Тема 4.2. Самокорректирующие коды

Аннотация: Самокорректирующие коды. Коды с исправлением одной ошибки. Верхняя и нижняя оценки мощности максимального кода. Коды Хэмминга, их свойства. Алгоритм декодирования для кодов Хэмминга. Линейные коды.

Требования к достоверности передачи по зашумленным каналам диктуют необходимость такого кодирования информации, при котором обеспечивалось бы возможность обнаружения и исправления ошибок. Это достигается путем введения избыточности, которая дает возможность обнаружить и исправить ошибки.

Коды, обладающие таким свойством, получили название помехоустойчивые. Они используются для обнаружения ошибок и для исправления ошибок (корректирующие коды).

Переход от теории, начало которой положил Шеннон, к практике стал возможен благодаря усилиям Ричарда Хэмминга, коллеги Шеннона по Bell Labs, получившего известность за открытие класса кодов, которые так и стали называть «кодами Хэмминга». Согласно легенде, к изобретению своих кодов Хэмминга подтолкнуло неудобство в работе с перфокартами на релейной счетной машине Bell Model V в середине 40-х годов. В отличие от контроля четности, способного «скорректировать» выпадение символа Хэмминг предложил коды, называемые «коды с исправлением ошибок» (Error-Correcting Code, ECC), способные корректировать ошибки замены в каналах связи.

Современные модификации этих кодов используются во всех системах хранения данных и для обмена между процессором и оперативной памятью. Среди новейших кодов ECC следует назвать коды LDPC (Low-Density Parity-check Code), которые хоть и не обладают стопроцентной достоверностью, но вероятность ошибки может быть доведена до необходимого значения, и при этом с максимальной полнотой используется пропускная способность канала.

Классификация помехоустойчивых кодов

В непрерывных кодах передаваемая информационная последовательность не разделяется на блоки. Избыточные элементы размещаются в определенном порядке между информационными.

Помехоустойчивое кодирование предполагает введение в передаваемое сообщение, наряду с информационными, так называемых проверочных разрядов, формируемых в устройствах защиты от ошибок. Избыточность позволяет отличить разрешенную и запрещенную (искаженную за счет ошибок) комбинации при приеме, иначе одна разрешенная комбинация переходила бы в другую.

Существующие помехоустойчивые коды можно разделить на ряд групп, только часть из которых используется для обнаружения ошибок в передаваемых по сети пакетах. В группе систематических (линейных) кодов общим свойством является то, что любая разрешенная комбинация может быть получена в результате линейных операций над линейно-независимыми векторами. Это способствует упрощению аппаратной и программной реализации данных кодов, повышает скорость выполнения необходимых операций.

