = 0, 1, \ldots, n-1. \quad (5.4)$$

При этом выражение (5.4) может быть обращено следующим образом

$$u_i = \frac{1}{n} \sum_{k=0}^{n-1} u'_k \alpha^{-ik}, \; i = 0, \ldots, n-1, \quad (5.5)$$

что демонстрирует полное совпадение (5.4), (5.5) соотношениям (5.1), (5.2), которые отвечают вещественным или комплексным сигналам. Таким образом, вектор $u' = \{u'_0, u'_1, \ldots, u'_{n-1}\}$ может трактоваться как ДПФ вектора $u = \{u_0, u_1, \ldots, u_{n-1}\}$ над полем $GF(p^m)$.

Пусть $u = \{u_0, u_1, \ldots, u_{n-1}\}$ – последовательность из n элементов конечного поля $GF(p)$, причем n делит $p^m - 1$ для некоторого m, и пусть α – примитивный элемент порядка n в расширении поля $GF(p^m)$.

Дискретным преобразованием Фурье вектора U в поле $GF(p^m)$ называется последовательность $v = \{v_0, v_1, \ldots, v_{n-1}\}$ элементов поля $GF(p^m)$ задаваемая равенством

$$v_j = \sum_{i=0}^{n-1} u_i \alpha^{ij}, \; j = 1, \ldots, n-1. \quad (5.6)$$

В случае дискретного преобразования Фурье преобразование S вещественно-значной временной функции s является комплексным. Аналогично если в случае преобразования в поле Галуа временная функция u принимает значения в поле $GF(p)$, то ее спектр V лежит в расширении поля $GF(p^m)$. Это используется при декодировании кодов, так как алгоритмы декодирования осуществляются в поле $GF(p^m)$.

В матричной форме ДПФ может быть записано следующим образом

$$v = (u_0 u_1 ... u_{n-1}) \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \\ 1 & \alpha^{n-1} & \alpha^{2(n-1)} & \cdots & \alpha^{(n-1)^2} \end{vmatrix}. \quad (5.7)$$

Учитывая ранее указанную аналогию, дискретный индексіназывается *дискретным временем*, вектор *u*– *временная функция* или *сигнал*, индекс *j* – *дискретная частота*, вектор *v* – *частотная функция* или *спектром*.

Преобразование Фурье обладает рядом замечательных свойств, которые переносятся и на случай преобразования в конечных полях. В отличие от поля комплексных чисел в поле Галуа преобразование Фурье существует не для любой длины *n*, так как не для любого *n* в поле существует элемент этого порядка. Если *m* – наименьшее целое, такое, что *n* делит p^m-1, то над полем *GF*(*p*) существует преобразование Фурье длины *n*и компоненты этого преобразования лежат в поле $GF(p^m)$.

Если векторы *u* и *v* связаны равенством (5.6) и однозначно определяют друг друга, то прямому преобразованию Фурье (5.6) соответствует обратное преобразование Фурье

$$u_i = \frac{1}{n} \sum_{j=0}^{n-1} v_i \alpha^{-ij}, \; j = 1, \ldots, n-1. \quad (5.8)$$

Заметим, что *n* есть число поля $GF(p^m)$, равное остатку от деления *n* mod *p*, где *p* – характеристика поля. В случае полей характеристики 2 число *n* можно заменить на 1, т.к. в этом случае *n* – всегда нечетное число.

Преобразование Фурье обладает многими сильными свойствами, которые переносятся на случай конечных полей. Примером является *свойство свертки*, приведенное ниже.

Пусть *u, g* и *w* – временные последовательности, причем $w_i = u_i \cdot g_i$, $i = 0, \ldots, n-1$.Частотные функции*U, G* и *W* являются преобразованием Фурье векторов *u, g* и *w*. Тогда компоненты дискретного преобразования Фурье *W* могут быть определены как

$$W_j = \frac{1}{n}\sum_{k=0}^{n-1} U_{((j-k))} G_k \ , j = 1, \ldots, n-1, \qquad (5.9)$$

где двойные скобки означают, что индексы вычисляются по модулю n.

При этом если U, G и W – частотные последовательности, причем $W_j = U_j G_j$, тогда компоненты вектора w могут быть определены, т.е. справедливо обратное утверждение

$$w_i = \sum_{j=0}^{n-1} u_{((i-j))} g_j \ , i = 0, \ldots, n-1 \ . \qquad (5.10)$$

Утверждения (5.9) и (5.10) носят название теорем о свертке в частотной и временной областях соответственно.

Как было показано выше иногда вектор $u(x)$ задается в виде многочлена (5.3). С помощью дискретного преобразования Фурье в поле $GF(p^m)$ многочлен $u(x)$ может быть преобразован в многочлен

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_0, \qquad (5.11)$$

который называется спектральным многочленом.

Свойства спектра тесно связаны с корнями α^i многочленов. Если вектор u во временной области и его преобразование Фурье v заданы в виде полиномов

$$u(x) = \sum_{i=0}^{n-1} u_i x^i \quad \text{è} \quad v(x) = \sum_{i=0}^{n-1} v_i x^i \ , \qquad (5.12)$$

тогда справедливы следующие утверждения:

– элемент α^i поля $GF(p^m)$ является корнем полинома $u(x)$ тогда и только тогда, когда i-я частотная компонента v_i равна нулю;

– элемент α^{-i} поля $GF(p^m)$ является корнем полинома $v(x)$ тогда и только тогда, когда i-я временная компонента u^i равна нулю.

5.2 Кодирование в спектральной области БЧХ кодов

Циклический код задается своим кодовым многочленом $v(x)$степени $n-1$, который образуется для несистематического кодирования в виде $v(x) = g(x)\,u(x)$, где $u(x)$ и $g(x)$ – информационный порождающий многочлен соответственно и $\{v_i,\ g_i,\ u_i\}$ є $GF(p)$. Тогда во временной области слово $v = \{v_0,\ v_1,\ \ldots,\ v_{n-1}\}$получается в результате циклической свертки

$$v_i = \sum_{k=0}^{n-1} g_{((i-k))} u_k \,. \qquad (5.13)$$

Обозначим через V_j, G_j и U_j как спектральные коэффициенты частотных векторов V, G и U. Тогда в частотной области из теоремы свертки операция кодирования может быть записана

$$V_j = G_j U_j \,,\ j = 1,\ \ldots,\ n-1. \qquad (5.14)$$

Удовлетворяющий этому равенству спектр задает в частотной области кодовое слово при условии, что во временной области все компоненты являются $GF(p)$-значными. Таким образом, в частотной области можно дать следующее определение циклического кода.

Циклическим кодом называется множество таких слов над $GF(p)$, у которых все спектральные компоненты, принадлежащие заданному множеству, так называемых *проверочных частотj*$_1, j_2, \ldots, j_{n-k}$ равны нулю.

Как известно БЧХ код задается корнями α^b, α^{b+1}, …, α^{b+2t-1}порождающего полинома $g(x)$, благодаря которым обеспечивается конструктивное расстояние d. Кроме того, поскольку $g(\alpha^i) = 0$ ($i = b,\ b+1,\ b+2t-1$), где α^i – корень порождающего полинома, то следовательно $V_i = 0$ для всех $i = b,\ b+1,\ b+2t-1$, т.е. спектр каждого кодового слова имеет нулевые значения на интервале $i = b,\ b+1,\ b+2t-1$.

Коды БЧХ являются такими циклическим кодами, у которых проверочные частоты выбираются последовательно. Исправляющий БЧХ код длиной $n = p^m-1$ определяется как множество кодовых слов над полем $GF(p^m)$, спектр которых равен нулю в заданном блоке из $2t$ последовательных частот.

Хотя каждое слово БЧХ кода является вектором над $GF(p)$, спектр кодового слова является вектором над $GF(p^m)$. Таким образом, БЧХ код может быть определен как множество $GF(p)$-значных обратных преобразований Фурье множества всех спектральных векторов, компоненты которых в заданном множестве частот равны нулю. Нельзя выбирать произвольный спектральный вектор, у которого стоят нули в заданном множестве частот, обратные преобразования Фурье некоторых таких векторов могут иметь компоненты, не принадлежащие полю $GF(p)$. Для того чтобы кодовое слово принадлежало полю $GF(p)$, выбирать нужно только спектр, который удовлетворяет *условиям сопряженности*. Эти условия сформулированы следующим образом.

Условие сопряженности. Пусть $V = \{V_0, V_1, \ldots, V_{n-1}\}$ есть частотный вектор с компонентами из поля $GF(p^m)$, где n делит $p^m - 1$. Тогда обратное преобразование Фурье v является вектором с компонентами $\{v_0, v_1, \ldots, v_{n-1}\}$ из поля $GF(p)$ тогда и только тогда, когда выполняется следующее равенство

$$V_j^q = V_{((jq))}, j = 1, \ldots, n-1. \qquad (5.15)$$

Кодирование в спектральной области выполняется автоматически за счет выбора спектрального вектора $V = \{V_0, V_1, \ldots, V_{n-1}\}$ удовлетворяющего условиям сопряженности и требованию $V_j = 0$ для всех j для которых $g(\alpha^j) = 0$. При кодировании $2t$ последовательных компонент спектра выбираются для того, чтобы в них записывались нули. Остальные компоненты представляют собой k информационных символов, который должны выбираться из $GF(p^m)$ такими p^k способами, чтобы обратное преобразование Фурье принимало значение в $GF(p)$.

Чтобы применить условие сопряженности при выборе ненулевых компонент кодового слова разобьем все числа по модулю n на подмножества известные под названием *циклотомические классы* или *классы сопряженных элементов* (п. 1.6). Все числа по модулю n разбиваются на циклотомические смежные классы, которые не пересекаются следующим образом

$$I_i = \{C_0, C_1, ..., C_{l-1}\} = \{i, pi, p^2 i, ..., p^{m-1} i\}.$$

Таким образом, числа по модулю распадаются на N множеств $I_1, I_2, \ldots, I_N$, которые соответствуют логарифмическому представлению

циклотомических классов. Пусть количество элементов каждого класса обозначается $l_1, l_2, \ldots, l_N$.

Например, если $p = 2$, $m = 3$ и $n = 7$, то имеет место разложение на $N = 3$ класса сопряженных элементов

$I_1 = \{C_0\}=\{0\}$, $I_2 = \{C_1, C_2, C_3\} = \{1, 2, 4\}$, $I_3 =\{C_1, C_2, C_3\}= \{3, 6, 5\}$.

Количество элементов каждого класса $l_1=1, l_2=3, l_3=3$.

Класс сопряженных элементов I_k выделяет в спектре множество частот. Учитывая условие сопряженности (5.15), можно утверждать, что если временной сигнал принимает значения в поле $GF(p)$, то значение спектра в одной из частот класса j определяет значение спектра при всех частотах этого класса. Таким образом, условия сопряженности требуют, чтобы спектральные коэффициенты C_j определялись для всех индексов j, которые принадлежат одному множеству I_k.

Для формирования БЧХ кода в качестве проверочных частот выбираются $2t$ спектральных компонент, которые полагаются равными нулю. Остальные значимые символы являются информационными и могут принимать произвольные значения с учетом условия сопряженности. И для этого множество $p^m - 1$ чисел разбивается на классы сопряженных элементов и выбирается по одному числу из каждого класса в качестве представителя C_j, которые единственным образом определяют значимые символы. Все остальные символы, индексы которых принадлежат тому же классу сопряженных элементов, не являются свободными, так как они образуют связанные частоты следующим образом.

Если в качестве представителя класса выбрать элемент C_j и задать ему значение α, так что $C_j = \alpha$, то остальные компоненты спектра, принадлежащие этому классу I_k определяются

$$C_{jp} = \alpha^p, \quad C_{p^2 j} = \alpha^{p^2}, ..., C_{p^{l_k-1} j} = \alpha^{p^{l_k-1}}. \qquad (5.15)$$

Так как при построении циклотомических классов $\alpha^{p^{l_k}} = \alpha$, то величина α и C_j принадлежат полю $GF(p^{l_k})$. Поэтому структура спектрального вектора $C = \{C_0, C_1, \ldots, C_{n-1}\}$, порождается N величинами, а именно порождающими элементами – $\alpha_1, \alpha_2, \ldots, \alpha_N$, причем α_k является элементом поля $GF(p^{l_k})$, которое в свою очередь является под полем $GF(p^m)$. Если спектральная компонента C_j для каждого класса задана с

помощью порождающего элемента α_k, то каждая другая спектральная компонента, индекс которой принадлежит классу сопряженных с j элементов, является степенью C_j или степенями порождающего элемента α_k. Если мощность этого класса сопряженных элементов равна l, то

$$C_j^{p^l} = C_j, \quad \text{ò.å.} \quad C_j^{p^l-1} = 1. \tag{5.17}$$

Следовательно, выбор спектральных компонент из $GF(p^m)$ в качестве возможного значения для C_j осуществляется следующим образом: допустимы или только те элементы поля, порядок которых делит $p^l - 1$ или нулевой элемент.

Количество спектральных векторов C с указанными свойствами в точности равно числу кодовых слов, а именно $M = \prod_{\alpha_k \neq 0} p^{l_k}$, где произведение берется по тем индексам k, которые соответствуют ненулевым порождающим элементам α_k.

Пример 5.1. Построить двоичный БЧХ кодер (7, 4, 3), который способен исправлять однократные ошибки ($t = 1$) с минимальным расстоянием $d = 3$, задаваемый набором корней $\{\alpha^1, \alpha^2\}$ и порождающим полиномом $g(x) = x^3 + x + 1$. Поле $GF(2^3)$ порождается примитивным элементом α, который является корнем многочлена $p(x) = x^3 + x + 1$.

Для построения БЧХ кодера в спектральной области необходимо построить частотный спектр, который состоит из $n = 7$ коэффициентов $C = \{C_0, C_1, C_2, C_3, C_4, C_5, C_6\}$.

Для описания БЧХ кодера в спектральной области разобьем множество $p^m - 1 = 2^3 - 1 = 7$ чисел на классы сопряженных элементов

$I_1 = \{0\}$, $I_2 = \{1, 2, 4\}$, $I_3 = \{3, 6, 5\}$.

В качестве проверочных частот выбираются $2t$ спектральных компонент, которые полагаются равными нулю. Так как код способен исправлять одну ошибку ($t = 1$), а корни порождающего полинома α^1 и α^2, то выберем в качестве проверочных частот $j = 1$ и $j = 2$. Таким образом, первая и вторая спектральные компоненты C_1 иC_2 равны нулю:$C_1 = 0$, $C_2 = 0$.

Тогда с учетом (5.17) любая компонента лежащая в множестве класса I_2 равна нулю и соответственно четвертая компонента $C_4 = 0$.

Спектральные компоненты C_0, C_3, C_5, C_6 являются информационными символами, которые необходимо задать с выполнением условия сопряженности. Поскольку компоненты C_0, C_3, C_5, C_6 принадлежат

циклотомическим классам I_1 и I_3 , то необходимо выбрать в каждом классе компоненты свободной частоты и задать порождающие элементы α_1 и α_3.

Для класса I_1 компонента свободной частоты $-C_0$, а для класса I_3 компонента свободной частоты $- C_3$. Тогда остальные спектральные коэффициенты связанных частот (C_5, C_6) порождаются двумя элементами $\alpha_1 \in GF(2)$, $\alpha_3 \in GF(2^3)$. При этом в качестве порождающего элемента для класса I_1 может выбираться элемент 0 или 1, т.к. α_1 принадлежит подполю $GF(2)$. Для класса I_3 в качестве порождающего элемента может быть задан любой элемент порядка 7, принадлежащий подполю $GF(2^3)$.

Выберем для порождающих элементов значения: $\alpha_1 = 1$ и $\alpha_3 = \alpha^2$, таким образом, нулевая и третья спектральные компоненты равны $C_0 = 1$ и $C_3 = \alpha^2$.

