А. Н. Мальчуков, А. Н. Осокин

Быстрое вычисление контрольной суммы CRC: таблица против матрицы

Контрольная сумма — это некоторое значение, вычисленное для последовательности байт данных с помощью определенного алгоритма, которое используется на приемной стороне для подтверждения корректности полученных данных. В статье рассматривается быстродействующий алгоритм вычисления контрольной суммы, легко реализуемый на комбинационных схемах.

Введение

Первоначально контрольная сумма использовалась в системах с наличием обратной связи (переспроса) для обнаружения ошибок, возникающих в зашумленных каналах связи. Позднее, с развитием криптографических хеш-функций (алгоритмов хеширования), контрольные суммы стали использоваться для подтверждения целостности и подлинности данных. Обычно контрольная сумма посылается (считывается) в конце сообщения:

<блок данных> <контрольная сумма>.

В настоящее время существует множество алгоритмов получения контрольной суммы: сложение байтов, CRC (англ. cyclic redundancy check — избыточный циклический код), MD5, SHA и т. д. CRC традиционно используется в проводных и беспроводных протоколах передачи данных (Ethernet, Bluetooth, ZigBee, CAN, Fibre Channel и т. д.) для контроля целостности управляющих фрагментов или кадров данных. Далее речь пойдет об алгоритмах вычисления контрольных сумм CRC.

«Стандартный» алгоритм

Под «стандартным» алгоритмом подразумевается алгоритм, вычисляющий контрольную сумму CRC побитно, т. е. в каждом такте (итерации) данные последовательно продвигаются в некотором регистре на один бит, и в итоге в этом регистре получают контрольную сумму. Этот алгоритм широко известен [1] по его аппаратной реализации на регистрах с обратной связью (рис. 1).

На рис. 1 схематично показана работа простого алгоритма. Для CRC8 его можно описать следующим образом.

Начало. Регистр (массив) 8 бит содержит нулевое значение, данные поступают в регистр через его младший разряд к старшим, начиная со старшего разряда данных последовательным сдвигом.

Шаг 1. Сдвигаем данные в регистре на один бит от младших к старшему разряду, в младший разряд регистра заносится бит из потока данных.

Шаг 2. Если выдвинутый бит из 8 разряда регистра равен 0, переходим на шаг 4.

Шаг 3. Складываем по модулю два выдвинутый бит с разрядами регистра 1, 2, 3 и результат записываем в соответствующие ячейки регистра, т. е. фактически инвертируем содержимое 1, 2 и 3 разрядов регистра.

Шаг 4. Если еще не все биты поступили в регистр из потока данных, переходим на шаг 1.

Конец. В регистре содержится контрольная сумма CRC8.

При вычислении контрольной суммы CRC8 для последовательности битов данных в конец добавляют 8 нулей. При проверке контрольной суммы через регистр пропускают последовательность битов данных вместе с контрольной суммой в конце. В итоге

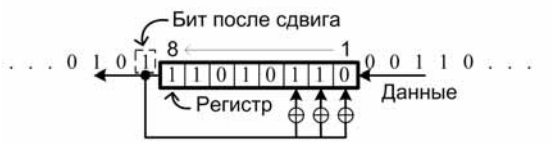

Рис. 1. Схематичное представление работы стандартного алгоритма на примере CRC8 ($x^8 + x^2 + x + 1$); стрелка со знаком ⊕ обозначает сложение по модулю два выдвинутого бита с разрядом регистра, на который указывает стрелка и запись результата в эту же ячейку регистра

проверки нулевое значение регистра соответствует безошибочному приему. Если регистр содержит ненулевое значение, то произошло искажение данных в информационном блоке и/или в контрольной сумме.

Данный алгоритм вычисления контрольной суммы CRC8 требует выполнения множества итераций, что существенно замедляет процесс вычисления. Для ускорения вычисления контрольной суммы CRC в [2] предложен табличный алгоритм, в котором данные сдвигаются не по 1 биту за итерацию, а по 8 бит (байту).

Табличный алгоритм

При последовательном сдвиге данных по байту (вместо одного бита) за итерацию (такт) необходимо знать изменения, которые должны были бы произойти в течение 8 сдвигов при обычном алгоритме (для CRC8 — инвертирование трех младших разрядов в случае единичного значения выдвинутого разряда), поэтому эти изменения нужно предварительно вычислить и занести в таблицу (которая потребует соответствующего своим размерам объема памяти запоминающего устройства при программной или аппаратной реализации). Адресом в такой таблице будет служить содержимое регистра до сдвига (выталкиваемый байт при сдвиге). Далее содержимое таблицы складывается по модулю два со значением регистра, в котором содержится байт данных после сдвига (рис. 2).

