Обнаружение ошибок

СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ

МЕТОДЫ ОБНАРУЖЕНИЯ ОШИБОК ПРИ ПОСЛЕДОВАТЕЛЬНОЙ ПЕРЕДАЧЕ ДАННЫХ

ГОСТ 28082—89

(СТ СЭВ 6185—88)

Издание официальное

- Каналы передачи данных не всегда надежны, да и само оборудование обработки информации работает со сбоями.
- По этой причине важную роль приобретают механизмы детектирования ошибок. Ведь если ошибка обнаружена, можно осуществить повторную передачу данных и решить проблему.

Чтобы найти ошибку, к сообщению добавляют контрольную сумму (**checksum**).

- В простом случае это сумма всех байтов сообщения.
- Например:
  Сообщение: 12 34 56
  Контрольная сумма: 12+34+56=102
- После передачи в это сообщение могут быть внесены помехи, например: искаженное сообщение 12 36 56 и контрольная сумма 102
- Если теперь сложить байты сообщения, результат не совпадет с контрольной суммой. Программа может сделать вывод, что данные были повреждены, и принять меры (например, еще раз запросить пакет или выдать сообщение, что он поврежден.

Чем плоха контрольная сумма

- Во-первых, если просто складывать байты, то старшие биты будут изменяться реже, чем младшие. То есть, если мы складываем
  12 34 56, контрольная сумма 102
  12 34 56 23, контрольная сумма 125
  12 34 56 23 45, контрольная сумма 170

  результат останется в пределах двух сотен. Если контрольная сумма записана в 32-разрядное слово, старшие байты будут содержать нули. Это не хорошо, так как все биты контрольной суммы должны быть равноценными.
- Во-вторых, сообщения с переставленными байтами будут иметь одну и ту же сумму:

  12 34 56, контрольная сумма 102
  56 12 34, контрольная сумма 102

Расстояние Хэмминга

- Пусть А и Б две двоичные кодовые последовательности равной длины. Расстояние Хэмминга между двумя этими кодовыми последовательностями равно числу символов, которыми они отличаются. Например, расстояние Хэмминга между кодами 00111 и 10101 равно 2.

- Для детектирования ошибок в n битах, схема кодирования требует применения кодовых слов с расстоянием Хэмминга не менее N+1, а для исправления ошибок в N битах необходима схема кодирования с расстоянием Хэмминга между кодами не менее 2N+1.

Широко распространены коды с одиночным битом четности

- К каждым М бит добавляется 1 бит, значение которого определяется четностью (или нечетностью) суммы этих М бит. Так, например, для двухбитовых кодов 00, 01, 10, 11 кодами с контролем четности будут 000, 011, 101 и 110. Если в процессе передачи один бит будет передан неверно, четность кода из М+1 бита изменится.

- Добавление бита четности уменьшает вероятность необнаруженной ошибки почти в 1000 раз. Использование одного бита четности типично для асинхронного метода передачи. В синхронных каналах чаще используется вычисление и передача битов четности как для строк, так и для столбцов передаваемого массива данных.

Штрих-коды (**barcode**).

- Банка консервированной кукурузы, на которой написан штрих-код:

  5 998483 710125
- Это код EAN-13 (тринадцать цифр). Правило для его проверки такое: сложить цифры на четных позициях, умножить сумму на три и добавь цифры на нечетных позициях. Должно получиться число, которое делится на десять:

  9 + 8 + 8+ 7 + 0 + 2 = 34
  5 + 9 + 4 + 3 + 1 + 1 + 5 = 28
  34 * 3 + 28 = 130, делится на 10
- А чтобы это условие выполнялось, последнюю цифру, 5, делают контрольной суммой.

Простой, короткий и быстрый (примерно в 10 раз быстрее CRC) способ подсчета контрольной суммы

- Берем по 4 байта из файла, и делаем с ними логическую операцию XOR. То, что получится в результате, — контрольная сумма.
- В этой контрольной сумме все байты используются полностью, среди них нет более значимых или менее значимых. Теоретическая вероятность того, что контрольная сумма окажется правильной для испорченного файла — 1 шанс на 2 ^ 32.
  Контроль по четности достаточно эффективен для выявления одиночных и множественных ошибок в условиях, когда они являются независимыми.

CRC

- ***cyclic redundancy code***, циклический избыточный код — способ цифровой идентификации некоторой последовательности данных, который заключается в вычислении контрольного значения её циклического избыточного кода.