Теперь необходимо вычислить значения для пятой и шестой спектральных компонент C_5, C_6. Так как индексы этих компоненты (5, 6) принадлежат классу I_3, то для вычисления C_5 и C_6 необходимо воспользоваться формулой (5.16)

$C_5 = \alpha_3^4 = (\alpha^2)^4 = \alpha^8 = \alpha$, $C_6 = \alpha_3^2 = (\alpha^2)^2 = \alpha^4$, $C_5 = \alpha$, $C_6 = \alpha^4$.

Таким образом, получены спектральные компоненты

$$\boldsymbol{C_0 = 1, C_1 = 0, C_2 = 0, C_3 = \alpha^2, C_4 = 0, C_5 = \alpha, C_6 = \alpha^4.}$$

Если для порождающих элементов α_1 и α_3 задать другие значения, например $\alpha_1 = 0$ и $\alpha_3 = \alpha^4$, тогда будут получены другие спектральные компоненты, а именно $C_0 = 0$ и $C_3 = \alpha^4$

$C_5 = \alpha_3^4 = (\alpha^4)^4 = \alpha^{16} = \alpha^2$, $C_6 = \alpha_3^2 = (\alpha^4)^2 = \alpha^8 = \alpha$, $C_5 = \alpha^2$, $C_6 = \alpha$.

И спектральный вектор будет иметь вид

$$\boldsymbol{C_0 = 0, C_1 = 0, C_2 = 0, C_3 = \alpha^4, C_4 = 0, C_5 = \alpha^2, C_6 = \alpha.}$$

В табл. 5.1 представлены все кодовые слова, полученные в частотной области для кода (7, 4) и выделены спектральные последовательности, которые вычислены в данном примере.

Таблица 5.1 – Кодовые слова в спектральной области для БЧХ кода (7, 4)

Кодовые слова в спектральной области							Кодовые слова во временной области						
C_0	C_1	C_2	C_3	C_4	C_5	C_6	u_6	u_5	u_4	u_3	u_2	u_1	u_0
0	0	0	0	0	0	0							
0	0	0	1	0	1	1							
0	0	0	α	0	α^4	α^2							
0	0	0	α^2	0	α	α^4							
0	0	0	α^3	0	α^5	α^6							
0	**0**	**0**	$\boldsymbol{\alpha^4}$	**0**	$\boldsymbol{\alpha^2}$	**α**	**1**	**1**	**1**	**0**	**1**	**0**	**0**
0	0	0	α^5	0	α^6	α^3							
0	0	0	α^6	0	α^3	α^5							
1	0	0	0	0	0	0							
1	0	0	1	0	1	1							
1	0	0	α	0	α^4	α^2							
1	**0**	**0**	$\boldsymbol{\alpha^2}$	**0**	**α**	$\boldsymbol{\alpha^4}$	**0**	**1**	**1**	**0**	**0**	**0**	**1**
1	0	0	α^3	0	α^5	α^6							
1	0	0	α^4	0	α^2	α^1							
1	0	0	α^5	0	α^6	α^3	1	1	0	0	0	1	0
1	0	0	α^6	0	α^3	α^5	1	0	1	1	0	0	0

Формируемые таким образом 2^4 кодовых слов являются в точности теми же 2^4 кодовыми словами, которые получаются кодированием во временной области. Для получения кодового слова во временной области необходимо выполнить обратное преобразование Фурье для соответствующей спектральной последовательности.

Например, для вектора C {1, 0, 0, α^2, 0, α, α^4} вычислим обратное преобразование Фурье с помощью формулы (5.8). Так как рассматривается поле характеристики $p = 2$, то в формуле $n = 1$.

$$u_0 = \sum_{j=0}^{6} C_j \alpha^{-0j} = C_0\alpha^0 + C_1\alpha^0 + C_2\alpha^0 + C_3\alpha^0 + C_4\alpha^0 + C_5\alpha^0 + C_6\alpha^0 =$$

$$= 1\cdot 1 + 0\cdot 1 + 0\cdot 1 + \alpha^2\cdot 1 + 0\cdot 1 + \alpha^1\cdot 1 + \alpha^4\cdot 1 =$$

$$= 1 + 0 + 0 + \alpha^2 + 0 + \alpha^1 + \alpha^4 = 1 + \alpha^2 + \alpha^1 + \alpha^4 = \alpha^6 + \alpha^2 = 1,$$

$$u_1 = \sum_{j=0}^{6} C_j \alpha^{-1j} = C_0\alpha^0 + C_1\alpha^{-1} + C_2\alpha^{-2} + C_3\alpha^{-3} + C_4\alpha^{-4} + C_5\alpha^{-5} + C_6\alpha^{-6} =$$
$$= 1\cdot\alpha^0 + 0\cdot\alpha^{-1} + 0\cdot\alpha^{-2} + \alpha^2\alpha^{-3} + 0\cdot\alpha^{-4} + \alpha^1\alpha^{-5} + \alpha^4\alpha^{-6} =$$
$$= 1 + 0 + 0 + \alpha^{-1} + 0 + \alpha^{-4} + \alpha^{-2} = 1 + \alpha^6 + \alpha^3 + \alpha^5 = \alpha^2 + \alpha^2 = 0,$$

$$u_2 = \sum_{j=0}^{6} C_j \alpha^{-2j} = C_0\alpha^0 + C_1\alpha^{-2} + C_2\alpha^{-4} + C_3\alpha^{-6} + C_4\alpha^{-8} + C_5\alpha^{-10} + C_6\alpha^{-12} =$$
$$= 1\cdot\alpha^0 + 0\cdot\alpha^{-1} + 0\cdot\alpha^{-2} + \alpha^2\alpha^{-6} + 0\cdot\alpha^{-8} + \alpha^1\alpha^{-10} + \alpha^4\alpha^{-12} =$$
$$= 1 + 0 + 0 + \alpha^{-4} + 0 + \alpha^{-9} + \alpha^{-8} = 1 + \alpha^3 + \alpha^5 + \alpha^6 = \alpha^1 + \alpha^1 = 0,$$

$$u_3 = \sum_{j=0}^{6} C_j \alpha^{-3j} = C_0\alpha^0 + C_1\alpha^{-3} + C_2\alpha^{-6} + C_3\alpha^{-9} + C_4\alpha^{-12} + C_5\alpha^{-15} + C_6\alpha^{-18} =$$
$$= 1\cdot\alpha^0 + 0\cdot\alpha^{-3} + 0\cdot\alpha^{-6} + \alpha^2\alpha^{-9} + 0\cdot\alpha^{-12} + \alpha^1\alpha^{-15} + \alpha^4\alpha^{-18} =$$
$$= 1 + 0 + 0 + \alpha^{-7} + 0 + \alpha^{-14} + \alpha^{-14} = 1 + 1 + 1 + 1 = 0,$$

$$u_4 = \sum_{j=0}^{6} C_j \alpha^{-4j} = C_0\alpha^0 + C_1\alpha^{-4} + C_2\alpha^{-8} + C_3\alpha^{-12} + C_4\alpha^{-16} + C_5\alpha^{-20} + C_6\alpha^{-24} =$$
$$= 1\cdot\alpha^0 + 0\cdot\alpha^{-4} + 0\cdot\alpha^{-8} + \alpha^2\alpha^{-12} + 0\cdot\alpha^{-16} + \alpha^1\alpha^{-20} + \alpha^4\alpha^{-24} =$$
$$= 1 + 0 + 0 + \alpha^{-10} + 0 + \alpha^{-19} + \alpha^{-20} = 1 + \alpha^4 + \alpha^2 + \alpha^1 = \alpha^5 + \alpha^4 = 1,$$

$$u_5 = \sum_{j=0}^{6} C_j \alpha^{-5j} = C_0\alpha^0 + C_1\alpha^{-5} + C_2\alpha^{-10} + C_3\alpha^{-15} + C_4\alpha^{-20} + C_5\alpha^{-25} + C_6\alpha^{-30} =$$
$$= 1\cdot\alpha^0 + 0\cdot\alpha^{-5} + 0\cdot\alpha^{-10} + \alpha^2\alpha^{-15} + 0\cdot\alpha^{-20} + \alpha^1\alpha^{-25} + \alpha^4\alpha^{-30} =$$
$$= 1 + 0 + 0 + \alpha^{-13} + 0 + \alpha^{-24} + \alpha^{-26} = 1 + \alpha^1 + \alpha^4 + \alpha^2 = \alpha^3 + \alpha^1 = 1,$$

$$u_6 = \sum_{j=0}^{6} C_j \alpha^{-6j} = C_0\alpha^0 + C_1\alpha^{-6} + C_2\alpha^{-12} + C_3\alpha^{-18} + C_4\alpha^{-24} + C_5\alpha^{-30} + C_6\alpha^{-36} =$$
$$= 1\cdot\alpha^0 + 0\cdot\alpha^{-6} + 0\cdot\alpha^{-12} + \alpha^2\alpha^{-18} + 0\cdot\alpha^{-24} + \alpha^1\alpha^{-30} + \alpha^4\alpha^{-36} =$$
$$= 1 + 0 + 0 + \alpha^{-16} + 0 + \alpha^{-29} + \alpha^{-32} = 1 + \alpha^5 + \alpha^6 + \alpha^3 = \alpha^4 + \alpha^4 = 0.$$

Таким образом, для спектрального вектора C {1, 0, 0, α^2, 0, α, α^4} получено обратное преобразование Фурье в виде временного вектора

$$U = \{u_6, u_5, u_4, u_3, u_2, u_1, u_0\} = \{0, 1, 1, 0, 0, 0, 1\}.$$

В виде многочлена временная кодовая последовательности выглядит следующим образом: $u(x) = x^5+x^4+x^3$.

Для проверки соответствия вектора C вектору U выполним дискретное преобразование Фурье (5.6) кодовой последовательности $u(x)$

$$v_0 = \sum_{i=0}^{6} u_i \alpha^{0i} = 1 + 1 + 1 = 1,$$

$$v_1 = \sum_{i=0}^{6} u_i \alpha^{1i} = \alpha^5 + \alpha^4 + 1 = 0,$$

$$v_2 = \sum_{i=0}^{6} u_i \alpha^{2i} = \alpha^{10} + \alpha^8 + 1 = \alpha^3 + \alpha + 1 = 0,$$

$$v_3 = \sum_{i=0}^{6} u_i \alpha^{3i} = \alpha^{15} + \alpha^{12} + 1 = \alpha + \alpha^5 + 1 = \alpha^2,$$

$$v_4 = \sum_{i=0}^{6} u_i \alpha^{4i} = \alpha^{20} + \alpha^{16} + 1 = \alpha^6 + \alpha^2 + 1 = 1 + 1 = 0,$$

$$v_5 = \sum_{i=0}^{6} u_i \alpha^{5i} = \alpha^{25} + \alpha^{20} + 1 = \alpha^4 + \alpha^6 + 1 = \alpha,$$

$$v_6 = \sum_{i=0}^{6} u_i \alpha^{6i} = \alpha^{30} + \alpha^{24} + 1 = \alpha^2 + \alpha^3 + 1 = \alpha^4.$$

Таким образом, полученный спектральный вектор $\{1, 0, 0, \alpha^2, 0, \alpha, \alpha^4\}$ полностью совпадает с вектором C, рассчитанным ранее.

Для спектрального вектора $C\{0, 0, 0, \alpha^4, 0, \alpha^2, \alpha^1\}$ обратное преобразование Фурье вычисляется

$$u_0 = \sum_{j=0}^{6} C_j \alpha^{-0j} = C_0\alpha^0 + C_1\alpha^0 + C_2\alpha^0 + C_3\alpha^0 + C_4\alpha^0 + C_5\alpha^0 + C_6\alpha^0 =$$
$$= 0\cdot 1 + 0\cdot 1 + 0\cdot 1 + \alpha^4\cdot 1 + 0\cdot 1 + \alpha^2\cdot 1 + \alpha^1\cdot 1 =$$
$$= 0 + 0 + 0 + \alpha^4 + 0 + \alpha^2 + \alpha^1 = 0 + \alpha^4 + \alpha^2 + \alpha^1 = \alpha^1 + \alpha^1 = 0,$$

$$u_1 = \sum_{j=0}^{6} C_j \alpha^{-1j} = C_0\alpha^0 + C_1\alpha^{-1} + C_2\alpha^{-2} + C_3\alpha^{-3} + C_4\alpha^{-4} + C_5\alpha^{-5} + C_6\alpha^{-6} =$$
$$= 0\cdot\alpha^0 + 0\cdot\alpha^{-1} + 0\cdot\alpha^{-2} + \alpha^4\alpha^{-3} + 0\cdot\alpha^{-4} + \alpha^2\alpha^{-5} + \alpha^1\alpha^{-6} =$$
$$= 0 + 0 + 0 + \alpha^1 + 0 + \alpha^{-3} + \alpha^{-5} = \alpha^1 + \alpha^4 + \alpha^2 = \alpha^2 + \alpha^2 = 0,$$

$$u_2 = \sum_{j=0}^{6} C_j \alpha^{-2j} = C_0\alpha^0 + C_1\alpha^{-2} + C_2\alpha^{-4} + C_3\alpha^{-6} + C_4\alpha^{-8} + C_5\alpha^{-10} + C_6\alpha^{-12} =$$
$$= 0\cdot\alpha^0 + 0\cdot\alpha^{-1} + 0\cdot\alpha^{-2} + \alpha^4\alpha^{-6} + 0\cdot\alpha^{-8} + \alpha^2\alpha^{-10} + \alpha^1\alpha^{-12} =$$
$$= 0 + 0 + 0 + \alpha^{-2} + 0 + \alpha^{-8} + \alpha^{-11} = \alpha^5 + \alpha^6 + \alpha^3 = \alpha^1 + \alpha^3 = 1,$$

$$u_3 = \sum_{j=0}^{6} C_j \alpha^{-3j} = C_0\alpha^0 + C_1\alpha^{-3} + C_2\alpha^{-6} + C_3\alpha^{-9} + C_4\alpha^{-12} + C_5\alpha^{-15} + C_6\alpha^{-18} =$$
$$= 0\cdot\alpha^0 + 0\cdot\alpha^{-3} + 0\cdot\alpha^{-6} + \alpha^4\alpha^{-9} + 0\cdot\alpha^{-12} + \alpha^2\alpha^{-15} + \alpha^1\alpha^{-18} =$$
$$= 0 + 0 + 0 + \alpha^{-5} + 0 + \alpha^{-13} + \alpha^{-17} = \alpha^2 + \alpha^1 + \alpha^4 = 0,$$

$$u_4 = \sum_{j=0}^{6} C_j \alpha^{-4j} = C_0\alpha^0 + C_1\alpha^{-4} + C_2\alpha^{-8} + C_3\alpha^{-12} + C_4\alpha^{-16} + C_5\alpha^{-20} + C_6\alpha^{-24} =$$
$$= 0\cdot\alpha^0 + 0\cdot\alpha^{-4} + 0\cdot\alpha^{-8} + \alpha^4\alpha^{-12} + 0\cdot\alpha^{-16} + \alpha^2\alpha^{-20} + \alpha^1\alpha^{-24} =$$
$$= 0 + 0 + 0 + \alpha^{-8} + 0 + \alpha^{-18} + \alpha^{-23} = \alpha^6 + \alpha^3 + \alpha^5 = \alpha^4 + \alpha^5 = 1,$$

$$u_5 = \sum_{j=0}^{6} C_j \alpha^{-5j} = C_0\alpha^0 + C_1\alpha^{-5} + C_2\alpha^{-10} + C_3\alpha^{-15} + C_4\alpha^{-20} + C_5\alpha^{-25} + C_6\alpha^{-30} =$$
$$= 0\cdot\alpha^0 + 0\cdot\alpha^{-5} + 0\cdot\alpha^{-10} + \alpha^4\alpha^{-15} + 0\cdot\alpha^{-20} + \alpha^2\alpha^{-25} + \alpha^1\alpha^{-30} =$$
$$= 0 + 0 + 0 + \alpha^{-11} + 0 + \alpha^{-23} + \alpha^{-29} = \alpha^3 + \alpha^5 + \alpha^6 = \alpha^2 + \alpha^6 = 1,$$

$$u_6 = \sum_{j=0}^{6} C_j \alpha^{-6j} = C_0\alpha^0 + C_1\alpha^{-6} + C_2\alpha^{-12} + C_3\alpha^{-18} + C_4\alpha^{-24} + C_5\alpha^{-30} + C_6\alpha^{-36} =$$
$$= 0\cdot\alpha^0 + 0\cdot\alpha^{-6} + 0\cdot\alpha^{-12} + \alpha^4\alpha^{-18} + 0\cdot\alpha^{-24} + \alpha^2\alpha^{-30} + \alpha^1\alpha^{-36} =$$
$$= 0 + 0 + 0 + \alpha^{-14} + 0 + \alpha^{-28} + \alpha^{-35} = 1 + 1 + 1 = 1.$$

При декодировании кодового слова декодеру не приходится заботиться о том, каким способом осуществлялось кодирование. Однако на последнем шаге, когда из исправленного кодового слова извлекается информация, декодеру необходимо знать, как эта информация была закодирована. Если кодирование осуществлялось в частотной области, то информационные символы должны вычисляться в частотной области.