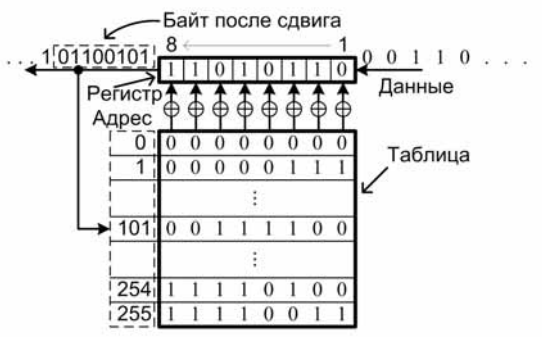

Рис. 2. Схематичное представление работы табличного алгоритма на примере CRC8 (x^8+x^2+x+1)

Табличный алгоритм вычисления контрольной суммы CRC8 можно описать следующим образом.

Начало. Регистр (массив) 8 бит содержит нулевое значение, данные поступают в регистр, начиная со старшего разряда данных последовательным сдвигом побайтно.

Шаг 1. Сдвигаем данные в регистре на один байт от младших к старшему разряду, в регистр заносится новый байт из потока данных.

Шаг 2. Выдвинутый из регистра байт задает адрес в предварительно подготовленной таблице.

Шаг 3. Выбранный по заданному адресу из таблицы байт складывается по модулю два с регистром.

Шаг 4. Если еще не все байты прошли через регистр из потока данных, переходим на шаг 1.

Конец. В регистре содержится контрольная сумма CRC8.

Процесс вычисления и проверки контрольной суммы не отличается от обычного алгоритма. В связи с тем, что в табличном алгоритме вместо побитового сдвига каждую итерацию данные сдвигаются на байт, длина потока данных должна быть кратной 8 битам, т. е. байту.

Однако, вместо использования таблицы из 256 байтов (для CRC8) можно обойтись матрицей в 8x8 бит (8 байт), которая легко реализуется на комбинационных схемах.

Матричный алгоритм

Процесс вычисления и проверки контрольной суммы CRC8 в матричном алгоритме осуществляется так же, как и в табличном, только вместо таблицы используется операция умножения вектора (выдвинутый байт) на матрицу (рис. 3). Матрица была специальным образом сформирована для кодового слова длиной 16 битов и образующего полинома x^8+x^2+x+1 [3].

Также необходимо отметить, что в результате умножения вектора на матрицу получаем значение, идентичное содержимому таблицы по соответствующему адресу в табличном алгоритме. На нашем примере: 11000111 ⊕ 11100000 ⊕ 00011100 ⊕ 00000111 = 00111100.

CRC32

Для удобства восприятия работа алгоритмов вычисления контрольной суммы была рассмотрена на примере CRC8, однако, наибольшее распространение получила контрольная сумма CRC32. Она используется для обнаружения ошибок, подтверждения целостности и подлинности данных в сетевом протоколе передачи данных IEEE 802.3; в стандартах цифрового кодирования видео и аудио сигналов MPEG-2; в формате хранения графической информации PNG; в алгоритмах сжатия данных RAR, ZIP

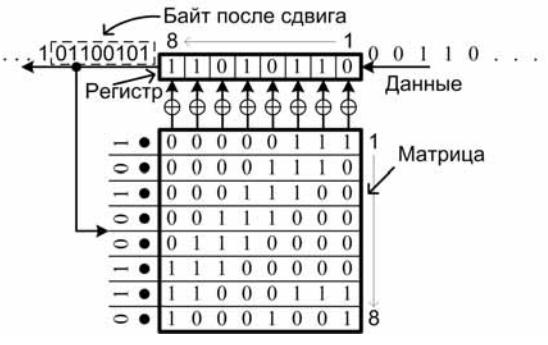

Рис. 3. Схематичное представление работы матричного алгоритма на примере CRC8 (x^8+x^2+x+1)

и др.; в устройствах хранения данных с интерфейсом SATA, UDMA; в стандартах обеспечения совместимости Unix-подобных ОС POSIX; в протоколе управления системами хранения данных, серверами и клиентами iSCSI; в модели обмена аэронавигационной информации AIXM и т. д.

Однобайтовый сдвиг

Выше был представлен быстродействующий матричный алгоритм вычисления контрольной суммы CRC8 при обработке одного байта за итерацию, в котором вместо одного бита за итерацию через результирующий регистр проходит 8 бит данных. Для CRC32 алгоритм будет тем же, за исключением матрицы, с помощью которой производится вычисление контрольной суммы (рис. 4).

