Помехоустойчивое кодирование в системах телекоммуникаций (ПКСТ)

Лекция 1

Список литературы

ОСНОВНАЯ

1. Макаров А.А., Прибылов В.П. Помехоустойчивое кодирование 2005г.
2. Макаров А.А.,Чернецкий Г.А. Корректирующие коды в системах передачи информации 2000г.

Список литературы

ДОПОЛНИТЕЛЬНАЯ:

1. Кларк Дж. и Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи 1987г.
2. Блейхут Р. Теория и практика кодов, контролирующих ошибки 1986г.
3. А.А. Макаров. Методы повышения помехоустойчивости систем связи 1991г.
4. А.А. Макаров, Л.А.Чиненков. Основы теории помехоустойчивости дискретных сигналов 1997г.

Историческая справка

1948г. К. Шеннон показал, что за счет кодирования передаваемой по каналу связи информации при незначительном уменьшении скорости можно практически полностью устранить воздействие помех на передаваемые данные.

1949г. М.Голей нашел код, исправляющий 3 ошибки в блоке из 23 бит.

Историческая справка

1950г. Р. Хэмминг открыл класс кодов, исправляющий 1 ошибку в блоке длины $2^m - 1$

1960г. Р. Боуз, Д. Рой-Чоудхури и

А. Хоквингем (коды БЧХ), исправляющие любое число ошибок

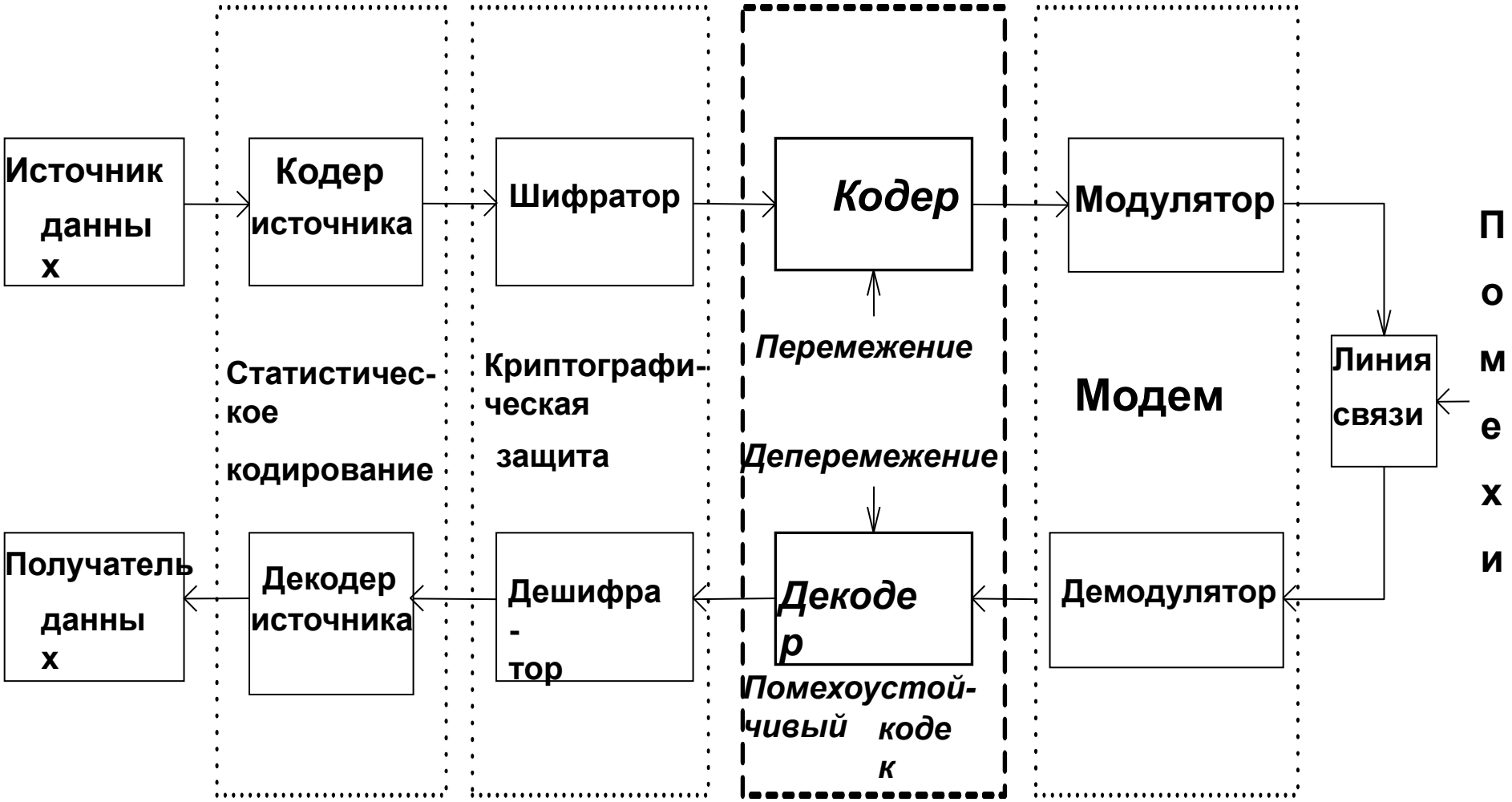
Несколько ранее, И.Рид и Г.Соломон открыли важный подкласс БЧХ-кодов.

Основные практические приложения теории помехоустойчивого кодирования включают:

- **Сотовая, транкинговая, пейджинговая и спутниковая связь;**
- **Сети передачи данных (*Ethernet*, *Wi-Fi*, *Bluetooth* – как правило, коды используются в режиме обнаружения ошибок);**
- **Модемы (протоколы *V*.32/*V*.90 используют решетчатое кодирование, *V*.42 – кодирование с обнаружением ошибок и переспросами);**

- **ОЗУ ЭВМ (в режиме обнаружения или в режиме исправления);**
- **Шины данных ЭВМ (*USB* и др., высокая скорость кода, малая избыточность);**
- **Магнитные и оптические носители информации (*Minidisks*, *CD*, *DVD*) и мн.др.**

Обобщенная структурная схема системы связи



Статистическое кодирование **используется для уменьшения первичной избыточности передаваемой информации.**

Криптографическая защита **используется для предотвращения несанкционированного доступа к информации.**

Помехоустойчивое кодирование **предназначено для защиты данных от ошибок и применяется в системах передачи и хранения данных либо только для обнаружения ошибок, либо для обнаружения и исправления ошибок (или ошибок и стираний в каналах со стиранием).**

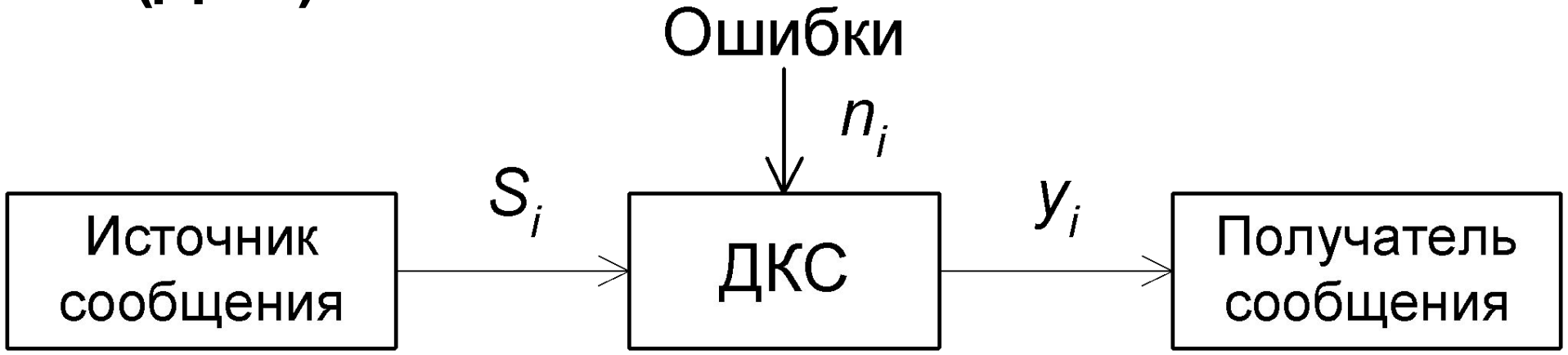
Основная задача *перемежителя* состоит в перестановке элементов потока данных с выхода кодера помехоустойчивого кода таким образом, чтобы деперемежитель на приемной стороне декоррелировал помехи, т.е. преобразовал пакет ошибок, происходящих в реальных дискретных каналах связи (ДКС) в поток независимых ошибок.

Модулятор **преобразует кодовые символы с выхода перемежителя в соответствующие аналоговые символы. Так как в канале связи возникают различного типа шумы, искажения и интерференция, то сигнал на входе *демодулятора (первое решающее устройство)* отличается от сигнала на входе модулятора.**

Декодер помехоустойчивого кода (второе решающее устройство) **использует избыточность кодового слова для того, чтобы обнаружить или обнаружить и исправить ошибки в принятом слове, и затем выдает оценку кодового слова источника сигнала потребителю.**

Дискретный канал связи

Если входные и выходные сигналы канала являются дискретными, то и канал называется *дискретным* (ДКС)



Обобщенная структурная схема системы передачи дискретных сообщений (СПДС)

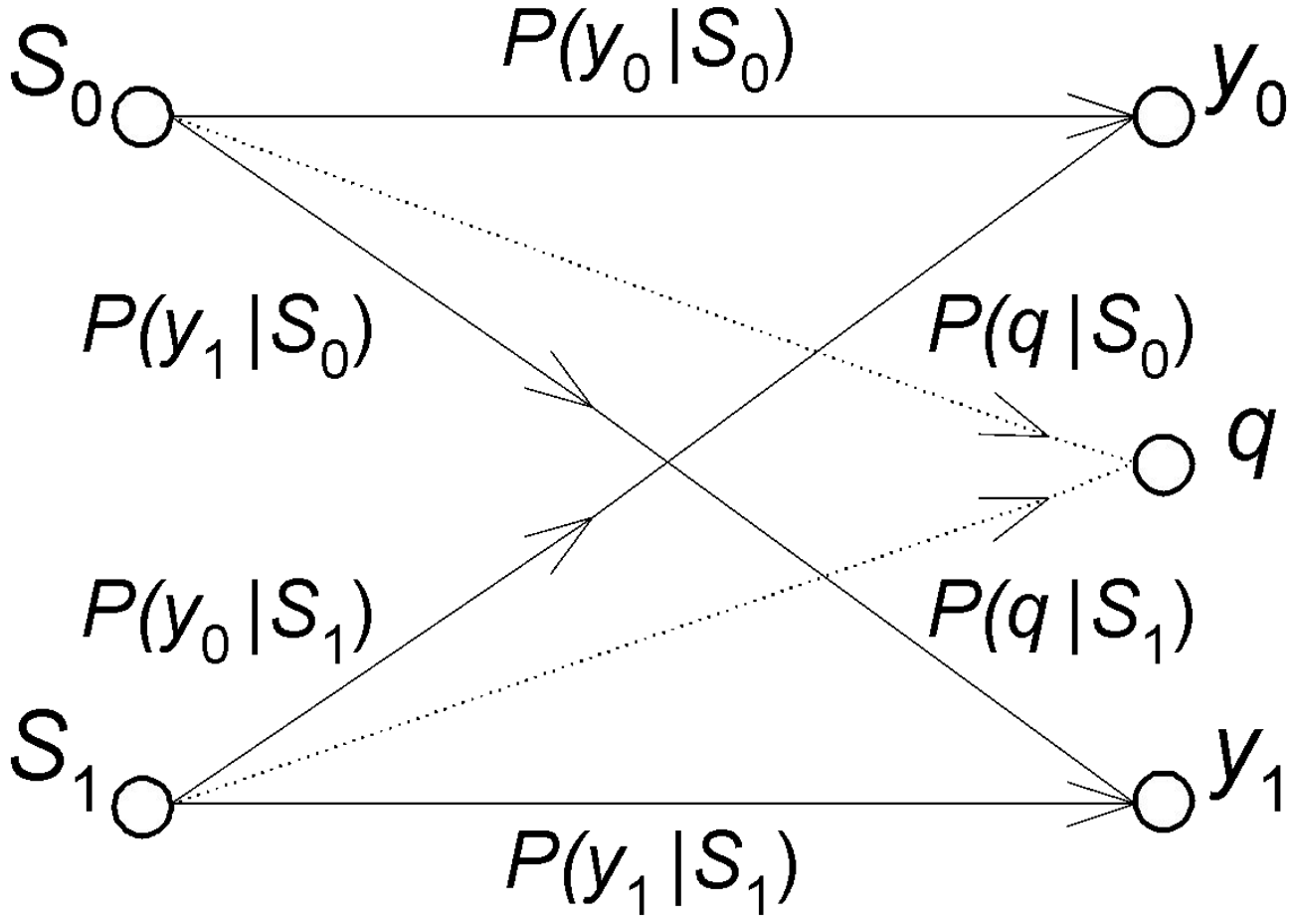
Математическая модель ДКС требует описания следующих параметров:

1) алфавитов входных и выходных сообщений (набор различных символов, из которых составляется сообщение, называется *алфавитом*, а их число – *объемом алфавита*);

2) скорости передачи элементов алфавита;

3) переходных вероятностей.

Диаграмма состояний и переходов для двоичного ДКС



S_0, S_1 – элементы алфавита источника;

y_0, y_1 – элементы алфавита на выходе канала; q – символ стирания; $p(y_i/S_j)$ и $p(q/S_j)$ – переходные вероятности, где $i, j \in \{0, 1\}$.

ДКС могут быть:

1) ***симметричными***, когда переходные вероятности $p(y_i/S_j)$ одинаковы для всех $i \neq j$ и, соответственно, несимметричными в противном случае;

2) ***без памяти***, когда переходные вероятности $p(y_i/S_j)$ не зависят от того, какие символы и с каким качеством передавались до данного символа S_j, и памятью в противном случае;

3) *<u>без стирания</u>*, когда алфавиты на входе канала и выходе демодулятора совпадают, в канале со стиранием алфавит на выходе демодулятора имеет дополнительный символ стирания *q*, формируемый тогда, когда демодулятор не может с заданной надежностью опознать переданный символ.

Основные понятия и определения теории кодирования

Избыточность кода

$$g = \frac{n-k}{n} = \frac{r}{n},$$

где *n* – количество элементов (символов) в кодовом слове;

k и *r* – количество информационных и проверочных символов, соответственно.

Скорость кода

$$R = k/n$$

Чем больше избыточность кода, тем меньше скорость кода, и наоборот.

Расстояние Хэмминга **между двумя кодовыми словами d_{ij}**

$$d_{ij} = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|$$

где x_{ik} , x_{jk} – координаты слов A_i , A_j в *n*-мерном неевклидовом пространстве.

Если код является двоичным, под расстоянием между парой кодовых слов понимается количество символов, в которых они отличаются между собой. Оно определяется сложением этих двух слов по mod 2 и равно числу единиц в этой сумме

$$d_{ij}=[(A_i+A_j)\text{mod } 2]\ \Sigma\text{“1”}\ ,$$

где (..+..)mod 2 – сложение по mod 2; ∑ “1” – означает, что после операции сложения по модулю два необходимо подсчитать количество единиц в полученном результате.

Пример

Дано: A_1=111, A_2=100

Решение: $d_{12} = (A_1 + A_2) \bmod 2 =$

111

100

⊕

011, ∑"1"=2.

Ответ: d_{12} = 2.

Таким образом, *расстоянием Хэмминга* между двумя двоичными последовательностями, называется число позиций, в которых они различны. Минимальное расстояние Хэмминга называется *кодовым расстоянием*

$$d = \min d_{ij}.$$

Число ошибочных символов в принятом кодовом слове называется *кратностью ошибки t*, при длине кодового слова *n* символов она изменяется в пределах от 0 до *n.* Так как *кратность ошибки t* в геометрическом представлении является расстоянием между переданным словом и принятым, то для обнаружения ошибок кратности t_o требуется кодовое расстояние

$$d \geq t_o + 1$$

Для исправления ошибок кратности $t_и$, требуется кодовое расстояние

$$d \geq 2 \cdot t_{\text{и}} + 1$$

Это означает, что для исправления ошибок искаженное кодовое слово должно располагаться ближе всего к соответствующему правильному слову. Кратность исправления $t_и$ определяет границу гарантированного исправления ошибок.

В случае исправления $t_и$ ошибок и t_q стираний (кратность стирания) кодовое расстояние d

$$d \geq 2 \cdot t_{\text{и}} + 1 + t_q$$

Спектр весов кода $N(w)$ **– это распределение весов** w **ненулевых кодовых слов, где** $w = w(i), \quad i = [1, n]$ **– вес** i**-го кодового слова, который равен числу ненулевых символов этого слова.**

Очевидно, что наилучшим как для исправления, так и для обнаружения ошибок будет код с наибольшим кодовым расстоянием. Для нахождения кодов с хорошими корректирующими свойствами используются *границы*. Так, например, известны границы Хэмминга и Плоткина, которые позволяют определить необходимое число проверочных символов в кодовом слове.

Граница Хэмминга **имеет вид:**

$$r \geq \log_m [\sum_{i=0}^{t_и} C_n^i \cdot (m-1)^i].$$

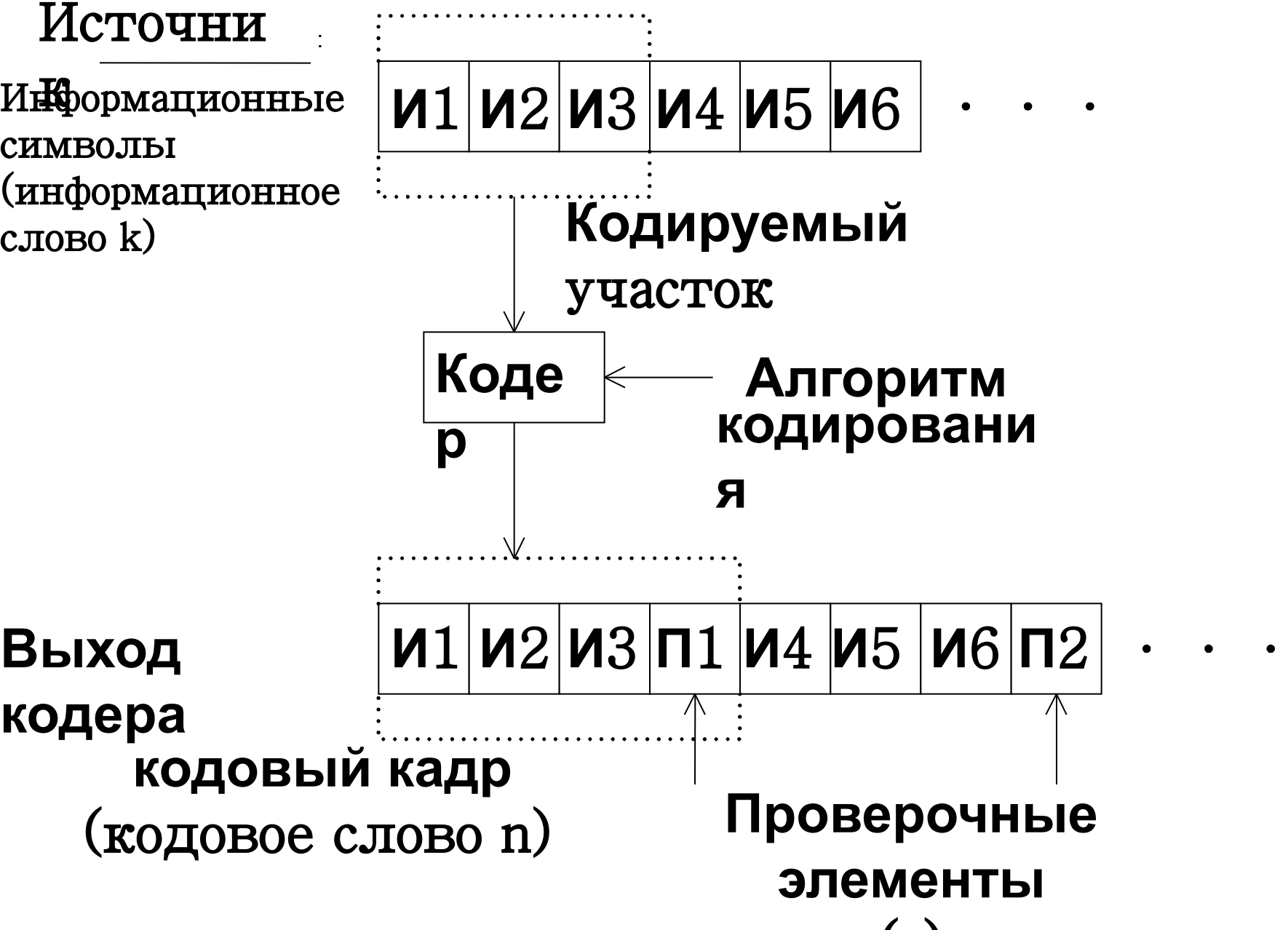
Граница Плоткина **имеет вид:**

$$r \geq \frac{m \cdot (d-1)}{m-1} - \log_m d.$$

Граница Хемминга утверждает, что не существует кодов с $n-k<r$ **гарантированно исправляющих ошибки кратности** $t_{\text{и}}$ **, а граница Плоткина утверждает, что могут быть построены (существуют) коды с** $n-k \geq r.$ **Граница Хэмминга обеспечивает избыточность кода, близкую к минимальной, при больших значениях** $R_k = k/n,$ **а граница Плоткина - при малых.**

Оптимальными обычно считаются такие $(n, k) -$ коды, которые обеспечивают в заданном канале *меньшую вероятность ошибки декодирования* p_b при одинаковой скорости кода и одинаковой вычислительной сложности декодирования.

Принцип образования кодового слова



Последовательность символов на выходе источника разбивается на блоки (кодовые слова или кодовые комбинации), содержащие одинаковое число символов *k*. При этом для двоичного кода ансамбль сообщений будет иметь объем $N_P = 2^K$. **При помехоустойчивом кодировании это множество из *Np* сообщений отображается на множество** $N = 2^n$ **возможных кодовых слов (*n* – число символов в кодовом слове после кодирования, иногда его называют длиной кодовых слов или значностью кода, n > k).**

В общем случае для равномерного блочного кода с основанием ***m*** код имеет $N = m^n$ возможных кодовых слов. Используемые для передачи сообщений кодовые слова из множества $N_{\text{р}} < N$ называют **разрешенными**, остальные кодовые слова из множества $N_{\text{з}} = (N - N_{\text{р}})$ не используются и называются **запрещенными (неразрешенными для передачи)**

Если в результате искажений в канале связи переданное кодовое слово

$\mathbf{a}_i\,(i=1,2,...,N)$ **превращается в одно из запрещенных слов** $\mathbf{b}_j\,(j=1,2,...,N_з)$ **, то декодер приёмника обнаруживает ошибку, так как такое слово не могло быть передано. Ошибка не обнаруживается только в том случае, когда передаваемое кодовое слово превращается в другое разрешенное, например,** $\mathbf{a}_{i+1}$ **, которое также могло быть передано.**

По сравнению с обнаружением *исправление* ошибок представляет собой более сложную операцию, поскольку в этом случае помимо обнаружения наличия ошибки в принятом кодовом слове декодер должен определить местоположение искаженного кодового символа и значение ошибки (для недвоичных кодов). Чтобы рассматриваемый код исправлял ошибки, необходимо часть или все множество $N_з$ запрещенных кодовых слов разбить на N непересекающихся подмножеств $N_з^{(i)} (i = 1, 2, ...N\ )$ по количеству разрешенных кодовых слов.

Каждое из этих подмножеств в декодере приемника приписывается одному из разрешенных кодовых слов. Если принятое кодовое слово принадлежит подмножеству $N_з^{(i)}$, то переданным считается разрешенное кодовое слово a_i. Ошибка не может быть исправлена, если переданное кодовое слово a_i в результате искажений превращается в кодовое слово любого другого подмножества $N_з^{(j)}, (j \neq i)$

Если передаваемая (разрешенная) комбинация может в результате искажений с одинаковой вероятностью превратиться в любую из N возможных комбинаций данного кода, то *коэффициент обнаружения Ко*, как доля обнаруживаемых (запрещенных) комбинаций, будет равен

$$K_O = \frac{N - N_P}{N} = 1 - \frac{N_P}{N}$$

Коэффициент исправления кода, как доля исправленных ошибок, когда разрешенная комбинация с одинаковой вероятностью превращается в любую из N кодовых комбинаций, равен

$$K_{И} = \frac{N - N_P}{N_P} \cdot \frac{1}{N}$$

Отношение $\frac{K_O}{K_И} = \frac{1}{N}$ **,**

следовательно коэффициент исправления кода Ки всегда меньше коэффициента обнаружения, что является общим условием для любых корректирующих кодов.

