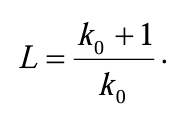
**ПОМЕХОЗАЩИТНОЕ КОДИРОВАНИЕ**

Задача обнаружения ошибки может быть решена довольно просто. Достаточно просто передавать каждую букву сообщения дважды. Например, при необходимости передачи слова «гора» можно передать «ггоорраа». При получении искаженного сообщения, например, «гготрраа» с большой вероятностью можно догадаться, каким было исходное слово. Конечно, возможно такое искажение, которое делает неоднозначным интерпретацию полученного сообщения, например, «гпоорраа», «ггоорреа» или «кгоорраа». Однако цель такого способа кодирования состоит не в исправлении ошибки, а в фиксации факта искажения и повторной передаче части сообщения в этом случае.

Недостаток данного способа обеспечения надежности состоит в том, что избыточность кода оказывается очень большой – очевидно, что L = 2.

Поскольку ошибка должна быть только обнаружена, можно предложить **другой способ кодирования**. Пусть имеется цепочка информационных бит длиной *k*0. Добавим к ним еще один бит, значение которого определяется тем, что новая кодовая цепочка из *k*0+1 бита должна содержать чётное количество единиц – по этой причине такой контрольный бит называется **битом чётности**. Например, для информационного байта 01010100 бит чётности будет иметь значение 1, а для байта 11011011 бит чётности равен 0. В случае одиночной ошибки передачи число 1 перестает быть чётным, что и служит свидетельством сбоя.

Например, если получена цепочка 110110111 (контрольный бит выделен подчеркиванием), ясно, что передача произведена с ошибкой, поскольку общее количество единиц равно 7, т.е. нечётно. В каком бите содержится ошибка при таком способе кодирования установить нельзя. Избыточность кода в данном случае, очевидно, равна:



На первый взгляд кажется, что путем увеличения *k*0 можно сколь угодно приближать избыточность к ее минимальному значению (*Lmin* = 1). Однако с ростом *k*0, во-первых, растет вероятность парной ошибки, которая контрольным битом не отслеживается; во-вторых, при обнаружении ошибки потребуется заново передавать много информации. Поэтому обычно *k*0 = 8 или 16 и, следовательно, L=1,125 (1,0625).

**Коды, исправляющие одиночную ошибку**

В 1948 г. Р.Хеммингом был предложен способ кодирования ин формации, который позволяет не только обнаружить наличие ошибки, но и локализовать её (т.е. определить, в каком бите она находится). Подобные коды стали называться кодами Хемминга. Основная идея состоит в добавлении к информационным битам не одного, а **нескольких битов чётности**, каждый из которых контролирует определенные информационные биты. Если пронумеровать все биты передаваемого кода Хемминга, начиная с 1 слева направо, то контрольными (проверочными) являются биты, номера которых равны степеням числа 2, а все остальные являются информационными. Например, для 12-битного кода Хемминга контрольными окажутся биты с номерами 1, 2, 4 и 8 (см. рисунок 1).