На рис. 4 матрица вычислена по алгоритму, который описан в работе [3], с образующим полиномом $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$.

При программной реализации матричного алгоритма вычисления контрольной суммы CRC32 для хранения матрицы потребуется объем запоминающего устройства равный 32 байтам, в случае реализации алгоритма, предложенного в работе [2], потребуется 1024 байта.

При аппаратной реализации матричного алгоритма вычисления контрольной суммы CRC32 запоминающего устройства не требуется, вместо этого на комбинационных схемах воспроизводится умножение вектора на матрицу по модулю два (рис. 5).

На рис. 5 обозначено ВБХ — бит X выдвинутого байта; РХХ — бит XX из регистра после сдвига; З-РХХ — данный бит будет записан на место XX бита в регистр по завершению итерации.

Стоит отметить, что в устройствах хранения данных широко применяется практика работы с блоками данных, равных не одному байту, а 1024 байтам, 4096 байтам и даже 1 048 576 байтам (1 Мб). Контрольная сумма вычисляется для целого блока. В та-

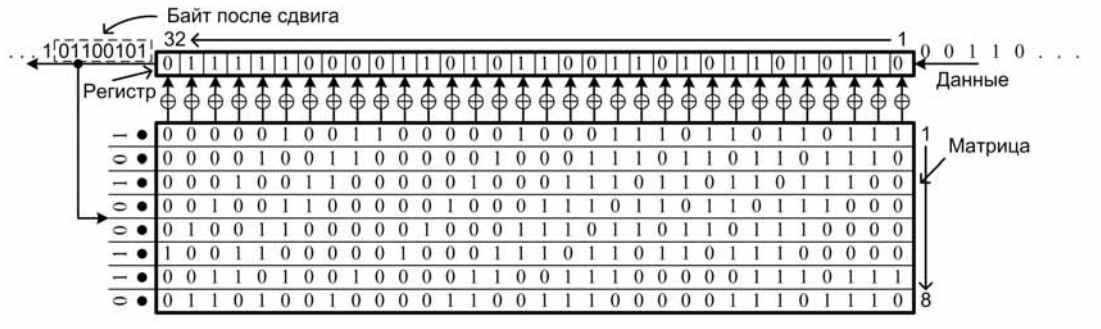

Рис. 4. Схематичное представление работы матричного алгоритма вычисления CRC32 ($x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$)

ких случаях блоки данных являются кратными 2, 4, 8 байтам, что в свою очередь позволяет быстрее вычислить контрольную сумму за счет обработки не одного байта за итерацию, а 2, 4 или 8 байтов. Рассмотрим пример обработки 4 байтов за итерацию.

Четырехбайтовый сдвиг

В случае вычисления контрольной суммы CRC32 сразу для 4 байтов данных за итерацию матрица будет иметь большее количество строк — 32 вместо 8 (рис. 3).

В этом случае при программной реализации матричного алгоритма вычисления контрольной суммы CRC32 для хранения матрицы потребуется объем запоминающего, устройства равный 128 байтам, а в случае реализации алгоритма, предложенного в работе [2], потребуется 16 Гб.

Также можно заметить, что на рис. 4 и 6 первые 8 строк в матрице совпадают. Это особенность алгоритма матричного деления — для одного образующего полинома матрица существует одна, количество строк зависит от длины блока данных, поэтому на рис. 4 представлена укороченная версия матрицы с рис. 6.

На рис. 7 показан пример реализации блока умножения вектора на матрицу по модулю два для данного случая. Обозначения те же, что и на рис. 5.

Заключение

Предложенный матричный алгоритм вычисления контрольной суммы CRC позволяет обходиться без использования запоминающего устройства при аппаратной реализации. А при программной реализации (например, при реализации на DSP) необходимый объем памяти для хранения матрицы значительно меньше, чем требуется для хранения таблицы при использовании табличного алгоритма. Например, для CRC8

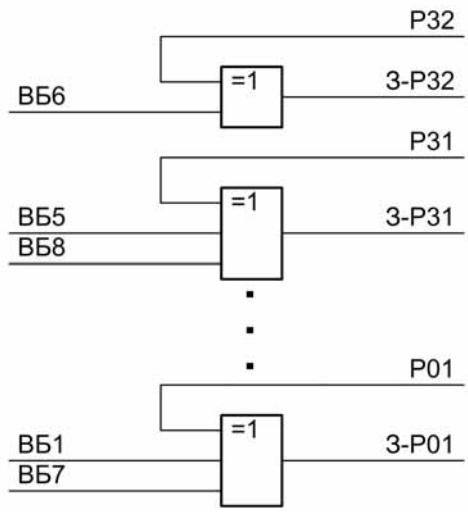

Рис. 5. Пример аппаратной реализации блока умножения вектора на матрицу по модулю два