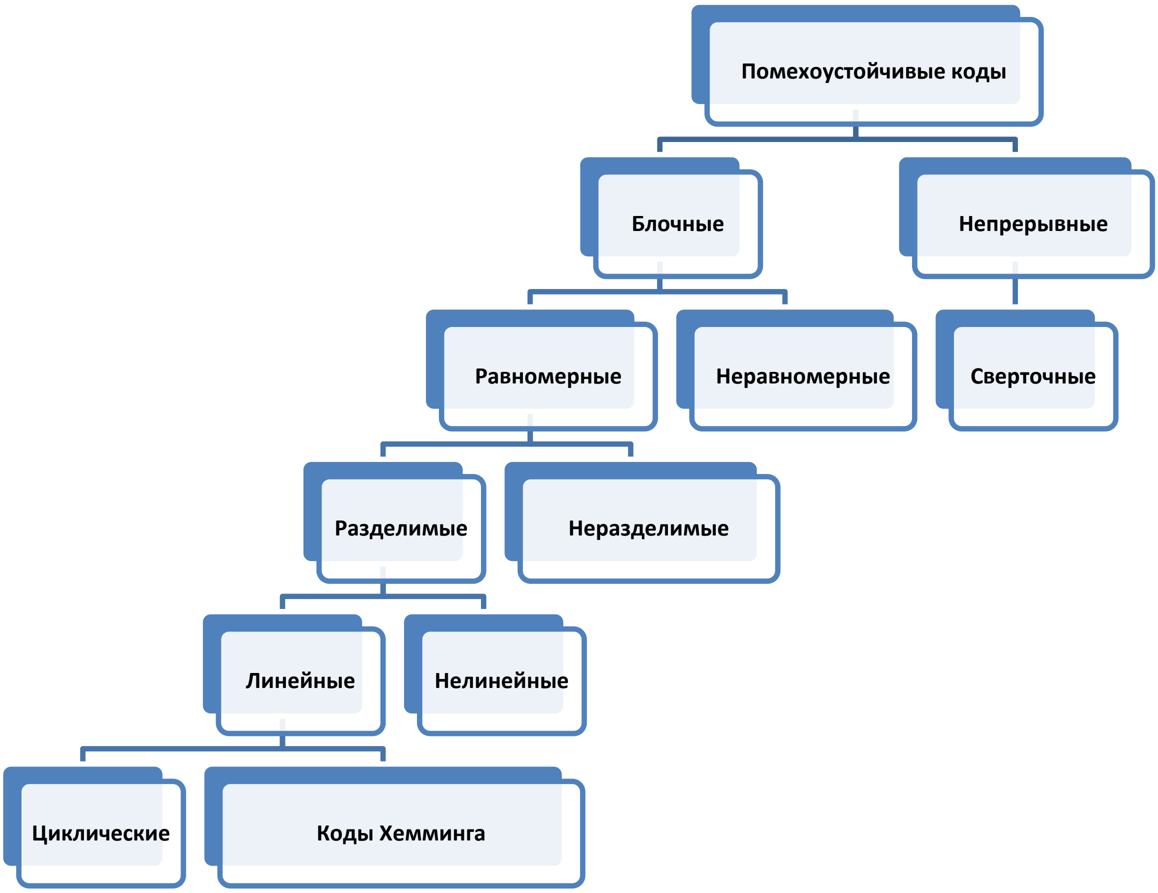


Рисунок 1 Классификация помехоустойчивых кодов

Все помехоустойчивые коды можно разделить на два основных класса: блочные и непрерывные (рекурентные или цепные).

В блочных кодах каждому сообщению (или элементу сообщения) сопоставляется кодовая комбинация (блок) из определенного количества разрядов. Блоки кодируются и декодируются отдельно друг от друга.

Блочные коды могут быть равномерными, когда длина кодовых комбинаций n постоянна, или неравномерными, когда n непостоянно.

В непрерывных кодах введение избыточности в последовательность входных символов осуществляется без разбивки ее на отдельные блоки. Процессы кодирования и декодирования в непрерывных кодах имеют также непрерывный характер.

Как блочные, так и непрерывные коды в зависимости от методов внесения избыточности подразделяются на разделимые и неразделимые. В разделимых кодах четко разграничена роль отдельных символов. Одни символы являются информационными, другие являются проверочными и служат для обнаружения и исправления ошибок.

Помехоустойчивый код характеризуется тройкой чисел (n, k, d_0), где n — общее число разрядов в передаваемом сообщении, включая проверочные (r), $k=n-r$ - число информационных разрядов, d_0 — минимальное кодовое расстояние между разрешенными кодовыми комбинациями, определяемое как минимальное число различающихся бит в этих комбинациях.

Разделимые коды, в свою очередь, делятся на систематические (линейные) и несистематические (нелинейные). Систематическими кодами называются блочные разделимые (n,k)-коды, в которых проверочные элементы представляют собой линейные комбинации информационных, несистематические коды таким свойством не обладают.

Неразделимые коды не имеют четкого разделения кодовой комбинации на информационные и проверочные символы.

Разделимые блочные коды делятся, в свою очередь, на несистематические и систематические. Несистематические разделимые коды строятся таким образом, что проверочные символы определяются как сумма подблоков длины l, на которые разделяется блок информационных символов.

Большинство известных разделимых кодов составляют систематические коды. У этих кодов значение проверочных символов определяется в результате проведения линейных операций над определенными информационными символами. Для случая двоичных кодов каждый проверочный символ выбирается таким, чтобы его сумма по модулю два с определенными информационными символами стала равной нулю (т.е. сумма единиц была четной). Декодирование сводится к проверке на четность определенных групп символов. В результате таких проверок дается информация о наличии ошибок, а в случае необходимости – о позиции символов, где имеются ошибки.