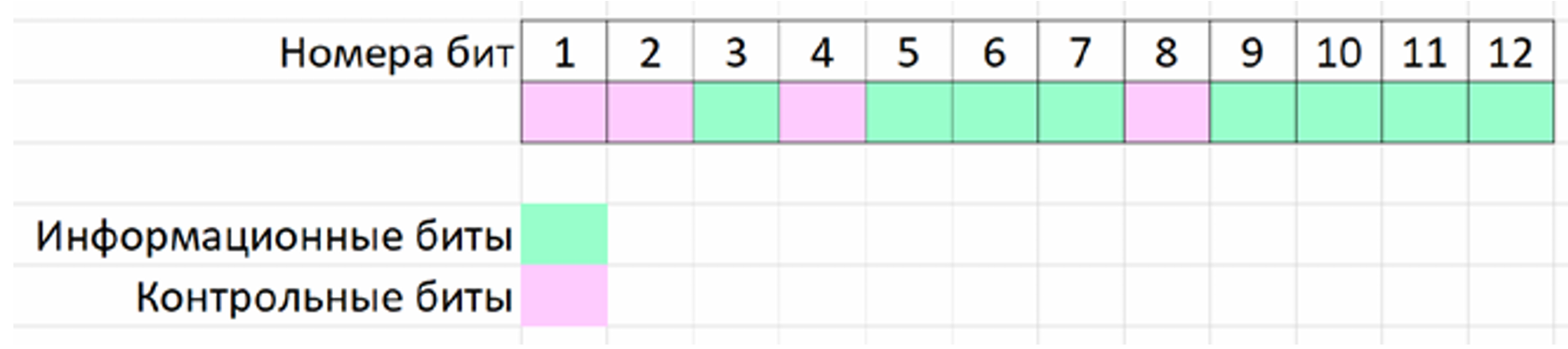


Рис. 1. Структура кода Хемминга

Номера контролируемых бит для каждого проверочного приведены в таблице 1. Состояние проверочного бита устанавливается таким образом, чтобы **суммарное количество единиц в контролируемых им битах было бы чётным.**

Таблица 1. Контролируемые биты

| **Пров. биты** | **Контролируемые биты** |
| --- | --- |
| 1 | 1 3 5 7 9 11 13 15 17 19 21 … |
| 2 | 2 3 6 7 10 11 14 15 18 19 22 … |
| 4 | 4 5 6 7 12 13 14 15 20 21 22 … |
| 8 | 8 9 10 11 12 13 14 15 24 25 26 … |
| 16 | 16 17 18 19 20 21 22 23 24 25 26 … |
| 32 | 32 33 34 35 36 37 38 39 40 41 42 … |

Для того чтобы легко запомнить номера контролируемых бит, существует такая закономерность:

1-ый контрольный бит контролирует 1 через 1 бит, начиная с самого себя (т.е. 1, 3, 5, 7, 9, 11 и т.д);

2-й контрольный бит контролирует 2 через 2 бита, начиная с самого себя (т.е. 2, 3, 6, 7, 10, 11 и т.д.) и т.д.

**Пример**. Составить код Хемминга для информационного байта 11010010. Контрольными являются биты с номерами 1, 2, 4 и 8. В остальные свободные биты заносим информационный байт (см. рисунок 2).

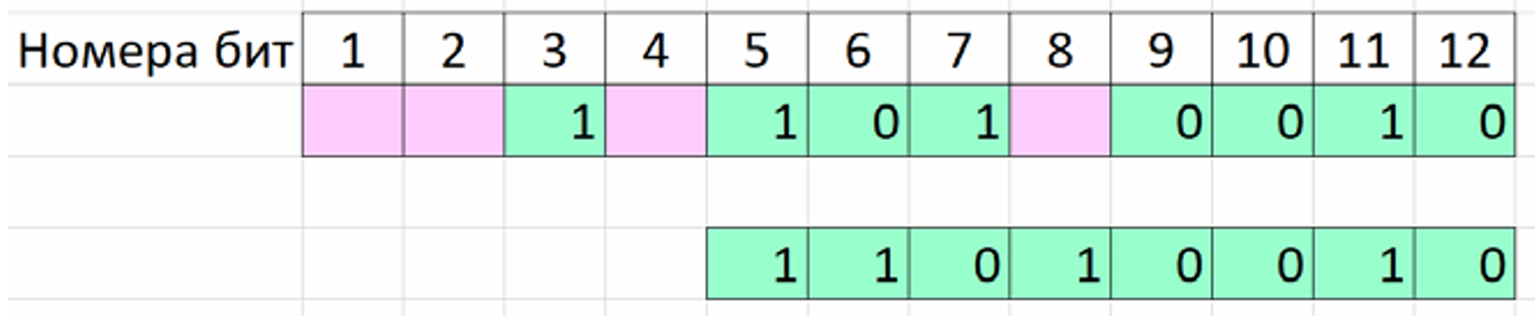


Рис. 2. Заполняем информационные биты

На рисунке 3 отмечено иксами (X), какие биты нужно использовать для вычисления первого контрольного бита (с номером 1).