Для недвоичных кодов Рида-Соломона кодовое слово и его спектр лежат в одном и том же поле. Используя для вычисления спектральных компонент информационные символы, можно осуществлять кодирование непосредственно в частотной области.

Поскольку любой кодовый полином $v(x)$ РС кода делится на порождающий полином $g(x)$, то для соответствующего полинома данных $u(x)$ выполняется соотношение$v(x) = g(x)\cdot u(x)$ и тогда k-й спектральный компонент слова может быть определен как

$$V_k = v(\alpha^k) = g(\alpha^k)u(\alpha^k).$$

Поскольку $g(\alpha^{b+1}) = 0$, где α^{b+i}, $i = 0, 1, \ldots, d-1$ – обязательныекорни порождающего многочлена, тогда $V_{b+i} = 0$ для всех $i = 0, 1, \ldots, d-1$.

РС код может быть определен как код, у которого спектры всех кодовых слов равны нулю в пределах окна длины $d-1$. Таким образом, каждый спектр, у которого $2t$ последовательных корней равны нулю, является кодовым словом.

Указанное свойство непосредственно ведет к следующему алгоритму несистематического кодирования в частотной области для РС кода. Кодирование осуществляется следующим образом. Выбираются $2t$ последовательных частот для обеспечения необходимого ограничения, т.е. значения в этих частотах V_b, V_{b+i},…,V_{b+2t-1} полагаются равными нулю. Остальные $n-2t$ спектральных коэффициентов заполняются информационными символами из поля GF(p^m). Тогда обратное преобразование Фурье дает несистематическое кодовое слово кода Рида-Соломона, как показано на рис. 5.1 при $b = 0$.

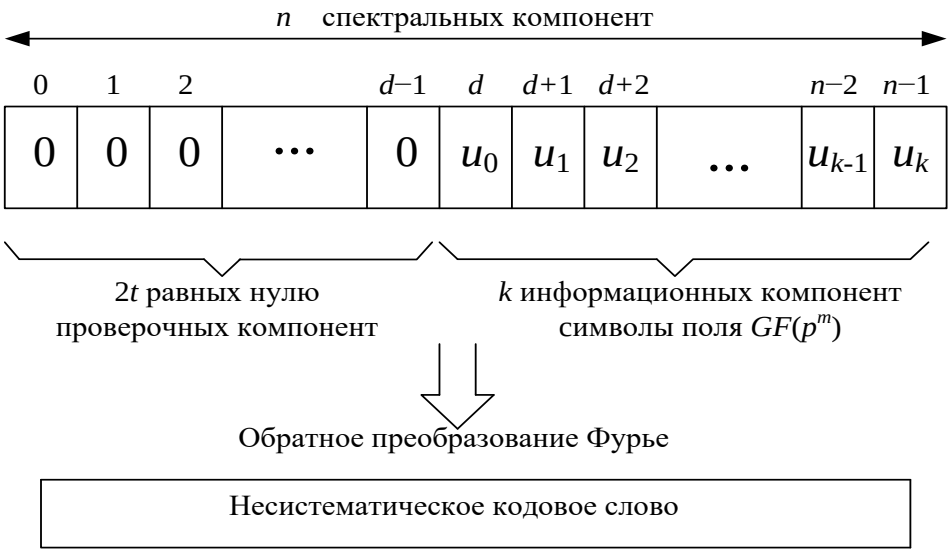

Рисунок 5.1 – Кодирование кодов РС с помощью преобразования Фурье

Например, нулевое спектральное окно (255, 191) РС кода, используемого на транспортном уровне DVB-H стандарта, имеет ширину, равную 64, тогда при $b = 0$ спектральные компоненты на «частотах» $k = 0$, 1, 2, …, 63 равны нулю для всех кодовых слов.

5.3 Декодирование БЧХ и РС кодов в спектральной области

Алгоритмы декодирования БЧХ и РС кодов во временной области были рассмотрены ранее. В этой главе представлены процедуры декодирования циклических кодов в частотной области, используя преобразование Фурье.

Обозначим принятое слов f на выходе канала связи как сумму кодового слова v и вектора ошибок: $f = v + e$, где $e = (e_1, e_2, \ldots, e_{n-1})$ – вектор искажений или, что эквивалентно полином ошибок

$$e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \quad \ldots \quad + e_1x + e_0 .$$

Задача декодера заключается в поиски значений коэффициентов вектора ошибок$e_1, e_2, \ldots, e_{n-1}$ и удалении этих значений из искаженного кодового слова f.

Известно, что синдромные компоненты для принятого кодового слова f задаются следующими равенствами

$$S_j = \sum_{i=0}^{n-1} f_i \alpha^{i(j+b-1)} = f(\alpha^{j+b-1}), \quad j = 1, 2, ..., 2t . \qquad (5.18)$$

Сравнив это равенство с формулой (5.6) можно утверждать, что компоненты синдромного вектора вычисляются как $2t$ компонент преобразования Фурье принятого кодового вектора f .

Для векторов f и e существуют дискретные преобразования Фурье, обозначаемые как $F = (F_0, F_2, \ldots, F_{n-1})$и $E = (E_0, E_2, \ldots, E_{n-1})$, с помощью которых определяются компоненты преобразования Фурье принятого кодового слова f в виде равенства $F_i = V_i + E_i$, где $i = 0, 1, \ldots, n-1$. При этом некоторые дискретные компоненты могут быть определены как значения векторов в нулях кода (корнях порождающего полинома)

$F_i = f(\alpha^j)$, $E_i = f(\alpha^j)$ при$i = 1, \ldots, 2t$; $j = b, b+1, \ldots, b+2t-1$.

Таким образом, спектральные компоненты с частотами $j = b, b+1, \ldots, b+2t-1$ определяются вектором ошибки

$$F_j = f(\alpha^{j+b-1}) = v(\alpha^{j+b-1}) + e(\alpha^{j+b-1}) = e(\alpha^{j+b-1}) = E_j . \qquad (5.19)$$

Известно, что спектральные компоненты кодового вектора V_j, которые расположены в проверочных частотах $j = b, b+1, \ldots, b+2t-1$, равны нулю. То есть

$V_j = 0$ для $j = b, b+1, \ldots, b+2t-1$.

Из определения кода БЧХ ясно, что преобразование Vкаждового кодового слова содержит $2t$ последовательных нулей. Таким образом,

можно получить $2t$ известных значений спектральных компонент вектора искажений

$$S_j = V_{j+b-1} = E_{j+b-1}, j = 1, 2, \ldots, 2t. \quad (5.20)$$

Блок компонент синдрома как бы образует «нулевое окно», через которое можно наблюдать $2t$ из n компонент спектра вектора ошибок, с помощью которых можно однозначно восстановить вектор ошибок.

Задача декодирования заключается в том, чтобы при наличии $t \leq (d-1)/2$ ошибок, по известным $2t$ спектральным компонентам вектора E. восстановить остальные $n-2t$ значений вектора искажений.

Таким образом, первой операцией, выполняемой при декодировании в частотной области, является вычисление компонент синдрома наблюдения на частотах, лежащих внутри нулевого окна

$$S_j = \sum_{i=0}^{n-1} f_i \alpha^{i(j+b-1)}, j = b, b+1, \ldots, b+2t-1. \quad (5.21)$$

Находим $2t$ спектральных компонент $\{E_0, E_1, \ldots, E_{2t}\}$ преобразования Фурье вектора ошибок E с помощью компонент синдрома S_j

$$E_{j+b-1} = S_j, \quad j = 1, 2, ..., 2t. \quad (5.22)$$

Все последующие шаги направлены на экстраполяцию спектра ошибок, определенного в пределах нулевого окна, на всю частотную область. Это оказывается возможным, если только фактическое число ошибочных символов vне превосходит исправляющей способности БЧХ или РС кода.

Как было указано ранее позиции ошибок в кодовом многочлене $f(x)$ могут быть найдены с помощью многочлена локаторов ошибок $\sigma(x)$

$$\sigma(x) = \prod_{j=1}^{v} (1 + \alpha^{i_j} x) = \sum_{k=0}^{v} \sigma_k x^k = \sum_{k=0}^{n-1} \sigma_k x^k, \quad (5.23)$$

где $\sigma_0 = 1$, $\sigma_v \neq 0$, $\sigma_k = 0$ для $k > v$.

Элемент α^{i_j} является корнем многочлена $\sigma(x)$, а все величины, обратные корням являются локаторами ошибок α^{-i}, где степень $i \in (i_1, i_2, \ldots, i_v)$ − есть значение позиции ошибочных символов.

Очевидно, что при подстановке $x = \alpha^{-i}$ в выражение (5.23) для $\sigma(x)$ мы получим компоненты обратного ДПФ, которые обозначаются $\Omega_0, \Omega_1, \ldots, \Omega_{n-1}$,для последовательности коэффициентов полинома локаторов ошибок $\sigma_0, \sigma_1, \ldots, \sigma_{n-1} = \{1, \sigma_1, \sigma_2, \ldots, \sigma_v, 0, 0, \ldots, 0\}$

$$\Omega_j = \sigma(\alpha^{-i}) = \sum_{k=0}^{n-1} \sigma_k \alpha^{-ik}, \; i = 0, 1, \ldots, n-1. \qquad (5.24)$$

Поскольку $\Omega_i = 0$ для любого i, при котором $e_i \neq 0$, то $\Omega_i e_i = 0$ при всех $i = 0, 1, \ldots, n-1$.

Используя теорему о свертке, которая утверждает, что свертка в частотной области равна нулю, и учитывая, что степень многочлена $\sigma(x)$ не превосходит t ($\sigma_i = 0$ для $i > v$) получим

$$\sum_{i=0}^{n-1} \sigma_i E_{k-i} = \sum_{i=0}^{v} \sigma_i E_{k-i} = 0, \; k = 0, 1, \ldots, n-1. \qquad (5.25)$$

Коэффициенты $\sigma_i (i = 0, 1, \ldots, v)$считаются известными, поскольку определен полином локаторов ошибок, причем $\sigma_0 = 1$. Поэтому последнее равенство можно интерпретировать как рекуррентное уравнение

$$E_j = -\sum_{i=1}^{v} \sigma_i E_{j-i}, \; j = 0, 1, \ldots, n-1, \qquad (5.26)$$

в котором все индексы берутся как вычеты по модулю $n-1$.

В этом уравнении знак «минус» может быть опущен в случае полей GF(2^m). Эта система содержит n уравнений, связывающих $n-t$ неизвестных (v коэффициентов многочлена $\sigma(x)$ и $n-2t$ компонент вектора E) и $2t$ известных компонент вектора E, заданных компонентами синдрома. Это равенство можно интерпретировать как уравнение для выходных символов V_j (5.20) линейного регистра сдвига с обратными связями, задаваемыми многочленом $\sigma(x)$. Таким образом, по точно известному отрезку $E_b, E_{b+1}, \ldots, E_{b+2t-1}$ можно вычислить всю последовательность $E_0, E_2, \ldots, E_{n-1}$.

С учетом формулы (5.20) система уравнений (5.26) содержит *v*уравнений с известными компонентами синдрома и *t* неизвестных компонент вектора *E*

$$S_k = -\sum_{j=1}^{v} \sigma_j S_{k-j}\,,\ k = v+1,\ \ldots,\ 2t. \qquad (5.27)$$

Данная система содержит *t* уравнений и может быть разрешена относительно σ, с помощью алгоритма Берлекэмпа-Месси, Питерсона-Горенстейна-Цирлера или Эвклида. Остальные компоненты спектра *S*можно получить рекуррентным продолжением, а именно используя выписанное выше уравнение для свертки, вычислить S_{2t+1} по уже известным компонентам *S* и σ, затем найти S_{2t+2} и т.д.

5.3.1 Алгоритм Питерсона-Горенстейна-Цирлера в частотной области

Если БЧХ или РС код ориентирован на исправление ошибок большой кратности, эффективным оказывается метод Берлекэмпа-Месси. Если же код рассчитан на исправление ошибок малой кратности, то используется *метод Питерсона-Горенстейна-Цирлера*, алгоритм которого представлен ниже.

1. Для принятого кодового слова *f*(*x*) вычислить компоненты синдрома S_j используя формулу (5.18).

2. С помощью синдромного вектора *S* в пределах нулевого окна, т.е. на частотах $j = b,\ b+1,\ \ldots,\ b+2t-1$, вычислить значения компонентов E_b, $E_{b+1}, \ldots, E_{b+2t-1}$ используя формулу (5.22).

3. Составить матрицу M_μразмерности μ x μ (начиная с $\mu = t$) следующего вида (вариант матрицы для $b = 0$)

$$M_\mu = \begin{vmatrix} E_0 & E_1 \ldots & E_{v-1} \\ E_1 & E_2 \ldots & E_v \\ \ldots & \ldots \ldots & \ldots \\ E_{v-1} & E_{v-2} \cdots & E_{2v-2} \end{vmatrix}$$

и уменьшая значение μ на единицу всякий раз, когда M_μ оказывается вырожденной. Если же M_μ – не вырождена, то становится известным величина *v*.

4. При известном значении v с помощью рекурсии (5.25) составляется линейная система из v уравнений с v неизвестными

$$\sigma_1 E_{v-1} + \sigma_2 E_{v-2} + ... + \sigma_v E_0 = -E_v,$$
$$\sigma_1 E_v + \sigma_2 E_{v-1} + \ ... \ + \sigma_v E_1 = -E_{v+1},$$
$$... \quad \dots \quad \dots$$
$$\sigma_1 E_{2v-2} + \sigma_2 E_{2v-1} + ... + \sigma_v E_{v-1} = -E_{2v-1}.$$

В матричной форме система уравнений представляет собой

$$M_v \sigma = -E_v,$$

где $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_v)$ и $E_v = (E_v, E_{v+1}, \dots, E_{2v-1})$. Для варианта $b = 0$ решение данной системы уравнений

$$\begin{bmatrix} \sigma_v \\ \sigma_{v-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = M_v^{-1} \begin{bmatrix} E_v \\ E_{v-1} \\ \vdots \\ E_{2v-1} \end{bmatrix}$$

дает коэффициенты полинома локаторов ошибок, т.е. коэффициенты рекурсии спектра ошибок.

5. С помощью рекурсии восстанавливается весь спектр ошибок

$$E_k = \sum_{i=1}^{v} \sigma_i E_{k-i}, \ k = v, v+1, \dots, n-1.$$

6. Для получения вектора ошибок и построения полинома ошибок $e(x)$ необходимо осуществить обратное дискретное преобразование Фурье

$$e_i = \sum_{k=0}^{n-1} E_k \alpha^{-ik}, \ i = 0, 1, \dots, n-1.$$

Если кодер реализован во ременной области, то для вычисления кодового слова во временной области надо воспользоваться обратным преобразованием Фурье применительно к исправленному спектру, а затем выделить из кодового слова переданную информацию. Если же кодирование производилось в частотной области, то информационные символы определяются непосредственно по исправленному спектру. В этом случае обратного преобразования в декодере делать не надо.