В общем случае передаваемая кодовая комбинация искажается случайным образом, что определяется случайным характером помех в канале связи. При этом доля комбинаций с ошибками различной кратности определяется вероятностью Рп, а доля комбинаций, в которых ошибка обнаруживается, вероятностью P_O, и коэффициент обнаружения равен

$$K_O = \frac{P_O}{P_П}$$

Вероятность необнаруживаемой ошибки при этом равна

$$P_Н = P_П - P_О$$

Коэффициент исправления кода

$$K_И = \frac{P_И}{P_П}$$

где Ри – вероятность исправления ошибки в кодовой комбинации

Так как в канале связи возникают различного типа шумы, искажения и интерференция, то сигнал на входе демодулятора отличается от сигнала на входе модулятора. При этом сигнал на выходе демодулятора является оценкой соответствующего переданного символа, а из-за шумов в канале возникают ошибки. В зависимости от характера оценки доверия к сигналу на выходе демодулятора различают *мягкое и жесткое решения.*

В соответствии с правилом жесткого решения сигнал на выходе демодулятора определен однозначно для каждого тактового интервала (“0” или “1” для двоичного канала).

Демодулятор с мягким решением обычно имеет два выхода: один из них представляет собой жесткое решение, на втором выходе формируется оценка качества этого решения в виде веса w_i, пропорционального отношению правдоподобия на выходе демодулятора.

Мягкое решение **демодулятора может быть использовано для дальнейшего повышения помехоустойчивости приема (например, мягкого декодирования кодовых слов помехоустойчивого кода). Наиболее часто используются 8-и или 16-и уровневые оценки качества (3 или 4 двоичных символа).**

Процесс декодирования можно представить в виде двух операций, в результате которых на выходе декодера получается оценка переданного информационного слова.

$$\hat{u} = D^{(2)} D^{(1)} (\hat{A})$$

где $D^{(1)}$ **– операция нейтрализации случайного действия помех в канале;**

$D^{(2)}$ **– операция декодирования.**

$\hat{A}$ **- оценка слова на выходе канала (после демодулятора) .**

Если $D^{(2)} = 1$, то информационные символы в кодовом слове имеют такой же вид, какой они имели на входе кодера (систематический кодер).

Оператор $D^{(1)}$ является статистическим и должен действовать так, чтобы в соответствии с выбранной мерой качества и критерием оптимальности обеспечивался минимум потерь информации в процессе преобразования полученной оценки в кодовое слово на выходе.

Критерием, позволяющим минимизировать потери информации, является <u>критерий максимального правдоподобия (МП).</u>

Правило решения по критерию МП можно записать в виде:

$$\text{если } \Lambda \geq \Lambda_0 \Leftrightarrow \frac{P(\hat{A}/A_i)}{P(\hat{A}/A_j)} \geq 1, \text{ то } \hat{A} = A_i, \text{ иначе } \hat{A} = A_j,$$

где Λ и Λ0 – отношение правдоподобия и пороговое отношение правдоподобия;

$P(\hat{A}/A_i)$ **и** $P(\hat{A}/A_j)$ **– функции правдоподобия (условные вероятности получения кодового слова** $\hat{A}$ **при передаче кодовых слов** A_i **и** A_j**,** $j \neq i$**).**

При этом обеспечивается минимум потерь информации, т.е.

***I*(** $\hat{A} \leftrightarrow$ ***Ai*) - *I*(** $\hat{A} \leftrightarrow$ ***Aj*) > 0 для всех** $j \neq i$

где *I*($\hat{A} \leftrightarrow$ ***Ai*), *I*(** $\hat{A} \leftrightarrow$ ***Aj*) – взаимная информация кодовых слов** $\hat{A}$ **и *Ai*,** $\hat{A}$ **и *Aj*.**

При известных априорных вероятностях *P*(*Aj*) и *P*(*Ai*) иногда используется <u>критерий максимальной апостериорной вероятности (МАВ)</u>, минимизирующий среднюю вероятность ошибки декодирования

$$p_{д} = \sum_{i} \sum_{j \neq i} P(A_i, A_j) \quad ; i, j=1, 2, \ldots, N,$$

где N – число кодовых слов данного кода; $P(A_i, A_j)$ – совместная вероятность передачи кодового слова A_i и принятия решения A_j.

Для критерия максимума апостериорной вероятности (МАВ) пороговое отношение правдоподобия принимает значение

$$\Lambda_0 = \frac{P(A_j)}{P(A_i)},$$

где $P(A_i)$ и $P(A_j)$- априорные вероятности передачи кодовых слов A_i и A_j

Классификация корректирующих кодов

Все коды делятся:

- ***Блочные*** **- передаваемое сообщение делится на блоки. Процесс кодирования и декодирования осуществляется отдельно на каждом блоке.**
- ***Непрерывные*** **- проверочные символы появляются путем непрерывного преобразования по определённому алгоритму информационных символов.**

- ***Разделимые*** – можно определить информационные и проверочные элементы.
- ***Неразделимые*** – этого сделать нельзя.
- ***Линейные*** – это разделимые коды, в которых проверочные элементы определяются линейными операциями с информационными элементами.

Все линейные коды делятся:

- ***Систематические (групповые)*****-** **если две комбинации принадлежат этой группе, то при сложении по модулю 2, получившиеся комбинации тоже принадлежат этой группе.**
- ***Несистематические*****- в противном случае.**

Простейшие корректирующие коды

Код с четным числом единиц

Код с четным числом единиц **является блочным кодом и образуется путем добавления к комбинации к–элементного кода одного избыточного элемента так, чтобы количество единиц в новой n-элементной комбинации было четным.**

Например, для к =2 $00 \rightarrow 000$

$01 \rightarrow 011$

$10 \rightarrow 101$

$11 \rightarrow 110$

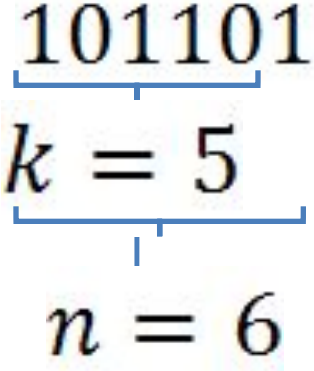
Этот код имеет очень высокую скорость, практически близкую к 1. Минимальное расстояние кода равно 2, а потому код не может исправить ни одной ошибки, но позволяет обнаружить одну ошибку, т.е. по принятому слову определить ситуацию, когда произошла одна ошибка. Коды проверки на четность часто используются в микросхемах оперативной памяти.

Простейшие корректирующие коды

Они только обнаруживают ошибку.

1) Код с проверкой на четность:

Добавляем к информационным символам один проверочный, чтобы число единиц было чётным:



Принцип обнаружения- проверка на чётность

Коэффициент обнаружения:

$$K_O = \frac{N - N_P}{N} = \frac{2^6 - 2^5}{2^6} = 0.5$$

Данный код обнаруживает ошибки нечетной кратности.

$$P_{0t} = \sum_t C_n^t P_0^t (1 - P_0)^{n-t}$$

Избыточность:

$$g = \frac{6-5}{6} = 0{,}17$$

Данный код разделимый и блочный

2) ***Код с постоянным весом:***

Вес- количество единиц кодовой комбинации.

Рассмотрим код 3 к 4: 1001100
1010100

Принцип обнаружения - определение веса (количество единиц)

В кодовом слове этого кода невозможно разделить символы на информационные и проверочные (избыточные).

$$K_O = \frac{N - N_P}{N} = \frac{128 - 35}{128} = 0.73$$

$$N = 2^7 = 128 \qquad N_P = C_7^3 = \frac{7!}{3!4!}$$

$$P_{\text{ош.}} = 1 - (1 - P_0)^n = 1 - (1 - P_0)^7$$

$$P_{\text{НО}} = C_3^1 P_0^1 (1-P_0)^2 C_4^1 P_0^1 (1-P_0)^3 +$$

$$+ C_3^2 P_0^2 (1-P_0)^1 C_4^2 P_0^2 (1-P_0)^2 +$$

$$+ C_3^3 P_0^3 C_4^3 P_0^3 (1-P_0)^1$$

Избыточность:

$$g = \frac{r}{n} = \frac{2}{7} \approx 0.3$$

Код блочный неразделимый систематический.

Обнаруживает все ошибки нечетной кратности и 50% ошибок вероятности четной кратности. Не обнаруживает ошибки четной кратности, когда количество искаженных единиц равно количеству искаженных нулей.

Инверсный код

Кодовые слова инверсного корректирующего кода образуются повторением исходного кодового слова. Если число единиц в исходном слове четное, оно повторяется в неизменном виде; если число единиц нечетное, то при повторении все символы исходного кодового слова инвертируются (нули заменяются единицами, а единицы - нулями).

Таблица кодирования

	k					r				
	1	2	3	4	5	6	7	8	9	10
а)	1	0	1	1	1	1	0	1	1	1
а)	0	1	1	0	0	0	1	1	0	0
б)	0	1	1	0	1	1	0	0	1	0
б)	1	0	0	0	0	0	1	1	1	1

а) в исходном кодовом слове четное число единиц,

б) в исходном кодовом слове нечетное число единиц.

Для обнаружения ошибок в кодовой комбинации, состоящей из n=r+k (в таблице n=10) производится две операции.

1. **Суммируются единицы, содержащиеся в первых k элементах комбинации.**
2. **Если число единиц четное, r последующих элементов сравниваются попарно с первыми k элементами; если число единиц в первых элементах нечетное, последующие элементы перед сравниванием инвертируются.**

Несовпадение хотя бы одной из пар сравниваемых кодовых элементов указывает на наличие ошибки в комбинации.

$$K_O = \frac{N - N_P}{N} = \frac{2^{10} - 2^5}{2^{10}} = 0.97$$

$$P_{\text{но}} = C_5^2 P_0^2 (1 - P_0)^3 P_0^2 (1 - P_0)^3$$

$$P_{\text{ош.}} = 1 - (1 - P_0)^n = 1 - (1 - P_0)^{10}$$

Код блочный, разделимый, систематический.

Энергетический выигрыш от кодирования

Энергетический выигрыш от кодирования (ЭВК) **указывает на улучшение качества системы связи от использования данного способа кодирования или метода защиты от ошибок и определяется выражением**

$$\text{ЭВК} = 10\lg\frac{\tilde{R}\cdot h_1^2}{h_2^2}[\text{дБ}],$$

где h_1^2 , h_2^2 **– отношения сигнал/шум в сравниваемых системах связи при одинаковой вероятности ошибок на выходе;** $\tilde{R}$ **– относительная скорость системы с защитой от ошибок.**

Например, если первая система является системой без помехоустойчивого кодирования, а вторая система – с обнаружением ошибок и переспросом,

то $\tilde{R} = (k/n) \cdot T/T_{ср}$;

здесь $T_{ср}$ **– средняя длительность передачи кодового слова (или символа длительности** T**) в системе с переспросом**

Для системы с кодом, исправляющим ошибки без переспроса, относительная скорость равна

$$\widetilde{R} = k/n = R,$$

т.е. равна скорости кода.

Если снять ограничения на длину кодового слова и полосу частот, то предельный выигрыш от помехоустойчивого кодирования при данной вероятности ошибки в канале связи с гауссовским шумом

$$10\lg\frac{E/N_0}{0{,}693}=10\lg\left(E/N_0\right)+1{,}6\ \text{дБ}$$

где *E*– энергия сигнала; N_0 – спектральная плотность мощности помехи.

В тех случаях, когда непрерывный канал, включая модулятор и демодулятор, заданы, выигрыш от помехоустойчивого кодирования можно оценить с помощью коэффициента исправления ошибок (повышения достоверности)

$$K_и = \frac{p}{p_д}$$

где p и $p_д$ – вероятности ошибок на символ на входе и выходе декодера.

Линейные двоичные блочные коды (групповые коды)

Двоичный блочный код является линейным тогда и только тогда, когда сумма по модулю 2 двух кодовых слов является также кодовым.

Двойные линейные блочные коды часто называют групповыми кодами, поскольку кодовые слова образуют математическую структуру, называемую группой.

Кодовые слова такого кода содержат *n* символов; причем *к* символов – являются информационными, а остальные *r =n-k* – проверочными символами.

Таким образом, любые кодовые слова данного кода можно записать как

A $_i$ **=(** $a_{i1}, a_{i2}, \ldots, a_{kk}, \ldots, a_{in}$**), (*k*+*r* =*n*).**

При этом все множество разрешенных кодовых слов равно 2^k **и составляет подмножество группы** N **порядка** 2^n **.**

Следовательно, количество запрещенных кодовых слов может быть найдено так:

$$N_З = N - N_P = 2^n - 2^k$$

Так как разрешенные кодовые слова группового линейного кода образуют подгруппу относительно операции сложения по модулю два, то для определения всех кодовых слов подгруппы достаточно найти *к* линейно-независимых кодовых слов (сумма этих слов по mod 2 не должно равняться нулю). Остальные $(2^k - k - 1)$ - кодовых слов находятся путем сложения по mod 2 этих *k* кодовых слов во всевозможных сочетаниях

$$\sum_{i=2}^{k} C_k^i = 2^k - k - 1$$

Для построения группового кода используют понятия: Производящая матрица и Проверочная матрица.

***Производящая матрица G* кода (*n*, *k*) имеет *n* столбцов и *k* строк, в канонической форме образуется путем дополнения (*r* =*n*-*k*)- столбцов к единичной матрице размерности *k* × *k*.**

$$G_{kxn} = \left| \begin{array}{ccccccccc} 1 & 0 & 0 & . & 0 & b_{11} & b_{12} & . & b_{1r} \\ 0 & 1 & 0 & . & 0 & b_{21} & b_{22} & . & b_{2r} \\ . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & 1 & b_{k1} & b_{k2} & . & b_{kr} \end{array} \right| \begin{array}{c} \\ \\ \\ \\ \} \\ \end{array} k =$$

$$\{\leftarrow \quad k \quad \rightarrow\} \quad \{\leftarrow \quad r \quad \rightarrow\}$$

$$= \left| \begin{array}{ccc} & . & \\ I_{kxk} & . & B_{kxr} \\ & . & \end{array} \right| = \left| \begin{array}{c} A_1 \\ A_2 \\ : \\ A_K \end{array} \right|$$

где *к* – количество столбцов в единичной подматрице, *r* – количество столбцов в символьной (дополнительной) подматрице, *к* – количество строк в матрице *G*

Символы $\{(b_{11}\ldots b_{1r});(b_{21}\ldots b_{2r});\ldots;(b_{k1}\ldots b_{kr})\}$ **могут быть найдены произвольно, однако должны выполняться условия:**

- вес каждой строки производящей матрицы $w_{A_i} \geq d$**;**

- $d_{A_iA_j} \geq d$ **, где** $d_{A_iA_j}$ **– расстояние между двумя кодовыми словами *Ai* и *Aj*, входящими в производящую матрицу; *d* – кодовое расстояние данного кода.**

Для определения алгоритмов кодирования и декодирования некоторого группового линейного кода строится проверочная матрица *H*. Проверочная матрица в каноническом виде записывается путем дополнения *k* столбцов к единичной матрице размерности (*r* x *r*) (дополнение приписывается слева от единичной матрицы).

$$H_{r\text{x}n} = \left| \begin{array}{cccccccc} a_{11} & a_{12} & . & . & a_{1k} & 1 & 0 & . & 0 \\ a_{21} & a_{22} & . & . & a_{2k} & 0 & 1 & . & 0 \\ . & . & . & . & . & . & . & . & . \\ a_{r1} & a_{r2} & . & . & a_{rk} & 0 & 0 & . & 1 \end{array} \right| \begin{array}{l} \\ \\ \\ \\ \} r \end{array} =$$

$$\begin{array}{cc} \{\leftarrow & k \rightarrow\} \{\leftarrow \quad r \quad \rightarrow\} \end{array}$$

$$= \left| \begin{array}{ccc} & . & \\ A_{r\text{x}k} & . & I_{r\text{x}r} \\ & . & \end{array} \right|$$

Эта матрица используется для составления проверочных уравнений.

Для построения группового (*n*, *k*) кода с заданными параметрами *n* и $t_и$, и определения правил кодирования и декодирования необходимо:

1) найти кодовое расстояние *d*;
2) найти количество проверочных элементов *r*;
3) построить производящую матрицу *G*;
4) построить проверочную матрицу *H* и систему проверочных уравнений.

Пример

Построить групповой двоичный код, исправляющий одиночные ошибки $t_и=1$, длина кодового слова $n=7$.

Решение задачи будем осуществлять поэтапно.

1 этап: Определение кодового расстояния *d*.

$$d \geq 2 \cdot t_{\text{и}} + 1$$

Тогда $d \geq 2 \cdot 1 + 1 = 3$ **, примем $d=3$.**

<u>2 этап</u>: Определение количества проверочных элементов *r* производится согласно границе Хэмминга.

$$2^r \geq \sum_{i=1}^{t_u} C_n^i + 1 \quad , r \geq \log_2\left(\sum_{i=1}^{t_u} C_n^i + 1\right) \; ;$$

$2^r \geq C_7^1 + 1 = 8,$ $r \geq \log_2 8 = 3$ **, примем *r* =3.**

Скорость кода *R* =4/7.

<u>3 этап:</u> Строим производящую матрицу G:

$$G_{k\text{x}n} = G_{4\text{x}7} = \left| \begin{array}{ccccccc} 1 & 0 & 0 & 0 & b_{11} & b_{12} & b_{13} \\ 0 & 1 & 0 & 0 & b_{21} & b_{22} & b_{23} \\ 0 & 0 & 1 & 0 & b_{31} & b_{32} & b_{33} \\ 0 & 0 & 0 & 1 & b_{41} & b_{42} & b_{43} \end{array} \right| = \left| \begin{array}{c} A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \right|$$

$$\{\leftarrow \quad 4 \quad \rightarrow\} \quad \{\leftarrow \quad 3 \quad \rightarrow\}$$

Проверочные символы записываем так, чтобы расстояния между кодовыми словами A_1, A_2, A_3 и A_4 и веса этих слов были не меньше трех. Т.о.,

$$G = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{vmatrix}$$

<u>4 этап</u>: Находим остальные разрешенных кодовых слов: $(2^k - k - 1) = 11$

$$A_5 = A_1 \oplus A_2 = (1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1)$$

$$A_6 = A_1 \oplus A_3 = (1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0)$$

$$A_7 = A_1 \oplus A_4 = (1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0)$$

$$A_8 = A_2 \oplus A_3 = (0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1)$$

$$A_9 = A_2 \oplus A_4 = (0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1)$$

$$A_{10} = A_3 \oplus A_4 = (0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$$

$$A_{11} = A_1 \oplus A_2 \oplus A_3 = (1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0)$$

$$A_{12} = A_1 \oplus A_2 \oplus A_4 = (1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0)$$

$$A_{13} = A_1 \oplus A_3 \oplus A_4 = (1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1)$$

$$A_{14} = A_2 \oplus A_3 \oplus A_4 = (0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0)$$

$$A_{15} = A_1 \oplus A_2 \oplus A_3 \oplus A_4 = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$$

Нулевое слово, хотя и не используется для передачи, также является разрешенным, так как оно определяется как сумма всех остальных разрешенных слов

$$A_0 = \left(\sum_{i=1}^{15} A_i \right) \bmod 2 = (0\,0\,0\,0\,0\,0\,0)$$

Таким образом, мы определили все множество $2^k = 2^4 = 16$ **разрешенных кодовых слов, являющихся элементами поля** $GF(2^3)$.

<u>5 этап:</u> Проверка расстояний между разрешенными словами $d_{A_i A_j} \geq d$

Эта проверка, в принципе, эквивалентна определению веса каждого разрешенного слова полученного кода *Wi*.

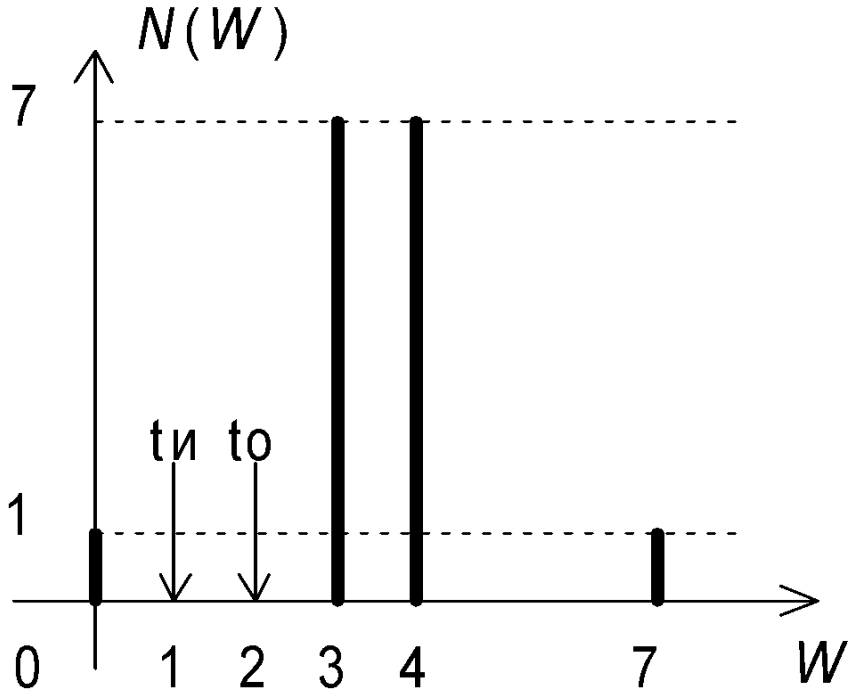
Построим таблицу весов *Wi* = $d_{A_i A_0}$

Количество слов A_i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}
Вес слова w_i	3	3	3	4	4	4	3	4	3	3	3	4	4	4	7

Таблица весов кода (7,4)

Как видно из таблицы , кодовое расстояние построенного кода *d*=3.

<u>6 этап</u>: Определение спектра весов *N*(*w*) и производящей функции *T*(*z*) кода. Подсчет количества одинаковых весов дает: *N*(0)=1; *N*(3)=7; *N*(4)=7; *N*(7)=1. Таким образом, спектр весов имеет вид, показанный на рисунке



Производящая функция будет иметь вид:

$$T(z) = 1 \cdot z^0 + 7 \cdot z^3 + 7 \cdot z^4 + 1 \cdot z^7$$

<u>7 этап</u>: Формирование проверочной матрицы *H*.

Для проверочной матрицы выбираем такие кодовые слова, проверочные символы которых образуют единичную матрицу, а число единиц четно.

$$H = \begin{vmatrix} A_{14} \\ A_{12} \\ A_{13} \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{vmatrix}$$

Таким образом, все строки проверочной матрицы будут удовлетворять проверкам на четность:

$$\left.\begin{aligned} a_2 \oplus a_3 \oplus a_4 \oplus b_1 = 0 \\ a_1 \oplus a_2 \oplus a_4 \oplus b_2 = 0 \\ a_1 \oplus a_3 \oplus a_4 \oplus b_3 = 0 \end{aligned}\right\}$$ проверочные уравнения для декодирования.

Операция кодирования, т.е. вычисление проверочных кодовых символов, определяется системой уравнений:

$$\left.\begin{aligned} b_1 = a_2 \oplus a_3 \oplus a_4 \\ b_2 = a_1 \oplus a_2 \oplus a_4 \\ b_3 = a_1 \oplus a_3 \oplus a_4 \end{aligned}\right\}$$ уравнения для кодирования.

Таким образом, если на вход кодера поступает последовательность вида {0110}, то на его выходе кодовое слово: {0110$b_1b_2b_3$}={0110011}.

Полученное кодовое слово передается по зашумленному каналу. Декодер принимает слово, в котором уже может быть одна ошибка, и вычисляет три проверочные суммы

$$\left.\begin{aligned} a_2' \oplus a_3' \oplus a_4' \oplus b_1' &= s_1 \\ a_1' \oplus a_2' \oplus a_4' \oplus b_2' &= s_2 \\ a_1' \oplus a_3' \oplus a_4' \oplus b_3' &= s_3 \end{aligned}\right\}$$

Вектор (S_1,S_2,S_3) называется *синдромом* и зависит только от конфигурации ошибок: в этих суммах значения всех неискаженных битов сокращаются, а один искаженный бит вносит 1 в каждую сумму, в которую он входит.