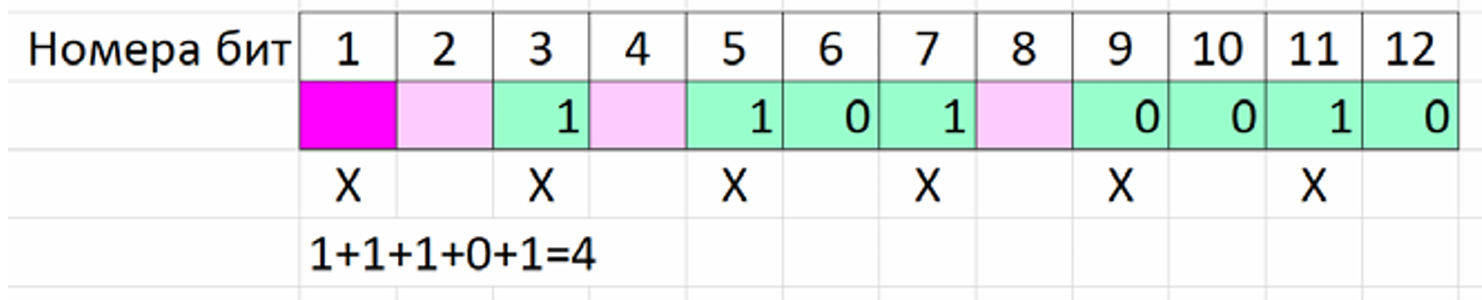


Рис. 3. Вычисление первого контрольного бита

Складываем контролируемые биты, и, если сумма получилась чётная, присваиваем контрольному биту значение 0, если нечётная, то значение 1.

Первому контрольному биту присваиваем значение 0.

Далее, на рисунке 4 отмечено, какие биты нужно использовать для вычисления контрольного бита с номером 2.

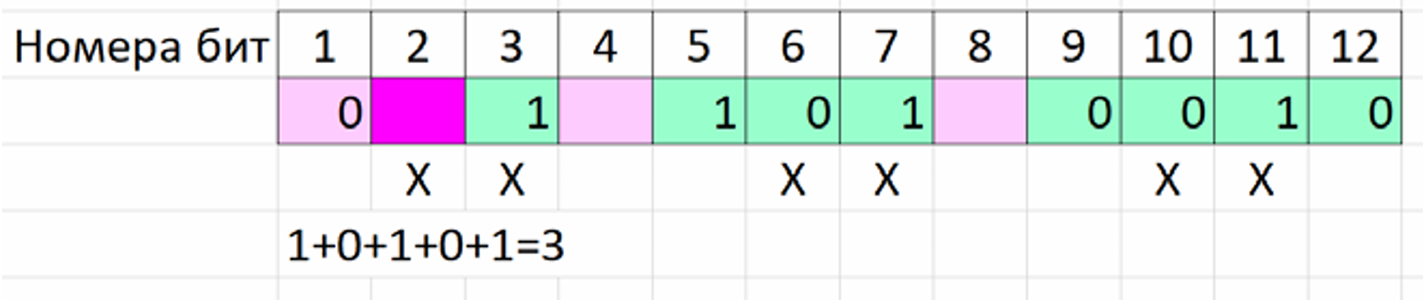


Рис. 4. Вычисление контрольного бита с номером 2

Сумма контролируемых бит нечётная, поэтому присваиваем биту с номером 2 значение 1.

Переходим к вычислению контрольного бита с номером 4. На рисунке 5 отмечены контролируемые им биты.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рис. 5. Вычисление контрольного бита с номером 4

Сумма контролируемых бит чётная, поэтому присваиваем биту с номером 4 значение 0.

И наконец, вычислим контрольный бит с индексом 8. На рисунке 6 отмечены контролируемые им биты.