- **Простая формула:** CRC = сообщение % полином

CRC

Циклический избыточный контроль (*Cyclic Redundancy Check, CRC*) является в настоящее время наиболее популярным методом контроля в вычислительных сетях (и не только в сетях, например, этот метод широко применяется при записи данных на диски и дискеты). Метод основан на рассмотрении исходных данных в виде одного многоразрядного двоичного числа. Например, кадр стандарта Ethernet, состоящий из 1024 байт, будет рассматриваться как одно число, состоящее из 8192 бит. В качестве контрольной информации рассматривается остаток от деления этого числа на известный делитель R. Обычно в качестве делителя выбирается семнадцати- или тридцати трехразрядное число, чтобы остаток от деления имел длину 16 разрядов (2 байт) или 32 разряда (4 байт). При получении кадра данных снова вычисляется остаток от деления на тот же делитель R, но при этом к данным кадра добавляется и содержащаяся в нем контрольная сумма.

Алгоритм CRC

- Алгоритм CRC базируется на свойствах деления с остатком двоичных многочленов, то есть является по сути остатком от деления многочлена, соответствующего входным данным, на некий фиксированный порождающий многочлен.
- Каждой конечной последовательности битов взаимооднозначно сопоставляется двоичный многочлен , последовательность коэффициентов которого таким образом представляет собой исходную последовательность. Например, десятичное число 90 (1011010 в двоичной записи) соответствует многочлену:

$$P(x) = 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0 = x^6 + x^4 + x^3 + x^1$$

Операция деления на примитивный полином эквивалентна следующей схеме:

- Пусть выбран примитивный полином, задающий цикл де Брейна 00101110010111100… и блок данных 0111110, построена таблица, верхняя строка заполнена блоком данных, а нижние строки — смещения на 0,1,2 бит цикла де Брейна

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- Тогда контрольная сумма будет равна операции XOR тех столбцов, над которыми в верхней строке расположена 1. В этом случае, 010 xor 101 xor 011 xor 111 xor 110 = 101 (CRC).

При «правильном» выборе порождающего многочлена (делителя), остатки от деления на него будут обладать нужными свойствами

- **хорошей перемешиваемостью и**
- **быстрым алгоритмом вычисления.**

- Второе обеспечивается тем, что степень порождающего многочлена обычно пропорциональна длине байта или машинного слова (например 8, 16 или 32).

Формализованный алгоритм расчёта CRC16

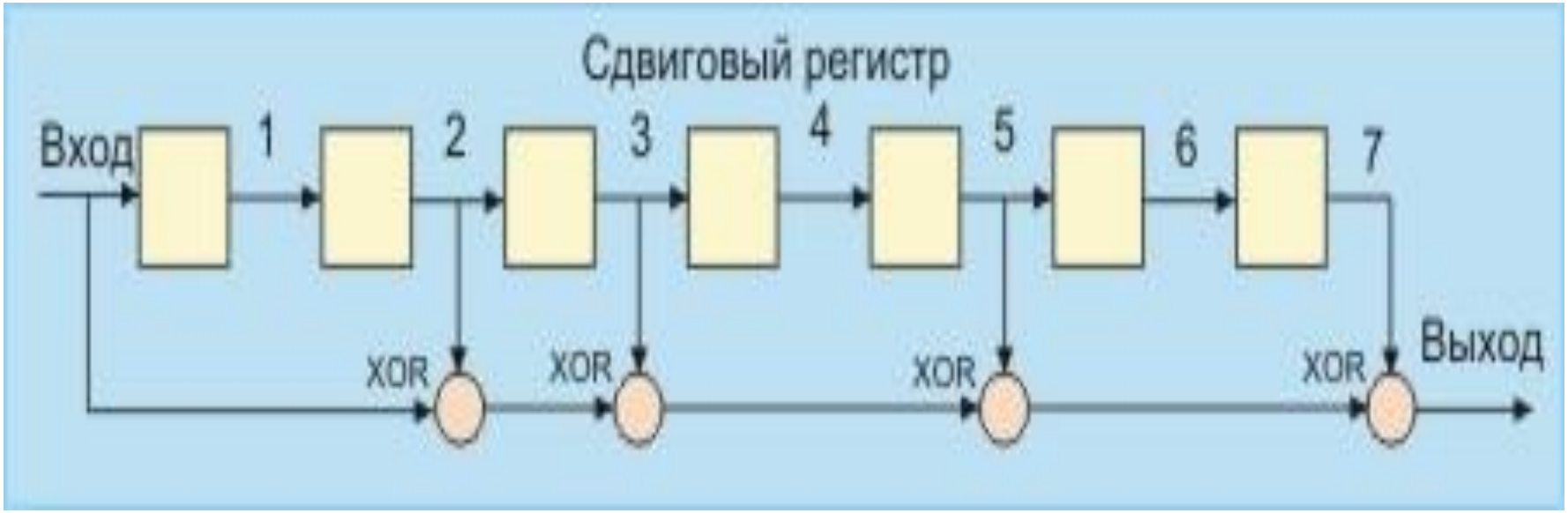
- Для получения контрольной суммы, необходимо сгенерировать полином. Основное требование к полиному: его степень должна быть равна длине контрольной суммы в битах. При этом старший бит полинома обязательно должен быть равен «1».
- Из файла берется первое слово. В зависимости от того, равен ли старший бит этого слова «1» или нет, выполняем (или нет) операцию XOR на полином. Полученный результат, вне зависимости от того, выполнялась ли операция XOR, сдвигаем на один бит влево (то есть умножаем на 2). После сдвига (умножения) теряется старый старший бит, а младший бит освобождается (обнуляется). На место младшего бита загружается очередной бит из файла. Операция повторяется до тех пор, пока не загрузится последний бит файла. После прохождения всего файла, в слове остается остаток, который и является контрольной суммой.