при вычислении по одному байту за итерацию матричному алгоритму потребуется 8 байтов, табличному — 256 байтов; при вычислении по два байта за итерацию матричному алгоритму потребуется 16 байтов, табличному — 64 Кб; при вычислении по 4 байта за итерацию матричному — 32 байта, табличному — 4 Гб; для CRC32 при вычислении по одному байту за итерацию матричному алгоритму потребуется 32 байта, табличному — 1024 байта (1 Кб); при вычислении по два байта за итерацию матричному алгоритму потребуется 64 байта, а табличному — 256 Кбайтов; при вычислении по 4 байта за итерацию матричному — 128 байтов, а табличному — 16 Гб.

Предлагаемый матричный алгоритм быстрого вычисления контрольной суммы CRC актуален при использовании более длиной контрольной суммы (CRC8, 16, 32, 64, 128). Особенно он полезен в случаях, когда блоки данных кратны 2, 3, 4 байтам и т.д., что позволяет быстрее вычислять контрольную сумму CRC.

Ускорение за счет использования сдвига на большее количество битов влечет за со-

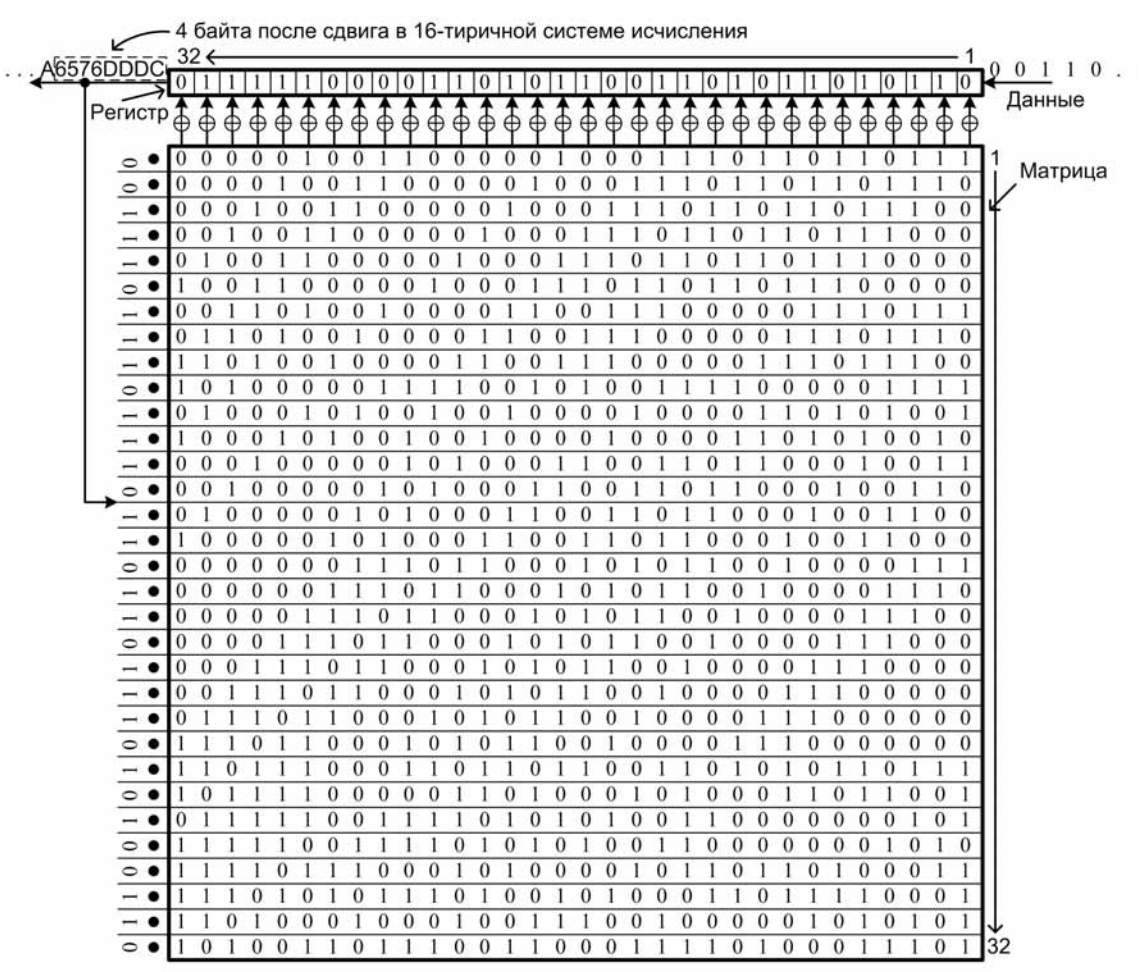

Рис. 6. Схематичное представление работы матричного алгоритма вычисления CRC32 при сдвиге по 4 байта за итерацию