Основные понятия

Контрольные разряды не передают информацию и в этом смысле бесполезны. Относительное число контрольных разрядов называется ***избыточностью помехоустойчивого кода***.

Код с контролем по четности позволяет обнаружить одиночные ошибки в блоках данных при передаче данных. Однако он не может обнаружить двукратные ошибки потому, что двукратная ошибка переводит кодовое слово через промежуток между допустимыми словами и превращает его в другое допустимое слово.

Будем рассматривать равномерные коды, длины всех слов, равные n, и ошибки типа замещения, то есть вида $0 \to 1$ и $1 \to 0$.

Код называется ***исправляющим r ошибок***, если при наличии в любом кодовом слове не более r ошибок типа замещения можно восстановить исходное кодовое слово.

Расстоянием Хэмминга (между 2 наборами длины n) называется число разрядов, в которых эти наборы различаются.

Коды, в которых возможно автоматическое исправление ошибок, называются самокорректирующимися. В настоящее время наибольший интерес представляют двоичные блочные корректирующие коды. При использовании таких кодов информация передаётся в виде блоков одинаковой длины и каждый блок кодируется и декодируется независимо друг от друга.

Основными характеристиками самокорректирующихся кодов являются:

1. Число разрешенных и запрещенных комбинаций. Если n — число символов в блоке, r — число проверочных символов в блоке, k — число информационных символов, то 2^n — число возможных кодовых комбинаций, 2^k — число разрешенных кодовых комбинаций, $2^n - 2^k$ — число запрещенных комбинаций.
2. Избыточность кода. Величину $\frac{r}{n}$ называют избыточностью корректирующего кода.
3. Минимальное кодовое расстояние. Минимальным кодовым расстоянием d называется минимальное число искаженных

символов, необходимое для перехода одной разрешенной комбинации в другую.

4. Число обнаруживаемых и исправляемых ошибок. Если g — количество ошибок, которое код способен исправить, то необходимо и достаточно, чтобы $d \geq 2g+1$.
5. Корректирующие возможности кодов.

Граница Плоткина даёт верхнюю границу кодового расстояния: $d \leq \frac{n \cdot 2^{k-1}}{2^k - 1}$.

Граница Хемминга устанавливает максимально возможное число разрешенных кодовых комбинаций: $2^k \leq \frac{2^n}{\sum_{i=0}^{(d-1)/2} C_n^i}$

где C_n^i — число сочетаний из n элементов по i элементам.

Код Хемминга

Построение кодов Хэмминга основано на принципе проверки на четность числа единичных символов: к последовательности добавляется такой элемент, чтобы число единичных символов в получившейся последовательности было четным.

Разобьем номера всех разрядов исходного слова на k классов:

$$D_i = \left\{ m \mid m = \left(m_{k-1} m_{k-2} \ldots m_0 \right)_2, m_i = 1 \right\}, 1 \leq m \leq n.$$

Кодом Хэмминга порядка n называется множество наборов

$$\tilde{a} = \left(a_1, a_2, \ldots, a_n \right) \in E_2^k,$$

удовлетворяющих системе уравнений (суммы по модулю 2):

$$\begin{cases} \sum_{j \in D_0} a_j = 0 \\ \sum_{j \in D_1} a_j = 0 \\ \vdots \\ \sum_{j \in D_{k-1}} a_j = 0 \end{cases}.$$

Теорема

Код Хэмминга порядка n содержит 2^{n-k} *наборов, где* $k = \log_2 n + 1$ *и исправляет одну ошибку.*

Доказательство:

➢ Рассмотрим систему уравнений:

$$\begin{cases} a_1 \oplus (a_3 \oplus ...) = 0; \\ a_2 \oplus (...) = 0; \\ \quad\quad \vdots \\ a_{2^{k-1}} \oplus (...) = 0. \end{cases}$$

Задаем произвольную a_j, кроме $a_1, a_2, a_3, a_4, ..., a_{2^{k-1}}$. Это можно сделать 2^{n-k} способами. Так как $a_1, a_2, a_3, a_4, ..., a_{2^{k-1}}$ в скобках не встречаются, то они однозначно определяются из системы.