Всего имеется 8 различных синдромов: один для случая отсутствия ошибок (нулевой синдром) и по одному для каждой из семи одиночных ошибок.

№	Позиция ошибки	Значение синдрома
1	0000000	(0,0,0)
2	1000000	(1,0,1)
3	0100000	(1,1,1)
4	0010000	(1,1,0)
5	0001000	(0,1,1)
6	0000100	(1,0,0)
7	0000010	(0,1,0)
8	0000001	(0,0,1)

Таким образом, по значению синдрома декодер может определить, в какой позиции произошла ошибка, и исправить ее.

Допустим, что в канале связи кодовое слово A_j было искажено.

1) Примем: $A_j=A_1=\{1000011\}$;

Последовательность ошибок: $e=\{0010000\}\rightarrow t=1$. Тогда слово на входе декодера имеет вид: $B_1=\{1010011\}$.

2) Применяя к B_1 проверочные соотношения, получим:

$$a_2 \oplus a_3 \oplus a_4 \oplus b_1 = 1 \to s_1,$$
$$a_1 \oplus a_2 \oplus a_4 \oplus b_2 = 0 \to s_2,$$
$$a_1 \oplus a_3 \oplus a_4 \oplus b_3 = 1 \to s_3,$$

где S_1, S_2 и S_3 – элементы синдрома ошибки, указывающие на факт наличия ошибки в случае появления единицы при проверке.

Однако эти элементы, сами по себе, не позволяют сделать вывод о том, какой из кодовых символов был искажен.

3) Устанавливаем, что искажен тот символ, который входит только в 1-ю и 3-ю проверки, как видно, это символ a_3.

4) Для исправления ошибки символ a_3 инвертируется с помощью исправляющей последовательности

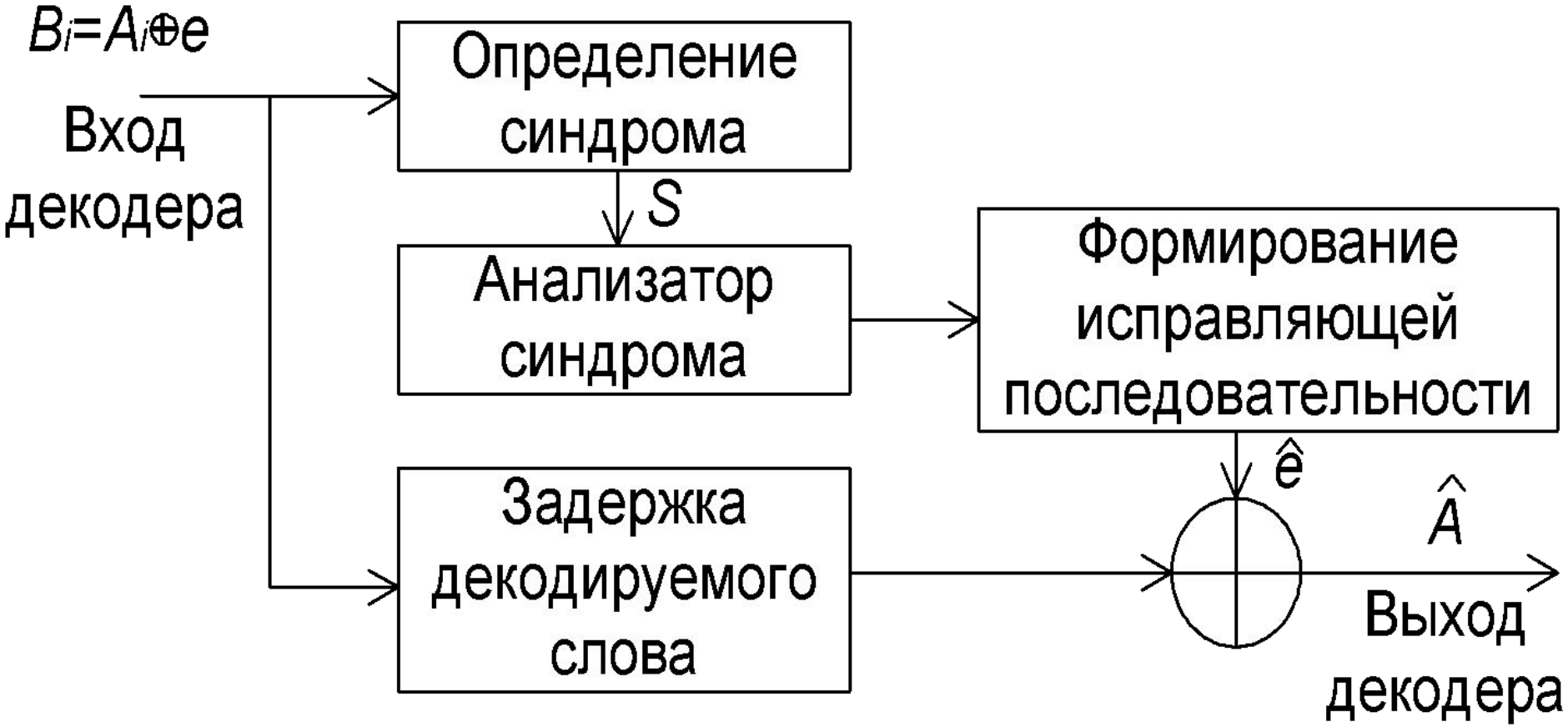
$$\hat{\vec{e}} = \{0010000\}$$

Таким образом, на выходе декодера получается оценка декодируемого кодового слова в виде

$$\hat{\vec{A}}_1 = B_1 \oplus \hat{\vec{e}} = \{1000011\} = A_1$$

На этом процесс декодирования заканчивается.

Обобщенная структурная схема синдромного декодера группового линейного кода



Если оценка последовательности ошибок точна ($\hat{e} = e$), то оценка декодируемого кодового слова является переданным кодовым словом ($\hat{A} = A$). Иначе $\hat{A}$ представляет собой другое разрешенное кодовое слово. Вероятность последнего события называют *вероятностью ошибки декодирования кодового слова* $P_{\text{нд}}$. Эта вероятность может быть определена из следующего соотношения

$$P_{\text{пп}} + P_{\text{оо/ио}} + P_{\text{нд}} = 1$$

В системах с обнаружением ошибок значение $P_{\text{нд}}$ меньше по сравнению с аналогичным значением для систем с исправлением ошибок.

Таким образом, для нахождения вероятности $P_{\text{нд}}$ следует вычесть из полной вероятности (1) вероятность правильного приема и вероятность обнаружения/исправления ошибок

$$P_{\text{нд}} = 1 - P_{\text{пп}} - P_{\text{оо/ио}}$$

Вероятность правильного приема кодового слова длины *n* в каналах с независимыми ошибками вероятности *p* определяется как

$$P_{\text{пп}} = (1-p)^n$$

<u>Режим обнаружения ошибок.</u>

Зная спектр весов кода $N(w)$ **можно определить** $P_{\text{оо}}$

$$P_{\text{оо}} = \sum_{i=1}^{n} \left[C_n^i - N(i) \right] \cdot p^i \cdot (1-p)^{n-i}$$

Соответственно, значение вероятности ошибки декодирования

$$P_{\text{нд}} = 1 - (1-p)^n - \sum_{i=1}^{n} \left[C_n^i - N(i) \right] \cdot p^i \cdot (1-p)^{n-i}$$

Без знания спектра весов кода вероятность обнаружения ошибок можно определить только примерно (нижняя граница), как

$$P_{\text{оо}} \geq \sum_{i=1}^{t_{\text{о}}} C_n^i \cdot p^i \cdot (1-p)^{n-i}$$

Тогда

$$P_{\text{нд}} \le 1 - (1-p)^n - \sum_{i=1}^{t_{\text{о}}} C_n^i \cdot p^i \cdot (1-p)^{n-i}$$

Режим исправления ошибок.

Вероятность неверного декодирования в этом случае полностью определяется кратностью исправления ошибок кода и может быть рассчитана как

$$P_{\text{нд}} = \sum_{i=t_{\text{и}}+1}^{n} C_n^i \cdot p^i \cdot (1-p)^{n-i}$$

или, с учетом больших значений *n*,

$$P_{\text{нд}} = 1 - (1-p)^n - \sum_{i=1}^{t_{\text{и}}} C_n^i \cdot p^i \cdot (1-p)^{n-i}$$

Зачастую необходимо оценивать помехоустойчивость устройства защиты от ошибок по эквивалентной вероятности ошибки бита, т.е. по вероятности ошибки декодирования кодового слова $P_{\text{нд}}$, приведенной к числу информационных битов k

$$p_{\partial} = \frac{P_{\text{нд}}}{k}$$

Число логических операций, необходимых для декодирования кодового слова длиной *n* (сложность декодера), обычно, увеличивается экспоненциально с ростом *n*.

Существенное упрощение процедуры декодирования достигается при использовании кодов Рида–Маллера, Хэмминга и циклических кодов.

Пример

Закодировать модифицированным кодом Хэмминга (7,4), d=3 комбинацию из четырех информационных символов {И1,И2,И3,И4}={1,0,0,1}.

Определяем значения проверочных символов.

П1 = (И1 + И2 + И4) mod 2=(1+0+1) mod 2 = 2 mod 2 = 0,

П2 = (И1 + И3 + И4) mod 2=(1+0+1) mod 2 = 2 mod 2 = 0,

П3 = (И2 + И3 + И4) mod 2=(0+0+1) mod 2 = 1 mod 2 = 1.

Таким образом, передаваемое кодовое слово кода Хэмминга имеет вид {И4,И3,И2, П3,И1,П2,П1}={1,0,0,1,1,0,0}.

Допустим, что в канале связи был искажен бит И1. Кодовое слово приняло вид {И'4,И'3,И'2,П'3, И'1,П'2,П'1}={1,0,0,1,*0*,0,0}. Процесс декодирования осуществляется в приемнике согласно проверочным уравнениям:

$S1 = (П'1 + И'1 + И'2 + И'4) \bmod 2=(0+0+0+1) \bmod 2= = 1;$

$S2 = (П'2 + И'1 + И'3 + И'4) \bmod 2=(0+0+0+1) \bmod 2= = 1;$

$S3 = (П'3 + И'2 + И'3 + И'4) \bmod 2=(1+0+0+1) \bmod 2= = 0.$

Записываем результат проверки в виде $S_3S_2S_1$=011, что равно десятичному числу 3, которое указывает номер искаженного разряда, начиная с младшего разряда. Следовательно, в этом разряде необходимо изменить 0 на 1.

После исправления искаженного символа в информационных разрядах получим последовательность символов 1001, которая совпадает с переданной комбинацией информационных символов.

Обнаруживающая способность кода Хэмминга может быть увеличена на единицу ($t_{\hat{1}} = 3$) введением дополнительной проверки на четность. В этом случае проверочная матрица расширенного кода Хэмминга (8,4) , будет иметь вид

$$H_{(8,4)} = \begin{vmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

а кодовое расстояние расширенного кода (8, 4) равно *d*=4. Однако следует отметить, что введение бита проверки на четность не сказывается на исправляющей способности внутреннего кода (7,4), $t_{\text{и}} = 1$ **.**

Для расширенного модифицированного кода Хэмминга кодовая последовательность имеет вид, представленный в таблице

$П_{общ}$	$И_4$	$И_3$	$И_2$	$П_3$	$И_1$	$П_2$	$П_1$
8	7	6	5	4	3	2	1

Распределение символов кодовой посылки кода (8,4)

Где Побщ=(П1+П2+И1+П3+И2+И3+И4)mod2.

ЦИКЛИЧЕСКИЕ КОДЫ

Циклическим кодом **называется линейный блочный код (*n*,*k*), который характеризуется свойством цикличности, то есть сдвиг влево на один шаг любого разрешенного кодового слова дает другое разрешенное кодовое слово, принадлежащее этому же коду:**

$$(a_1, a_2, \ldots a_n) \rightarrow (a_n, a_1, a_2, \ldots a_{n-1})$$

Множество кодовых слов циклического кода представляется совокупностью полиномов степени (*n*-1) и менее, делящихся на некоторый полином *g*(*x*) степени *r*=*n*-*k*, который является сомножителем двучлена $x^n + 1$. Полином *g*(*x*) называется *порождающим* (или производящим). Существуют специальные таблицы по выбору *g*(*x*) в зависимости от предъявляемых требований к корректирующим свойствам кода. Поэтому при изучении конкретных циклических кодов будут рассматриваться соответствующие способы построения *g*(*x*).

Производящая матрица **циклического кода определяется путем умножения g(x) степени (n-k) последовательно на** $1, x, x^2 \ldots x^{k-1}$**.**

Проверочная матрица **совершенного циклического кода определяется**

$$\mathrm{H} = \begin{vmatrix} h_1(x) = \dfrac{x^n + 1}{g(x)} \\ h_2(x) = h_1(x) \cdot x \\ \text{---------} \\ h_r(x) = h_1(x) \cdot x^{r-1} \end{vmatrix},$$

где $h_i(x), i = 1, 2, \ldots r$ **– строки проверочной матрицы (проверочные соотношения).**

Умножение и деление многочленов производится по обычным правилам алгебры, но с приведением подобных членов по модулю 2.

Например.

Построить проверочную матрицу для циклического кода (15,11) и порождающего полинома

$$g(x) = x^4 + x + 1$$

$$
\begin{array}{rll|l}
 & x^{15} & +1 & g(x) = x^4 + x + 1 \\
\oplus & \underline{x^{15} + x^{12} + x^{11}} & & \hline
 & x^{12} + x^{11} & +1 & x^{11} + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1 \\
\oplus & \underline{x^{12} + x^9 + x^8} & & \\
 & x^{11} + x^9 + x^8 & +1 & \\
\oplus & \underline{x^{11} + x^8 + x^7} & & \\
 & x^9 + x^7 & +1 & \\
\oplus & \underline{x^9 + x^6 + x^5} & & \\
 & x^7 + x^6 + x^5 & +1 & \\
\oplus & \underline{x^7 + x^4 + x^3} & & \\
 & x^6 + x^5 + x^4 + x^3 & +1 & \\
\oplus & \underline{x^6 + x^3 + x^2} & & \\
 & x^5 + x^4 + x^2 & +1 & \\
\oplus & \underline{x^5 + x^2 + x} & & \\
 & x^4 + x & +1 & \\
\oplus & \underline{x^4 + x} & +1 & \\
 & 0 & &
\end{array}
$$

Полином $x^{11}+x^8+x^7+x^5+x^3+x^2+x+1$ **, полученный в результате деления** $x^{15}+1$ **( ) на** ***g*(*x*), и будет первой строкой проверочной матрицы. В двоичной записи этот полином выглядит как {000100110101111}, следовательно, проверочная матрица будет иметь вид**

$$\mathrm{H}(x)=\begin{vmatrix}000100110101111\\001001101011110\\010011010111100\\100110101111000\end{vmatrix}.$$

Проверочная матрица позволяет вычислить синдром ошибки в виде полинома степени (*n*-*k*-1)

$$S(x) = B(x) \cdot \mathrm{H}^{\mathrm{T}} = [A(x) + e(x)] \cdot \mathrm{H}^{\mathrm{T}} = e(x) \cdot \mathrm{H}^{\mathrm{T}},$$

где *B*(*x*) – полином принятого кодового слова степени (*n*-1), *e*(*x*) – полином ошибок в канале степени (*n*-1).

АЛГОРИТМ ПОЛУЧЕНИЯ РАЗРЕШЕННОЙ КОДОВОЙ КОМБИНАЦИИ ЦИКЛИЧЕСКОГО КОДА ИЗ КОМБИНАЦИИ ПРОСТОГО КОДА

Пусть задан полином

$$g(x) = a_{r-1}x^{r} + a_{r-2}x^{r-1} + \ldots + 1$$

определяющий корректирующую способность кода и число проверочных разрядов r, а также исходная комбинация простого к-элементного кода в виде многочлена $A_{k-1}(x)$

Требуется определить разрешенную кодовую комбинацию циклического кода (n,k).

1. Умножаем многочлен исходной кодовой комбинации x^r:

$$A_{k-1}(x) \cdot x^r$$

2. Определяем проверочные разряды, дополняющие исходную информационную комбинацию до разрешенной, как остаток от деления полученного в предыдущем пункте произведения на порождающий полином:

$$A_{k-1}(x) \cdot x^r / g_r(x) \Rightarrow R(x)$$

3.Окончательно разрешенная кодовая комбинация циклического кода определяется так:

$$A_{n-1}(x) = A_{k-1}(x) \cdot x^r + R(x)$$

Для обнаружения ошибок в принятой кодовой комбинации достаточно поделить ее на производящий полином. Если принятая комбинация – разрешенная, то остаток от деления будет нулевым. Ненулевой остаток свидетельствует о том, что принятая комбинация содержит ошибки. По виду остатка (синдрома) можно в некоторых случаях также сделать вывод о характере ошибки, ее местоположении и исправить ошибку.

Пример. Пусть требуется закодировать комбинацию вида 1101, что соответствует

$$A(x) = x^3 + x^2 + 1$$

Определяем число проверочных символов r=3. Из таблицы возьмем многочлен $g(x) = x^3 + x + 1$ **, т.е. 1011.**

Умножим А(х) на x^r

$$A(x) \cdot x^r = (x^3 + x^2 + 1) \cdot x^3 = x^6 + x^5 + x^3 \Rightarrow 11010000$$

Разделим полученное произведение на образующий полином g(x)

$$A(x)\cdot x^r / g(x) = (x^6 + x^5 + x^3)/(x^3 + x + 1) =$$

$$= x^3 + x^2 + x + 1 + 1/(x^3 + x + 1) \Rightarrow 1111 + 001/1011$$

В результате получим закодированное сообщение:

$$F(x) = (x^3 + x^2 + 1)\cdot(x^3 + x + 1) =$$

$$= (x^3 + x^2 + 1)\cdot x^3 + 1 \Rightarrow 1101001$$

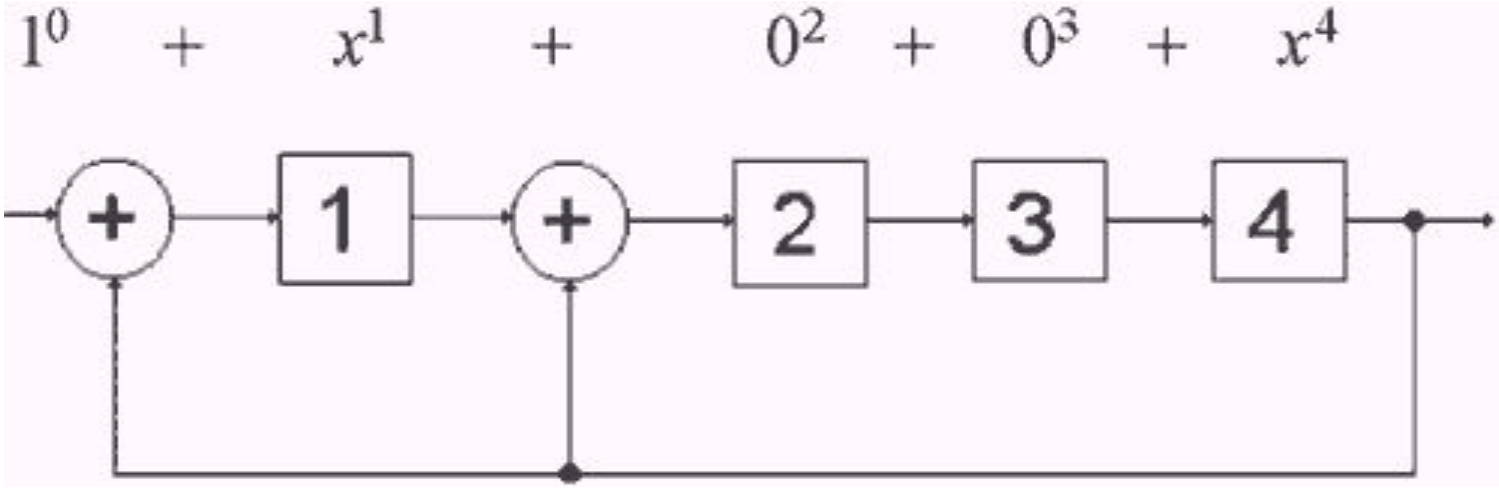
В полученной кодовой комбинации циклического кода информационные символы A(x)=1101, а проверочные 001. Закодированное сообщение делится на образующий полином без остатка.

Построение формирователя остатка циклического кода

Структура устройства осуществляющего деление на полином полностью определяется видом этого полинома. Существуют правила позволяющие провести построение однозначно.

Сформулируем правила построения ФПГ.

1.Число ячеек памяти равно степени образующего полинома r.
2.Число сумматоров на единицу меньше веса кодирующей комбинации образующего полинома.
3.Место установки сумматоров определяется видом образующего полинома.

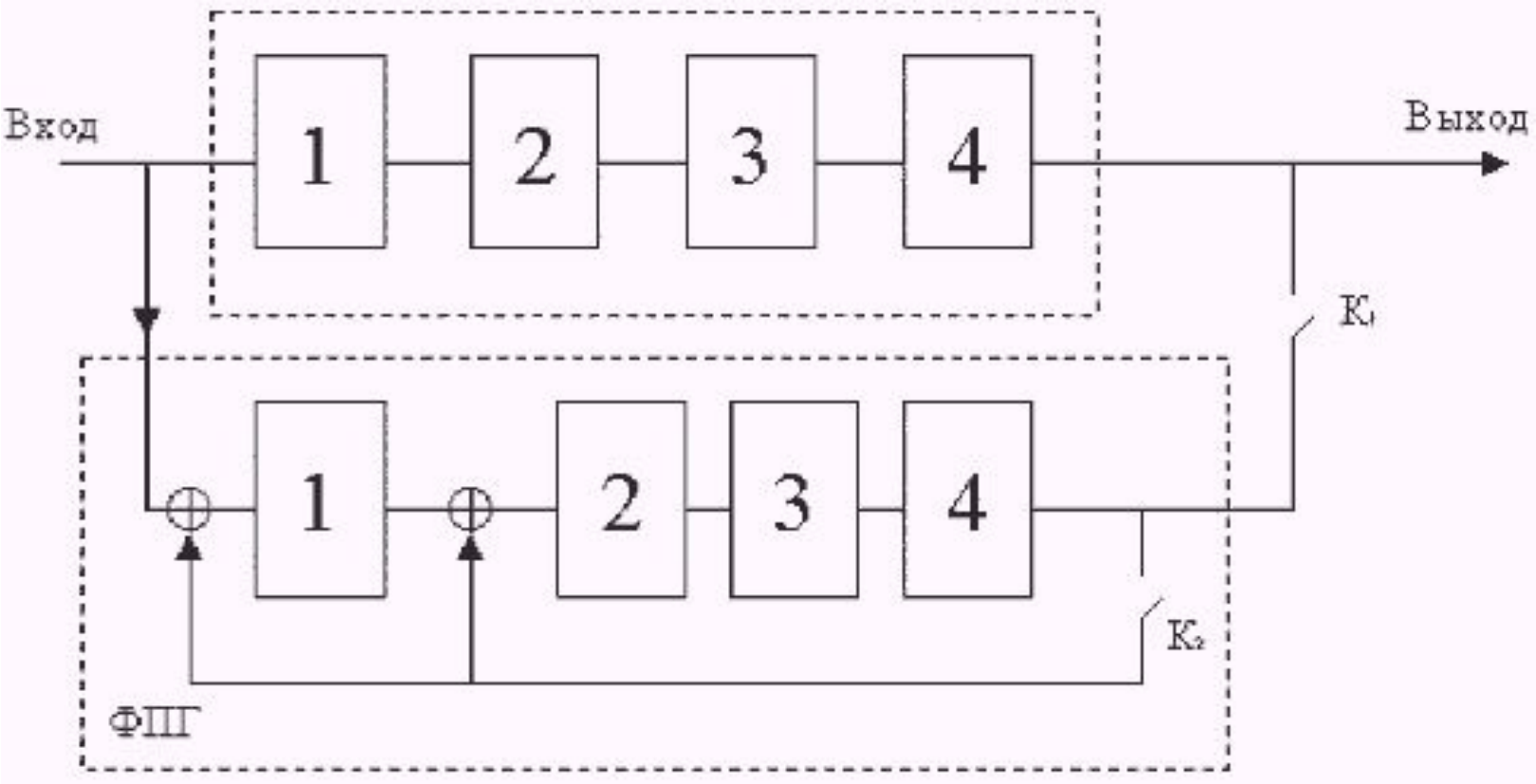


Сумматоры ставят после каждой ячейки памяти, (начиная с нулевой) для которой существует НЕнулевой член полинома. Не ставят после ячейки для которой в полиноме нет соответствующего члена и после ячейки старшего разряда.