Пример 5.2. Рассмотрим РС код (7, 3, 5) из примера 4.3, использующий конструкцию поля $GF(2^3)$ и способного исправлять любые ошибки кратности один или два ($t = 2$). Корнями порождающего полинома $g(x)$являются элементы поля α^0, α^1, α^2, α^3(нули кода для $b = 0$). Принятая кодовая последовательность с ошибками имеет вид

$f(x) = \alpha x^6 + x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha^5 x + \alpha^6$, при этом вектор ошибок равен $e(x) = \alpha x^6 + \alpha^2 x^2$. Необходимо осуществить декодирование кодовой последовательности $f(x)$ в частотной области используя преобразование Фурье.

Синдромные компоненты для последовательности $f(x)$ из примера 4.3 равны$S_1 = \alpha^4$, $S_2 = \alpha^5$, $S_3 = 0$, $S_4 = \alpha^6$.

Спектральные компоненты вектора ошибок (5.22) равны

$E_0 = S_1 = \alpha^4$,

$E_1 = S_2 = \alpha^5$,

$E_2 = S_3 = 0$,

$E_3 = S_4 = \alpha^6$.

Остальные спектральные компоненты E_4, E_5, E_6 необходимо найти.

Предположим, что при передаче кодовой последовательности произошло две ошибки ($\mu = 2$). Строим матрицу размером (2 x 2)M_2.

$$M_2 = \begin{vmatrix} E_0 & E_1 \\ E_1 & E_2 \end{vmatrix} = \begin{vmatrix} \alpha^4 & \alpha^5 \\ \alpha^5 & 0 \end{vmatrix}.$$

Определитель матрицы $\det M_2 = \alpha^4 \cdot 0 + \alpha^5 \cdot \alpha^5 = \alpha^{10} = \alpha^3$не равен нулю. Таким образом, произошло две ошибки и $\nu = 2$.

Для нахождения коэффициентов σ_1, σ_2 полинома локаторов ошибок $\sigma(x)$ решим систему уравнений

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = M_2^{-1} \begin{bmatrix} E_2 \\ E_3 \end{bmatrix}.$$

Построение обратной матрицы $M_2 = \begin{vmatrix} \alpha^4 & \alpha^5 \\ \alpha^5 & 0 \end{vmatrix}$, осуществляется через миноры $a_{11} = 0$, $a_{12} = \alpha^5$, $a_{21} = \alpha^5$, $a_{22} = \alpha^4$.

$$M_2^{-1} = \frac{1}{|M|} \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = \frac{1}{\alpha^3} \begin{vmatrix} 0 & \alpha^5 \\ \alpha^5 & \alpha^4 \end{vmatrix} = \alpha^4 \begin{vmatrix} 0 & \alpha^5 \\ \alpha^5 & \alpha^4 \end{vmatrix} = \begin{vmatrix} 0 & \alpha^2 \\ \alpha^2 & \alpha \end{vmatrix}.$$

Таким образом, получим

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} 0 & \alpha^2 \\ \alpha^2 & \alpha \end{bmatrix} \begin{bmatrix} 0 \\ \alpha^6 \end{bmatrix},$$

$\sigma_2 = 0 + \alpha^2 \cdot \alpha^6 = \alpha^8 = \alpha$, $\sigma_1 = 0 + \alpha \cdot \alpha^6 = \alpha^7 = 1$.

Используя найденные значения коэффициентов $\sigma_1 = 1$ и $\sigma_2 = \alpha$, а также спектральные компоненты вектора ошибок $E_0 = \alpha^4$,$E_1 = \alpha^5$,$E_2 = 0$ и

$E_3 = \alpha^6$ вычислим по формуле (5.26) остальные спектральные компоненты E_4, E_5 и E_6.

$E_4 = \sigma_1 E_3 + \sigma_2 E_2 = 1\cdot\alpha^6 + \alpha\cdot 0 = \alpha^6$,

$E_5 = \sigma_1 E_4 + \sigma_2 E_3 = 1\cdot\alpha^6 + \alpha\cdot\alpha^6 = \alpha^2$,

$E_6 = \sigma_1 E_5 + \sigma_2 E_4 = 1\cdot\alpha^2 + \alpha\cdot\alpha^6 = \alpha^6$.

Применяя к полученному спектральному вектору $E = \{\alpha^4, \alpha^5, 0, \alpha^6, \alpha^6, \alpha^2, \alpha^6\}$ обратное преобразование Фурье $e_i = E(\alpha^{-i})$ для многочлена $E(x) = \alpha^6x^6+\alpha^2x^5+\alpha^6x^4+\alpha^6x^3+\alpha^5x+\alpha^4$получим

$e_0 = E(1) = \alpha^6+\alpha^2+\alpha^6+\alpha^6+\alpha^5+\alpha^4 = 0$,

$e_1 = E(\alpha^{-1}) = \alpha^6\alpha^{-6}+\alpha^2\alpha^{-5}+\alpha^6\alpha^{-4}+\alpha^6\alpha^{-3}+\alpha^5\alpha^{-1}+\alpha^4 = 1+\alpha^4+\alpha^2+\alpha^3+\alpha^4+\alpha^4 = 0$,

$\boldsymbol{e_2 = E(\alpha^{-2}) = \alpha^{-6}+\alpha^{-8}+\alpha^{-2}+1+\alpha^3+\alpha^4 = \alpha^1+\alpha^6+\alpha^5+1+\alpha^3+\alpha^4 = \alpha^2,}$

$e_3 = E(\alpha^{-3}) = \alpha^{-12}+\alpha^{-13}+\alpha^{-6}+\alpha^{-3}+\alpha^2+\alpha^4 = \alpha^2+\alpha^1+\alpha^1+\alpha^4+\alpha^2+\alpha^4 = 0$,

$e_4 = E(\alpha^{-4}) = \alpha^{-18}+\alpha^{-18}+\alpha^{-10}+\alpha^{-6}+\alpha^1+\alpha^4 = \alpha^3+\alpha^3+\alpha^4+\alpha^1+\alpha^1+\alpha^4 = 0$,

$e_5 = E(\alpha^{-5}) = \alpha^{-24}+\alpha^{-23}+\alpha^{-14}+\alpha^{-9}+\alpha^0+\alpha^4 = \alpha^4+\alpha^5+1+\alpha^5+1+\alpha^4 = 0$,

$\boldsymbol{e_6 = E(\alpha^{-6}) = \alpha^{-30}+\alpha^{-28}+\alpha^{-18}+\alpha^{-12}+\alpha^{-1}+\alpha^4 = \alpha^5+1+\alpha^3+\alpha^2+\alpha^6+\alpha^4 = \alpha.}$

В результате получен полином ошибок $e(x) = \alpha x^6 +\alpha^2x^2$, что соответствует действительному расположению и величинам ошибок.

Пример 5.3. Принятая кодовая последовательность с ошибками имеет вид $f(x) = \alpha x^5 + \alpha^6x^4 + \alpha^4$, при этом вектор ошибок равен $e(x) = \alpha^4x^6 +\alpha^5x$. Необходимо осуществить декодирование кодовой последовательности $f(x)$ в частотной области используя преобразование Фурье. Корнями порождающего полинома $g(x)$являются элементы поля α^4, α^5, α^6, α^0(нули кода для $b = 4$).

Синдромные компоненты принятой последовательности

$S_1 = f(\alpha^4) = \alpha^{21}+\alpha^{22}+\alpha^4 = 1+\alpha+\alpha^4 = \alpha^6$,

$S_2 = f(\alpha^5) = \alpha^{26}+\alpha^{26}+\alpha^4 = \alpha^4$,

$S_3 = f(\alpha^6) = \alpha^{31}+\alpha^{30}+\alpha^4 = \alpha^3+\alpha^2+\alpha^4 = 1$,

$S_4 = f(\alpha^0) = \alpha+\alpha^6+\alpha^4 = 1$.

Спектральные компоненты вектора ошибок (5.22) равны

$E_4 = S_1 = \alpha^6$,

$E_5 = S_2 = \alpha^4$,

$E_6 = S_3 = 1$,

$E_0 = S_4 = 1$.

Необходимо найти остальные спектральные компоненты E_1, E_2, E_3 .

Предположим, что при передаче кодовой последовательности произошло две ошибки ($\mu = 2$). Строим матрицу размером (2 x 2) M_2.

$$M_2 = \begin{vmatrix} E_4 & E_5 \\ E_5 & E_6 \end{vmatrix} = \begin{vmatrix} \alpha^6 & \alpha^4 \\ \alpha^4 & 1 \end{vmatrix}.$$

Определитель матрицы det $M_2 = \alpha^6 \cdot 1 + \alpha^4 \cdot \alpha^4 = \alpha^6 + \alpha^1 = \alpha^5$не равен нулю. Таким образом, произошло две ошибки и $\nu = 2$.

Для нахождения коэффициентов σ_1, σ_2 полинома локаторов ошибок $\sigma(x)$ решим систему уравнений

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = M_2^{-1} \begin{bmatrix} E_6 \\ E_0 \end{bmatrix}.$$

Миноры матрицы M_2 равны

$a_{11} = 1$, $a_{12} = \alpha^4$, $a_{21} = \alpha^4$, $a_{22} = \alpha^6$.

Обратная матрица

$$M_2^{-1} = \frac{1}{|M|} \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = \frac{1}{\alpha^5} \begin{vmatrix} 1 & \alpha^4 \\ \alpha^4 & \alpha^6 \end{vmatrix} = \alpha^2 \begin{vmatrix} 1 & \alpha^4 \\ \alpha^4 & \alpha^6 \end{vmatrix} = \begin{vmatrix} \alpha^2 & \alpha^6 \\ \alpha^6 & \alpha \end{vmatrix}.$$

Найдем коэффициенты полинома локаторов ошибок

$$\begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \alpha^2 & \alpha^6 \\ \alpha^6 & \alpha \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$\sigma_2 = \alpha^2 \cdot + \alpha^6 = 1$, $\sigma_1 = \alpha^6 + \cdot \alpha = \alpha^5$.

Спектральные компоненты вектора ошибок

$E_1 = \sigma_1 E_0 + \sigma_2 E_6 = \alpha^5 \cdot 1 + 1 \cdot 1 = \alpha^4$,

$E_2 = \sigma_1 E_1 + \sigma_2 E_0 = \alpha^5 \cdot \alpha^4 + 1 \cdot 1 = \alpha^6$,

$E_3 = \sigma_1 E_2 + \sigma_2 E_1 = \alpha^5 \cdot \alpha^6 + 1 \cdot \alpha^4 = 0$.

Таким образом, спектральные компоненты вектора ошибок равны $E = \{1, \alpha^4, \alpha^6, 0, \alpha^6, \alpha^4, 1\}$ или в виде полинома $E(x) = x^6 + \alpha^4 x^5 + \alpha^6 x^4 + \alpha^6 x^2 + \alpha^4 x + 1$, Тогда для получения вектора ошибок выполним обратное преобразование Фурье $e_i = E(\alpha^{-i})$

$e_0 = E(1) = 1 + \alpha^4 + \alpha^6 + \alpha^6 + \alpha^4 + 1 = 0$,

$\boldsymbol{e_1 = E(\alpha^{-1}) = E(\alpha^6) = \alpha^{36} + \alpha^{34} + \alpha^{30} + \alpha^{18} + \alpha^{10} + 1 = \alpha^1 + \alpha^6 + \alpha^2 + \alpha^4 + \alpha^3 + 1 = \alpha^5}$.

$e_2 = E(\alpha^{-2}) = E(\alpha^5) = \alpha^{30} + \alpha^{29} + \alpha^{26} + \alpha^{16} + \alpha^9 + 1 = \alpha^2 + \alpha^1 + \alpha^5 + \alpha^2 + \alpha^2 + 1 = 0$,

$e_3 = E(\alpha^{-3}) = E(\alpha^4) = \alpha^{24} + \alpha^{24} + \alpha^{22} + \alpha^{14} + \alpha^8 + 1 = \alpha^3 + \alpha^3 + \alpha^1 + 1 + \alpha^1 + 1 = 0$,

$e_4 = E(\alpha^{-4}) = E(\alpha^3) = \alpha^{18} + \alpha^{19} + \alpha^{18} + \alpha^{12} + \alpha^7 + 1 = \alpha^4 + \alpha^5 + \alpha^4 + \alpha^5 + 1 + 1 = 0$,

$e_5 = E(\alpha^{-5}) = E(\alpha^2) = \alpha^{12} + \alpha^{14} + \alpha^{14} + \alpha^{10} + \alpha^6 + 1 = \alpha^5 + \alpha^3 + \alpha^6 + 1 = 0$,

$\boldsymbol{e_6 = E(\alpha^{-6}) = E(\alpha^1) = \alpha^6 + \alpha^9 + \alpha^{10} + \alpha^8 + \alpha^5 + 1 = \alpha^6 + \alpha^2 + \alpha^3 + \alpha + \alpha^5 + 1 = \alpha^4}$,

В результате получен полином ошибок $e(x) = \alpha^4 x^6 + \alpha^5 x$, что соответствует действительному расположению и величинам ошибок.

5.3.2 Алгоритм Берлекэмпа-Месси в частотной области

С практической точки зрения рациональный путь нахождения значений количества ошибок v и коэффициентов $\sigma_1, \sigma_2, \ldots, \sigma_v$ рекурсии предоставляет *алгоритм Берлекэмпа-Месси*, основой которого служит структура обычного регистра сдвига с линейной обратной связью, генерирующего линейную рекуррентную последовательность $E_0, E_1, \ldots, E_{n-1}$, представленная на рис. 5.2. При начальной загрузке $E_0, E_1, \ldots, E_{v-1}$ данная схема формирует на выходе остальные элементы $E_v, E_{v+1}, \ldots, E_{n-1}$ согласно соотношению

$$E_k = \sum_{i=1}^{v} \sigma_i E_{k-i}, \; k = v, v+1, \ldots, n-1. \qquad (5.27)$$

Алгоритм Берлекэмпа-Месси представляет собой итеративную процедуру, начинающуюся со значения $v = 1$ и начальной загрузке E_0. На j-м шаге итерации алгоритм берет следующий по порядку компонент синдрома E_j и строит регистр сдвига минимальной длины, генерирующий последовательность $E_0, E_1, \ldots, E_j$. Если регистр, построенный на предыдущем шаге итераций, способен сформировать указанную последовательность, алгоритм прямо переходит к следующему шагу итераций. В противном случае, перед переходом осуществляется перерасчет коэффициентов и, если необходимо, увеличивается значение памяти.

Процедура декодирования завершается правильным декодированием, если фактическое число ошибок не превышает исправляющей способности кода.

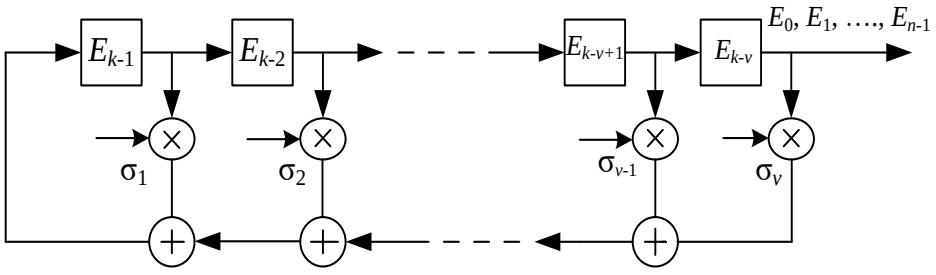

Рисунок 5.2 – Регистр сдвига с линейной обратной связью

На рисунке 5.3 приведена блок-схема реализации описанной процедуры декодирования. Декодирование начинается с вычисления преобразования Фурье для принятого кодового вектора $f(x)$

$$F_j = \sum_{i=0}^{n-1} \alpha^{ij} f_i \, , j = 0, 1, \ldots, n-1. \qquad (5.28)$$

Полученные компоненты преобразования Фурье F_j задают начальные значения для компонент синдрома S

$$S_j = F_{j+b-1}, j = 0, 1, \ldots, n-1. \qquad (5.29)$$

Далее выполняется алгоритм Берлекэмпа-Месси для нахождения многочлена локаторов ошибок $\sigma(x)$ с учетом $2t$известных компонент вектора E. Остальные компоненты вектора $S=\{S_{2v+1}, S_{2v+2}, \ldots, S_{n-1}\}$ можно получить рекуррентным продолжением, а именно используя уравнение для свертки (5.27), вычислить S_{2v+1} по уже известным компонентам S и σ, затем найтиS_{2v+2}и т.д. Это вычисление может быть выполнено с помощью регистра сдвига с линейной обратной связью, весовые множители, в отводах которого совпадают с компонентами вектора σ, а начальное состояние задается компонентами синдрома $S_1, S_2, \ldots, S_{2t}$. Это позволяет вычислить компоненты S_j для всех j, а затем исправить ошибки $F_j = V_j + E_j$ и с помощью обратного преобразования Фурье найти исправленную кодовую последовательность.