Пусть теперь передано кодовое слово $\tilde{a} = (a_1 a_2 ... a_n)$, ошибка произошла в разряде с номером d и пусть $d = (\gamma_{k-1}\gamma_{k-2}...\gamma_1\gamma_0)_2$. Пусть на выходе получено слово $\tilde{\beta} = (\beta_1\beta_2...\beta_n)$, при этом $\beta_j = \alpha_j$ при $i \neq d$, $\beta_d = \alpha_d \oplus 1$. Обозначим $\delta_0 = \sum\limits_{j \in D_0} \beta_j, \delta_1 = \sum\limits_{j \in D_1} \beta_j, ..., \delta_{k-1} = \sum\limits_{j \in D_{k-1}} \beta_j$. Докажем, что $(\delta_{k-1}\delta_{k-2}...\delta_1\delta_0)_2 = d$.

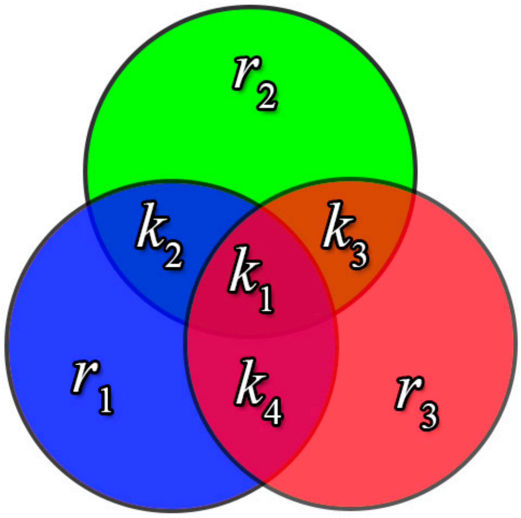
Пусть $\gamma_i = 0 \Rightarrow d \notin D_j$, тогда $\sum\limits_{j \in D_j} \beta_j = \sum\limits_{j \in D_j} \alpha_j$, следовательно, $\delta_j = 0$ и $\delta_j = \gamma_j$. Пусть теперь $\gamma_j = 1$ и $\mathrm{d} \in \mathrm{D}_{i.}$. Тогда $\sum\limits_{j \in D_i} \beta_i = \sum\limits_{j \in D_i} \alpha_i \oplus 1 = 1 \Rightarrow \delta_i = 1 \Rightarrow \delta_i = \gamma_i$.

Теорема доказана. ≺

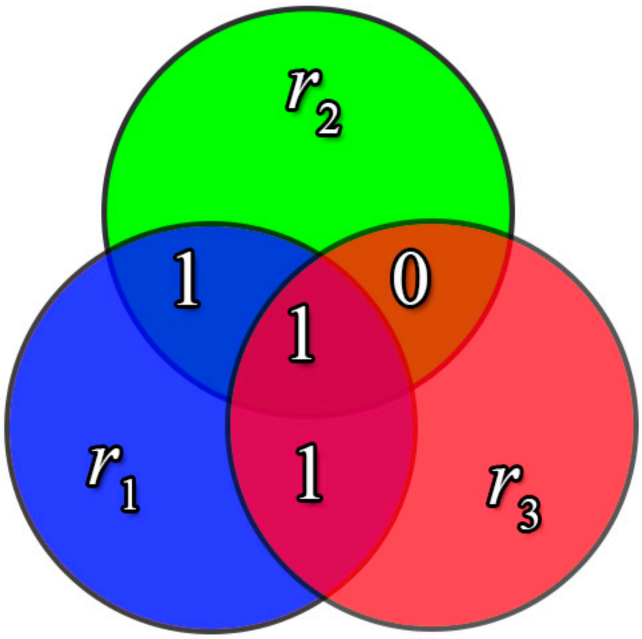
Рассмотрим теперь более подробно и наглядно как работает код Хемминга. Для каждого числа проверочных символов *r*=3,4,5, … существует классический код Хэмминга с маркировкой $(n,k) = (2^r - 1, 2^r - 1 - r)$, то есть существуют коды хемминга (7,4),(15,11),(31,26) и т.д.