4. В цепь обратной связи необходимо поставить ключ, обеспечивающий правильный ввод исходных элементов и вывод результатов деления.

Структурная схема кодера циклического кода (9,5)

Полная структурная схема кодера приведена на следующем рисунке. Она содержит регистр задержки и рассмотренный выше формирователь проверочной группы.



Рассмотрим работу этой схемы

1. На первом этапе $К_1$– замкнут, $К_2$ – разомкнут. Идет одновременное заполнение регистров задержки и сдвига информ. элементами (старший вперед!) и через 4 такта старший разряд в ячейке №4

2. Во время пятого такта $К_2$ – замыкается, а $К_1$ – размыкается с этого момента в ФПГ формируется остаток.

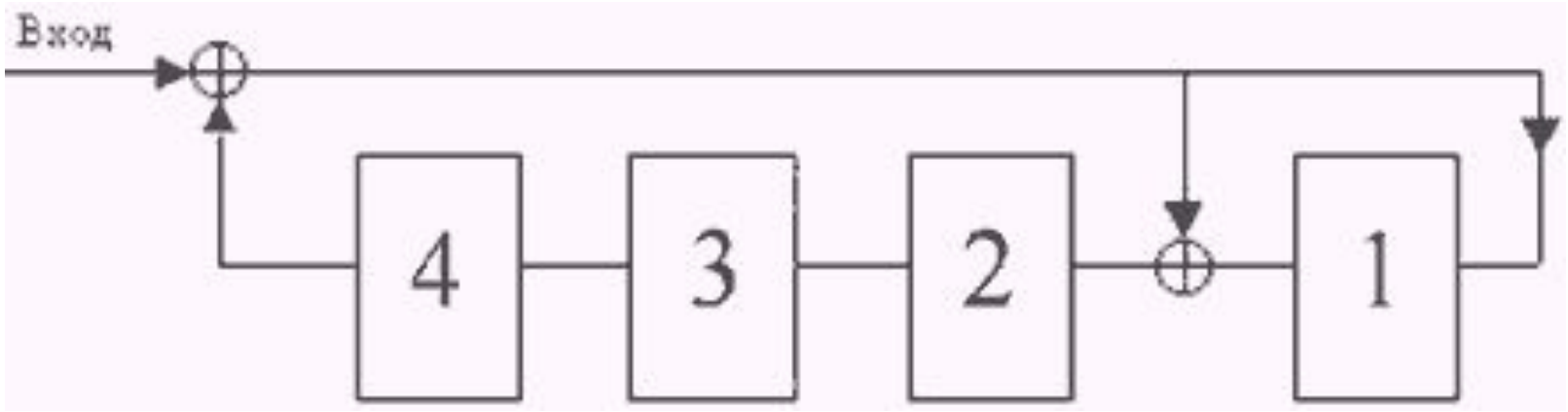
Одновременно из РЗ на выход выталкивается задержанные информационные разряды.
За 5 тактов (с 5 по 9 включительно) в линию уйдут все 5-информационных элементов. К этому времени в ФПГ сформируется остаток.

3. $К_2$ – размыкается, $К_1$ – замыкается и вслед за информационными в линию уйдут элементы проверочной группы.
4. Одновременно идет заполнение регистров новой комбинацией.

Второй вариант построения кодера ЦК.

Рассмотренный выше кодер очень наглядно отражает процесс деления двоичных чисел. Однако можно построить кодер содержащий меньшее число элементов, т.е. более экономичный.

Устройство деления на производящий $P_4(x) = X^4 + X + 1$ **полином можно реализовать в следующем виде:**

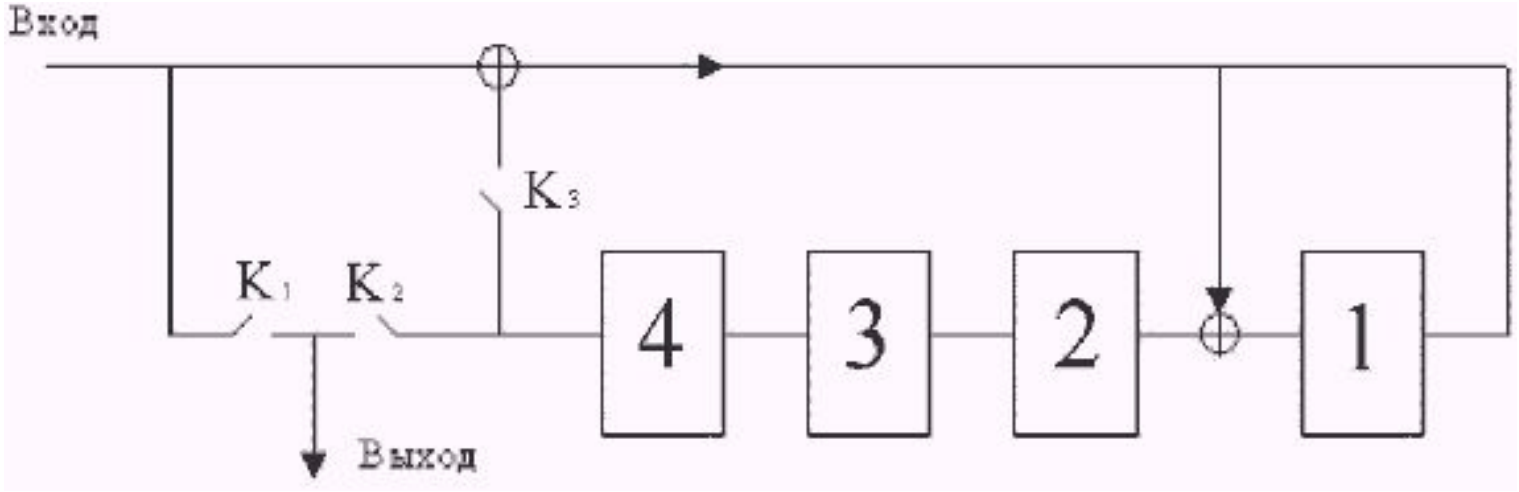

Вход
4
3
2
1

За пять тактов в ячейках будет сформирован такой же остаток от деления, что и в рассмотренном выше Формирователе проверочной группы (ФПГ).

Далее вслед за информационными уходят проверочные из ячеек устройств деления.

Но важно отключить обратную связь на момент вывода проверенных элементов, иначе они исказятся.

Окончательно структурная схема экономичного кодера выглядит так.



- На первом такте Кл.1 и Кл.3 замкнуты, информационные элементы проходят на выход кодера и одновременно формируются проверочные элементы.
- После того, как в линию уйдет пятый информационный элемент, в устройстве деления сформируются проверочные;

- на шестом такте ключи 1 и 3 размыкаются (разрываются обратная связь), а ключ 2 замыкается и в линию уходят проверочные разряды. Ячейки при этом заполняются нулями и схема возвращается в исходное состояние.

Определение ошибочного разряда в ЦК.

Пусть A(x)-многочлен, соответствующий переданной кодовой комбинации.

H(x)- многочлен, соответствующий принятой кодовой комбинации. Тогда сложение данных многочленов по модулю два даст многочлен ошибки.

E(x)=A(x) Å H(x)

При однократной ошибке E(x) будет содержать только один единственный член, соответствующий ошибочному разряду.

Остаток, полученный от деления принятого многочлена H(x) на производящей Pr(x), равен остатку полученному при делении соответствующего многочлена ошибок E(x) на Pr(x).

$$\frac{H(x_1)}{P_r(x)} = \frac{E(x_1)}{P_r(x)}$$

При этом ошибке в каждом разряде будет соответствовать свой остаток R(x) (он же синдром), а значит, получив синдром можно однозначно определить место ошибочного разряда.

Алгоритм определения ошибки.

Пусть имеем n-элементные комбинации (n = k + r) тогда:

1. Получаем остаток от деления E(x), соответствующего ошибке в старшем разряде [1000000000], на образующий полином Pr(x).

$$\frac{E_1(x)}{P_r(x)} = R_0(x)$$

2. Делим полученный полином H(x) на $P_r(x)$ и получаем текущий остаток R(x).

3. Сравниваем $R_0(x)$ и $R(x)$.

- Если они равны, то ошибка произошла в старшем разряде.

Если "нет", то увеличиваем степень принятого полинома на X и снова проводим деление.

$$\frac{H(x) \bullet x}{P_r(x)} = R(x)$$

4. Опять сравниваем полученный остаток с $R_0(x)$.

- Если они равны, то ошибки во втором разряде.
- Если нет, то умножаем $H(x)x^2$ и повторяем эти операции до тех пор, пока R(x) не будет равен $R_0(x)$.

Ошибка будет в разряде, соответствующем числу, на которое повышена степень H(x) плюс один.

Например:

$$\frac{H(x)x^3}{P_r(x)} = R_0(x)$$

номер ошибочного разряда 3+1=4

Пример декодирования комбинации ЦК.

Положим, получена комбинация H(x)=111011010

Проанализируем её в соответствии с вышеприведенным алгоритмом.

Реализуя алгоритм определения ошибок, определим остаток от деления вектора, соответствующего ошибке в старшем разряде x^8 , на производящий полином P(x)=X4+X+1

$$
\begin{array}{l}
\oplus \quad x^8 \\
\underline{x^8 + x^5 + x^4} \\
\qquad\quad x^5 + x^4 \\
\qquad \oplus \\
\qquad \underline{x^5 + x^2 + x} \\
\qquad\qquad x^4 + x^2 + x \\
\qquad\qquad \oplus \\
\qquad\qquad \underline{x^4 + x + 1} \\
\qquad\qquad\quad x^2 + 1
\end{array}
\qquad
\left|
\begin{array}{c}
g(x) = x^4 + x + 1 \\
\hline
x^4 + x + 1
\end{array}
\right.
$$

$$x^8$$

$$x^8 + x^5 + x^4\, x^4 + x + 1$$

$$x^5 + x^4$$

$$x^5 + x^2 + x$$

$$x^4 + x^2 + x$$

$$x^4 + x + 1$$

$$x^2 + 1 = R_0(x) = 0101$$

Разделим принятую комбинацию на образующий полином H(x) · x

1110110100 10011

10011 111111

5-т 11101

10011

6-т 11100

10011

7-т 11111

10011

8-т 11000

10011

9-т 10110 = R(x) ≅ $R_0(x)$

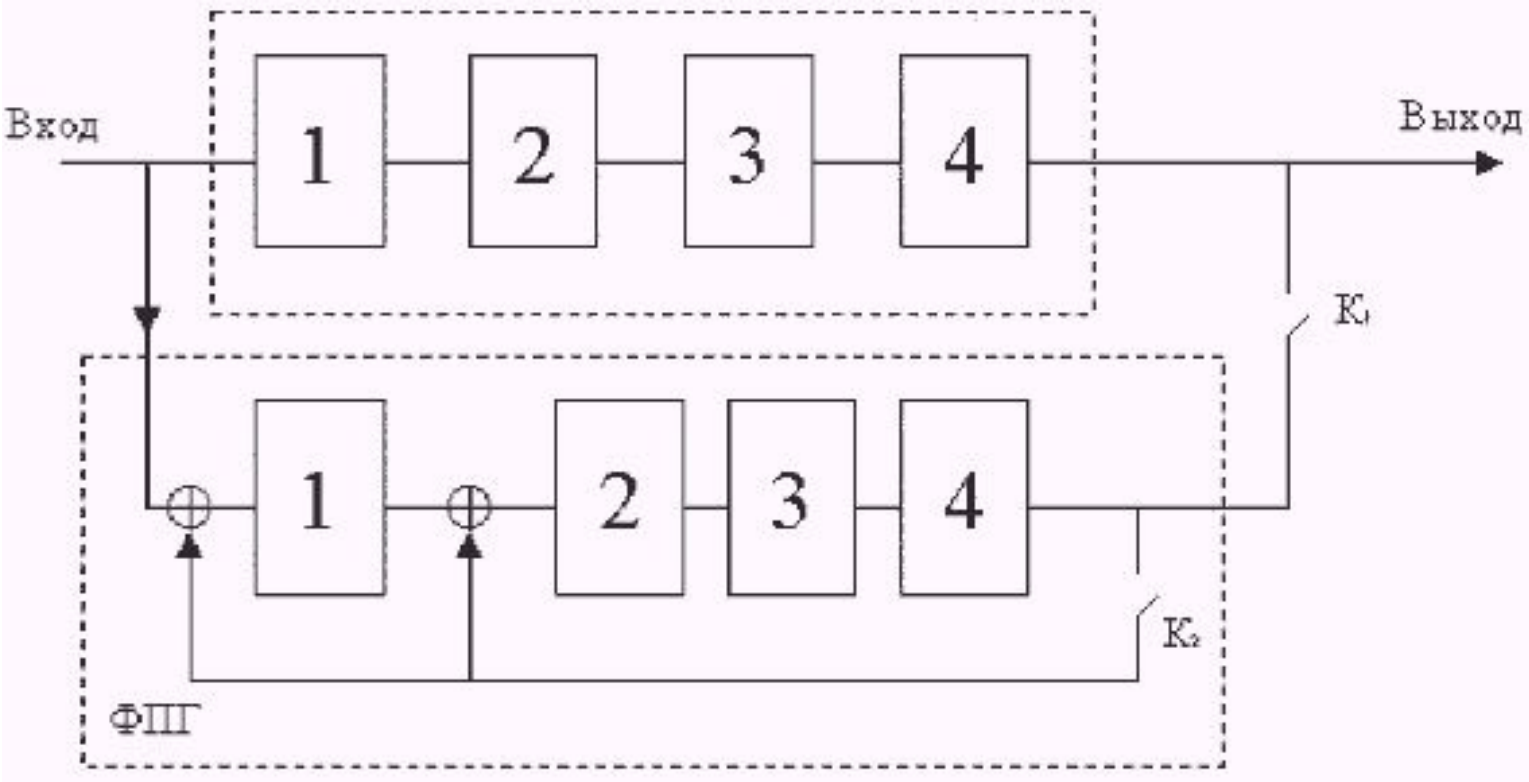
10011

10-т 0101=$R_0(x)$

Полученный на 9-м такте остаток, как видим, не равен $R_0(x)$. Значит необходимо умножить принятую комбинацию на X и повторить деление. Однако результаты деления с 5 по 9 такты включительно будут такими же, значит необходимо продолжить деление после девятого такта до тех пор, пока в остатке не будет $R_0(x)$.

В нашем случае это произойдет на 10 такте, при повышении степени на 1. Значит ошибки во втором разряде.

Декодер циклического с исправлением ошибки



Если ошибка в первом разряде, то остаток $R_0(x)=10101$ появляется после девятого такта в ячейках ФПГ.

Если во втором по старшинству то после 10го;

в третьем по старшинству то после 11го;

в четвертом по старшинству то после 12го

в пятом по старшинству то после 13го
в шестом по старшинству то после 14го
в седьмом по старшинству то после 15го
в восьмом по старшинству то после 16го
в девятом по старшинству то после 17го.
На 10 такте старший разряд покидает регистр задержки и проходит через сумматор по модулю 2.

Если и этому моменту остаток в ФПГ=$R_0(x)$, то логическая 1 с выхода дешифратора поступит на второй вход сумматора и старший разряд инвертируется.

В нашем случае инвертируется второй разряд на 11 такте.

Действия над многочленами.

При формировании комбинаций циклического кода часто используют операции сложения многочленов и деления одного многочлена на другой. Так,

$$A_1(x)+A_2(x)=\left(x^3+x^2+1\right)+\left(x^3+x\right)=x^2+x+1$$

поскольку $x^3+x^3=(1\oplus 1)x^3=0$ **.**

Следует отметить, что действия над коэффициентами полинома (сложение и умножение) производятся по модулю 2.

Рассмотрим операцию деления на следующем примере:

$$
\begin{array}{l|l}
x^7+x^5+x^4+x^2+1 & x^3+x^2+1 \\
\underline{x^7+x^6+x^4} & \\
x^6+x^5+x^2 & x^4+x^3+1 \\
\underline{x^6+x^5+x^3} & \\
x^3+x^2+1 & \\
\underline{x^3+x^2+1} & \\
0 \quad 0 \quad 0 &
\end{array}
$$

Деление выполняется, как обычно, только вычитание заменяется суммированием по модулю два.

Отметим, что запись кодовой комбинации в виде многочлена, не всегда определяет длину кодовой комбинации. Например, при n = 5, многочлену $x+1$ **ответствует кодовая комбинация 00011.**

Умножение многочленов производится по обычным правилам алгебры, но с приведением подобных членов по модулю 2. Например,

$$
\begin{array}{r}
x^3 + x + 1 \\
\underline{x^2 + 1} \\
x^3 + 0 + x + 1 \\
\oplus \quad \underline{x^5 + 0 + x^3 + x^2} \\
x^5 + 0 + 0 + x^2 + x + 1 = x^5 + x^2 + x + 1
\end{array}
$$

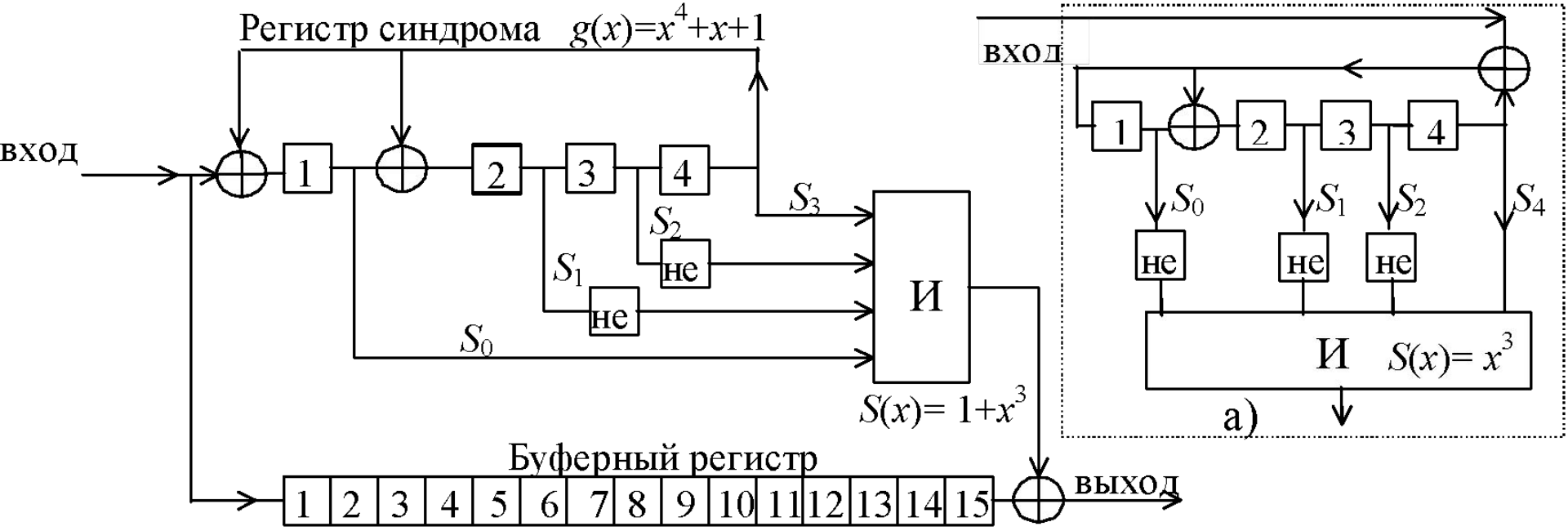
Выбор образующего полинома

Рассмотрим вопрос выбора образующего полинома, который определяет корректирующие свойства циклического кода. В теории циклических кодов показано, что образующий полином представляет собой произведение так называемых минимальных многочленов $m_i(x)$, являющихся простыми сомножителями (то есть делящимся без остатка лишь на себя и на 1) бинома $x^n + 1$:

Декодер Меггита

Декодер Меггита **представляет собой синдромный декодер, исправляющий одиночные ошибки. В нем хранится только один синдром ошибки:** $S_{15}(x) = x^3 + 1$ **(соответствует конфигурации ошибки** $e^{15}(x) = x^{14}$**). Синдромы остальных одиночных ошибок циклически сдвигаются в регистре синдрома до совпадения с** $S_{15}(x)$**; число циклов сдвига плюс единица равно номеру искаженного кодового символа. Поэтому такие декодеры иногда называются** ***декодерами с вылавливанием ошибок*****.**

Декодер Меггита для циклического кода (15,11)



Декодер работает следующим образом: кодовое слово (с ошибками или без них) в виде последовательности из 15 двоичных символов поступает в буферный регистр и одновременно в регистр синдрома, где производится деление этого слова на производящий полином кода

$$g(x) = x^4 + x + 1$$

в результате чего вычисляется синдром ошибки $S_j(x) : S_{0\text{j}}, S_{1\text{j}}, S_{2\text{j}}, S_{3\text{j}}$ **– элементы синдрома.**

Ошибка обнаруживается, если хотя бы один элемент синдрома не равен нулю.

Исправление ошибок производится в следующих 15 циклах сдвига. В каждом *i*-ом цикле проверяется равенство

$$S_{j+i}(x) = S_{15}(x)$$

и в благоприятном случае на выходе схемы “И” появляется импульс коррекции ошибки, инвертирующий символ на выходе буферного регистра.

Полный процесс работы декодера занимает 30 тактов.

В пунктирном квадрате показана возможная модификация регистра синдрома, упрощающая реализацию схемы И. Для этого принимаемая последовательность до ввода в регистр синдрома умножается на x^4 , тогда синдром ошибки в первом символе кодового слова будет равен $S_{15}(x) = x^3$.

СВЕРТОЧНЫЕ КОДЫ

Сверточными кодами **являются древовидные коды, на которые накладываются дополнительные ограничения по линейности и постоянстве во времени. Для сверточных кодов справедлива линейная свертка**

$$a_i = \sum_{k=0}^{n-1} g_{i-k} \cdot d_k$$

или в виде полиномов $a(x) = g(x)\, d(x)$,

где a_i – символы кода, g_{i-k} – весовые коэффициенты (коэффициенты производящего полинома кода $g(x)$), d_k – информационные символы кода.

Сверточные (n,k) коды, которые иногда называют *скользящими блочными кодами*, обычно задаются скоростью

$$R = k_0 / n_0$$

и, в отличие от блочных кодов, требуют для своего описания несколько порождающих (производящих) полиномов.

Эти полиномы могут быть объединены в матрицу

$$\mathbf{G}(x) = [\, g_{ij}(x) \,] = \begin{vmatrix} g_{11}(x) & g_{12}(x) & \dots & g_{1n_0}(x) \\ g_{21}(x) & g_{22}(x) & \dots & g_{2n_0}(x) \\ \dots & \dots & \dots & \dots \\ g_{k_0 1}(x) & g_{k_0 2}(x) & \dots & g_{k_0 n_0}(x) \end{vmatrix}$$

где i=1...k_0, j=1...n_0, k_0 и n_0 – целые числа: k_0 – число информационных символов, необходимых для формирования одного кадра n_0 на выходе кодера

При использовании сверточных кодов поток данных разбивается на гораздо меньшие блоки длиной k символов, которые называются кадрами информационных символов.

Основными характеристиками сверточных кодов являются величины:

-k_0 – размер кадра информационных символов;

-n_0 - размер кадра кодовых символов;

- m- длина памяти кода;

- $k=(m+1)k_0$- информационная длина слова;

- $n=(m+1)n_0$- кодовая длина блока.