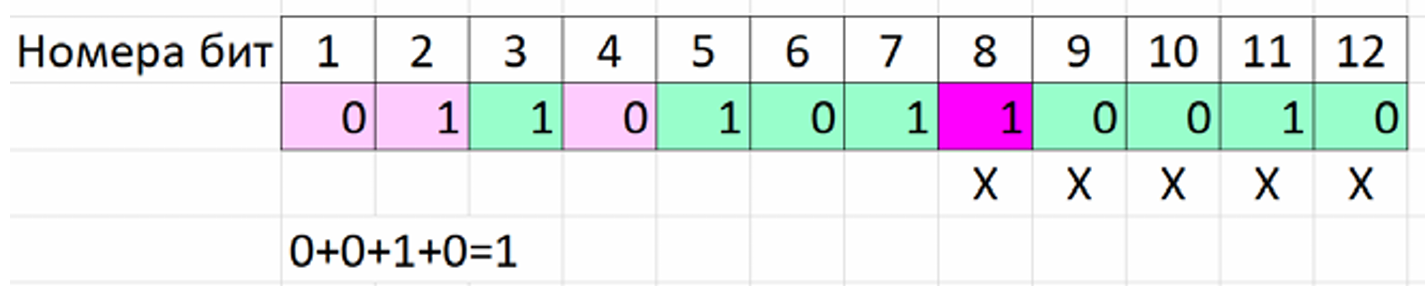


Рис. 6. Вычисление контрольного бита с номером 8

Сумма контролируемых бит нечётная, поэтому присваиваем биту с номером 8 значение 1.

Таким образом, код Хемминга для информационного байта 11010010 – это 011010110010.

Допустим, в процессе передачи по реальному каналу связи произошла ошибка и вместо указанной выше последовательности пришла следующая: 011000110010 (в 5-м бите 1 заменилась на 0, см. рисунок 7).

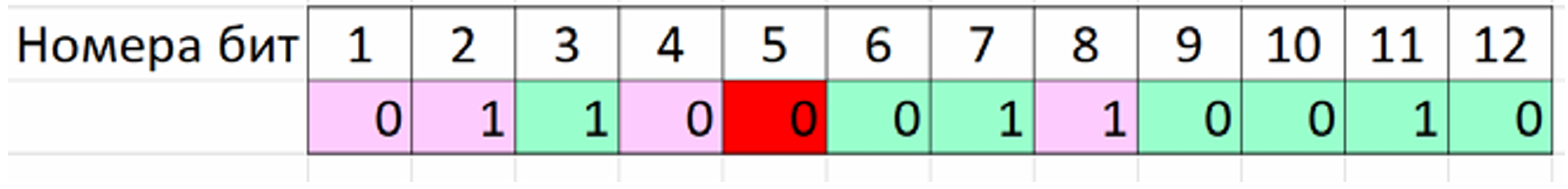


Рис. 7. Код Хемминга с ошибкой в одном бите

Для того чтобы проверить код Хемминга на наличие или отсутствие ошибки, необходимо проверить все контрольные биты на правильность. Сумма всех контролируемых бит каждого конкретного контрольного бита (включая его самого) должна быть чётной! Если сумма нечётна, значит данный контрольный бит сигнализирует об ошибке среди бит, которые он контролирует.

В нашем примере бит 1 указывает на наличие ошибки в каком-либо бите с нечётным номером (1, 3, 5, 7, 9 или 11), поскольку сумма контролируемых им бит нечётная (см. рисунок 8).



Рис. 8. Контрольный бит с номером 1 сигнализирует об ошибке

Бит 2 свидетельствует о том, что в битах 2, 3, 6, 7, 10 и 11 ошибки нет, т.е. ошибка в 1, 5 или 9 битах (см. рисунок 9).