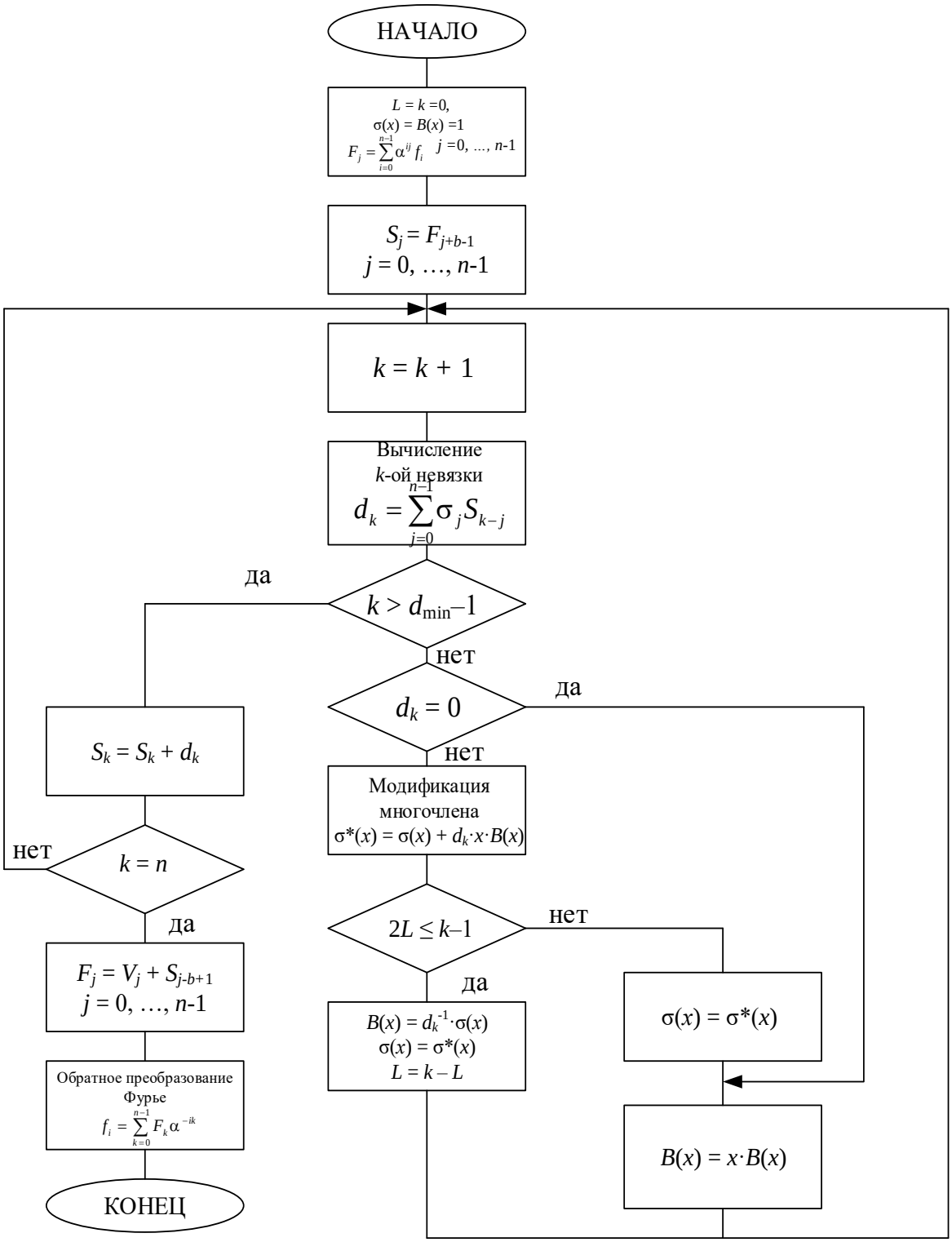

Рисунок 5.3 – Частотный декодер Берлекэмпа-Месси

Декодирование двоичных БЧХ кодов и кодов Рида-Соломона может быть организовано различными способами, с учетом того каким образом

работает кодер и декодер (во временной или частотной области). Схема кодера и декодера, работа которых осуществляется во временной области была рассмотрена ранее и представлена на рисунке 3.6. Однако в каналах передачи работа системы кодирования может быть комбинированной (временной кодер и частотный декодер, или наоборот). На рис. 5.4 представлена схема защиты информации, в которой кодирование осуществляется в частотной области и соответственно, как только вычислен исправленный вектор, декодирование закончено.

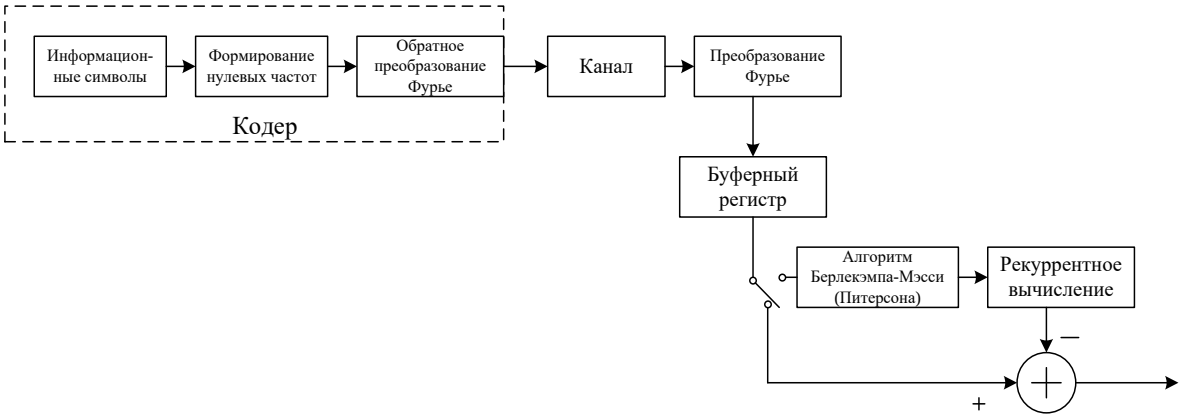

Рисунок 5.4 – Схема частотного кодера/декодера для БЧХ кодов

Если работа кодера организована во временной области, то декодирование может быть осуществлено в частотной области и с помощью рекуррентного вычисления находится конфигурация ошибок. При этом окончательное исправление ошибок можно производить как во временной, так и в частотной области. На рис. 5.5 представлена схема смешанной реализации временного кодера и частотного декодера, который реализован с помощью алгоритма Берлекэмпа-Месси (рис. 5.3) и исправление ошибок осуществляется во временной области.

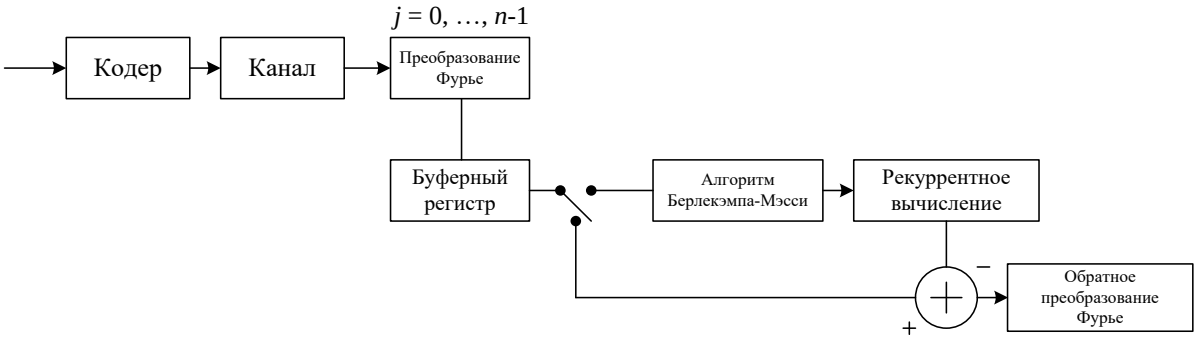

Рисунок 5.5 – Схема временного кодера и частотного декодера для БЧХ кодов

На рис. 5.6 представлена схема смешанной реализации временного кодера и частотного декодера, в которой преобразование Фурье аналогично процедуре вычисления синдромных компонент, а обратное преобразование Фурье аналогично процедуре Ченя. При этом алгоритм Форни, используемый во временном декодере, заменяется рекуррентным вычислением.

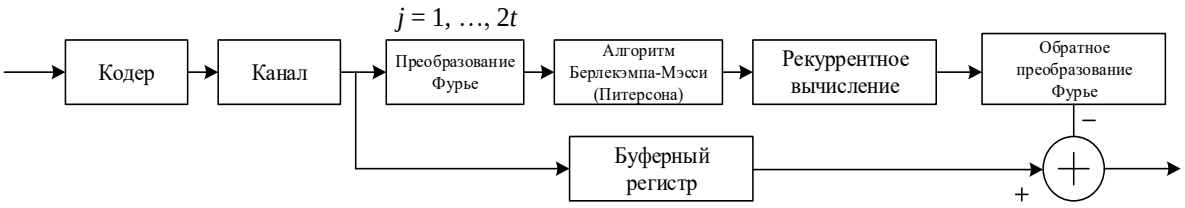

Рисунок 5.6–Схема смешанной реализации временного кодера и частотного декодера для БЧХ кодов

Выбор одной из представленных схем осуществления кодирования и декодирования определяется возможными методами вычисления преобразования Фурье и способа реализации декодера. При этом схема может быть так модернизирована, чтобы вычислять только часть спектральных компонент, как например, на рис. 5.6.

5.3.3 Алгоритм Евклида в частотной области

Для решения ключевого уравнения (5.27) возможно применение традиционного алгоритма Евклида, рассмотренного в 4.6.2, с целью нахождения вектора искажений. В случае декодирования в частотной области найденный многочлен локаторов ошибок $\sigma(x)$ и начальные синдромы S_1, S_2, …, S_{2t} используются для вычисления компонент преобразования Фурье E_j вектора искажений.

Начальные значения для v компонент вектора искажений E_j находятся с помощью синдромных компонент S_j

$$E_{j+b-1} = S_j, j = 1, \ldots, 2t. \tag{5.30}$$

Многочлен $\sigma(x)$ указывает отводы регистра сдвига с обратными связями, с помощью которых можно получить преобразование вектора ошибок, который удовлетворяет условию (5.27) для заданного v-компонентного блока.

Таким образом, рекуррентная процедура (5.27) дает преобразование вектора искажений в виде компонент E_0, E_1, …,E_{n-1}. Далее для получения векторной последовательности e_0, e_2, …, e_{n-1} осуществляется обратное преобразование Фурье вектора E

$$e_i = \frac{1}{n}\sum_{j=0}^{n-1} \alpha^{-ij} E_j, j = 0, \ldots, n-1. \tag{5.31}$$

Далее формируется с помощью коэффициентов $e_0, e_1, \ldots, e_{n-1}$ многочлен ошибок $e(x)$, с помощью которого исправляются ошибки в принятой последовательности $v(x) = f(x) + e(x)$. Блок-схема декодирования для кодов БЧХ, основанный на алгоритме Евклида в частотной области представлена на рисунке 5.7.

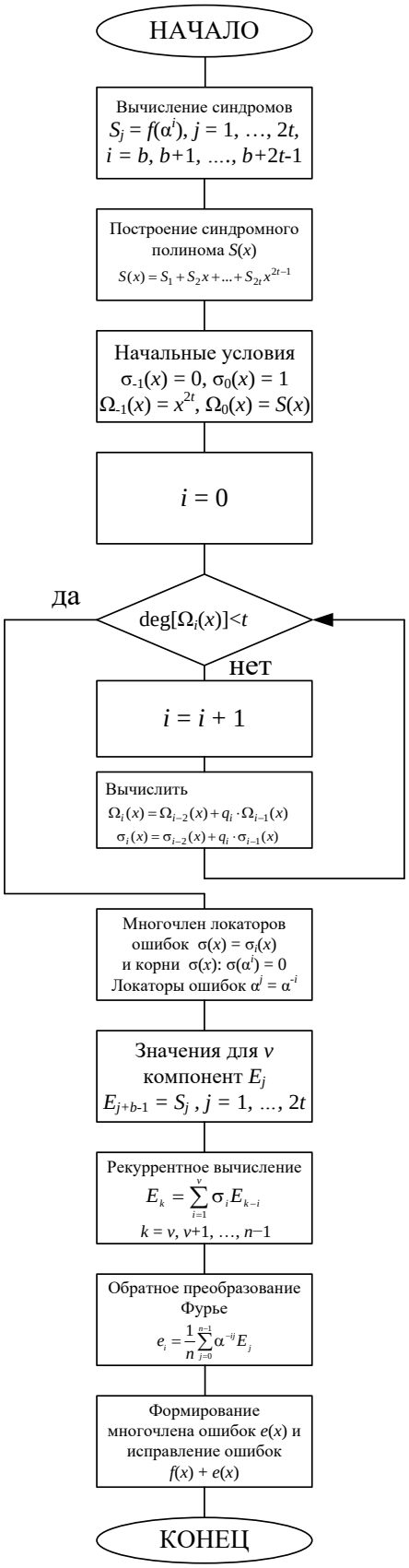

Рисунок 5.7 – Алгоритм Евклида для БЧХ кодов в частотной области

Пример 5.4. Рассмотрим РС код (7, 3, 5) из примера 4.3, использующий конструкцию поля $GF(2^3)$ и способного исправлять любые ошибки кратности один или два ($t = 2$). Корнями порождающего полинома $g(x) = x^4+\alpha^3x^3+x^2+\alpha x+\alpha^3$являются элементы поля α^1, α^2, α^3, α^4(нули кода для $b = 1$).Кодовая последовательность $v(x) = g(x)\,u(x) = (x^4+\alpha^3x^3+x^2+\alpha x+\alpha^3)\cdot$ $\cdot(\alpha^4x^2+\alpha^2x+\alpha) = \alpha^4x^6+\alpha^6x^5+\alpha^3x^4+\alpha^6x^3+\alpha^3x+\alpha^4$. Принятая кодовая последовательность с ошибками имеет вид $f(x) = \alpha^6x^5 + \alpha^3x^4 + \alpha^3x + \alpha^4$, при этом вектор ошибок равен $e(x) = \alpha^4x^6 +\alpha^6x^3$. Необходимо осуществить декодирование кодовой последовательности $f(x)$ в частотной области используя преобразование Фурье.

Синдромные компоненты для последовательности $f(x)$ равны

$S_1 = \alpha^5$, $S_2 = \alpha^5$, $S_3 = 0$, $S_4 = \alpha^6$.

Далее работает алгоритм Евклида для нахождения полинома локаторов ошибок.

Установить начальные значения.

$$\sigma_{-1}(x) = 0,\ \sigma_0(x) = 1,\ \Omega_{-1}(x) = x^4,\ \Omega_0(x) = S(x) = \alpha^5 + \alpha^3x + \alpha^5x^3$$

Выполнить деление $q_1(x) = \left[\dfrac{\Omega_{-1}(x)}{\Omega_0(x)}\right] = \left[\dfrac{x^4}{\alpha^5 + \alpha^3x + \alpha^5x^3}\right] = \alpha^2x$.