Вместо длины кодового слова часто используется понятие «*длина кодового ограничения*» n_a, которая показывает максимальное расстояние между позициями информационных символов, участвующих в формировании проверочного символа данного кода (например, при R =1/2 длина кодового ограничения равна числу ячеек памяти регистра сдвига кодера)

Входная последовательность из *k* информационных символов представляется вектором-строкой

$$D(x) = \left[d_i(x)\right] = \left[d_1(x), d_2(x), \ldots d_{k0}(x)\right];$$

а кодовое слово на выходе кодера

$$A(x) = \left[a_j(x)\right] = \left[a_1(x) \ldots a_{n0}(x)\right]$$

Операция кодирования представляется в виде произведения

A(*x*) = D(*x*)·G(*x*)

Проверочная матрица H(*x*) должна удовлетворять условию

$$G(x) \cdot H^T(x) = 0$$

а вектор синдромов ошибки (синдромных полиномов) равен

$$S(x) = B(x) \cdot H^T(x) = \left[S_j(x)\right] = \left[S_1(x) \ldots S_{n0-k0}(x)\right]$$

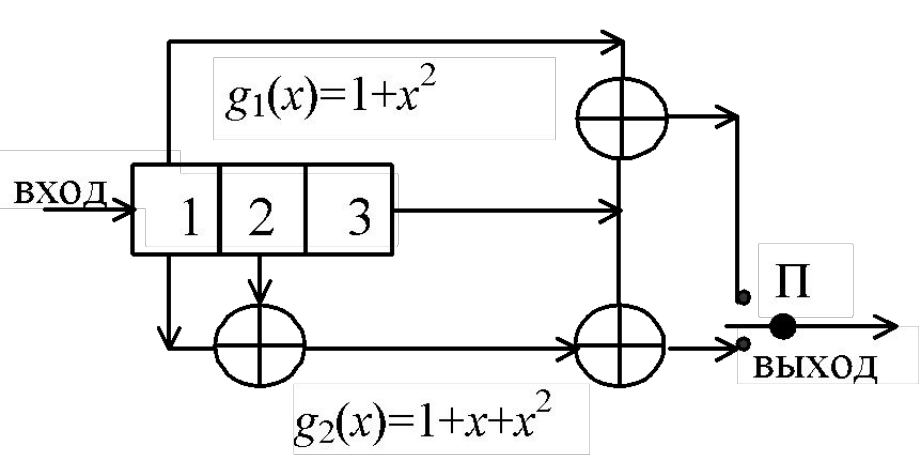
т.е. (*n*0 - *k*0)-мерный вектор-строка из полиномов, а B(*x*) =A(*x*)+e(*x*),
где e(*x*) −вектор ошибок в декодируемой последовательности B(*x*).
Очевидно, что

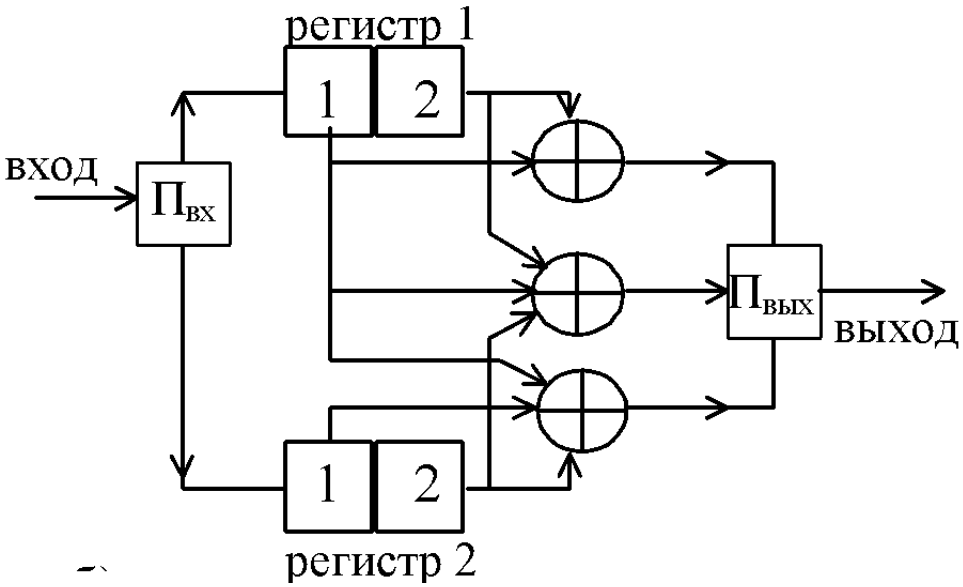
$$S(x) = \left[A(x) + e(x)\right] \cdot H^T(x) = e(x) \cdot H^T(x) \quad .$$

Как и блочные, сверточные коды могут быть систематическими и несистематическими и обозначаются как линейные сверточные (n,k) - коды.

Систематическим сверточным кодом **является такой код, для которого в выходной последовательности кодовых символов содержится без изменения породившая его последовательность информационных символов. В противном случае сверточный код является несистематическим.**

Кодирование сверточных кодов производится аналогично блочным циклическим кодам с помощью регистров сдвига, у которых структура обратных связей определяется производящим полиномом кода. Различие только в том, что при $k_0 > 1$ сверточный код имеет несколько производящих полиномов, а кодер должен иметь соответствующее число регистров сдвига.





Кодеры сверточных кодов а) *R*=1/2 и б) *R*=2/3

Кодеры работают следующим образом.

На вход регистра сдвига кодера (а) из 3 ячеек памяти подается двоичная последовательность информационных символов, из которых с помощью регистра сдвига и сумматоров по модулю 2 формируется две двоичных последовательности. Символы этих последовательностей с помощью переключателя П поочередно подключаются к выходу кодера; скорость переключения должна быть в два раза больше скорости ввода информационных символов.

Матрица производящих полиномов кода R=2/3 (рисунок б) имеет вид

$$\mathrm{G}(x)=\begin{bmatrix} g_{11}(x), g_{12}(x), g_{13}(x) \\ g_{21}(x), g_{22}(x), g_{23}(x) \end{bmatrix}=\begin{bmatrix} 1+x, & 1+x, & 1 \\ 0, & x, & 1+x \end{bmatrix}.$$

Регистры сдвига 1 и 2 (число регистров равно k_0) имеют по две ячейки памяти и три сумматора по модулю 2 (число сумматоров равно n_0), формирующих символы кода в соответствии с видом производящих полиномов.

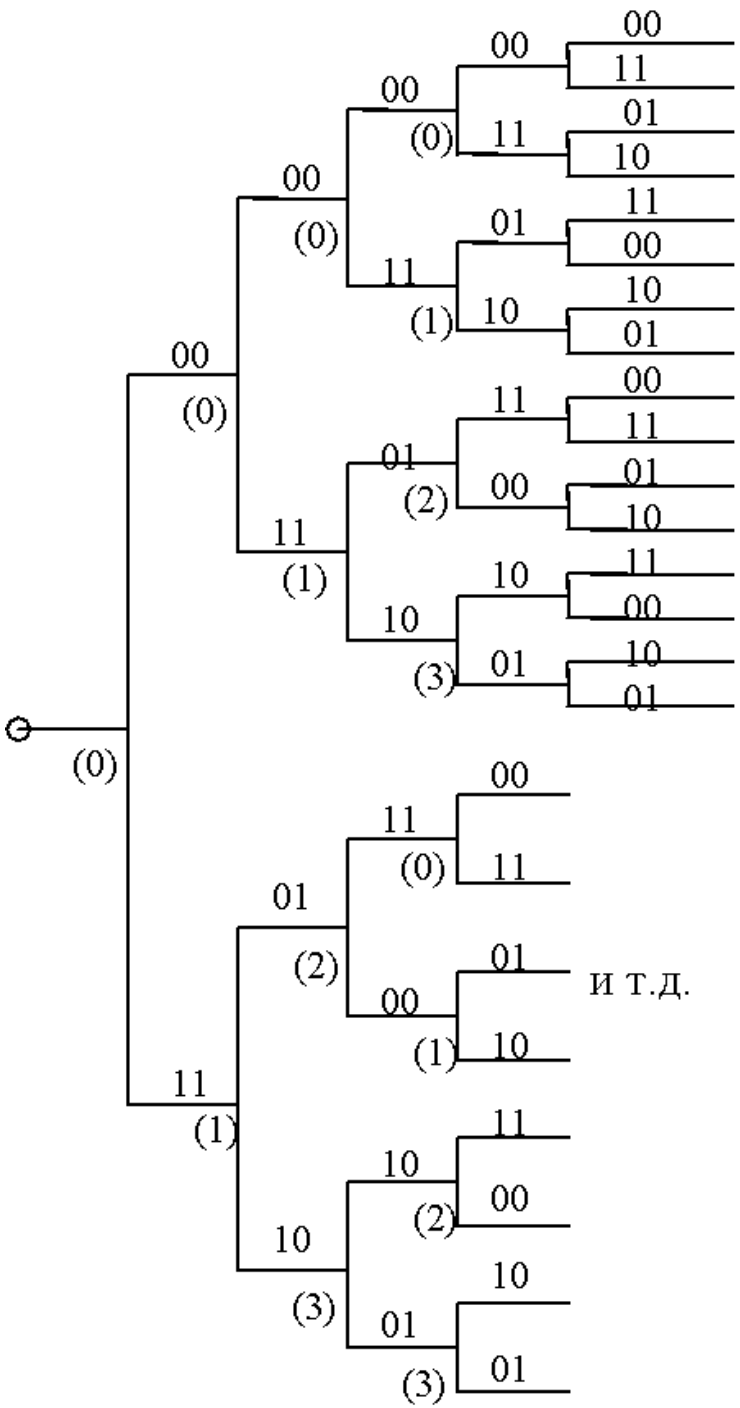
Переключатель Пвх разделяет входные информационные символы между регистрами, переключатель Пвых формирует кодовую последовательность на выходе кодера из выходных символов сумматоров.

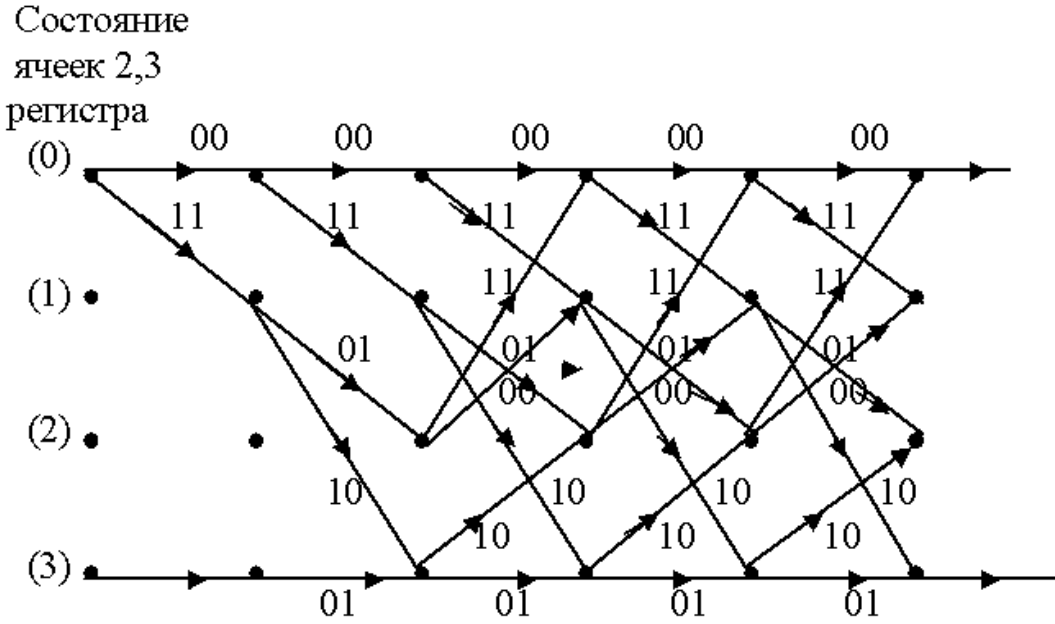
Для описания сверточных кодов применяются способы :

– с помощью кодового дерева или решетчатой структуры;

– с помощью разностных треугольников.

Кодовое дерево рассматриваемого кода R=1/2, и соответствующая ему кодовая решетка имеют вид, показанный на рисунке.


(0)
00
(0)
11
(1)
00
(0)
11
(1)
00
(0)
11
(1)
01
(2)
10
(3)
00
11
01
10
11
00
10
01
00
11
01
00
10
11
10
01
00
11
01
10
(0)
01
(2)
10
(3)
11
(1)
00
(0)
11
(1)
01
(2)
00
(1)
11
10
(2)
10
01
(3)
00
11
01
10
и т.д.


Состояние
ячеек 2,3
регистра
(0)
(1)
(2)
(3)
00
11
01
10

Кодовое дерево строится таким образом, что информационному символу «0» соответствует перемещение на верхнюю ветвь (ребро) дерева, а информационному символу «1» – на нижнюю ветвь. Можно обратить внимание, что после формирования четырех вершин (на рисунке отмечены цифрами 0,1,2,3 в скобках) структура ветвей дерева повторяется.

Это обстоятельство определяется состоянием двух последних ячеек памяти регистра сдвига кодера (00,01,10,11); в общем случае число состояний зависит от кодового ограничения кода и равно

$$2^{n_a - 1}$$

Решетка сверточного кода представляет состояния кодера в виде четырех уровней, а ветви дерева являются ребрами решетки, в результате чего избыточные части дерева отождествляются. Такое представление кода является более удобным при разработке и описании процессов декодирования.

Если сверточный код является систематическим, то *g*1(*x*) = 1 (в верхней ветви кодера рисунка (а) отсутствует член x^2 суммы) и информационная последовательность становится частью выходной последовательности без кодирования.

Для кодера рис. (а) длина кодового ограничения равна 3. Эта величина означает, что поступивший на вход регистра символ на протяжении трех тактовых интервалов входного сигнала будет оказывать влияние на формируемые выходные (т.е. кодовые) символы.

Таким образом, действие одного информационного символа, поступившего на вход кодера, ограничено тремя тактовыми интервалами, т.е. от момента поступления символа на вход первой ячейки регистра до момента его выхода из регистра.

Существует несколько способов описания связей между разрядами в регистре сдвига и сумматорами по модулю 2:

1. Один из этих способов заключается в определении n–векторов связи $\overline{g}_1, \overline{g}_2, \ldots, \overline{g}_n$, где n-количество сумматоров в составе кодера.

Каждый вектор имеет n_a составляющих из нулей и единиц (количество разрядов в регистре сдвига) и описывает связь разрядов регистра сдвига кодера с соответствующим сумматором по модулю 2.

Единица (1) на i-й позиции вектора означает, что разряд с номером i связан с сумматором, а нуль (0) означает, что связи между разрядом с номером i и сумматором не существует.

Так, для кодера на рис.(а) число сумматоров n=2 и будет вектор связи для верхнего сумматора $\overline{g}_1$ и вектор связи $\overline{g}_2$ для нижнего сумматора. С учетом сказанного эти векторы связи будут иметь вид

$$\overline{g}_1 = 101$$

$$\overline{g}_2 = 111$$

2. Второй способ позволяет представить связи между разрядами регистра и сумматорами в виде набора из n полиноминальных генераторов $g_1(x), g_2(x), ..., g_n(x)$, где n-количество сумматоров.

В зависимости от того, имеется ли связь между соответствующими разрядами регистра сдвига и сумматором, в каждом слагаемом полинома коэффициенты принимают только два значения 1 и 0.

Для кодера рис.(а) полиноминальные генераторы будут иметь следующий вид:

$$g_1(x) = 1 \cdot x^0 + 0 \cdot x^1 + 1 \cdot x^2 = 1 + x^2$$

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

С помощью полиноминальных генераторов легко определить кодовые символы на выходе кодера, когда на его вход поступает заданная последовательность информационных символов.

Пусть, например, на вход кодера поступает последовательность информационных символов $\overline{a} = 111000...$

Этой последовательности соответствует полином

$$a(x) = 1 \cdot x^0 + 1 \cdot x^1 + 1 \cdot x^2 + 0 \cdot x^3 + 0 \cdot x^4 +$$

$$+ 0 \cdot x^5 + ... = 1 + x + x^2$$

Полином b(x), соответствующий кодовым символам на выходе кодера, можно определить следующим образом.

Сначала найдем произведения

$$a(x)\cdot g_1(x) \quad , \quad a(x)\cdot g_2(x)$$

$$a(x)\cdot g_1(x) = (1+x+x^2)\cdot(1+x^2) =$$

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

$$= 1+1\cdot x+0\cdot x^2+1\cdot x^3+1\cdot x^4$$

$$a(x)\cdot g_2(x)=(1+x+x^2)\cdot(1+x+x^2)=$$

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

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

$$=1+0\cdot x+1\cdot x^2+0\cdot x^3+1\cdot x^4$$

(значения сумм в круглых скобках определяем по модулю 2).

Полином b(x), коэффициентами которого будут кодовые символы на выходе кодера, определим сложением

$a(x)\cdot g_1(x)$ **и** $a(x)\cdot g_2(x)$ **(здесь сложение не по модулю 2)**

$$a(x)\cdot g_1(x)=1+1\cdot x+0\cdot x^2+1\cdot x^3+1\cdot x^4$$

$$a(x)\cdot g_2(x)=1+0\cdot x+1\cdot x^2+0\cdot x^3+1\cdot x^4+$$

$$b(x)=(1,1)+(1,0)x+(0,1)x^2+(1,0)x^3+(1,1)x^4$$

Последовательность $\overline{b}$ кодовых символов определяется двойными коэффициентами в круглых скобках полинома b(x), т.е.

$\overline{b}$ **=11 10 01 10 11 00 00….**

Построение решетчатой диаграммы

Решетчатая диаграмма также состоит из ребер и узлов. Все узлы решетки, расположенные вдоль верхней горизонтали, имеют одно и тоже состояние a=00. Узлы, расположенные вдоль второй горизонтали, также имеют одно состояние b=10, вдоль третьей – c=01, вдоль четвертой – d=11.

При построении решетки, как и для древовидной диаграммы, предполагается, что первоначально ячейки регистра сдвига кодера содержали одни нули, т.е. кодер вначале находится в состоянии a=00. Поэтому построение решетки начинается с левого узла a=00 (в верхней горизонтали решетки).

Если на вход кодера, находящегося в состоянии a=00, поступает информационный символ 0 или 1, то на выходе появляются соответственно 00 или11. Поэтому из узла «*a*» проводим два ребра, обозначенных соответственно 00 и 11. При этом ребро 00, соответствующее отклику кодера на символ 0, идет выше ребра 11, соответствующее отклику кодера на символ 1.

На 1-м уровне имеется два узла «a» и «b», из которых выходит 4 ребра. Из узла «a» выходят ребра 00 и 11.
Из узла «b» выходят ребра 10 (отклик кодера на нулевой символ и это ребро идет выше ребра 01) и 01(отклик кодера на единичный символ).
На 2-м уровне уже задействованы 4 узла с состояниями a, b, c, d.

На 3-м уровне наблюдается принципиальное отличие древовидной и решетчатой диаграмм. На древовидной на 3-м уровне расположено 8 узлов: по два каждого узла. На решетчатой диаграмме – количество узлов не изменилось по сравнению со 2-м уровнем, т.е. осталось равным 4. Два узла «а» на древовидной диаграмме отождествляются и превращаются в один узел «а» на решетчатой диаграмме.

Аналогично происходит и с другими узлами «b»,«c» и «d».

На 4-м уровне на древовидной диаграмме отождествляются четыре узла «a», четыре узла «b», четыре узла «c», четыре узла «d» и превращаются соответственно в один узел «a», в один узел «b», в один узел «c» и в один узел «d» на решетчатой диаграмме.

Таким образом, в результате описанных отождествлений получается решетчатая диаграмма, на которой на любом уровне после 3-го имеется всего 4 узла.

В общем случае (при любом n_a) число вершин в решетчатой диаграмме не растет, а остается равным

