Алгоритмическое и программное обеспечение

УДК 681.3.06

ИССЛЕДОВАНИЕ ПРОГРАММНЫХ РЕАЛИЗАЦИЙ АЛГОРИТМОВ ВЫЧИСЛЕНИЯ CRC, СОВМЕСТИМЫХ С PKZIP, WINRAR, ETHERNET

Е.А. Мыцко, А.Н. Мальчуков

Томский политехнический университет
E-mail: EvgenRus70@mail.ru; jgs@tpu.ru

Приведены численные результаты компьютерного эксперимента по определению быстродействия программных реализаций алгоритмов вычисления CRC32. Показано, что быстродействующий четырёхбайтовый матричный алгоритм следует применять во встраиваемых системах, промышленных системах передачи данных.

Ключевые слова:
Контрольная сумма, циклический избыточный код, табличный алгоритм, матричный алгоритм, программная реализация.

Key words:
Check sum, cyclic redundancy code, table-driven algorithm, matrix-driven algorithm, software implementation.

Существуют различные алгоритмы вычисления контрольной суммы CRC и способы их реализации. Стандартный алгоритм подразумевает побитное вычисление контрольной суммы путём деления полинома, представляющего данные, на образующий полином, используемый для вычисления CRC в различных протоколах передачи данных (Ethernet, ZigBee) и архиваторах (Pkzip, WinRAR). Также существует табличный алгоритм [1], ускоряющий расчёт контрольной суммы за счёт побайтного вычисления. Далее речь пойдёт о программных реализациях алгоритмов вычисления CRC32, используемых в Pkzip, WinRAR, Ethernet.

Классическая реализация алгоритма вычисления CRC

Основой для данного алгоритма является полиномиальная арифметика [2]. Классическая реализация расчёта контрольной суммы заключается в побитовом делении полинома, представляющего данные (файл) на образующий полином. Образующий полином вычисляется в зависимости от сферы применения алгоритма. В данном случае это будет полином, используемый в Pkzip, WinRAR, Ethernet.

Классический алгоритм реализуется с помощью последовательного итерационного сдвига данных в регистре с обратной связью на один бит [3]. В результате проделанных операций в регистре вычисляется контрольная сумма CRC.

После обработки всех данных в регистре будет находиться контрольная сумма CRC32 (рис. 1). Основным недостатком данного алгоритма является то, что в один момент времени происходят вычисления для одного бита данных, что значительно замедляет его работу. В связи с этим для повышения быстродействия используются алгоритмы, выполняющие вычисления побайтно.

Табличный алгоритм

Особенностью табличного алгоритма является то, что данные считываются побайтно, что значительно ускоряет вычисление CRC. При расчете контрольной суммы используется таблица с предвычисленными значениями на основе образующе-

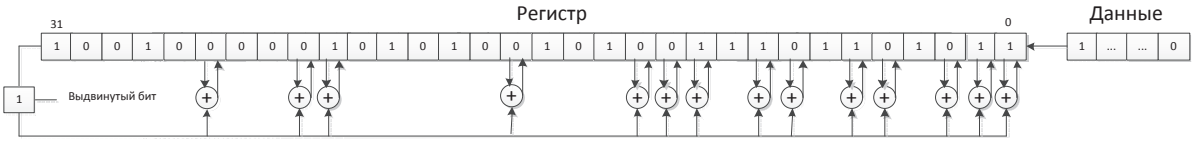

Рис. 1. *Схематичное представление реализации классического алгоритма вычисления CRC32*