Используя описанную выше возможность замены полинома на бинарную последовательность,
рассмотрим алгоритм подсчета контрольной суммы CRC8:
Исходный массив данных: 1001 0110 0100 1011.
Порождающий многочлен: 1101 0101.

Пример расчета контрольной суммы CRC - 8

В Ethernet Вычисление crc производится аппаратно.

Реализация аппаратного расчета CRC для образующего полинома
B(x)= 1 + x2 + x3 +x5 + x7.
В этой схеме входной код приходит слева.



Эффективность CRC для обнаружения ошибок на многие порядки выше простого контроля четности.

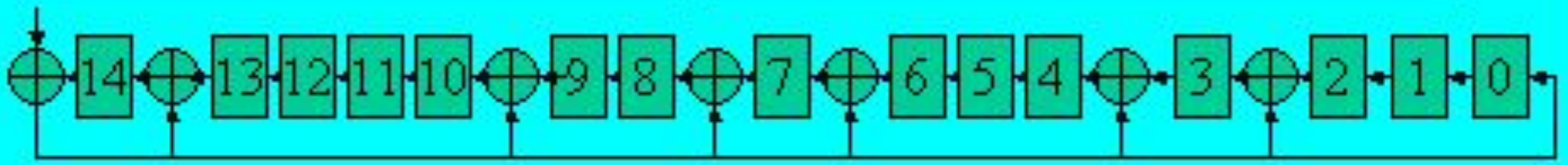
- В настоящее время стандартизовано несколько типов образующих полиномов. Для оценочных целей можно считать, что вероятность невыявления ошибки в случае использования CRC, если ошибка на самом деле имеет место, равна (1/2)r, где r - степень образующего полинома.
- CRC-12 $x12 + x11 + x3 + x2 + x1 + 1$
- CRC-16 $x16 + x15 + x2 + 1$
- CRC-CCITT $x16 + x12 + x5 + 1$

Формирование CRC сдвигающим регистром

В стандарте CAN

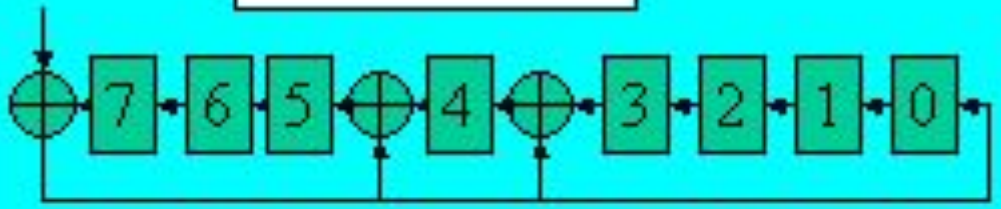
$x^{15}+x^{14}+x^{10}+x^8+x^7+x^4+x^3+1$

100 0101 1001 1001 B = 4599 H



В стандарте 1-wire

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



V.41 CCITT

$x^{16}+x^{12}+x^5+1$

IEEE 802 и др.