$$2^{n_a-1}$$

В рассматриваемом примере кодера

$$n_a = 3$$

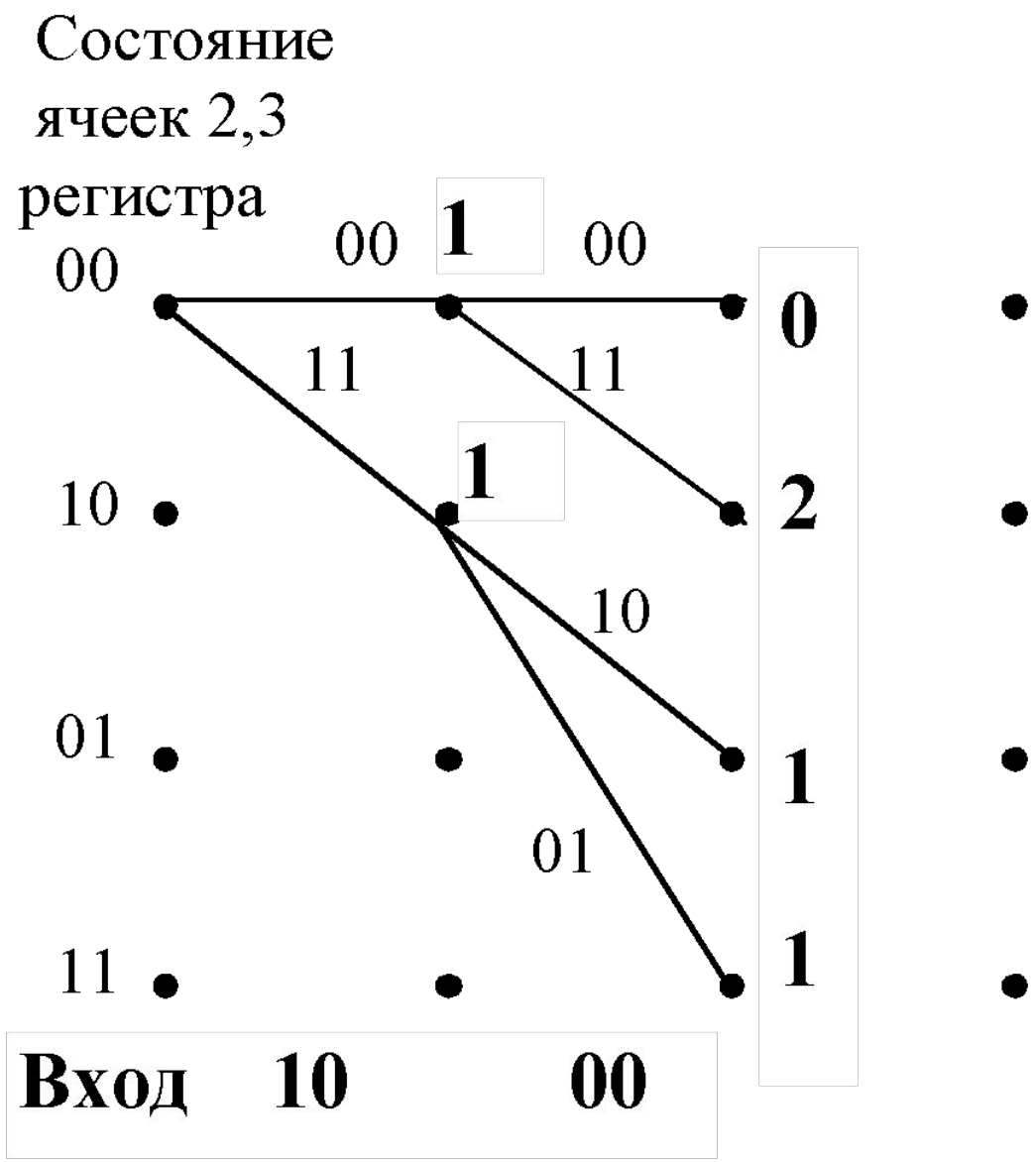
Поэтому получаем

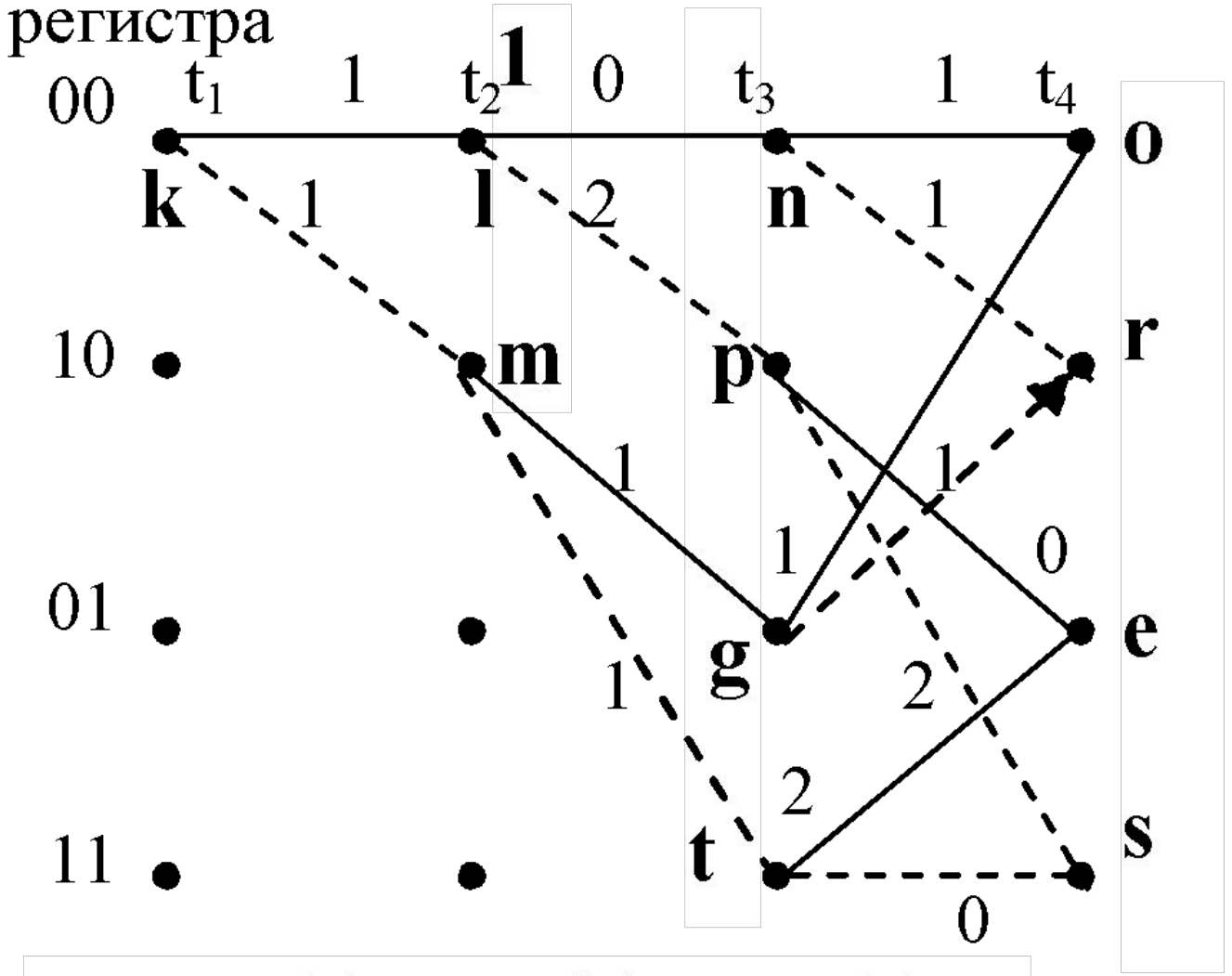
$$2^{n_a - 1} = 2^{3-1} = 4$$

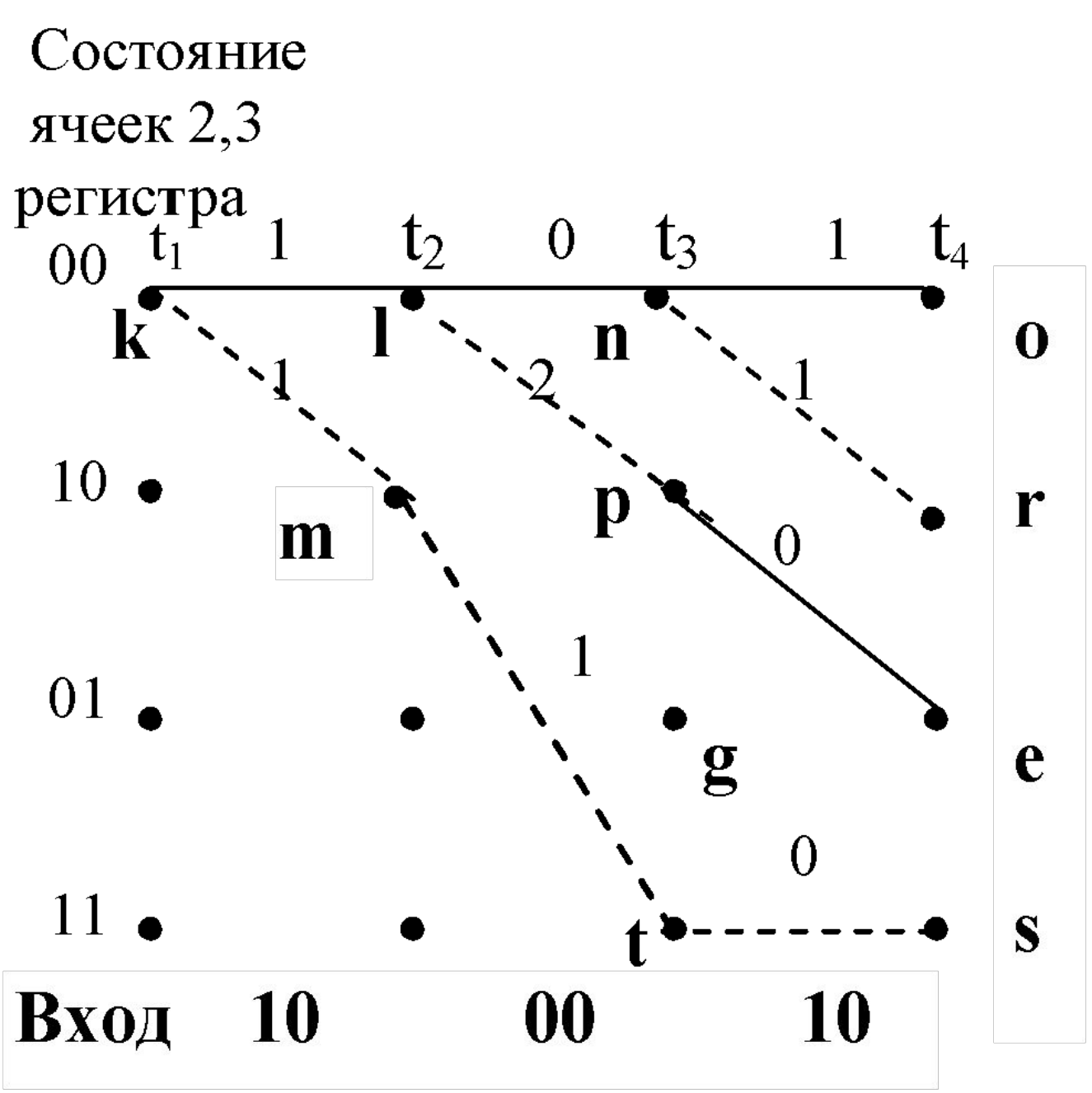
Алгоритм сверточного декодирования Витерби

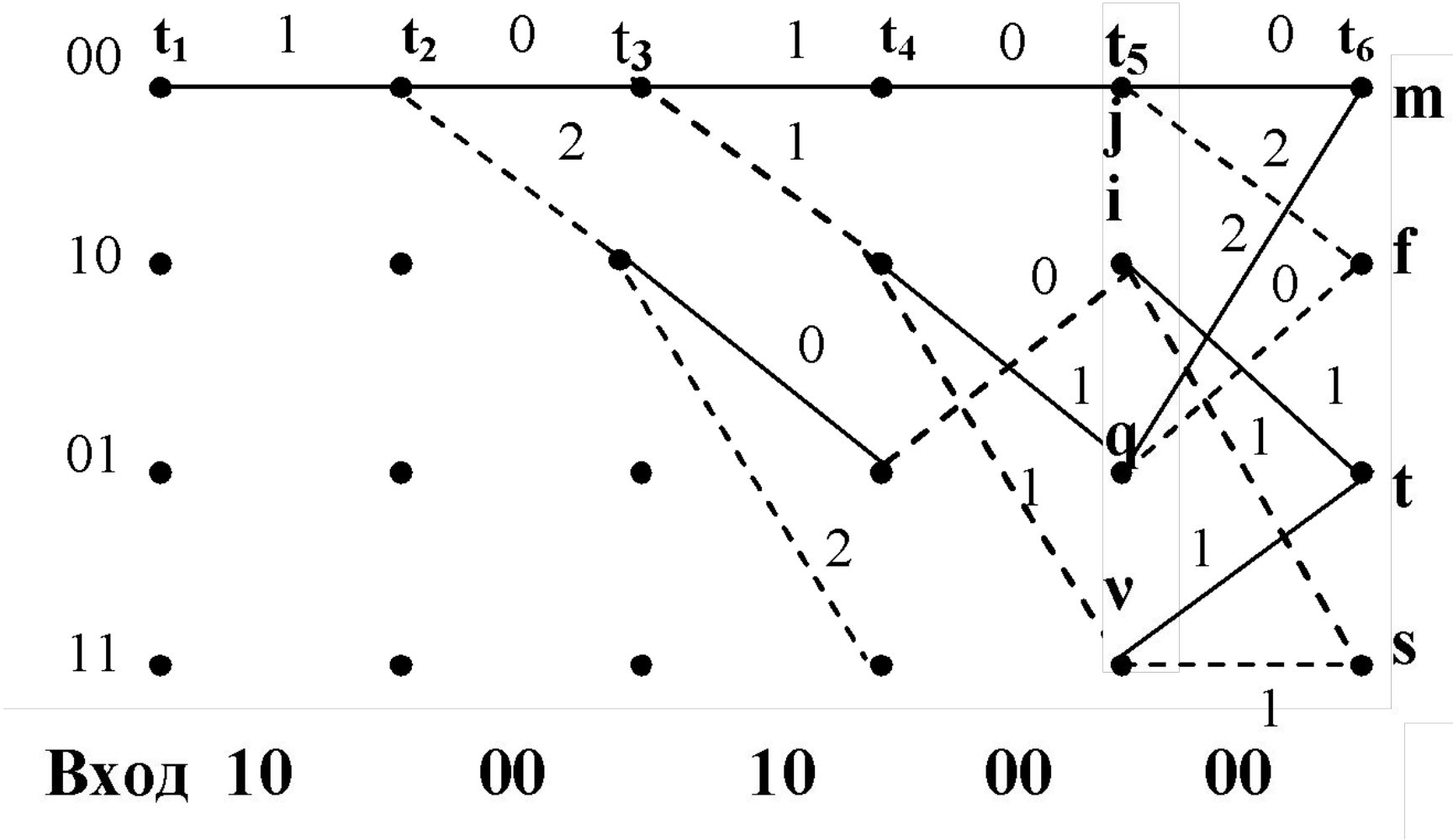
Пример декодирования по алгоритму Витерби R=1/2

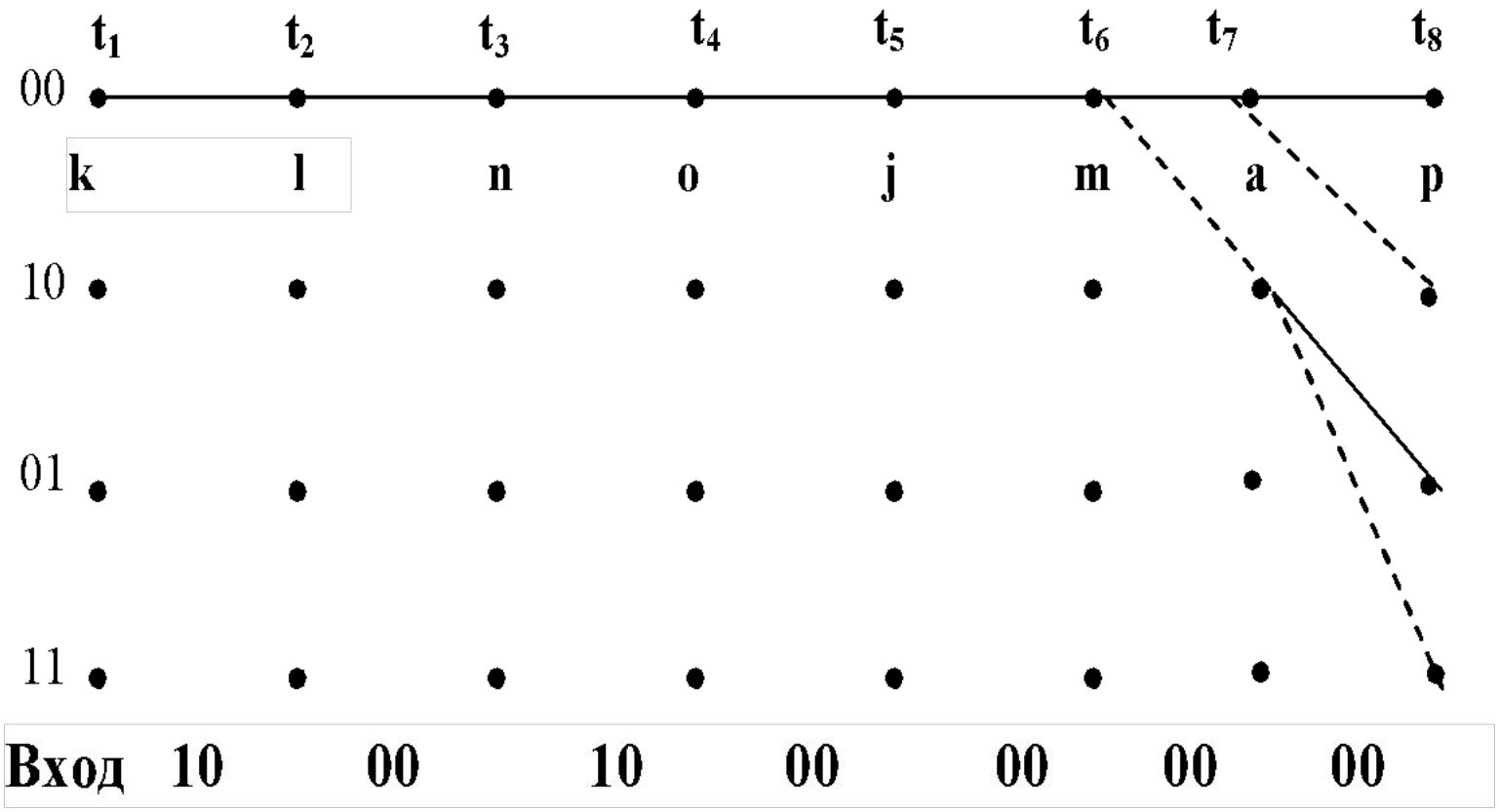
Предположим, что передавалось нулевая кодовая последовательность вида {00 00 00 00 …}, а принятая последовательность на выходе демодулятора с жестким решением имеет вид {10 00 10 00 00 …} (произошло две ошибки в информационных символах).


Состояние
ячеек 2,3
регистра
00
10
01
11
00
1
00
11
11
1
10
01
0
2
1
1
Вход
10
00

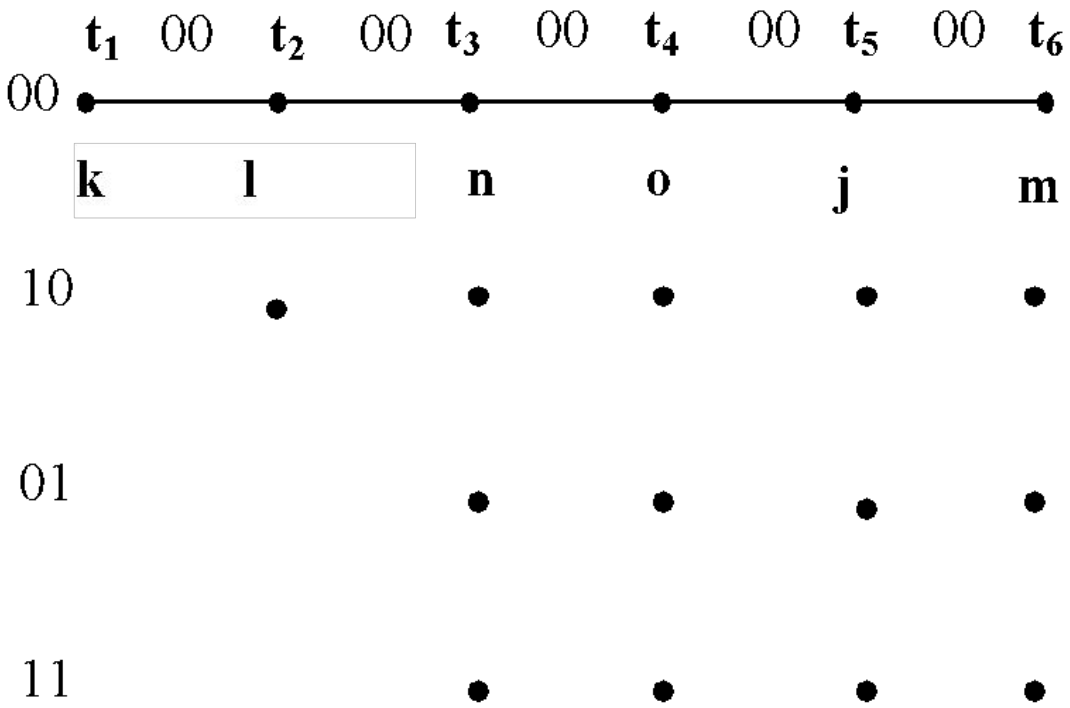

Состояние
ячеек 2,3
регистра
00
10
01
11
t_1
t_2
t_3
t_4
1
0
1
k
l
n
o
r
m
p
e
g
s
t
2
Вход 10 00 10


Состояние
ячеек 2,3
регистра
00
10
01
11
t_1
t_2
t_3
t_4
1
0
1
k
l
n
o
m
p
r
g
e
t
s
1
2
1
0
1
0
Вход
10
00
10


00
10
01
11
t1
t2
t3
t4
t5
t6
1
0
1
0
0
m
j
i
f
q
t
v
s
2
1
2
2
0
0
0
1
1
1
1
1
2
1
Вход 10 00 10 00 00


t1 t2 t3 t4 t5 t6 t7 t8
00
k l n o j m a p
10
01
11
Вход 10 00 10 00 00 00 00

Из построенной диаграммы декодера видно, что от момента t_1 до момента t_6 выжил только один путь k,l,n,o,j,m. Теперь перенесем этот один выживший путь с диаграммы декодера на диаграмму кодера. Этому пути соответствуют обозначения ребер: 00 00 00 00 00.


t1 00 t2 00 t3 00 t4 00 t5 00 t6
00
k l n o j m
10
01
11

Декодер принимает решение, что на интервале от t1 до t6 по каналу передавалась последовательность кодовых символов, соответствующая выжившему пути k,l,n,o,j,m, т.е. 00 00 00 00 00. Таким образом, ошибки, возникшие на выходе демодулятора, оказываются исправленными.

Алгоритм сверточного декодирования Витерби

1.При декодировании используются как решетчатая диаграмма кодера, так и решетчатая диаграмма декодера.

Когда из демодулятора поступает пара принятых символов между моментами времени t_i и t_{i+1}, то определяются расстояния Хэмминга между этой парой символов и парами символов, которыми отмечены ребра решетчатой диаграммы кодера между теми же моментами времени.

Эти расстояния пишут над соответствующими ребрами решетчатой диаграммы декодера. Обозначения на ребрах решетки декодера накапливаются декодером в процессе декодирования.

Т.о. решетчатая диаграмма кодера всегда одна и та же (она не зависит от принятой последовательности), а решетчатая диаграмма декодера определяется как диаграммой кодера, так и принятой последовательностью, т.е. ее вид зависит от принятой последовательности.

2. С помощью пометок (цифр) на ребрах решетчатой диаграммы декодера для момента времени ti определяются расстояния Хэмминга между принятой последовательностью и путями по диаграмме декодера. Все пути начинаются в точке *a* (которой соответствует состояние a=00) и заканчиваются в узлах решетки декодера для момента ti .

Для каждого момента времени t_i (где i >3) имеем четыре узла и в каждый узел приходят два пути (все они начинаются в одной и той же точке *a),* т.е. всего путей будет 8, исходящих из точки a.

Декодирование Витерби состоит в том, что из двух путей, приходящих в один узел, при продолжении операции декодирования выживает только один – тот, которому соответствует меньшее расстояние Хэмминга. Если эти два расстояния имеют одинаковую величину, то произвольно выбирается любой из двух.

Отсекание одного из двух путей, сходящихся в узле решетки, гарантирует, что число продолжающихся путей будет равно числу состояний (т.е. четырем для рассматриваемого кодера). В этом заключается существенное преимущество решетчатой диаграммы при сравнении древовидной диаграммой при декодировании.

В результате использования алгоритма декодирования Витерби находится наиболее вероятный (с минимальным расстоянием Хэмминга) путь через решетку декодера. При определении этого пути происходит исправление ошибок, возникших при приеме передаваемой кодовой последовательности.

Основные трудности при реализации алгоритма Витерби определяются тем, что сложность декодера экспоненциально растет с увеличением кодового ограничения (число состояний декодера равно 2^{n_a-1}); поэтому значение кодового ограничения кодов, применяемых на практике, не превышает $n_a \le 15$. Недвоичные коды декодировать алгоритмом Витерби еще сложнее.

Декодирование по алгоритму Витерби кода R=2/3 оказывается существенно сложнее по сравнению с кодами R=1/2. В каждую вершину кодовой решетки будет входит теперь не два пути, а четыре и для определения выживших путей необходимо производить восьмеричное сравнение.

В общем случае для кодов $R=k_0/n_0$ требуется 2^{k_0} -ичное сравнение, что приводит к значительным усложнениям практической реализации декодеров. С целью упрощения алгоритма декодирования сверточные коды $R=(n_0-1)/n_0$ получают *выкалыванием* кодов $R=1/2$.

Например, для получения кода *R*=3/4 выкалывается каждый третий символ, сформированный кодером, и на выход поступают, соответственно, 1,2,4,5,7,8,10, … символы.

При декодировании выколотых кодов по алгоритму Витерби выколотые ребра решетки воспринимаются как стирания этих ребер в канале, соответственно уменьшается кодовое расстояние для исправления ошибок и увеличивается размер кадра.

Сверточные коды $R=1/n_0$ строятся аналогично кодам $R=1/2$, но имеют большее кодовое расстояние и кодовое ограничение. В кодере вместо двух сумматоров по модулю 2 на выходе ставится n сумматоров. Например, кодер $R=1/3$ должен иметь три сумматора по модулю 2 на выходе по сравнению со схемой, показанной на рисунке (а).

В качестве дополнительного производящего полинома $g_3(x)$ может быть взят $g_1(x)$ или $g_2(x)$.

Изменения алгоритма Витерби состоят в том, что метрики на ребрах решетки вычисляются из расчета n_0 символов на ребре вместо двух. Число состояний решетки остается тем же.

Последовательное декодирование

В отличие от алгоритма Витерби при последовательном декодировании производится продолжение и обновление метрики только одного пути, который представляется наиболее вероятным, при этом делается попытка принять решение о декодируемом символе, принятом в начале пути.

Если решение о декодируемом символе, находящемся в начале пути, принять невозможно, то производится либо движение вперед, т.е. прием очередного символа и обработка метрики данного пути, или возврат назад (выбор другого пути), если значения метрики увеличиваются. Процедура продолжается до тех пор, пока не будет принято решение о декодировании символа, расположенного в начале пути.

<u>Основное достоинство</u>
последовательного алгоритма заключается в том, что в среднем длина пути, достаточная для правильного декодирования меньше, чем у алгоритма Витерби.

Недостатки определяются тем, что длина пути, приводящего к правильному декодированию, является случайной величиной.

Это вызывает затруднения при реализации декодера, так как нельзя определить заранее какой объем памяти потребуется для сохранения метрики рассматриваемого пути, а это значит, что всегда существует вероятность переполнения памяти и сбой декодера.

В практике построения последовательных декодеров применяются два варианта алгоритма последовательного декодирования, позволяющих получить приемлемые для реализации сложность декодирования и объем памяти: алгоритм Фано и стек-алгоритм.

Алгоритм Фано работает с малыми значениями длин кодового ограничения. Стек-алгоритм более прост для понимания и реализации на микропроцессорах, но требует несколько большего объема памяти.

Декодер создает стек, состоящий из просмотренных ранее путей (могут иметь различную длину) и упорядочивает их в соответствии со значением метрики. На каждом шаге продолжается путь, находящийся наверху стека, что порождает 2^{k_0} новых путей со своей метрикой. Затем стек опять упорядочивается. Процедура продолжается до тех пор, пока длина пути, находящегося наверху стека, не станет равной длине декодируемой последовательности.

Пример.

Дано:

передаваемая информационная последовательность *I*(*x*)=100, скорость кода *R*=1/2, *n*$_a$=3, производящие полиномы кода

$$g_1(x) = 1 + x + x^2$$

$$g_2(x) = 1 + x^2$$

последовательность ошибок *e*(x)=010000.