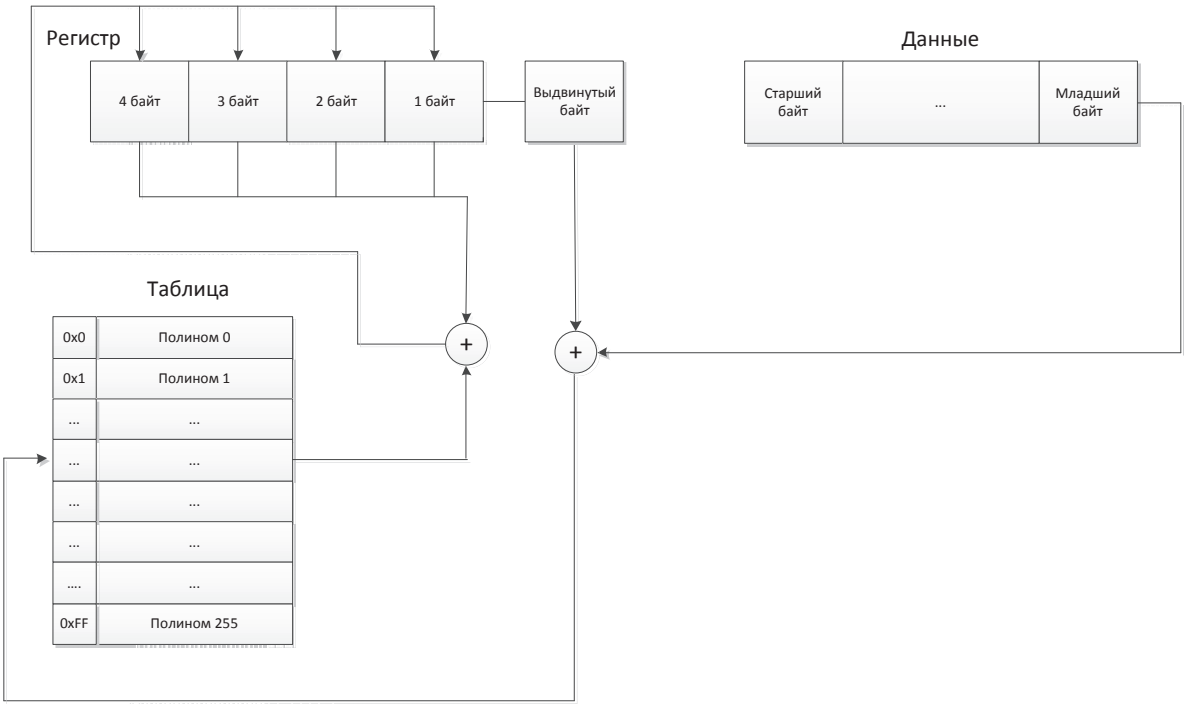

Рис. 2. *Схематичное представление табличного алгоритма*

го полинома. В алгоритмах Pkzip, WinRAR, Ethernet используется образующий полином 104C11DB7h [4], представленный в шестнадцатеричной системе счисления. На рис. 2 схематично представлен табличный алгоритм.

Матричный алгоритм

Процесс вычисления контрольной суммы CRC32 в матричном алгоритме осуществляется так же, как и в табличном, за исключением того, что вместо таблицы используется операция умножения вектора (выдвинутый байт) на матрицу по модулю 2 (рис. 3, 4).

При программной реализации матричного алгоритма (однобайтовый сдвиг) вычисления контрольной суммы CRC32 потребуется объём памяти равный 32 байтам (для хранения матрицы).

Матричный алгоритм можно ускорить, если вместо однобайтового сдвига использовать сдвиг по 2 или 4 байта. Сдвиг по 3 байта в данном случае не рассматривается, так как при программной реализации отсутствует тип данных размером 3 байта, что вызывает трудности и потерю скорости при реализации алгоритма.

При увеличении числа байт, обрабатываемых за итерацию, будет увеличиваться размер матрицы,