Вычислить значения $\sigma_1(x)$, $\Omega_1(x)$.

$$\sigma_1(x) = \sigma_{-1}(x) + q_1\cdot\sigma_0(x) = 0 + \alpha^2x\cdot 1 = \alpha^2x,$$

$$\Omega_1(x) = \Omega_{-1}(x) + q_1\cdot\Omega_0(x) = x^4 + \alpha^2x\cdot(\alpha^5 + \alpha^3x + \alpha^5x^3) =$$

$$x^4 + \alpha^7x^4 + \alpha^5x^2 + x = x + \alpha^5x^2.$$

Так как условие $\deg[\Omega_1(x)] < t$ не выполняется и степень полинома $\Omega_1(x)$ больше чем количество исправляемых ошибок кодом $\Omega_1(x)<2 \rightarrow (2<2)$, то выполняется деление

$$q_2(x) = \left[\frac{\Omega_0(x)}{\Omega_1(x)}\right] = \left[\frac{\alpha^5 + \alpha^3x + \alpha^5x^3}{\alpha^5x^2 + x}\right] = x + \alpha^2.$$

Вычислить значения $\sigma_2(x)$, $\Omega_2(x)$.

$$\sigma_2(x) = \sigma_0(x) + q_2\cdot\sigma_1(x) = 1 + (x + \alpha^2)\cdot\alpha^2x = 1 + \alpha^2x^2 + \alpha^4x,$$

$$\Omega_2(x) = \Omega_0(x) + q_2\cdot\Omega_1(x) = (\alpha^5 + \alpha^3x + \alpha^5x^3) + (x + \alpha^2)\cdot(\alpha^5x^2 + x) =$$

$$= \alpha^5x^3 + \alpha^3x + \alpha^5 + \alpha^5x^3 + x^2 + x^2 + \alpha^2x = \alpha^5x + \alpha^5.$$

Условие $\deg[\Omega_2(x)] < t$ выполняется, т.к. степень полинома $\Omega_2(x)$ меньше количества исправляемых ошибок $\Omega_2(x)<2\rightarrow(1<2)$, таким образом

процедура декодирования завершается и получен многочлен локаторов ошибок $\sigma(x) = 1 + \alpha^4 x + \alpha^2 x^2$.

Вычислим компоненты вектора преобразования E используя (5.30) и синдромные компоненты

$E_1 = S_1 = \alpha^5$,

$E_2 = S_2 = \alpha^3$,

$E_3 = S_3 = 0$,

$E_4 = S_4 = \alpha^5$.

Для нахождения остальных компонент вектора E_5, E_6 и $E_0 = E_7$ используем рекуррентное продолжение (5.27)

$$E_k = \sum_{i=1}^{2} \sigma_i E_{k-i} = \sigma_1 E_{k-1} + \sigma_2 E_{k-2},\ k = 5, 6, 7.$$

$E_5 = \sigma_1 E_4 + \sigma_2 E_3 = \alpha^4\alpha^5 + \alpha^2 \cdot 0 = \alpha^2$;

$E_6 = \sigma_1 E_5 + \sigma_2 E_4 = \alpha^4\alpha^2 + \alpha^2 \cdot \alpha^5 = \alpha^2$;

$E_0 = E_7 = \sigma_1 E_6 + \sigma_2 E_5 = \alpha^4\alpha^2 + \alpha^2 \cdot \alpha^2 = \alpha^3$.

Таким образом, спектральные компоненты вектора ошибок равны $E = \{\alpha^3, \alpha^5, \alpha^3, 0, \alpha^5, \alpha^2, \alpha^2\}$ или $E(x) = \alpha^3+\alpha^5x+\alpha^3x^2+\alpha^5x^4+\alpha^2x^5+\alpha^2x^6$. Для нахождения вектора ошибок $e(x)$выполним обратное преобразование Фурье $e_i = E(\alpha^{-i})$, где $E(x) = \sum_{j=0}^{n-1} E_j x^j = \alpha^2x^6+\alpha^2x^5+\alpha^5x^4+\alpha^3x^2+\alpha^5x+\alpha^3$

$e_0 = E(1) = \alpha^3+\alpha^5+\alpha^3+\alpha^5+\alpha^2+\alpha^2 = 0$,

$e_1 = E(\alpha^{-1}) = E(\alpha^1) = \alpha^8+\alpha^7+\alpha^9+\alpha^5+\alpha^6+\alpha^3 = \alpha^1+1+\alpha^2+\alpha^5+\alpha^6+\alpha^3 = 0$,

$e_2 = E(\alpha^{-2}) = E(\alpha^5) = \alpha^{32}+\alpha^{27}+\alpha^{25}+\alpha^{13}+\alpha^{10}+\alpha^3 = \alpha^4+\alpha^6+\alpha^4+\alpha^6+\alpha^3+\alpha^3 = 0$,

$\boldsymbol{e_3 = E(\alpha^{-3}) = E(\alpha^4) = \alpha^{26}+\alpha^{22}+\alpha^{21}+\alpha^{11}+\alpha^9+\alpha^3 = \alpha^5+\alpha^1+1+\alpha^4+\alpha^2+\alpha^3 = \alpha^6}$,

$e_4 = E(\alpha^{-4}) = E(\alpha^3) = \alpha^{20}+\alpha^{17}+\alpha^{17}+\alpha^9+\alpha^8+\alpha^3 = \alpha^6+\alpha^3+\alpha^3+\alpha^2+\alpha^1+\alpha^3 = 0$,

$e_5 = E(\alpha^{-5}) = E(\alpha^2) = \alpha^{14}+\alpha^{12}+\alpha^{13}+\alpha^7+\alpha^7+\alpha^3 = 1+\alpha^5+\alpha^6+1+1+\alpha^3 = 0$,

$\boldsymbol{e_6 = E(\alpha^{-6}) = E(\alpha^1) = \alpha^8+\alpha^7+\alpha^9+\alpha^5+\alpha^6+\alpha^3 = \alpha^1+1+\alpha^2+\alpha^5+\alpha^6+\alpha^3 = \alpha^4}$.

В результате получен полином ошибок $e(x) = \alpha^4x^6 + \alpha^6x^3$, что соответствует действительному расположению и величинам ошибок.

5.3.4 Исправление стираний и ошибок в частотной области

Если кодовое расстояние кода равно d и принятое слово содержит s символов, соответствующих стираниям, то при декодировании s «стертых» символов могут быть проигнорированы. Однако даже если игнорировать эти символы, то два кодовых слова будут отличаться друг от друга по

меньшей мере в $d-s$ оставшихся позициях, что позволяет в дополнение к стираниям исправлять по меньшей мере

$$t = \left[\frac{d - s - 1}{2} \right]$$

ошибок в канале. Таким образом, код с минимальным расстоянием d позволяет декодировать любую конфигурацию, содержащую t ошибок s стираний, при условии, что

$$d \geq 2t + s + 1.$$

Как было изложено выше класс кодов БЧХ допускает многие сильные алгоритмы декодирования. Все они могут быть обобщены на случай декодирования ошибок и стираний в частотной области.

Рассмотрим процедуру декодирования БЧХ кодов с ошибками и стиранием с помощью алгоритма Берлекэмпа-Месси в частотной области.

Задача состоит в декодировании вектора $f = v + e + w$ и вычисления модифицированного вектора ошибок, который содержит значения ошибок и значения стираний

$$e(x) = \sum_{l=1}^{t} e_{j_l} x^{j_l} + \sum_{m=1}^{s} w_{j_m} x^{j_m}, \qquad (5.32)$$

где $e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + ... + e_{j_v} x^{j_v}$ – многочлен ошибок, $w(x) = w_{j_1} x^{j_1} + w_{j_2} x^{j_2} + \ldots + w_{j_s} x^{j_s}$ – многочлен стираний.

С помощью корней *многочлен локаторов стираний* $\upsilon(x)$, задаются позиции стертых символов, которые декодеру заранее известны. В частотной области преобразование Фурье многочлена локаторов стирания определяется с помощью локаторов стираний $y_l = \alpha^{j_l}$ для $l = 1, \ldots, s$

$$\upsilon(x) = \sum_{k=0}^{n-1} \upsilon_k x^k = \prod_{l=1}^{s} (1 - y_l x). \qquad (5.33)$$

Этот многочлен определен так, чтобы компоненты υ_i обратного преобразования вектора V были равны нулю, как только $w_i \neq 0$. Таким образом, имеем $\upsilon_i \cdot w_i = 0$ и в частотной области получим кодовый вектор с ошибками и стиранием

$$F = (V \cdot U) + E.$$

При этом произведение $V*U$ равно нулю в блоке длины $2t-s$, т.к. вектор V отличен от нуля только на блоке длины $s+1$, а вектор U имеет нулевые компоненты в блоке длины $2t$. Тогда модифицированный синдром с учетом ($T_j = F_j$) на этом блоке $2t$ равен

$$T_j = E_j.$$

Полином локаторов искажений (ошибок и стираний) $\tau(x) = \sigma(x)\cdot\upsilon(x)$ при обратном преобразовании Фурье содержит нуль в позиции каждой ошибки и каждого стираний $\tau_i = 0$, если $e_i \neq 0$ или $w_i \neq 0$. Таким образом, произведение преобразований Фурье вектора локаторов искажений и сумму векторов ошибок и стираний равен

$$\tau * (E + W) = 0,$$

и многочлен $\tau(x)$ отличен от нуля в блоке длины, не превышающей $t + s + 1$. Используя известные преобразования компоненты вектора τ и известные компоненты $E + W$ можно воспользоваться этим содержащим свертку уравнением для того, чтобы рекуррентно продолжить вектор $E + W$ до всех n компонент и тогда вычисляются значения компонент вектора кодового слова

$$F_i = V_i + (E_i + F_i).$$

Завершается процесс декодирование обратным преобразованием Фурье.

Для решения ключевого уравнения и вычисления многочлена локаторов ошибок используется интерактивный алгоритм Берлекэмпа-Месси в частотной области.

Для инициализации процесса декодирования используется полином локаторов стираний $\upsilon(x)$: $\sigma(x) = B(x) = \upsilon(x)$ и синдромные компоненты, которые находятся с помощью компонент преобразования Фурье принятого кодового многочлена $f(x)$

$$F_j = \sum_{i=0}^{n-1} \alpha^{ij} f_i, \; j = 0, \ldots, n-1. \qquad (5.34)$$

Тогда синдромный компонент равен $S_j = F_{j+b-1}$ для $j = 0, 1, \ldots, n-1$. Обозначим многочлен локаторов ошибок через многочлен искажений $\tau(x) = B(x) = \upsilon(x)$. При начальных условиях, даваемых многочленом $\upsilon(x)$,

алгоритм Берлекэмпа-Месси рекуррентно порождает многочлен локаторов искажений в соответствии с уравнением

$$\tau^{(k)}(x) = \tau^{(k-1)}(x) + d_k\, x B^{(k-1)}(x),$$

$$B^{(k)}(x) = x B^{(k-1)}(x) + d_k^{-1}\tau^{(k-1)}(x).$$

Невязка на каждом шаге итерации вычисляется по формуле

$$d_k = \sum_{j=0}^{n-1} \tau_j S_{k-j}. \qquad (5.35)$$

На рисунке 5.8 представлен алгоритм Берлекэмпа-Месси исправляющий ошибки и стирания в принятой кодовой последовательности. После инициализации работы декодера осуществляется вычисление многочлена локаторов стираний с помощью, показанной на рис. 5.8 специальной вспомогательной цепи, содержащей первые s итераций. Далее осуществляется вычисление многочлена стираний до тех пор пока k не превзойдет $2t$. При каждом стирании длина регистра сдвига L возрастает на единицу и в конце алгоритма происходит замена $k = k-$ sи $L = L-$s.После нахождения всех компонент вектора ошибок и стираний осуществляется обратное преобразование Фурье.

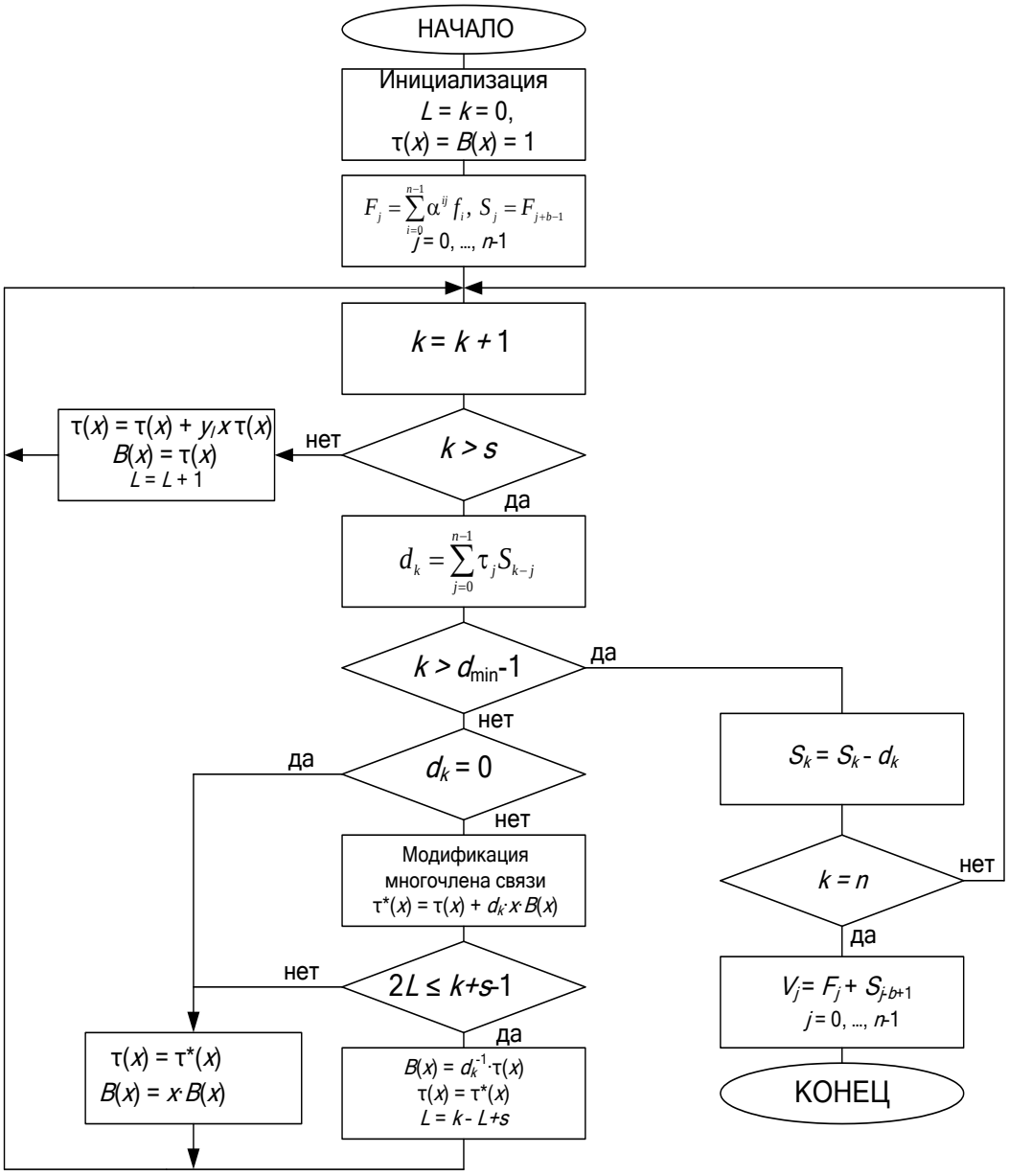

Рисунок 5.8 – Алгоритм Берлекэмпа-Месси в частотной области

Пример 5.5.Рассмотрим РС код (7, 3, 5) использующий конструкцию поля $GF(2^3)$ и способного исправлять однократные ошибки $t = 1$ и двукратные стирания $s = 2$, так как минимальное кодовое расстояние кода равно $d = 5$ и соответственно $d = 2t+s+1$. Корнями порождающего полинома $g(x) = x^4+\alpha^3x^3+x^2+\alpha x+\alpha^3$ являются элементы поля α^1, α^2, α^3, α^4 (нули кода для значения $b = 1$). Кодовая последовательность $v(x) = g(x)u(x) = (x^4+\alpha^3x^3+x^2+\alpha x+\alpha^3)\cdot(\alpha^4x^2+\alpha^2x+\alpha) = \alpha^4x^6+\alpha^6x^5+\alpha^3x^4+\alpha^6x^3+\alpha^3x+\alpha^4$. Принятая кодовая последовательность с ошибками имеет вид $f(x) = \alpha^6x^5 + \alpha\ ^6x^3 + \alpha^4$, при этом вектор ошибок равен $e(x) = \alpha^6x^5 +\alpha^6x^3+\alpha^4$.