Графическая интерпретация

Рассмотрим сначала графическую интерпретацию для кода Хемминга (7,4).



Информационные биты – это k_1, k_2, k_3, k_4, проверочные - r_1, r_2, r_3. Подставляя вместо k_i конкретные значения вычисляем проверочные биты так, чтобы в каждом круге сумма бит была четной (т.е. чтобы сумма по модулю два была равна нулю). Например, для кодирования сообщения 1101 получим:



Откуда найдем проверочные биты:

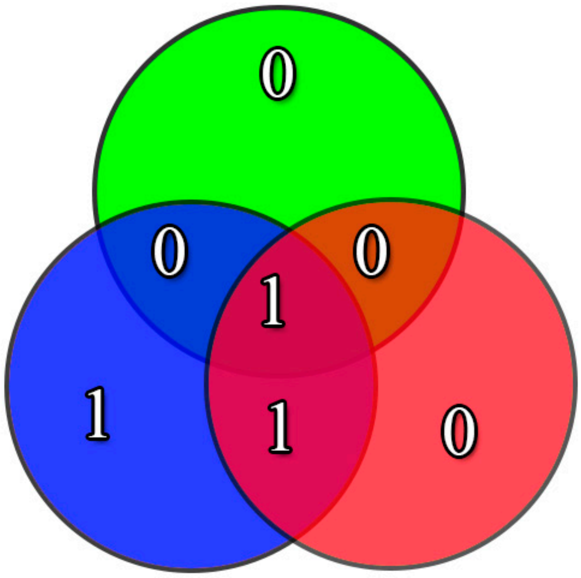
$r_1 = 1 + 1 + 1 = 1,$

$r_2 = 1 + 1 + 0 = 0,$

$r_3 = 1 + 0 + 1 = 0.$

Таким образом получили код Хемминга: 1101 100 (здесь мы пробелом отделили информационные биты от проверочных для наглядности).

Предположим теперь, что при передаче произошла ошибка замены и было получено сообщение, например, 1001 100 (замена возможна в любом бите, как информационном, так и контрольном). Тогда выполним проверку по каждому из кругов:



Контроль четности не сходится в левом нижнем круге и в верхнем круге, но сходится в правом нижнем круге. Построив область ошибки как пересечение двух кругов с ошибками минус круг без ошибки получим область (бит) в которой произошла ошибка – это область k_2, т.е. заменяем k_2 с 0 на 1, тогда верное сообщение будет 1101 100, что и должно быть.

<u>Вспомогательные таблицы</u>

Рассмотрим теперь получение кода Хемминга для слов произвольной длины при помощи таблиц. Рассмотрим сразу на примере.

Пример: Для заданного сообщения Х = 0110101 построить код Хэмминга, внести одиночную ошибку и произвести декодирование

Построим сначала вспомогательную таблицу:

0	1	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	1	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	1	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	1	1	1	1	1	1	…

При необходимости ее можно «тянуть» вниз и вправо до бесконечности.

Теперь убираем из рассмотрения первый столбец (он соответствует нулевой позиции, в которой не может быть никаких битов), а те столбцы, которым соответствует **первое** появление единицы в каждой строке выделим – эти биты будут проверочными:

0	**1**	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	**1**	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	**1**	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	**1**	1	1	1	1	1	…

Теперь в верхней строке во все не выделенные ячейки внесем наше число (последовательно, слева направо):*0110101*

			0		1	1	0		1	0	1			
0	**1**	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	**1**	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	**1**	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	**1**	1	1	1	1	1	…

У нас остались незанятые ячейки – они лишние. Можно убрать их из рассмотрения.