Найти: оценку переданного слова с использованием стек-алгоритма декодирования сверточных кодов.

Решение:

переданная кодовая последовательность равна $A(x)=110111$.

С учетом последовательности ошибок $e(x)$ принятое слово имеет вид $B(x)=100111$.

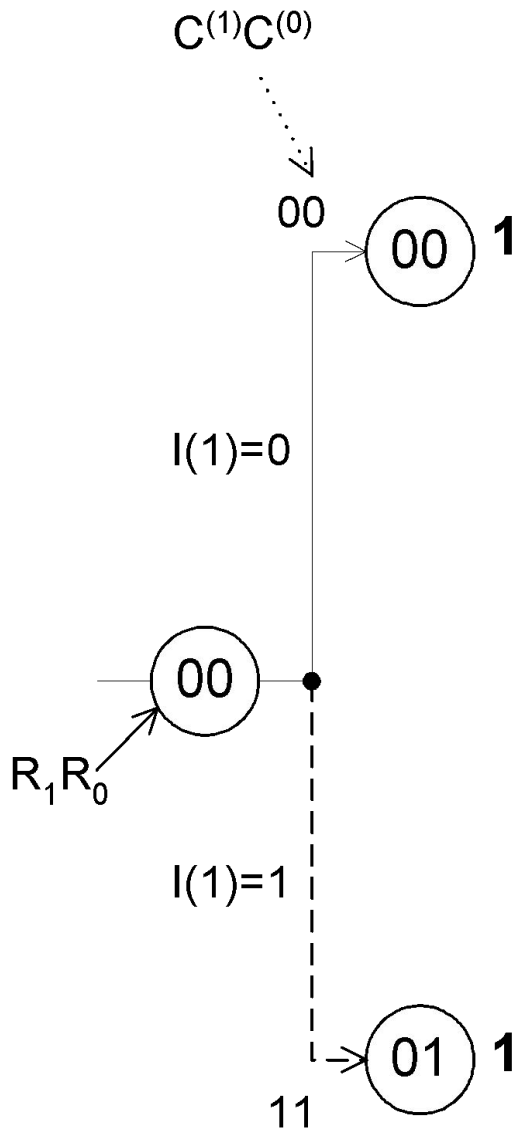
Декодирование осуществляется за несколько шагов, где состояния представляют собой возможные состояния триггеров регистра кодера $R_1(x)$ и $R_0(x)$, ветви соответствуют выходам кодера $C_1(x)$ и $C_0(x)$. После каждого состояния показана метрика ветви. Метрика последней ветви данного пути соответствует метрике пути.

Принятая последовательность:

B=<u>10</u>.01.11

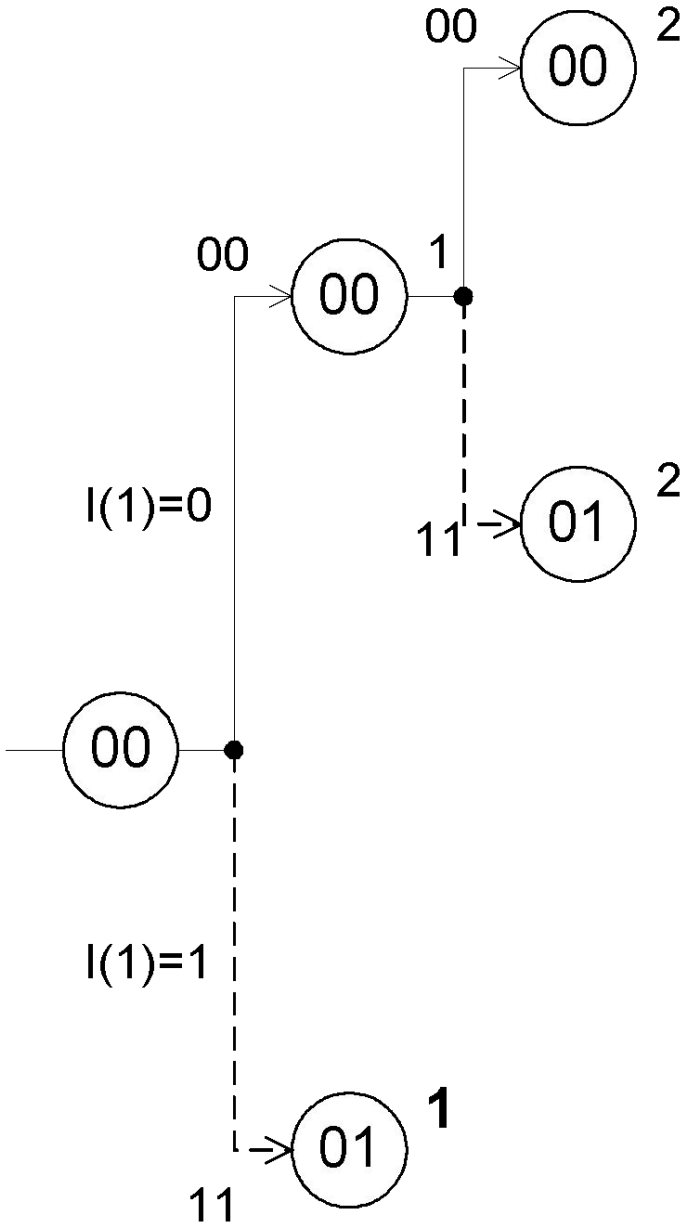
Стек «оценка информационной последовательности – метрика пути»:

- 0 – 1
- 1 – 1



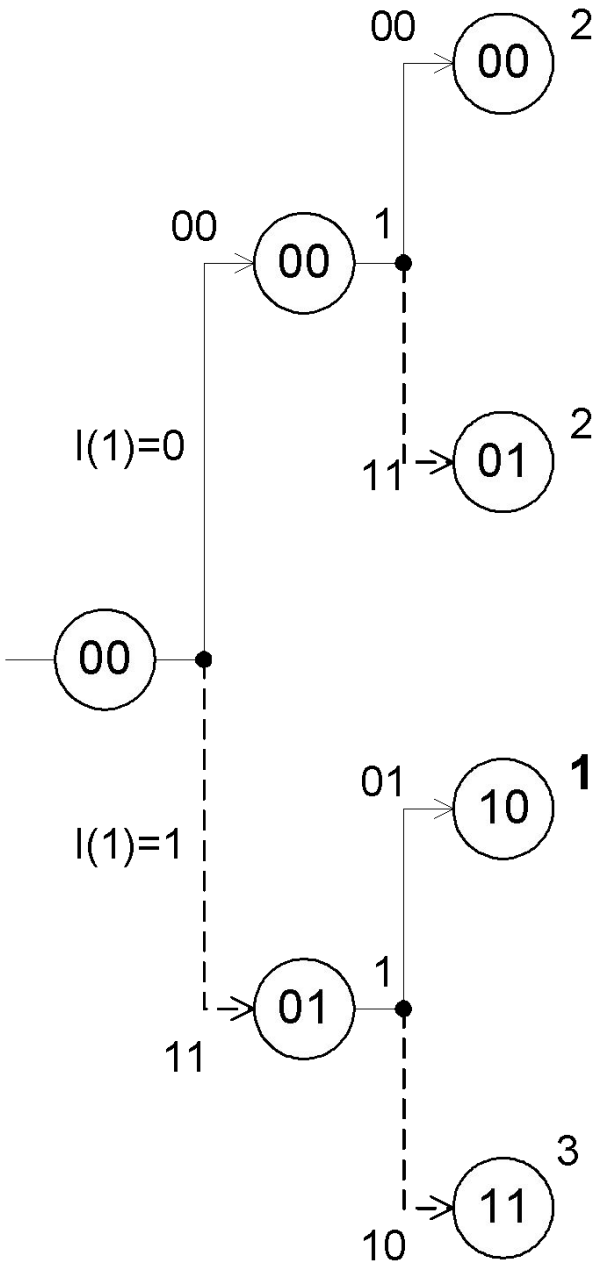
B=10.**<u>01</u>**.11

- 1 – 1
- 00 – 2
- 01 – 2



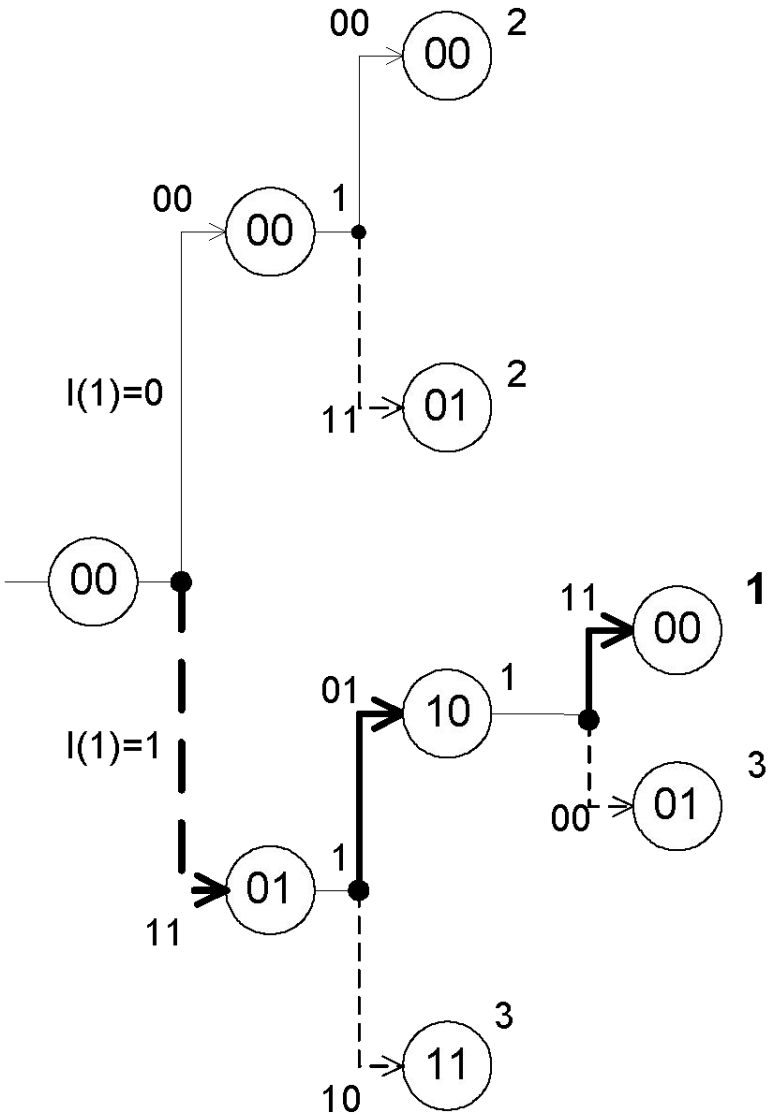
B=10.**<u>01</u>**.11

- 10 – 1
- 00 – 2
- 01 – 2
- 11 – 3



B=10.01.**11**

- 100– 1 (декод.)
- 00 – 2
- 01 – 2
- 101 – 3
- 11 – 3



Выводы:

- **Последовательное декодирование позволяет использовать большие значения длины кодового ограничения (на практике, 100>K>10), что повышает помехоустойчивость приема.**
- **Продолжительность декодирования зависит от значения отношения сигнал/шум.**
- **Стек-алгоритм применяется в приложениях с низкой скоростью передачи, возможна реализация декодера практической системы в программном исполнении.**

Синдромное декодирование

Синдромное декодирование сверточных кодов, в принципе, не отличается от синдромного декодирования циклических кодов.

Вначале декодер по принимаемой последовательности символов вычисляет вектора синдромов ошибки, когда $R \neq 1/2$, или один синдром ошибки, когда $R = 1/2$. Затем путем анализа синдромов определяется вектор (или символ) ошибки и производится коррекция соответствующих информационных символов входной последовательности.

Практически используются, в основном, два метода синдромного декодирования:

- **декодирование с табличным поиском;**
- **пороговое декодирование.**

Декодирование с табличным поиском **заключается в том, что вычисленный синдром ошибки сравнивается с таблицей всевозможных синдромов ошибок данного кода, для каждого из которых символ ошибки заранее определен. После обнаружения подобного синдрома в таблице остается только выполнить коррекцию информационной последовательности.**

Применение:

Сверточные коды с малым кодовым ограничением ($n_a \leq 10 \div 20$) и с малым энергетическим выигрышем (1÷2,5 дБ)

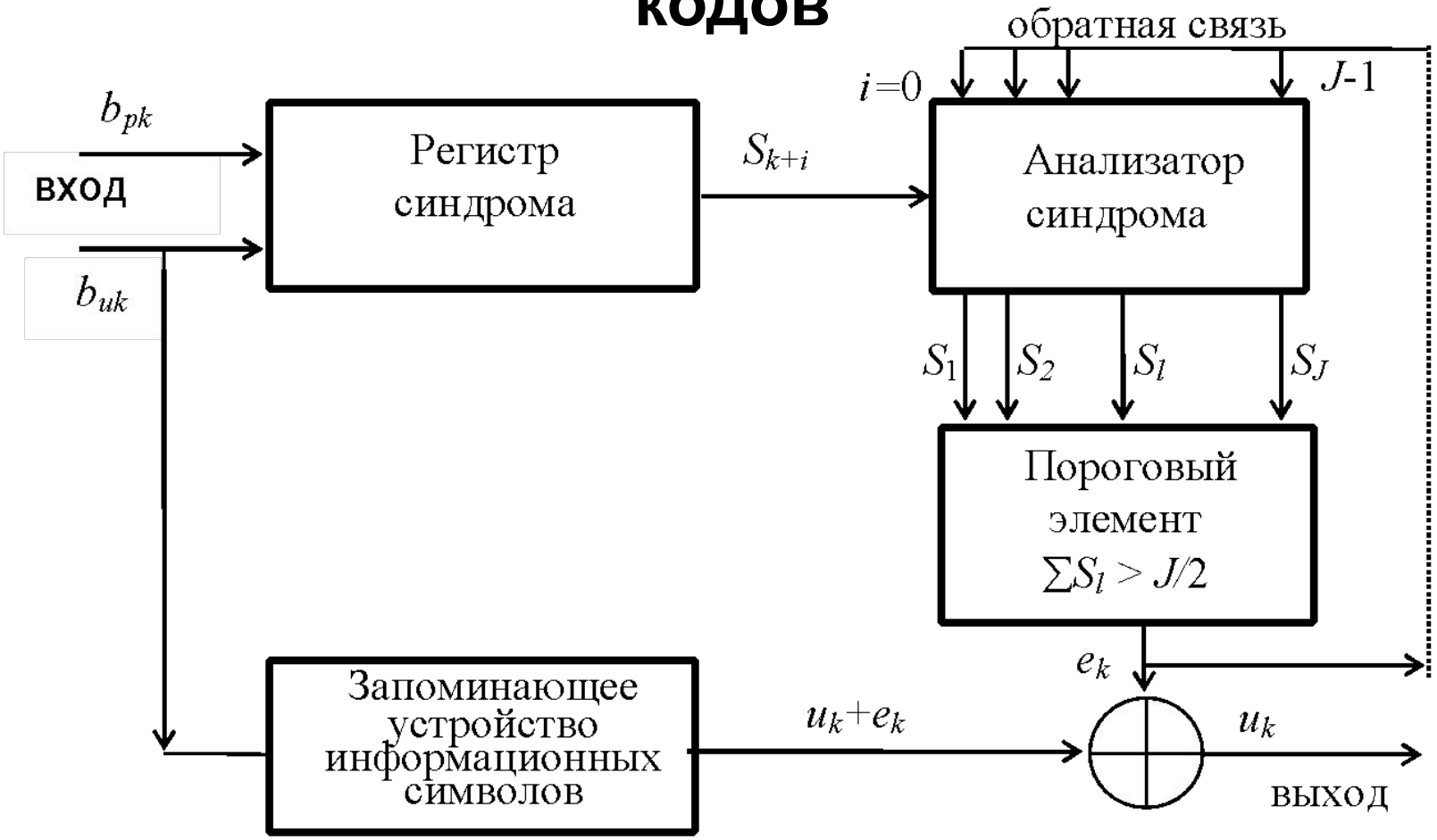
Пороговое декодирование **возможно для определенного класса линейных кодов как блочных, так и сверточных, позволяющих получить так называемые разделенные (или ортогональные)** ***проверки на четность*****.**

Число проверочных уравнений *J* и число исправляемых ошибок *t* связаны соотношением

$$J \geq 2t,$$

кодовое расстояние - $d = J + 1$.

Пороговое декодирование сверточных кодов



Структурная схема порогового декодера

Последовательность символов канала b_k после разделения на информационные b_{uk} и проверочные b_{pk} поступает в регистр синдрома, где производится вычисление элементов синдрома S_k .

В анализаторе синдрома формируются ортогональные проверки S_l (l=1…J), сумма которых в пороговом устройстве сравнивается с уровнем порога и определяется значение символа ошибки e_k .

Исправление ошибки происходит в сумматоре по модулю два, на второй вход которого подаются информационные символы канала из запоминающего устройства, выполняющего функцию хранения этих символов на время вычисления символа ошибки.

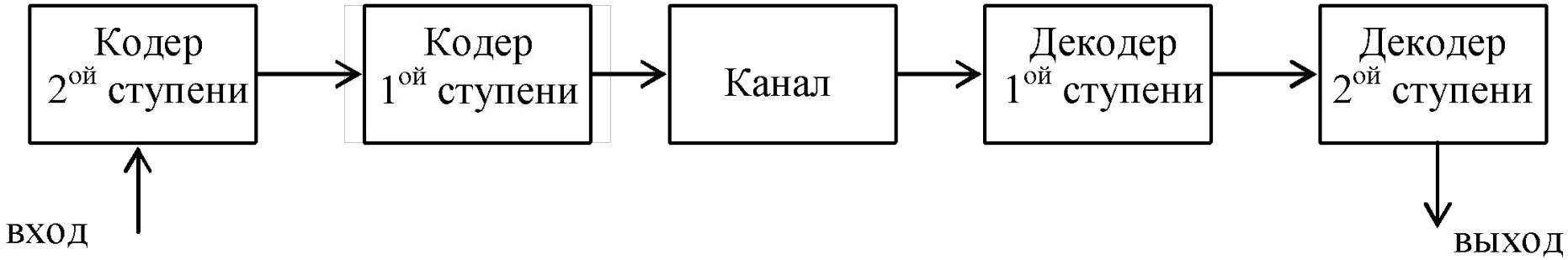
КАСКАДНЫЕ КОДЫ

Каскадные коды **используются в практике передачи дискретных сигналов в качестве методов реализации кодов большой длины и высокой корректирующей способности.**

Эта цель может быть достигнута применением нескольких ступеней кодирования.

Наибольшее распространение получили две ступени кодирования различными кодами.

Последовательные каскадные коды



Вторая ступень кодирования в последовательном каскадном кодеке формирует k_2 строк, каждая из которых состоит из k_1 информационных символов кода (n_1, k_1) 1-ой ступени (внутреннего кода).

В каскадном коде каждая последовательность из k_1 двоичных символов внутреннего кода представляется как один символ недвоичного кода (основание этого кода равно $g = 2^{k_1}$) 2-ой ступени (внешнего кода);

К k_2 информационным символам внешнего кода приписывается (n_2-k_2) проверочных символов, каждый из которых также состоит из k_1 двоичных символов; в результате образуется кодовое слово внешнего кода (n_2, k_2).

Затем каждая строка из k_1 двоичных символов кодируется кодом (n_1, k_1) и к каждой строке приписывается (n_1- k_1) проверочных символов кода 1-ой ступени (внутреннего кода).

При этом следует иметь в виду, что кодовое слово кода 1-ой ступени может составлять только часть символа кода 2-ой ступени, тогда основание этого кода будет равно $g = 2^{b \cdot k_1}$, где *b*- целое число.

В другом методе формирования каскадного кода (иногда называемого *итеративным* кодом)

k_2 информационных блоков, каждый из которых состоит из k_1 двоичных символов кода 1-ой ступени (внутреннего кода) записываются в виде строк матрицы

Каждый ее столбец образует k_2 символов, которые являются информационными символами кода (n_2, k_2) 2-ой ступени (внешнего кода). Затем, как и при каскадном кодировании, каждая строка из k_1 двоичных символов кодируется кодом (n_1, k_1) и к каждой строке приписывается (n_1-k_1) проверочных символов кода 1-ой ступени (внутреннего кода).

В результате и в том и другом случае образуется блок двоичных символов (матрица $n_1 \times n_2$) длиной $n_1 \cdot n_2$, содержащий $k_1 \cdot k_2$ информационных символов, а общее кодовое расстояние равно произведению кодовых расстояний внешнего и внутреннего кодов.

$$
\left|\begin{array}{cccccc}
a_{11} & a_{12}\ldots & \ldots a_{1k_1} & b_{1(k_1+1)} & b_{1(k_1+2)}\ldots & \ldots b_{1n_1} \\
a_{21} & a_{22}\ldots & \ldots a_{2k_1} & b_{2(k_1+1)} & b_{2(k_1+2)}\ldots & \ldots b_{2n_1} \\
---- & ------ & ------ & ------ & ------ & ----- \\
a_{k_2 1} & a_{k_2 2}\ldots & \ldots a_{k_2 k_1} & b_{k_2(k_1+1)} & b_{k_2(k_1+2)}\ldots & \ldots b_{k_2 n_1} \\
b_{(k_2+1)1} & b_{(k_2+1)2}\ldots & \ldots b_{(k_2+1)k_1} & b_{(k_2+1)(k_1+1)} & b_{(k_2+1)(k_1+2)}\ldots & \ldots b_{(k_2+1)n_1} \\
b_{(k_2+2)1} & b_{(k_2+2)2}\ldots & \ldots b_{(k_2+2)k_1} & b_{(k_2+2)(k_1+1)} & b_{(k_2+2)(k_1+2)}\ldots & \ldots b_{(k_2+2)n_1} \\
---- & ----- & ----- & ------ & ------- & ----- \\
b_{n_2 1} & b_{n_2 2}\ldots & \ldots b_{n_2 k_1} & b_{n_2(k_1+1)} & b_{n_2(k_1+2)}\ldots & \ldots b_{n_2 n_1}
\end{array}\right|,
$$

где a_{ij} – информационные символы, b_{ij} – проверочные символы

В качестве кодов 1-ой и 2-ой ступеней могут использоваться как блочные, так и сверточные коды. При этом на первой ступени целесообразно применить такой код, который обеспечивает уменьшение вероятности ошибки при возможно меньшем отношении сигнал/шум в канале связи. Это могут быть ортогональные (или биортогональные) коды при $g = 2^{k_1}$ - ичной модуляции, короткие блочные или сверточные коды.

В качестве внешнего кода могут применяться как блочные (например, циклические), так и сверточные коды. Одним из кодов 2-ой ступени (внешним кодом) часто используются недвоичные циклические коды Рида-Соломона (коды РС), которые являются кодами с максимальным кодовым расстоянием

$$d = n_2 - k_2 + 1$$

Вероятность ошибки на выходе декодера кодов РС в каналах с независимыми ошибками

$$p_д < \frac{2^{k_1-1}}{2^{k_1}-1} \sum_{i=t+1}^{n_2} \frac{t+i}{n_2} C_{n_1}^{i} \left(1-p_1\right)^{n_2-i},$$

где t – кратность ошибок, исправляемых внешним кодом, p_1 – вероятность ошибки на выходе декодера 1-ой ступени, $(2^{k_1-1}/2^{k_1}-1)$ – множитель, учитывающий среднее число ошибок в двоичных символах на одну ошибку недвоичного символа кода РС.