бой серьезный рост объема требуемой памяти запоминающего устройства для алгоритма из работы [2] и лишь небольшой рост объема требуемой памяти (для хранения матрицы) при программной реализации или незначительное усложнение комбинационной схемы при аппаратной реализации для предлагаемого матричного алгоритма.

Совместное использование ускоренного метода вычисления контрольных сумм CRC с быстродействующими кодеками помехоустойчивых полиномиальных кодов [4] позволит обнаруживать ошибки большей кратности, которые кодек помехоустойчивого полиномиального кода не способен исправлять.

Также возможно использование матричного алгоритма в микроконтроллерах жестких дисков для обеспечения проверки на целостность больших блоков данных с помощью новых, более длинных контрольных сумм CRC, например, CRC256 или CRC512.

Работа выполнена на кафедре вычислительной техники института «Кибернетический центр» Томского политехнического университета и поддержана грантом «У. М. Н. И. К» Фонда содействия развитию малых форм предприятий в научно-технической сфере.

Список литературы

1. *Темников Ф. Е., Афонин В. А., Дмитриев В. И.* Теоретические основы информационной техники: учебное пособие. — 2-е изд., испр. и доп. — М.: Энергия, 1979. — 512 с.
2. *Ross N. Williams*. A Painless Guide to CRC Error Detection Algorithms [Электронный ресурс]. — Режим доступа: http://www.ross.net/crc/download/crc_v3.txt, свободный.
3. *Буркатовская Ю. Б., Мальчуков А. Н., Осокин А. Н.* Быстродействующие алгоритмы деления полиномов в арифметике по модулю два. // Известия Томского политехнического университета. 2006 — № 1 — С. 19–24.
4. *Мальчуков А. Н., Осокин А. Н.* Реализация системы автоматизированного проектирования кодеков помехоустойчивых кодов короткой длины // Прикладная информатика, 2008 — № 6 (18). — С. 106–112.

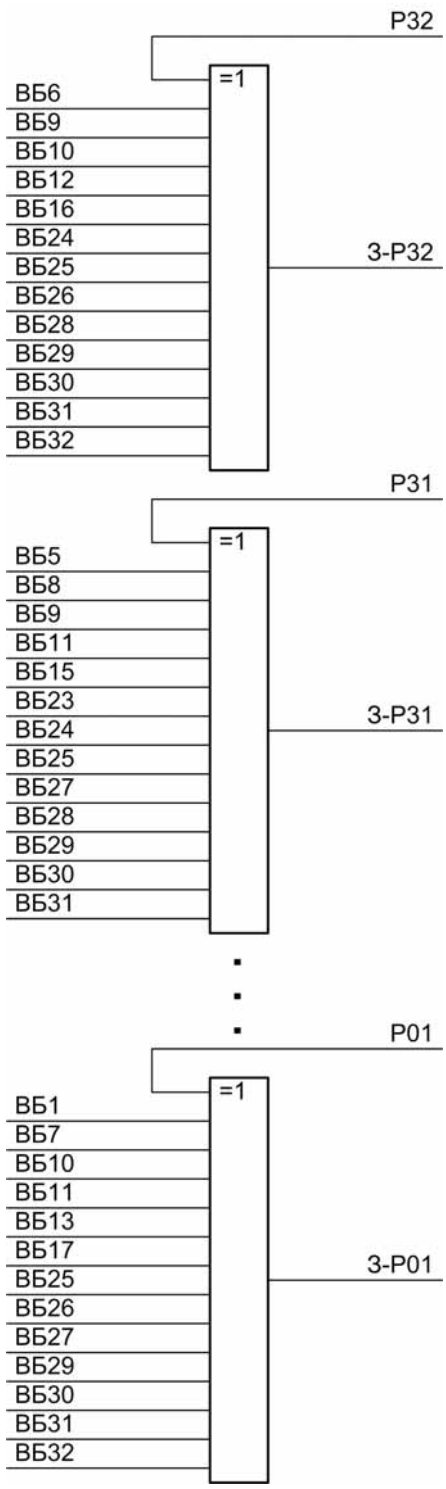

Рис. 7. Пример аппаратной реализации блока умножения вектора на матрицу по модулю два