$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}\ x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

• Обнаружение ошибок

Этот·метод·может·быть·развит·путем·формирования·блока·данных·с·N·строками,·M столбцами·и·K·слоями.·Здесь·биты·четности·формируются·для·всех·строк·и·столбцов каждого·из·слоев,·а·также·битов,·имеющих·одинаковые·номера·строк·и·столбцов·i,j. Полное·число·битов·четности·в·этом·случае·равно·(N+M+1)×K·+(N+1)×(M+1).·Если·M+N+K=8, число·бит·данных·составит·512,·а·число·бит·четности·-·217.·Нетрудно·видеть,·что·в·этом случае·число·исправляемых·ошибок·будет·больше·1.·Смотри·рис.·1.¶

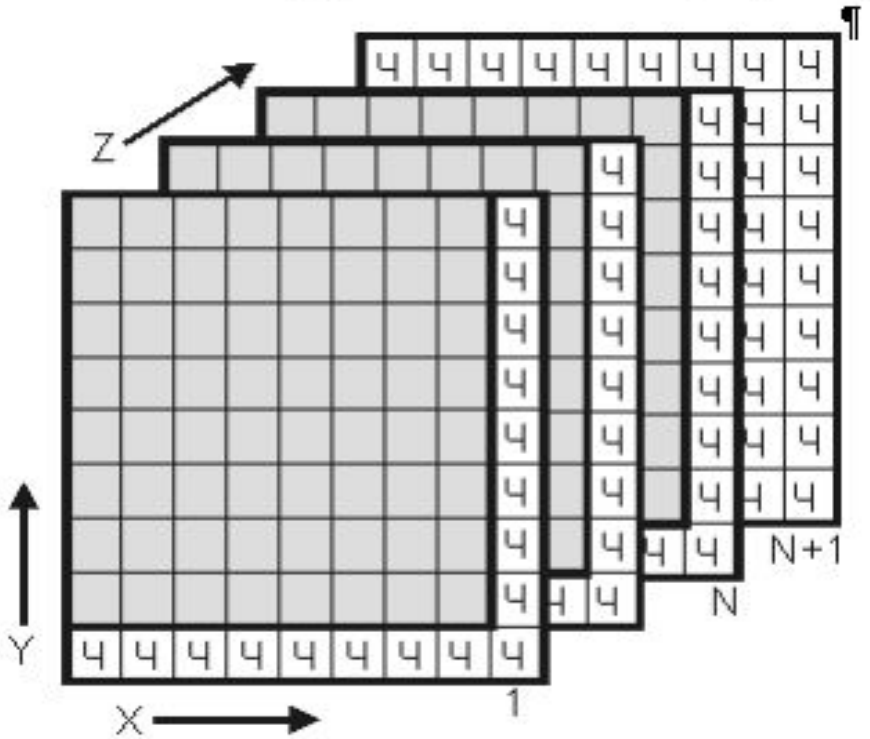


Рис.·1.·Метод·коррекции·более·одной·ошибки·в·блоке·данных·(битам·данных соответствуют·окрашенные·квадраты)¶

Коды Рида-Соломона были предложены в 1960 Ирвином Ридом (Irving S. Reed) и Густавом Соломоном (Gustave Solomon), являвшимися сотрудниками Линкольнской лаборатории

Популярным кодом Рида-Соломона является RS(255,223) с 8-битными символами. Каждое кодовое слово содержит 255 байт, из которых 223 являются информационными и 32 байтами четности.

Декодер может исправить любые 16 символов с ошибками в кодовом слове: то есть, ошибки могут быть исправлены, если число искаженных байт не превышает 16.

Коррекция ошибок

- Процедура коррекции ошибок предполагает два совмещенных процесса:
- обнаружение ошибки и
- определение места (идентификация сообщения и позиции в сообщении).
- После решения этих двух задач, исправление тривиально – надо инвертировать значение ошибочного бита.

- В наземных каналах связи, где вероятность ошибки невелика, обычно используется метод детектирования ошибок и повторной пересылки фрагмента, содержащего дефект.
- Для спутниковых каналов с типичными для них большими задержками системы коррекции ошибок становятся привлекательными. Здесь используют коды Хэмминга или коды свертки.