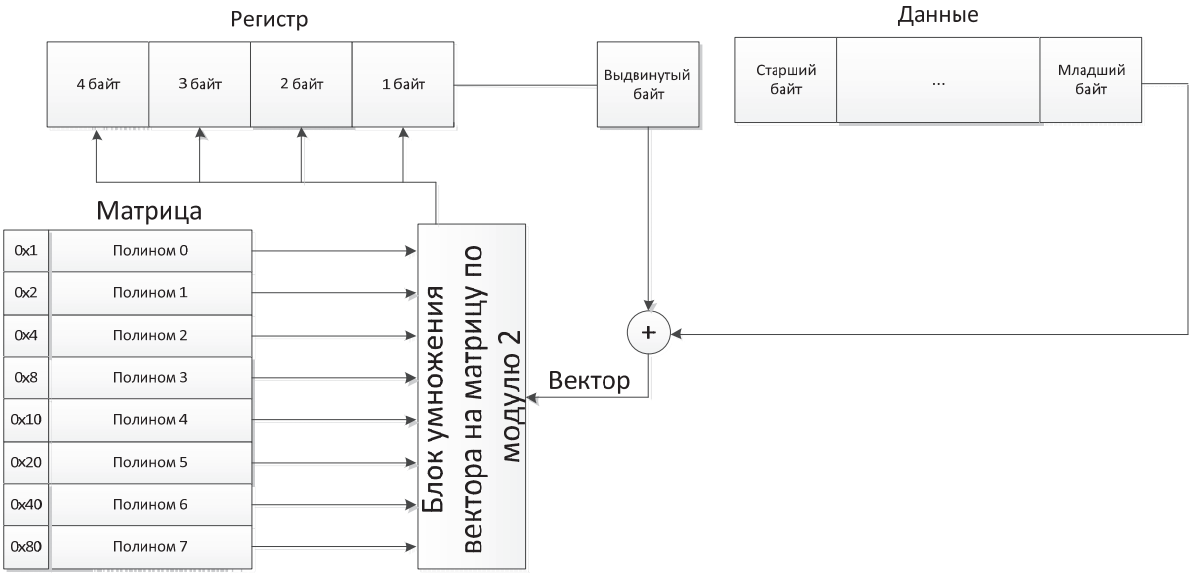

Рис. 3. *Схематичное представление матричного алгоритма со сдвигом 1 байт*

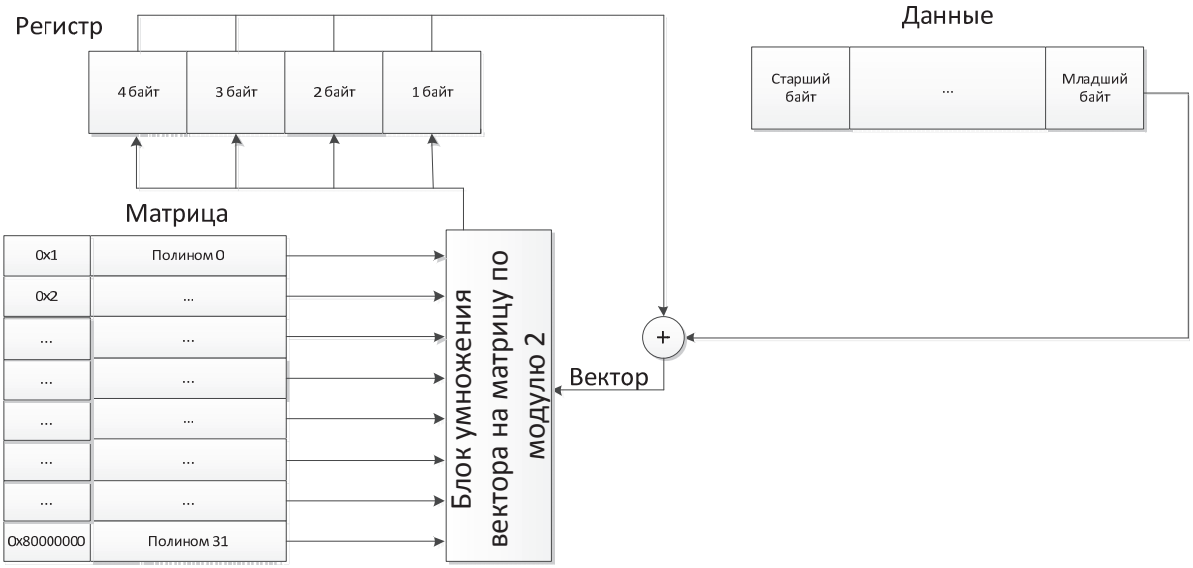

Рис. 4. *Схематичное представление матричного алгоритма со сдвигом 4 байта*

используемой при вычислениях. Для двухбайтового сдвига матрица расширится до 64-х байт; для четырёхбайтового сдвига – до 128 байт [5].