Теперь посчитаем проверочные биты. Для этого выбираем вспомогательную строку и везде, где в этой строке есть единица мы смотрим на строку значений и считаем кол-во единиц на указанных позициях (фактически мы находим конъюнкцию строки значений и соотвествующей вспомогательной строки). Для первой вспомогательной строки будет:

			0		1	1	0		1	0	1			
0	**1**	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	**1**	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	**1**	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	**1**	1	1	1	1	1	…

Число единиц 3 – нечетно, значит проверочный бит ставим 1 (незаполненная ячейка):

	1		0		1	1	0		1	0	1			
0	**1**	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	**1**	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	**1**	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	**1**	1	1	1	1	1	…

Теперь для второй вспомогательной строки:

	1		0		1	1	0		1	0	1			

0	1	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	1	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	1	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	1	1	1	1	1	1	…

Кол-во единиц в заданных позициях в первой строке (строке значений) – 2 – четное, значит проверочный бит 0:

	1	0	0		1	1	0		1	0	1			
0	1	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	1	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	1	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	1	1	1	1	1	1	…

Аналогично для третьей вспомогательной строки:

	1	0	0	0	1	1	0		1	0	1			
0	1	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	1	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	1	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	1	1	1	1	1	1	…

И для четвертой:

	1	0	0	0	1	1	0	0	1	0	1			
0	1	0	1	0	1	0	1	0	1	0	1	0	1	…
0	0	1	1	0	0	1	1	0	0	1	1	0	0	…
0	0	0	0	1	1	1	1	0	0	0	0	1	1	…
0	0	0	0	0	0	0	0	1	1	1	1	1	1	…

В итоге получили код: 10001100101.

Замечание: на практике, проверочные биты ставят после информационных, т.е., на самом деле, получим код 0110101 1000 – здесь мы отделили информационные биты от проверочных пробелом.

Для объяснения того, как работает коррекция, вернемся к записи, полученной по вспомогательным таблицам (это элементарно делается из кода 0110101 1000, если вставлять проверочные бита на позиции целой степени двойки).

Пусть при передаче сообщения $X^{/}$ произошла ошибка замещения в 7–м разряде (уже в подготовленном для таблицы виде), т.е. получено сообщение $X^{//}$ = 10001110101. Докажем это, для этого вычислим также по таблице, но при этом учитываем контрольные значения (т.е. тоже считаем их). Рассчитываемые значения называют синдромами. По первой вспомогательной строке получим:

	1	0	0	0	1	1	0	0	1	0	1
0	**1**	0	1	0	1	0	1	0	1	0	1
0	0	**1**	1	0	0	1	1	0	0	1	1
0	0	0	0	**1**	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	**1**	1	1	1

$S_1 = 1 + 0 + 1 + 1 + 1 + 1 = 1$

По второй:

	1	0	0	0	1	1	0	0	1	0	1
0	**1**	0	1	0	1	0	1	0	1	0	1
0	0	**1**	1	0	0	1	1	0	0	1	1
0	0	0	0	**1**	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	**1**	1	1	1

$S_2 = 0 + 0 + 1 + 1 + 0 + 1 = 1$

По третьей и четвертой:

$S_3 = 0 + 1 + 1 + 1 = 1$

$S_4 = 0 + 1 + 0 + 1 = 0.$

Запишем полученные значения в обратном порядке. Получим двоичное число. Переведем его в десятичную систему. Разряд, в котором произошла ошибка, равен $S = 0111_2 = 7$.

Код Хемминга (7,4) с помощью матриц

Код Хемминга может быть получен с помощью операций над матрицами. Для кода (7,4) получение кодового слова будет выглядеть так:

$$\begin{pmatrix} k_1 & k_2 & k_3 & k_4 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & r_1 & r_2 & r_3 \end{pmatrix}$$

Синдромы получаются по формуле:

$$\begin{pmatrix} k_1 & k_2 & k_3 & k_4 & r_1 & r_2 & r_3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} S_1 & S_2 & S_3 \end{pmatrix}.$$

Проверьте, что по данным формулам можно получить требуемый код и синдромы.

Следует отметить, что в коде Хемминга нулевой синдром означает отсутствие однократной ошибки при передаче.

Вопросы для самоконтроля:

1. Классификация помехоустойчивых кодов
2. Самокорректирующие коды. Основные понятия.
3. Коды с исправлением одной ошибки. Верхняя и нижняя оценки мощности максимального кода.
4. Коды Хэмминга, их свойства.
5. Графическая интерпретация кода Хэмминга (7,4).
6. Построение кода Хэмминга при помощи вспомогательных таблиц.
7. Алгоритм декодирования для кодов Хэмминга. Линейные коды.