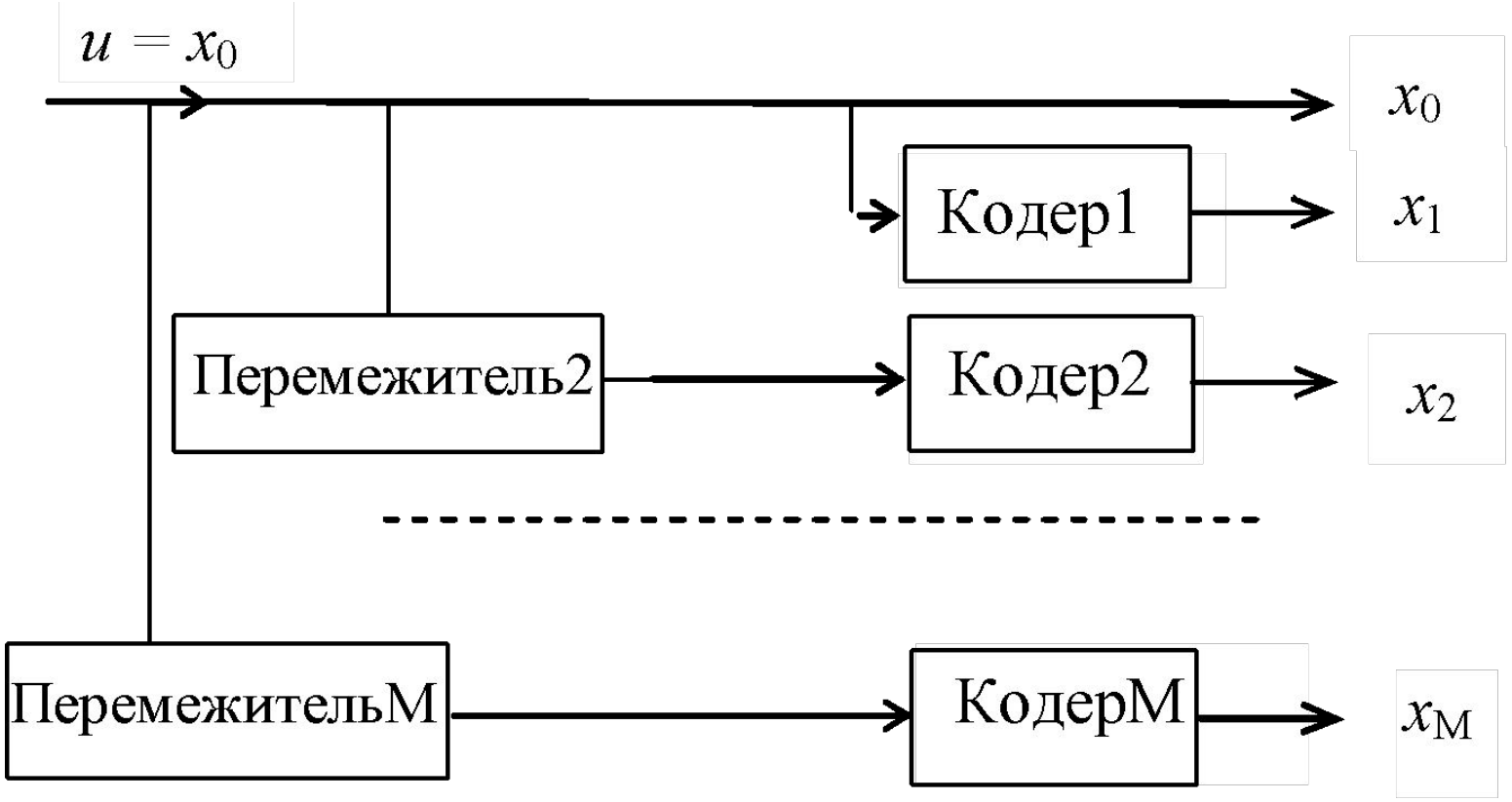
Каскадное кодирование с внешним кодом РС (или сверточным кодом с итерационным декодированием) и внутренним сверточным кодом *R*=1/2 (с декодером Витерби) позволяют работать в канале с отношением сигнал/шум 2,0…2,5дБ, обеспечивая вероятность ошибки декодирования $p_d = 10^{-5}$ **.**

Параллельные каскадные коды

Параллельные каскадные коды с итерационным декодированием, обычно называют *турбокодами*.

Турбокод может рассматриваться как блочный код с очень большой длиной блока. В компонентных кодерах турбокодека могут использоваться коды Хемминга, БЧХ, Рида-Соломона и сверточные, работающие в блочном режиме.

Турбокод образуется при параллельном каскадировании двух или более систематических кодов. Обычно используются два или три первичных (компонентных) кода, а соответствующие им турбокоды называются *двумерными или трехмерными.*



Структурная схема турбокодера

Блок данных *u* длиной *k* символов поступает сначала на вход турбокодера. Последовательность символов $x_0 = u$ поступает на систематический выход турбокодера и параллельно на M ветвей, состоящих из последовательно включенных перемежителя и компонентного кодера.

В этой схеме передаваемая информация совместно используется во всех компонентных кодерах. Каждый перемежитель преобразует структуру последовательности символов X_0 по псевдослучайному закону и выдает пакет на вход соответствующего кодера.

Возможность исправления ошибок зависят не только от минимального кодового расстояния, но и от распределения весов кода, в частности, от числа кодовых слов с низким весом.

Поэтому задачей перемежителя является преобразование входной последовательности таким образом, чтобы последовательности, приводящие к кодовым словам с низким весом на выходе первого компонентного кодера, были преобразованы в другие, порождающие различные кодовые слова с высоким весом на выходах остальных кодеров.

На выходах компонентных кодеров каждой из M ветвей образуются последовательности проверочных символов $X_1 \ldots X_M$. Поскольку информационные последовательности (систематическая часть) на выходах кодеров всех ветвей идентичны с точностью до линейной операции перемежения, в канал передается только одна из них, что существенно повышает скорость передачи и эффективность системы кодирования.

Эта информационная последовательность X_0 мультиплексируется с проверочными последовательностями X_1... X_M, образуя кодовое слово, которое подлежит передаче по каналу.

Скорость кода на выходе турбокодера $R = 1/(M+ 1)$.

Для повышения скорости кода применяют выкалывание (перфорацию) определенных проверочных символов выходной последовательности кодера. В типичном случае после выкалывания в канал передается только половина проверочных символов каждой ветви. Тогда скорость кода возрастает до R =1/(M/2+1).

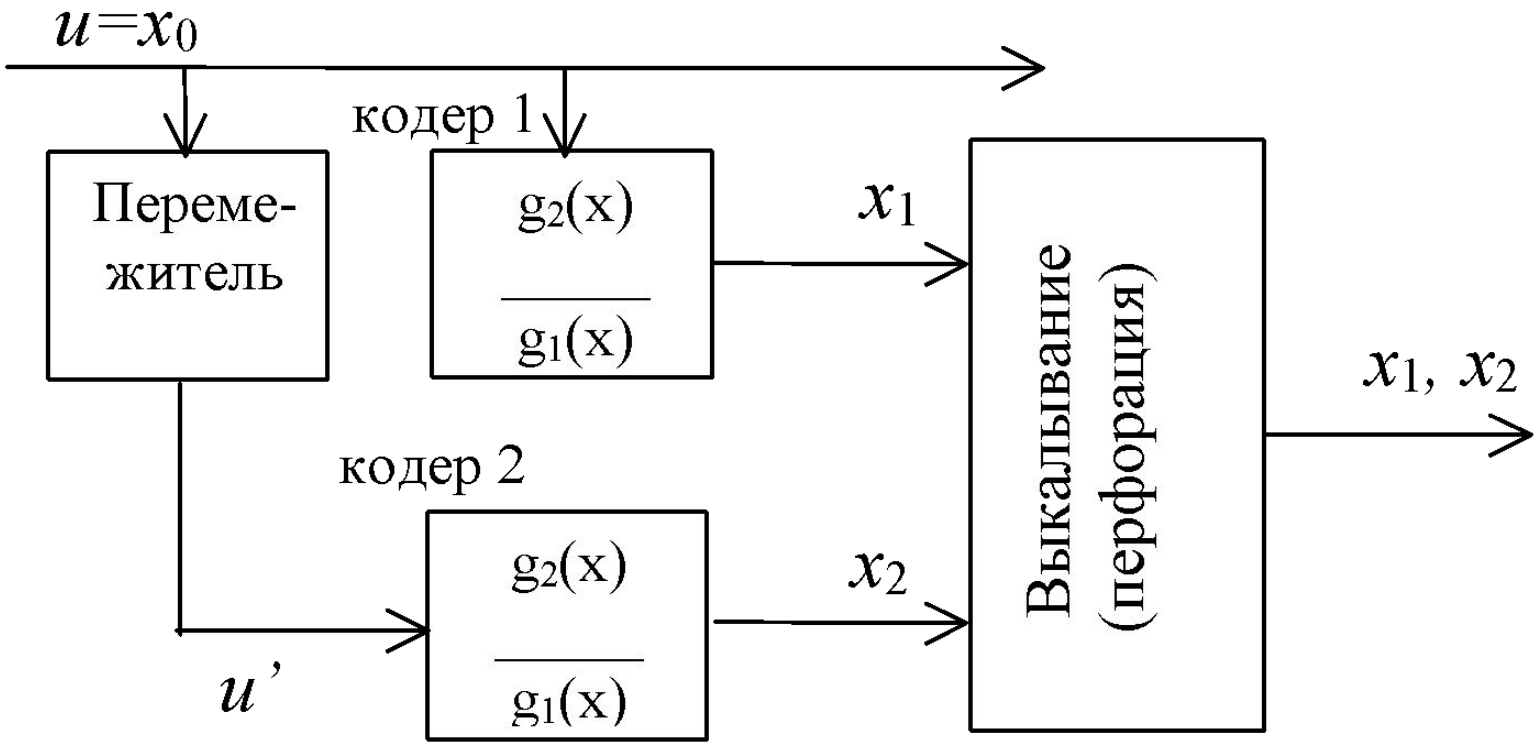
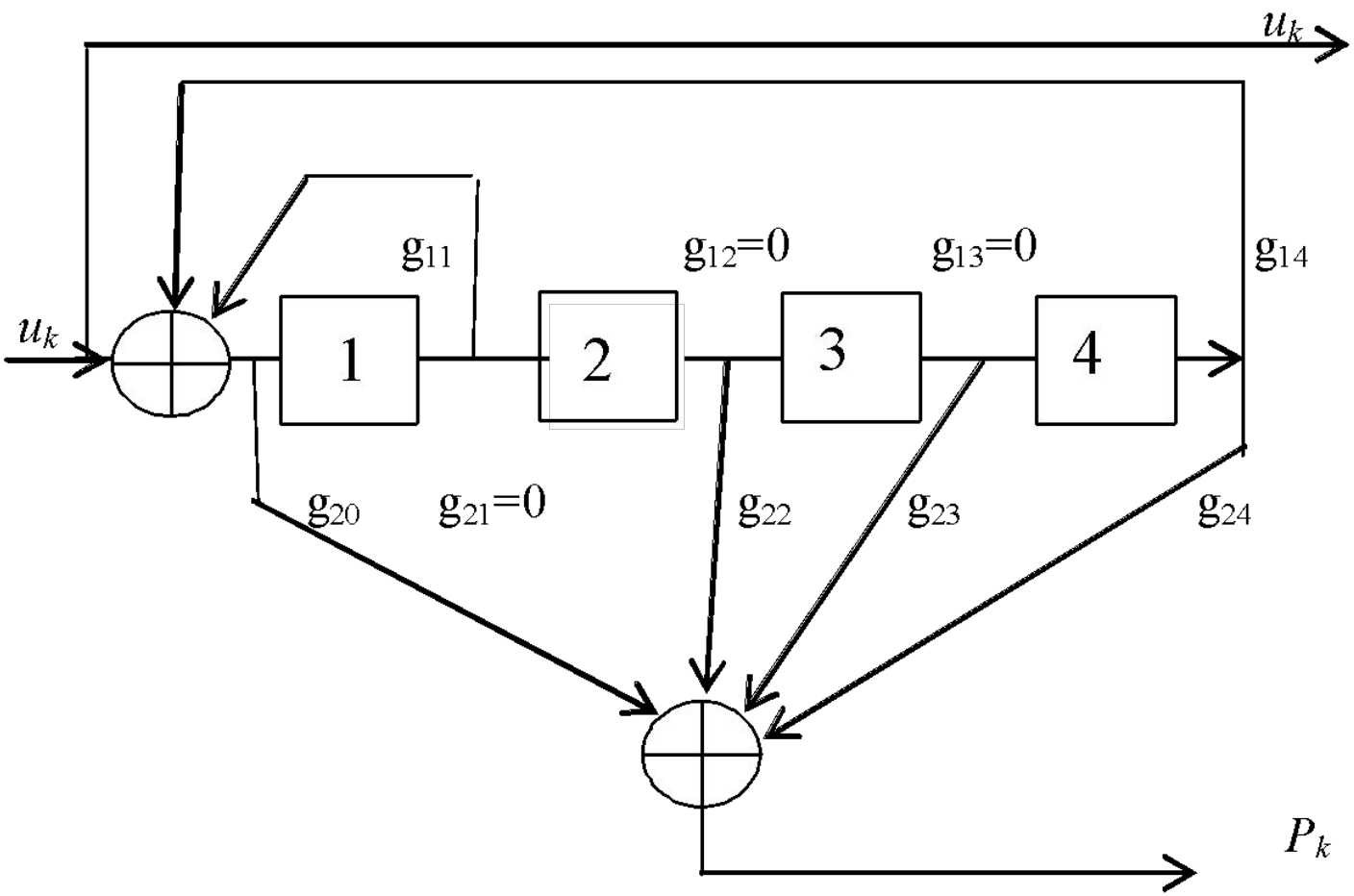


Схема турбокодера
с двумя одинаковыми кодерами

Операция выкалывания сводится к передаче в канал нечетных проверочных символов первого кодера и четных проверочных символов второго.

В совокупности с информационной последовательностью это приводит к результирующей кодовой скорости $R = 1/2$.

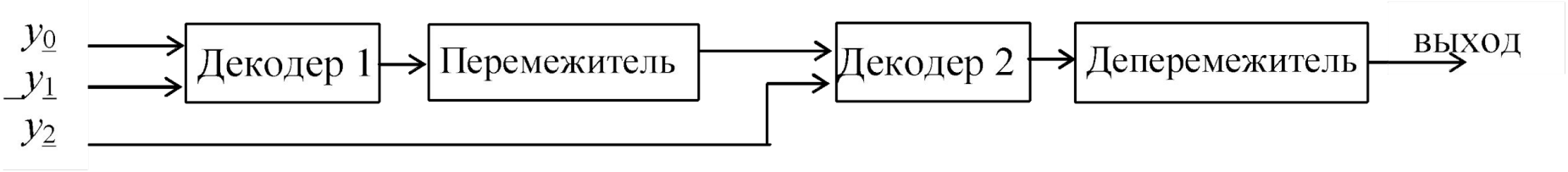
Выкалывание позволяет устанавливать произвольное значение кодовой скорости и даже адаптировать параметры кодера к свойствам канала. Если в канале возрастает уровень шумов, то уменьшение степени выкалывания вносит в кодированный поток дополнительную избыточность и повышает исправляющую способность кода.


u_k
u_k
g_{11}
$g_{12}=0$
$g_{13}=0$
g_{14}
1
2
3
4
g_{20}
$g_{21}=0$
g_{22}
g_{23}
g_{24}
P_k

Декодирование турбокодов.

В основе декодирования кодов, исправляющих ошибки, лежит сравнение вероятностных характеристик различных кодовых слов, а применительно к сверточным кодам с декодированием по алгоритму Витерби — различных путей на решетчатой диаграмме.

Если имеется некоторое предварительное знание о принимаемом сигнале до его декодирования, то такая информация называется априорной и ей соответствует априорная вероятность. В противном случае имеет место только апостериорная информация. При декодировании турбокодов существенным является использование обоих видов информации.



Структурная схема турбодекодера

Рассмотрим процесс декодирования для случая, когда кодирование осуществляется двухкомпонентным турбокодером.

При этом в результате демультиплексирования на входе декодера имеется информационная y_0 и две кодированные проверочные последовательности y_1 и y_2.

После декодирования информационной и первой проверочной последовательностей получается начальная оценка информационной последовательности, которая может использоваться как априорная информация при декодировании второй проверочной последовательности во втором компонентном декодере.

Такой подход требует, чтобы компонентный декодер мог использовать мягкое решение для входных данных ("мягкий" вход) и выдавать данные в непрерывном диапазоне амплитуд или, по крайней мере, без грубого квантования ("мягкий" выход).

КОДЫ БЧХ И РИДА-СОЛОМОНА

Коды *Боуза* — *Чоудхури* — *Хоквингема* (БЧХ) являются подклассом циклических кодов. Их отличительное свойство — возможность построения кода БЧХ с минимальным расстоянием не меньше заданного. Это важно, потому что, вообще говоря, определение минимального расстояния кода есть очень сложная задача.

Кодирующий многочлен g(x) для БЧХ-кода, длина кодовых слов n которого , строится так. Находится примитивный многочлен минимальной степени q такой, $n \leqslant 2^q - 1$ что или $q \geqslant \log_2(n+1)$. Пусть α - корень этого многочлена, тогда рассмотрим кодирующий многочлен $g(x) = \mathrm{НОК}(m_1(x), \ldots, m_{d-1}(x))$,

где - $m_1(x), \ldots, m_{d-1}(x)$ многочлены минимальной степени, имеющие корнями соответственно

$$\alpha, \alpha^2, \ldots, \alpha^{d-1}$$

Построенный кодирующий многочлен производит код с минимальным расстоянием между кодовыми словами, не меньшим d , и длиной кодовых слов n.

Для кодирования кодами БЧХ применяются те же методы, что и для кодирования циклических кодов.

Пример

Построить код БЧХ при n=7 для исправления независимых ошибок кратности $t_{И}$=1.

Решение:

1) Для исправления независимых ошибок кратности $t_{И}$=1 требуется кодовое расстояние $d \geq 2t_u + 1 = 3$.

2) Находим m: $n = 2^m - 1$; m=3.

3) Число проверочных символов

$$r \geq \frac{(d-1)\cdot m}{2} \quad ; \quad r = \frac{(3-1)\cdot 3}{2} = 3 \text{ .}$$

4) Находим производящий полином, учитывая, что $g(x)=$ $\text{НОК}\left[\prod_i m_i(x)\right]$, где i=1, 3, 5,…,(2tи-1) – номера минимальных функций $m_i(x)$.

Тогда $i=d-2=3-2=1$; $g(x) = m_1(x) = x^3 + x + 1 \to 1011$ и проверочный полином .

$$h(x) = \frac{x^7 + 1}{x^3 + x + 1} = x^4 + x^2 + x + 1 \to 10111$$

5)Строим производящую матрицу, которая образуется добавлением *n-k* нулей к производящему полиному, записанному в виде двоичного числа; остальные строки матрицы представляют собой циклический сдвиг первой строки

$$\begin{matrix} g(x)\cdot x^0 \\ g(x)\cdot x^1 \\ g(x)\cdot x^2 \\ g(x)\cdot x^3 \end{matrix} = \left|\begin{matrix} 0001011 \\ 0010110 \\ 0101100 \\ 1011000 \end{matrix}\right| \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

6)Суммируя по модулю два во всевозможных сочетаниях строки производящей матрицы, получим остальные 11 разрешенных кодовых слов (из общего числа $2^k - 1 = 2^4 - 1 = 15$).

Также к ним применимы все методы, используемые для декодирования циклических кодов. Однако существуют гораздо лучшие алгоритмы, разработанные именно для БЧХ-кодов.

Исторически первым методом декодирования был найден Питерсоном для двоичного случая g=2, затем Горенстейном и Цирлером для общего случая. Упрощение алгоритма было найдено Берлекэмпом, а затем усовершенствовано Мэсси (алгоритм БерлекэмпаИсторически первым методом декодирования был найден Питерсоном для двоичного случая g=2, затем Горенстейном и Цирлером для общего случая. Упрощение

Код Рида-Соломона является частным случаем БЧХ-кода.

Коды *Рида — Соломона* — недвоичные циклические коды, позволяющие исправлять ошибки в блоках данных. Элементами кодового вектора являются не биты, а группы битов (блоки). Очень распространены коды Рида-Соломона, работающие с байтами — недвоичные циклические коды, позволяющие исправлять ошибки в блоках данных. Элементами кодового вектора являются не биты, а

Построение недвоичных (*q*-ичных) кодов БЧХ мало отличается от построения двоичных кодов и сводится к определению производящего полинома *g*(*x*), который либо неприводим, либо представляет собой произведение неприводимых (примитивных) полиномов.

Производящий полином кода РС определяется просто

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)\ldots(x-\alpha^{2t}),$$

где α - примитивный элемент, а длина m-ичного кодового слова равна $n = 2^m - 1$.

Код Рида — Соломона, исправляющий *t* ошибок, требует 2*t* проверочных символов и с его помощью исправляются произвольные пакеты ошибок длиной *t* и меньше. Согласно теореме о границе Рейгера, коды Рида — Соломона являются оптимальными с точки зрения соотношения длины пакета и возможности исправления ошибок — используя 2*t* дополнительных проверочных символов исправляются *t* ошибок (и менее).

Кодирование с помощью кода Рида — Соломона может быть реализовано двумя способами: систематическим и несистематическим.

При несистематическом кодировании информационное слово умножается на некий неприводимый полином. Полученное закодированное слово полностью отличается от исходного и для извлечения информационного слова нужно выполнить операцию декодирования и уже потом можно проверить данные на содержание ошибок. Такое кодирование требует большие затраты ресурсов только на извлечение информационных данных, при этом они могут быть без ошибок.

При систематическом кодировании к информационному блоку из k символов приписываются $2t$ проверочных символов, при вычислении каждого проверочного символа используются все k символов исходного блока. В этом случае нет затрат ресурсов при извлечении исходного блока, если информационное слово не содержит ошибок, но кодировщик/декодировщик должен выполнить $k(n - k)$ операций сложения и умножения для генерации проверочных символов.

При операции кодирования информационный полином умножается на порождающий многочлен. Умножение исходного слова *S* длины *k* на неприводимый полином при систематическом кодировании можно выполнить следующим образом:

К исходному слову приписываются $2t$ нулей, получается полином $T = Sx^{2t}$.
Этот полином делится на порождающий полином *G*, находится остаток *R*, $Sx^{2t} = QG + R$, где *Q* — частное.
Этот остаток и будет корректирующим кодом Рида—Соломона, он приписывается к исходному блоку символов. Полученное кодовое слово $C = Sx^{2t} + R$.
Кодировщик строится из сдвиговых регистров, сумматоров и умножителей. Сдвиговый регистр состоит из ячеек памяти.

Декодировщик, работающий по авторегрессивному спектральному методу декодирования, последовательно выполняет следующие действия:

- Вычисляет синдром ошибки
- Строит полином ошибки
- Находит корни данного полинома
- Определяет характер ошибки
- Исправляет ошибки

Вычисление синдрома ошибки

Вычисление синдрома ошибки выполняется синдромным декодером, который делит кодовое слово на порождающий многочлен. Если при делении возникает остаток, то в слове есть ошибка. Остаток от деления является синдромом ошибки.

Построение полинома ошибки

Вычисленный синдром ошибки не указывает на положение ошибок. Степень полинома синдрома равна $2t$, что много меньше степени кодового слова n. Для получения соответствия между ошибкой и ее положением в сообщении строится полином ошибок. Полином ошибок реализуется с помощью алгоритма Берлекэмпа — Месси, либо с помощью алгоритма Евклида. Алгоритм Евклида имеет простую реализацию, но требует больших затрат ресурсов. Поэтому чаще применяется более сложный, но менее затратоемкий алгоритм Берлекэмпа — Месси.

Коэффициенты найденного полинома непосредственно соответствуют коэффициентам ошибочных символов в кодовом слове.

Нахождение корней

На этом этапе ищутся корни полинома ошибки, определяющие положение искаженных символов в кодовом слове. Реализуется с помощью процедуры Ченя, равносильной полному перебору. В полином ошибок последовательно подставляются все возможные значения, когда полином обращается в ноль — корни найдены.

Определение характера ошибки и ее исправление

По синдрому ошибки и найденным корням полинома с помощью алгоритма Форни определяется характер ошибки и строится маска искаженных символов. Эта маска накладывается на кодовое слово с помощью операции XOR и искаженные символы восстанавливаются. После этого отбрасываются проверочные символы и получается восстановленное информационное слово.

Применение

В настоящий момент коды Рида — Соломона имеют очень широкую область применения благодаря их способности находить и исправлять многократные пакеты ошибок.

Коды РС применяются для исправления ошибок и стираний в каналах с пакетирующимися ошибками и в качестве внешнего кода при каскадном кодировании в аппаратуре радиорелейной, сотовой и космической связи с учетом того, что после декодирования внутреннего кода ошибки группируются. Важной его особенностью является то, что этот код имеет кодовое расстояние

$$d = (n-k+1) = r+1$$

и является кодом с *максимальным кодовым расстоянием*. Поэтому коды РС всегда оказываются короче всех других циклических кодов над тем же алфавитом и являются оптимальными с точки зрения эффективности использования избыточности, так как максимальная кратность исправляемых ошибок t равна половине проверочных символов в *m*-ичном кодовом слове.