- Код Хэмминга представляет собой блочный код, который позволяет выявить и исправить ошибочно переданный бит в пределах переданного блока. Обычно код Хэмминга характеризуется двумя целыми числами, например, (11,7) используемый при передаче 7-битных ASCII-кодов. Предполагается, что имела место ошибка в одном бите и что ошибка в двух или более битах существенно менее вероятна.

- Например, пусть возможны следующие правильные коды (все они, кроме первого и последнего, отстоят друг от друга на расстояние 4):
- 00000000
11110000
00001111
11111111
- При получении кода 00000111 можно предположить, что правильное значение полученного кода равно 00001111. Другие коды отстоят от полученного на большее расстояние Хэмминга.

В общем случае код имеет N=M+C бит

- Пусть M=4, а N=7, тогда слово-сообщение будет иметь вид: M4, M3, M2, C3, M1, C2, C1. Теперь попытаемся вычислить значения C1, C2, C3. Для этого используются уравнения, где все операции представляют собой сложение по модулю 2:
- C1 = M1 + M2 + M4
  C2 = M1 + M3 + M4
  C3 = M2 + M3 + M4
- Для определения того, доставлено ли сообщение без ошибок, вычисляем следующие выражения (сложение по модулю 2):
- C11 = C1 + M4 + M2 + M1
  C12 = C2 + M4 + M3 + M1
  C13 = C3 + M4 + M3 + M2
- C11C12C13 Значение
- 1 2 4 Позиция бит
- 0 0 0 Ошибок нет
- 0 0 1 Бит C3 не верен
- 0 1 0 Бит C2 не верен
- 0 1 1 Бит M3 не верен
- 1 0 0 Бит C1 не верен
- 1 0 1 Бит M2 не верен
- 1 1 0 Бит M1 не верен
- 1 1 1 Бит M4 не верен

Циклические коды BCH (Bose-Chadhuri-Hocquenghem).

- Пусть общее число бит в блоке равно N, из них полезную информацию несут в себе K бит, тогда в случае ошибки, имеется возможность исправить m бит. Таблица содержит зависимость m от N и K для кодов BCH.

Общее число бит N	Число полезных бит M	Число исправляемых бит m
31	26	1
	21	2
	16	3
63	57	1
	51	2
	45	3
127	120	1
	113	2
	106	3

Процент обнаруживаемых ошибок

Число полезных бит M	Число избыточных бит (n-m)		
	6	7	8
32	48%	74%	89%
40	36%	68%	84%
48	23%	62%	81%

Рассмотрим пример передачи кода буквы s = 0x073 = **1110011** с использованием кода Хэмминга (11,7).

- Позиция бита: 11 10 9 8 7 6 5 4 3 2 1
- Значение бита: **1 1 1 * 0 0 1 * 1 * ***
- 11 =1011 10 =1010 09 =1001 05 =0101 03 =0011 контр. сумма=1110
- Приемник получит код
- Позиция бита: 11 10 9 8 7 6 5 4 3 2 1
- Значение бита: **1 1 1 1 0 0 1 1 1 1 0**

- Просуммируем снова коды позиций ненулевых битов и получим нуль.
- 11 =1011 10 =1010 09 =1001 08 =1000 05 =0101 04 =0100 03 =0011 02 =0010 **сумма =0000**

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

Код RS(255,223) может исправить до 16 ошибок в символах.

В худшем случае, могут иметь место 16 битовых ошибок в разных символах (байтах). В лучшем случае, корректируются 16 полностью неверных байт, при этом исправляется 16 x 8=128 битовых ошибок.

Очень распространены коды Рида-Соломона, работающие с байтами (октетами).

Циклы де Брейна названы по имени голландского математика Н. Г. де БрейнаЦиклы де Брейна названы по имени голландского математика Н. Г. де Брейна (англ.Циклы де Брейна названы по имени голландского математика Н. Г. де Брейна (англ.) (англ.Циклы де Брейна названы по имени голландского математика Н. Г. де Брейна (англ.) (англ. Nicolaas Govert de BruijnЦиклы де Брейна названы по имени голландского математика Н. Г. де Брейна (англ.) (англ. Nicolaas Govert de Bruijn), который рассматривал их в 1946 году

Примеры циклов де Брейна для *k*= 2 с периодом 2, 4, 8, 16:

01 (содержит подпоследовательности 0 и 1)

0011 (содержит подпоследовательности 00, 01, 11, 10)

00010111 (000, 001, 010, 101, 011, 111, 110, 100)

0000100110101111