Необходимо осуществить декодирование кодовой последовательности $f(x)$ в частотной области используя преобразование Фурье.

Декодеру известно расположение стираний в кодовой последовательности – это позиции 4 и 1. Таким образом, локаторы стираний равны $y_1 = \alpha^4$ и $y_2 = \alpha^1$.

Для инициализации алгоритма Берлекэмпа-Месси необходимо вычислить компоненты преобразования Фурье для принятого многочлена $f(x)$ используя (5.34)

$$F_j = \sum_{i=0}^{6} \alpha^{ij} f_i, \quad j = 0, \ldots, 6.$$

$F_0 = f(\alpha^0) = \alpha^6+\alpha^6+\alpha^4 = \alpha^4$;
$F_1 = f(\alpha^1) = \alpha^{11}+\alpha^9+\alpha^4 = \alpha^4+\alpha^2+\alpha^4 = \alpha^2$;
$F_2 = f(\alpha^2) = \alpha^{16}+\alpha^{12}+\alpha^4 = \alpha^2+\alpha^5+\alpha^4 = \alpha^6$;
$F_3 = f(\alpha^3) = \alpha^{21}+\alpha^{15}+\alpha^4 = 1+\alpha+\alpha^4 = \alpha^6$;
$F_4 = f(\alpha^4) = \alpha^{26}+\alpha^{18}+\alpha^4 = \alpha^5+\alpha^4+\alpha^4 = \alpha^5$;
$F_5 = f(\alpha^5) = \alpha^{31}+\alpha^{21}+\alpha^4 = \alpha^3+1+\alpha^4 = \alpha^2$;
$F_6 = f(\alpha^6) = \alpha^{36}+\alpha^{24}+\alpha^4 = \alpha+\alpha^3+\alpha^4 = \alpha^5$.

Для $b = 1$ синдромные компоненты равны $S_j = F_j$

$\boldsymbol{S_0 = \alpha^4; S_1 = \alpha^2; S_2 = \alpha^6; S_3 = \alpha^6; S_4 = \alpha^5; S_5 = \alpha^2; S_6 = \alpha^5.}$

Инициализация алгоритма

$L = k = 0$; $\tau(x) = B(x) = 1$.

Шаг 1. $k = 1$.

$k > s \rightarrow 1 > 2$? – нет.

$\tau(x) = 1+\alpha^4 x \cdot 1 = 1+\alpha^4 x$;

$B(x) = \tau(x) = 1+\alpha^4 x \cdot 1 = 1+\alpha^4 x$;

$L = L + 1 = 1$.

Шаг 2. $k = 2$.

$k > s \rightarrow 2 > 2$? – нет

$\tau(x) = 1+\alpha^4 x+\alpha^1 x \cdot (1+\alpha^4 x) = 1+\alpha^4 x+ \alpha^1 x+\alpha^5 x^2 = \alpha^5 x^2 + \alpha^2 x+1$;

$B(x) = \tau(x) = \alpha^5 x^2 + \alpha^2 x+1$;

$L = L + 1 = 2$.

Шаг3. $k = 3$.

$k > s \rightarrow 3 > 2$? – да;

$$d_3 = \sum_{j=0}^{6} \tau_j S_{3-j} = \tau_0 S_3 + \tau_1 S_2 + \tau_2 S_1 = 1 \cdot \alpha^6 + \alpha^2 \cdot \alpha^6 + \alpha^5 \cdot \alpha^2 = \alpha^6 + \alpha + 1 = \alpha^4.$$

$k > d-1 \rightarrow 3 > 4$? – нет;

$d_3 = 0$? – нет;

$\tau^*(x) = \tau(x) + d_3 xB(x) = \alpha^5 x^2 + \alpha^2 x + 1 + \alpha^4 x(\alpha^5 x^2 + \alpha^2 x + 1) = \alpha^5 x^2 + \alpha^2 x + 1 +$

$+ \alpha^2 x^3 + \alpha^6 x^2 + \alpha^4 x = \alpha^2 x^3 + \alpha x^2 + \alpha x + 1;$

$2L \le k + s - 1 \rightarrow 4 \le 4$ – да;

$B(x) = d_3^{-1}\tau(x) = \alpha^3(\alpha^5 x^2 + \alpha^2 x + 1) = \alpha x^2 + \alpha^5 x + \alpha^3;$

$\tau(x) = \tau^*(x) = \alpha^2 x^3 + \alpha x^2 + \alpha x + 1;$

$L = k - L + s = 3.$

Шаг4. $k = 4$.

$k > s \rightarrow 4 > 2$? – да;

$$d_4 = \sum_{j=0}^{6} \tau_j S_{4-j} = \tau_0 S_4 + \tau_1 S_3 + \tau_2 S_2 + \tau_3 S_1 = 1 \cdot \alpha^5 + \alpha \cdot \alpha^6 + \alpha \cdot \alpha^6 + \alpha^2 \cdot \alpha^2 =$$

$= \alpha^5 + 1 + 1 + \alpha^4 = 1;$

$k > d - 1 \rightarrow 4 > 4$? – нет;

$d_4 = 0$? – нет;

$\tau^*(x) = \tau(x) + d_4 xB(x) = (\alpha^2 x^3 + \alpha x^2 + \alpha x + 1) + x(\alpha x^2 + \alpha^5 x + \alpha^3) =$

$= \alpha^2 x^3 + \alpha x^2 + \alpha x + 1 + \alpha x^3 + \alpha^5 x^2 + \alpha^3 x = \alpha^4 x^3 + \alpha^6 x^2 + x + 1;$

$2L \le k + s - 1 \rightarrow 6 \le 5$ – нет;

$\tau(x) = \tau^*(x) = \alpha^4 x^3 + \alpha^6 x^2 + x + 1;$

$B(x) = xB(x) = x(\alpha x^2 + \alpha^5 x + \alpha^3) = \alpha x^3 + \alpha^5 x^2 + \alpha^3 x;$

Шаг 5. $k = 5$.

$k > s \rightarrow 5 > 2$? – да;

$$d_5 = \sum_{j=0}^{6} \tau_j S_{5-j} = \tau_0 S_5 + \tau_1 S_4 + \tau_2 S_3 + \tau_3 S_2 = 1 \cdot \alpha^2 + 1 \cdot \alpha^5 + \alpha^6 \cdot \alpha^6 + \alpha^4 \cdot \alpha^5 =$$

$= \alpha^2 + \alpha^5 + \alpha^5 + \alpha^3 = \alpha^5;$

$k > d - 1 \rightarrow 5 > 4$? – да;

$S_5 = S_5 + d_5 = \alpha^2 + \alpha^5 = \alpha^3;$

$k = n \rightarrow 5 = 7$ – нет.

Шаг 6. $k = 6$.

$k > s \rightarrow 6 > 2$? – да;

$$d_6 = \sum_{j=0}^{6} \tau_j S_{6-j} = \tau_0 S_6 + \tau_1 S_5 + \tau_2 S_4 + \tau_3 S_3 = 1 \cdot \alpha^5 + 1 \cdot \alpha^3 + \alpha^6 \cdot \alpha^5 + \alpha^4 \cdot \alpha^6 =$$

$= \alpha^5 + \alpha^3 + \alpha^4 + \alpha^3 = 1;$

$S_6 = S_6 + d_6 = \alpha^5 + 1 = \alpha^4;$

$k = n \rightarrow 6 = 7$ – нет.

Шаг 7. $k = 7$.

$k > s \rightarrow 7 > 2$? – да;

$$d_7 = \sum_{j=0}^{6} \tau_j S_{6-j} = \tau_0 S_7 + \tau_1 S_6 + \tau_2 S_5 + \tau_3 S_4 = \tau_0 S_0 + \tau_1 S_6 + \tau_2 S_5 + \tau_3 S_4 =$$

$$= 1 \cdot \alpha^4 + 1 \cdot \alpha^4 + \alpha^6 \cdot \alpha^3 + \alpha^4 \cdot \alpha^5 = \alpha^4 + \alpha^4 + \alpha^2 + \alpha^2 = 0;$$

$S_0 = S_0 + d_7 = \alpha^4 + 0 = \alpha^4$;

$k = n \rightarrow 7 = 7$ – да.

Таким образом, синдромные компоненты, вычисленные с помощью алгоритма Берлекэмпа-Месси равны

$\boldsymbol{S_0 = \alpha^4; S_1 = \alpha^2; S_2 = \alpha^6; S_3 = \alpha^6; S_4 = \alpha^5; S_5 = \alpha^3; S_6 = \alpha^4.}$

Вычислим компоненты преобразования Фурье кодовой последовательности $v(x)$

$$V_j = F_j + S_{j-b+1}, \text{ для b} = 1 \; V_j = F_j + S_j,$$

$V_0 = F_0 + S_0 = \alpha^4 + \alpha^4 = 0$;

$V_1 = F_1 + S_1 = \alpha^2 + \alpha^2 = 0$;

$V_2 = F_2 + S_2 = \alpha^6 + \alpha^6 = 0$;

$V_3 = F_3 + S_3 = \alpha^6 + \alpha^6 = 0$;

$V_4 = F_4 + S_4 = \alpha^5 + \alpha^5 = 0$;

$V_5 = F_5 + S_5 = \alpha^2 + \alpha^3 = \alpha^5$;

$V_6 = F_6 + S_6 = \alpha^5 + \alpha^4 = 1$.

Для нахождения кодовой последовательности $v(x)$ выполним обратное преобразование Фурье для компонент вектора V

$$v_i = \frac{1}{n} \sum_{j=0}^{6} \alpha^{-ij} V_j \,,$$

$v_0 = \alpha^0 \cdot V^5 + \alpha^0 \cdot V^6 = 1 \cdot \alpha^5 + 1 \cdot 1 = \alpha^4$;

$v_1 = \alpha^{-5} \cdot V^5 + \alpha^{-6} \cdot V^6 = \alpha^{-5} \cdot \alpha^5 + \alpha^{-6} \cdot 1 = 1 + \alpha = \alpha^3$;

$v_2 = \alpha^{-2 \cdot 5} \cdot V^5 + \alpha^{-2 \cdot 6} \cdot V^6 = \alpha^{-10} \cdot \alpha^5 + \alpha^{-12} \cdot 1 = \alpha^2 + \alpha^2 = 0$;

$v_3 = \alpha^{-3 \cdot 5} \cdot V^5 + \alpha^{-3 \cdot 6} \cdot V^6 = \alpha^{-15} \cdot \alpha^5 + \alpha^{-18} \cdot 1 = \alpha^4 + \alpha^2 = \alpha^6$;

$v_4 = \alpha^{-4 \cdot 5} \cdot V^5 + \alpha^{-4 \cdot 6} \cdot V^6 = \alpha^{-20} \cdot \alpha^5 + \alpha^{-24} \cdot 1 = \alpha^6 + \alpha^4 = \alpha^3$;

$v_5 = \alpha^{-5 \cdot 5} \cdot V^5 + \alpha^{-5 \cdot 6} \cdot V^6 = \alpha^{-25} \cdot \alpha^5 + \alpha^{-30} \cdot 1 = \alpha + \alpha^5 = \alpha^6$;

$v_6 = \alpha^{-6 \cdot 5} \cdot V^5 + \alpha^{-6 \cdot 6} \cdot V^6 = \alpha^{-30} \cdot \alpha^5 + \alpha^{-36} \cdot 1 = \alpha^3 + \alpha^6 = \alpha^4$.

Исправленная кодовая последовательность равна

$v(x) = \alpha^4 x^6 + \alpha^6 x^5 + \alpha^3 x^4 + \alpha^6 x^3 + \alpha^3 x + \alpha^4$.

Контрольные вопросы

1. Преобразование Фурье в поле Галуа.
2. Спектральное представление БЧХ кодов.
3. Условие сопряженности для преобразования Фурье.
4. Кодирование БЧХ кодов с помощью преобразования Фурье.
5. Кодирование кодов РС с помощью преобразования Фурье.
6. Построить двоичный БЧХ кодер(7, 4, 3), который способен исправлять однократные ошибки ($t = 1$) с минимальным расстоянием $d = 3$, задаваемый набором корней $\{\alpha^1, \alpha^2\}$и порождающим полиномом $g(x) = x^3 + x + 1$ в частотной области.
7. Для вектора $C\{1, 0, 0, \alpha^2, 0, \alpha, \alpha^4\}$ вычислить обратное преобразование Фурье.
8. Декодирование БЧХ кодов в частотной области.
9. Преобразование Фурье вектора ошибки.
10. Теорема о свертки и рекуррентное уравнение для вектора ошибки.
11. Декодирование БЧХ кодов в частотной области с помощью алгоритма Берлекэмпа-Месси.
12. Алгоритм Евклида для декодирования в частотной области.
13. Исправление стираний и ошибок БЧХ кодов в частотной области.
14. Алгоритм Питерсона-Горенстейна-Цирлера в частотной области.
15. Вычислить полином ошибок для РС кода длины $n = 7$, синдромные компоненты которого равны $S_1 = \alpha^4$, $S_2 = \alpha^2$, $S_3 = 0$, $S_4 = \alpha^6$. Значение коэффициентов многочлена локаторов ошибок равны $\sigma_1 = 1$, $\sigma_2 = \alpha$.