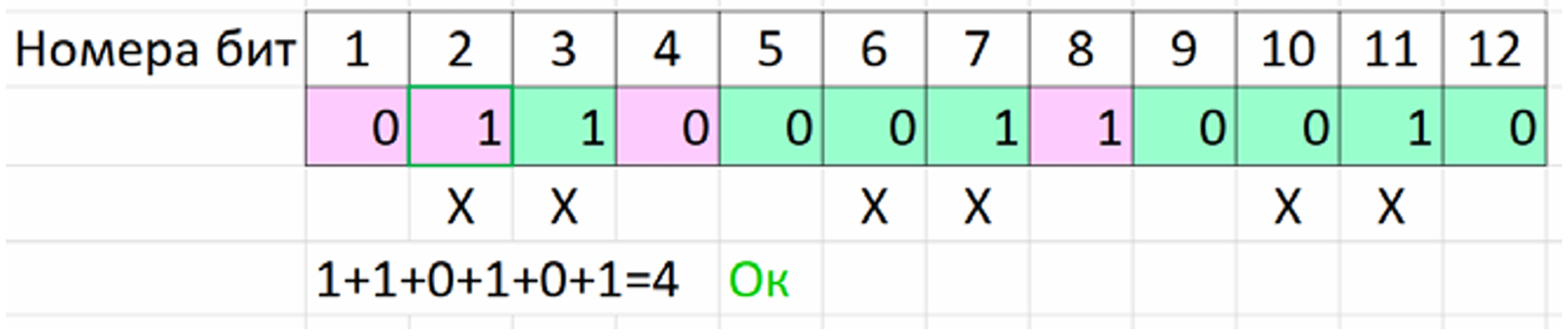


Рис. 9. Контрольный бит с номером 2 верный

Бит 4 указывает, что ошибка находится в одном из бит с номерами 4, 5, 6, 7, 12 (см. рисунок 10). В то же время нам известно, что ошибка в одном из бит с номерами 1, 5, 9. Следовательно, ошибка в 5 бите!



Рис. 10. Контрольный бит с номером 4 сигнализирует об ошибке

Таким образом, однозначно устанавливается, что ошибочным является 5-й бит – можно исправить его значение на противоположное и, тем самым, восстановить правильную последовательность.

Контрольный бит с номером 8 дополнительно сообщает, что ошибки в 9 бите нет (см. рисунок 11).

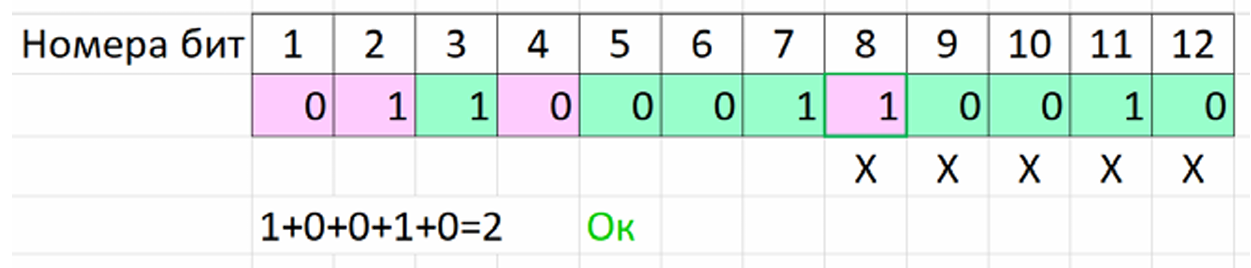


Рис. 11. Контрольный бит с номером 8 верный

Более детальное рассмотрение кодов Хемминга позволяет сформулировать простой алгоритм проверки и исправления передаваемой последовательности бит:

1) произвести проверку всех битов чётности;

2) если все биты чётности верны, то перейти к п.5;

3) вычислить сумму номеров всех неправильных битов чётности;

4) инвертировать содержимое бита, номер которого равен сумме, найденной в п.3;

5) исключить биты чётности, передать правильный информационный код.

На рассмотренном выше примере легко убедиться в справедливости данного алгоритма. В примере 1 и 4 контрольные биты сигнализировали об ошибке, 1+4=5. В 5 бите присутствовала ошибка! Избыточность кодов Хемминга для различных длин передаваемых последовательностей приведена в таблице 2.

Таблица 2. Зависимость избыточности кода от числа контрольных бит

| Число  информационных бит | Число  Контрольных бит | Избыточность |
| --- | --- | --- |
| 8 | 4 | 1,50 |
| 16 | 5 | 1,31 |
| 32 | 6 | 1,06 |

Из сопоставления видно, что выгоднее передавать и хранить более длинные последовательности бит. Безусловно, данный способ кодирования требует увеличения объема памяти компьютера приблизительно на одну треть при 16-битной длине машинного слова, однако он позволяет автоматически исправлять одиночные ошибки. Поэтому, оценивая время наработки на отказ, следует исходить из вероятности появления парной ошибки в одной последовательности (т.е. сбои должны произойти в двух битах одновременно). Расчёты показывают, что для указанного ранее количества ячеек в памяти объемом 1 Мбайт среднее время появления ошибки составляет более 80 лет, что, безусловно, можно считать вполне приемлемым с практической точки зрения.