ПРИЛОЖЕНИЕ А

n	**Делители полинома $x^n + 1$**
7	$(x + 1)(x^3 + x^2 + 1)$
9	$(x + 1)(x^2 + x + 1)(x^6 + x^3 + 1)$
15	$(x + 1)(x^2 + x + 1)(x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1)$
17	$(x + 1)(x^8 + x^5 + x^4 + x^3 + x + 1)(x^8 + x^7 + x^6 + x^4 + x^2 + x + 1)$
21	$(x + 1)(x^2 + x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)(x^6 + x^4 + x^2 + x + 1)(x^6 + x^5 + x^4 + x^2 + 1)$
23	$(x + 1)(x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1)(x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1)$
25	$(x + 1)(x^4 + x^3 + x^2 + x + 1)(x^{20} + x^{15} + x^{10} + x^5 + 1)$
27	$(x + 1)(x^2 + x + 1)(x^6 + x^3 + 1)(x^{18} + x^9 + 1)$
31	$(x + 1)(x^5 + x^3 + 1)(x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1)(x^5 + x^4 + x^3 + x + 1)$x $x(x^5 + x^4 + x^2 + x + 1)(x^5 + x^3 + x^2 + x + 1)$

ПРИЛОЖЕНИЕ Б
НЕПРИВОДИМЫЕ ПОЛИНОМЫ

Степень	Многочлен	Двоичный вектор
1	$x + 1$	11
2	$x^2 + x + 1$	111
3	$x^3 + x + 1$ $x^3 + x^2 + 1$	1011 1101
4	x^4+x+1 x^4+x^3+1 $x^4+x^3+x^2+x+1$	10011 11001 11111
5	x^5+x^2+1 x^5+x^3+1 $x^5+x^3+x^2+x+1$ $x^5+x^4+x^2+x+1$ $x^5+x^4+x^3+x+1$ $x^5+x^4+x^3+x^2+x+1$	100101 101001 101111 110111 111011 111101
6	x^6+x+1 x^6+x^3+1 $x^6+x^4+x^2+x+1$ $x^6+x^4+x^3+x+1$ x^6+x^5+1 $x^6+x^5+x^2+x+1$ $x^6+x^5+x^3+x^2+1$ $x^6+x^5+x^4+x+1$ $x^6+x^5+x^4+x^2+1$	1000011 1001001 1010111 1011011 1100001 1100111 1101101 1110011 1110101
7	x^7+x+1 x^7+x^3+1 $x^7+x^3+x^2+x+1$ $x^7+x^4+x^3+x^2+1$ $x^7+x^5+x^2+x+1$ $x^7+x^5+x^3+x+1$ $x^7+x^6+x^3+x+1$ $x^7+x^6+x^4+x+1$	10000011 10001001 10001111 10011101 10100111 10101011 11001011 11010011
8	$x^8+x^4+x^3+x+1$ $x^8+x^4+x^3+x^2+1$ $x^8+x^5+x^3+x+1$ $x^8+x^5+x^3+x^2+1$ $x^8+x^6+x^5+x^2+1$ $x^8+x^7+x^3+x+1$ $x^8+x^7+x^5+x^3+1$	100011011 100011101 100101011 100101101 101100101 110001011 110101001
9	x^9+x+1 x^9+x^4+1 $x^9+x^4+x^2+x+1$ $x^9+x^4+x^3+x+1$ $x^9+x^5+x^4+x+1$ $x^9+x^6+x^5+x^2+1$	1000000011 1000010001 1000010111 1000011011 1000110011 1001100101
10	$x^{10}+x^3+1$	10000001001

ПРИЛОЖЕНИЕ В
ПОРОЖДАЮЩИЕ МНОГОЧЛЕНЫ ДЛЯ БЧХ КОДОВ

n	***k***	***t***	***r***	**Порождающий полином в восьмеричной форме**
7	4	1	3	13
15	11	1	4	23
	7	2	8	721
	5	3	10	2467
31	26	1	5	45
	21	2	10	3551
	16	3	15	107657
	11	5	20	5423325
	6	7	25	313365047
63	57	1	6	103
	51	2	12	12471
	45	3	18	1701317
	39	4	24	166623567
	36	5	27	1033500423
	30	6	33	1574641656547
	24	7	39	17323260404441
	18	10	45	1363026512351725
127	120	1	7	211
	113	2	14	41567
	106	3	21	11554743
	99	4	28	3447023271
	92	5	35	624730022327
	85	6	42	130704476332273
	78	7	49	26230002166130115
	71	9	56	6255010713253127753
	64	10	63	1206534025570773100045
255	247	1	8	435
	239	2	16	267543
	231	3	24	156720665
	223	4	32	75626641375
	215	5	40	2315754726421
	207	6	48	16176560567636227
	199	7	56	7633031270420722341
	191	8	64	2663470176115333714567
	187	9	68	527553135400013222363351
	179	10	76	22624710717340432416300455

ПРИЛОЖЕНИЕ Г
АРИФМЕТИЧЕСКИЕ ТАБЛИЦЫ ПОЛЕЙ ГАЛУА

Таблица Г.1 – Сложение в расширенном поле $GF(2^4)$ ($p(x) = x^4+x+1$)

+	**0**	**1**	$\boldsymbol{\alpha^1}$	$\boldsymbol{\alpha^2}$	$\boldsymbol{\alpha^3}$	$\boldsymbol{\alpha^4}$	$\boldsymbol{\alpha^5}$	$\boldsymbol{\alpha^6}$	$\boldsymbol{\alpha^7}$	$\boldsymbol{\alpha^8}$	$\boldsymbol{\alpha^9}$	$\boldsymbol{\alpha^{10}}$	$\boldsymbol{\alpha^{11}}$	$\boldsymbol{\alpha^{12}}$	$\boldsymbol{\alpha^{13}}$	$\boldsymbol{\alpha^{14}}$
0	0	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
1	1	0	α^4	α^8	α^{14}	α^1	α^{10}	α^{13}	α^9	α^2	α^7	α^5	α^{12}	α^{11}	α^6	α^3
$\boldsymbol{\alpha^1}$	α^1	α^4	0	α^5	α^9	1	α^2	α^{11}	α^{14}	α^{10}	α^3	α^8	α^6	α^{13}	α^{12}	α^7
$\boldsymbol{\alpha^2}$	α^2	α^8	α^5	0	α^6	α^{10}	α	α^3	α^{12}	1	α^{11}	α^4	α^9	α^7	α^{14}	α^{13}
$\boldsymbol{\alpha^3}$	α^3	α^{14}	α^9	α^6	0	α^7	α^{11}	α^2	α^4	α^{13}	α	α^{12}	α^5	α^{10}	α^8	1
$\boldsymbol{\alpha^4}$	α^4	α	1	α^{10}	α^7	0	α^8	α^{12}	α^3	α^5	α^{14}	α^2	α^{13}	α^6	α^{11}	α^9
$\boldsymbol{\alpha^5}$	α^5	α^{10}	α^2	α	α^{11}	α^8	0	α^9	α^{13}	α^4	α^6	1	α^3	α^{14}	α^7	α^{12}
$\boldsymbol{\alpha^6}$	α^6	α^{13}	α^{11}	α^3	α^2	α^{12}	α^9	0	α^{10}	α^{14}	α^5	α^7	α	α^4	1	α^8
$\boldsymbol{\alpha^7}$	α^7	α^9	α^{14}	α^{12}	α^4	α^3	α^{13}	α^{10}	0	α^{11}	1	α^6	α^8	α^2	α^5	α^1
$\boldsymbol{\alpha^8}$	α^8	α^2	α^{10}	1	α^{13}	α^5	α^4	α^{14}	α^{11}	0	α^{12}	α	α^7	α^9	α^3	α^6
$\boldsymbol{\alpha^9}$	α^9	α^7	α^3	α^{11}	α	α^{14}	α^6	α^5	1	α^{12}	0	α^{13}	α^2	α^8	α^{10}	α^4
$\boldsymbol{\alpha^{10}}$	α^{10}	α^5	α^8	α^4	α^{12}	α^2	1	α^7	α^6	α	α^{13}	0	α^{14}	α^3	α^9	α^{11}
$\boldsymbol{\alpha^{11}}$	α^{11}	α^{12}	α^6	α^9	α^5	α^{13}	α^3	α	α^8	α^7	α^2	α^{14}	0	1	α^4	α^{10}
$\boldsymbol{\alpha^{12}}$	α^{12}	α^{11}	α^{13}	α^7	α^{10}	α^6	α^{14}	α^4	α^2	α^9	α^8	α^3	1	0	α	α^5
$\boldsymbol{\alpha^{13}}$	α^{13}	α^6	α^{12}	α^{14}	α^8	α^{11}	α^7	1	α^5	α^3	α^{10}	α^9	α^4	A	0	α^2
$\boldsymbol{\alpha^{14}}$	α^{14}	α^3	α^7	α^{13}	1	α^9	α^{12}	α^8	α	α^6	α^4	α^{11}	α^{10}	α^5	α^2	0

Таблица Г.2 – Умножение в расширенном поле $GF(2^4)$ ($p(x) = x^4+x+1$)

*	0	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
$\boldsymbol{\alpha^1}$	0	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1
$\boldsymbol{\alpha^2}$	0	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1
$\boldsymbol{\alpha^3}$	0	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2
$\boldsymbol{\alpha^4}$	0	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3
$\boldsymbol{\alpha^5}$	0	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4
$\boldsymbol{\alpha^6}$	0	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5
$\boldsymbol{\alpha^7}$	0	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6
$\boldsymbol{\alpha^8}$	0	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7
$\boldsymbol{\alpha^9}$	0	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8
$\boldsymbol{\alpha^{10}}$	0	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9
$\boldsymbol{\alpha^{11}}$	0	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}
$\boldsymbol{\alpha^{12}}$	0	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}
$\boldsymbol{\alpha^{13}}$	0	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}
$\boldsymbol{\alpha^{14}}$	0	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}

Таблица Г.3 – Сложение в расширенном поле $GF(2^4)$ ($p(x) = x^4+x^3+1$)

+	**0**	**1**	**α^1**	**α^2**	**α^3**	**α^4**	**α^5**	**α^6**	**α^7**	**α^8**	**α^9**	**α^{10}**	**α^{11}**	**α^{12}**	**α^{13}**	**α^{14}**
0	0	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
1	1	0	α^{12}	α^9	α^4	α^3	α^{10}	α^8	α^{13}	α^6	α^2	α^5	α^{14}	α^1	α^7	α^{11}
α^1	α^1	α^{12}	0	α^{13}	α^{10}	α^5	α^4	α^{11}	α^9	α^{14}	α^7	α^3	α^6	1	α^2	α^8
α^2	α^2	α^9	α^{13}	0	α^{14}	α^{11}	α^6	α^5	α^{12}	α^{10}	1	α^8	α^4	α^7	α^1	α^3
α^3	α^3	α^4	α^{10}	α^{14}	0	1	α^{12}	α^7	α^6	α^{13}	α^{11}	α^1	α^9	α^5	α^8	α^2
α^4	α^4	α^3	α^5	α^{11}	1	0	α^1	α^{13}	α^8	α^7	α^{14}	α^{12}	α^2	α^{10}	α^6	α^9
α^5	α^5	α^{10}	α^4	α^6	α^{12}	α^1	0	α^2	α^{14}	α^9	α^8	1	α^{13}	α^3	α^{11}	α^7
α^6	α^6	α^8	α^{11}	α^5	α^7	α^{13}	α^2	0	α^3	1	α^{10}	α^9	α^1	α^{14}	α^4	α^{13}
α^7	α^7	α^{13}	α^9	α^{12}	α^6	α^8	α^{14}	α^3	0	α^4	α^1	α^{11}	α^{10}	α^2	1	α^5
α^8	α^8	α^6	α^{14}	α^{10}	α^{13}	α^7	α^9	1	α^4	0	α^5	α^2	α^{12}	α^{11}	α^3	α^1
α^9	α^9	α^2	α^7	1	α^{11}	α^{14}	α^8	α^{10}	α^1	α^5	0	α^6	α^3	α^{13}	α^{12}	α^4
α^{10}	α^{10}	α^5	α^3	α^8	α^1	α^{12}	1	α^9	α^{11}	α^2	α^6	0	α^7	α^4	α^{14}	α^{13}
α^{11}	α^{11}	α^{14}	α^6	α^4	α^9	α^2	α^{13}	α^1	α^{10}	α^{12}	α^3	α^7	0	α^8	α^5	1
α^{12}	α^{12}	α^1	1	α^7	α^5	α^{10}	α^3	α^{14}	α^2	α^{11}	α^{13}	α^4	α^8	0	α^9	α^6
α^{13}	α^{13}	α^7	α^2	α^1	α^8	α^6	α^{11}	α^4	1	α^3	α^{12}	α^{14}	α^5	α^9	0	α^{10}
α^{14}	α^{14}	α^{11}	α^8	α^3	α^2	α^9	α^7	α^{13}	α^5	α^1	α^4	α^{13}	1	α^6	α^{10}	0

Таблица Г.4 – Умножение в расширенном поле $GF(2^4)$ ($p(x) = x^4+x^3+1$)

*****	**0**	**1**	**α^1**	**α^2**	**α^3**	**α^4**	**α^5**	**α^6**	**α^7**	**α^8**	**α^9**	**α^{10}**	**α^{11}**	**α^{12}**	**α^{13}**	**α^{14}**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
α^1	0	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1
α^2	0	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1
α^3	0	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2
α^4	0	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3
α^5	0	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4
α^6	0	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5
α^7	0	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6
α^8	0	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7
α^9	0	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8
α^{10}	0	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9
α^{11}	0	α^{11}	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}
α^{12}	0	α^{12}	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}
α^{13}	0	α^{13}	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}
α^{14}	0	α^{14}	1	α^1	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}

ПРИЛОЖЕНИЕ Д
МИНИМАЛЬНЫЕ МНОГОЧЛЕНЫ ДЛЯ ЭЛЕМЕНТОВ ПОЛЕЙ ГАЛУА

Таблица Д.1 – Минимальные многочлены и распределение элементов по циклатомическим классам поля $GF(2^4)$, порождаемое многочленом

$p(x) = x^4 + x^3 + 1$

Циклотомические классы	Сопряженные элементы. Корни многочлена $\varphi_i(x)$	Минимальные многочлены $\varphi_i(x)$
$C_0=\{0\}$	α^0	$\varphi_0(x) = x + 1$
$C_1=\{1, 2, 4, 8\}$	$\alpha^1, \alpha^2, \alpha^4, \alpha^8$	$\varphi_1(x) = x^4 + x^3 + 1$
$C_3=\{3, 6, 9, 12\}$	$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$\varphi_3(x) = x^4 + x^3 + x^2 + x + 1$
$C_5=\{5, 10\}$	α^5, α^{10}	$\varphi_5(x) = x^2 + x + 1$
$C_7=\{7, 11, 13, 14\}$	$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	$\varphi_7(x) = x^4 + x + 1$

СПИСОК ЛИТЕРАТУРЫ

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

2. Берлекэмп Э. Алгебраическая теория кодирования. Перевод с англ. И.И. Грушко. Москва: «Мир», 1971. – 478 с.

3. Васильев К.К. Теория электрической связи: учебное пособие / К.К. Васильев, В.А. Глушков, А.В. Дормидонтов // Ульяновск: УлГТУ, − 2008, − 452 с.

4. Васин В.А. Радиосистемы передачи информации. Учебное пособие для вузов / В.А. Васин, В.В. Калмыков // «Горячая линия – Телеком», 2005.

5. Вернер М. Основы кодирования. Учебник для ВУЗов. Перевод с нем. Д.К. Зигангирова, Москва: Техносфера, 2004. – 288 с.

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

7. Когновицкий О.С. Теория помехоустойичвого кодирования: практикум / О.С. Когновицкий, В.М. Охорзин // СПбГУТ: - Спб., 2013. 68с.

8. Когновицкий О.С. Циклические коды БЧХЭ как рекурсивные последовательности / Труды учебных заведений связи // СПбГУТ: - Спб., 2016.

9. Морелос-Сарагоса Р. Исскуство помехоустойчивого кодирования. Методы, алгоритмы, применение. Москва: Техносфера, 2006. – 320 с.

10. Марков А.А. Введение в теорию кодирования. Москва: Наука, 1982.

11. Никитин Г.И. Помехоустойчивые циклические коды: Учебное пособие / Г.И. Никитин, Поддубный С.С. // СПбГУТ: − Спб., 1998. – 71 с.

12. Скляр Б. Цифровая связь. Теоретические основы и практическое применение / Б. Скляр // М.: Издательский дом «Вильямс», 2003. – 1104 с.

13. Нечаев А.А. Кольца Галуа в приложениях к кодам и линейным рекурентам / А.А. Нечаев, А.С. Кузьмин, А.А. Куракин // Москва, 1998.

14. Прокис Джон. Цифровая связь / Пер. с англ. под ред. Д.Д. Кловского. М.: Радио и связь, 2000.

15. Шульгин В.И. Основы теории передачи информации. Часть 2. Помехоустойчивое кодирование. Учебное пособие. Харьков: Нац. аэрокосм. ун-т «Харьк. авиац. ин-т», 2003. – 87 с.

СОДЕРЖАНИЕ

Навчальне видання

КРИЛОВА Вікторія Анатоліївна

ЗАВАДОСТІКІ КОДИ. МЕТОДИ ТА АЛГОРИТМИ ЦИКЛІЧНИХ БЧХ КОДІВ

Навчальний посібник

Російською мовою

Роботу рекомендував до видання проф. *Борисенко А.М.*

Редактор

План 2016 р., поз. 4

Підп. до друку 05.04. 2017 р. Формат 60×84 1/16. Папір офісний.
Riso – друк. Гарнітура Таймс. Ум. друк. арк. 22,5. Наклад 100 пр.
Зам. № . Ціна договірна.

Видавець і виготовлювач
Видавничий центр НТУ «ХПІ»,
вул. Фрунзе, 21, м. Харків-2, 61002

Свідоцтво суб'єкта видавничої справи ДК № 3656 від 24.12.2009 р.