Компьютерный эксперимент

Для постановки компьютерного эксперимента по определению быстродействия программных реализаций различных алгоритмов вычисления контрольной суммы CRC32 использовался суперкомпьютерный кластер «СКИФ-политех1». В табл. 1 представлена конфигурация суперкомпьютерного кластера «СКИФ-политех1» [6]. Для получения временной оценки вычисления контрольной суммы над конкретным файлом конкретной программной реализацией производилось по 50 запусков для файлов объёмом от 10 до 1010 мегабайт с шагом 200 мегабайт. На основе данных по 50-ти запускам каждого алгоритма составлены таблицы средних значений по времени расчёта контрольной суммы, по доверительным интервалам и по размерам исполняемых файлов.

Исследование проводилось в 3 этапа: с буфером в 1 Мб, который установлен в исходном коде табличной реализации [7]; с буферами 8 и 1 байт – для моделирования вычисления контрольной суммы для микроконтроллеров.

Таблица 1. *Конфигурация «СКИФ-политех1»*

Аппаратная конфигурация	Программная конфигурация
Количество вычислительных узлов: 24 Количество процессоров: 48 (Intel XEON 5150) Количество вычислительных ядер: 96 (2,66 Ghz) Общий объем ОП: 192 ГБ Общий объем HDD: 2880 ГБ Объем СХД: 5 ТБ Системная сеть: Infiniband 4x, 24 порта Вспомогательная сеть: Gigabit Etherhet, 48 портов Сервисная сеть: ServNet, 25 портов Пиковая производительность: 1,02 ТФЛОПС	Novell SLES 10 Microsoft Windows HPC Server 2008 Установлена на узлы КВК В том числе в комплекте ОС Программное обеспечение трансляции исходных текстов параллельных программ для языков C, C++ и FORTRAN E диная система авторизации пользователей. Поддержка одновременного исполнения команд на всех или выделенных узлах кластера

Буфер 1 мегабайт. Использование суперкомпьютерного кластера позволило сэкономить время на проведение эксперимента и получить временные оценки с достаточно высокой достоверностью (табл. 2).

Таблица 2. *Средние значения времени расчёта CRC32 и доверительные интервалы результата*

Алгоритм	Среднее значение времени расчёта CRC32, с (Доверительные интервалы, %)					
	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
Табличный	0,057 (±1,4)	1,194 (±0,217)	2,299 (±0,203)	3,227 (±0,162)	4,334 (±0,211)	6,184 (±0,155)
Матричный (сдвиг 1 байт)	0,125 (±0,732)	2,81 (±0,112)	5,399 (±0,067)	7,602 (±0,166)	10,23 (±0,161)	14,557 (±0,048)
Матричный (сдвиг 2 байта)	0,092 (±3,882)	2,084 (±0,133)	4,02 (±0,09)	5,645 (±0,105)	7,566 (±0,097)	10,835 (±0,072)
Матричный (сдвиг 4 байта)	0,08 (±0,827)	1,729 (±0,172)	3,32 (±0,011)	4,694 (±0,314)	6,272 (±0,188)	8,965 (±0,064)

Таблица 3. *Ускорение относительно однобайтового сдвига*

Алгоритм	Ускорение относительно матричного алгоритма со сдвигом 1 байт, %						Среднее ускорение, %
	Объём использованных файлов, Мб						
	10	210	410	610	810	1010	
МС 2 байта	26,4	25,836	25,541	25,743	26,041	25,568	25,85
МС 4 байта	36	37,919	38,507	38,253	38,69	38,414	37,96

Для того чтобы выяснить, как влияет на скорость вычисления увеличение числа байт, обрабатываемых за итерацию, в матричном алгоритме была вычислена разница для данных из табл. 2 для матричных алгоритмов с 2- и 4-байтными сдвигами относительно однобайтового (табл. 3).

Как видно из табл. 3, матричный алгоритм с 4-байтным сдвигом вычисляет контрольную сумму на 37,96 % быстрее, чем матричный алгоритм с однобайтовым сдвигом.

Однако даже матричный алгоритм с 4-байтовым сдвигом уступает по скорости табличному алгоритму на 44,11 % (табл. 4).

В связи с увеличением числа байт до 4, обрабатываемых за итерацию, в матричном алгоритме разрыв по быстродействию относительно табличного алгоритма сокращается до ~44 %. Следует помнить, что четырёхбайтовый матричный алгоритм требует в 8 раз меньше памяти (128 байт) для хранения матрицы, чем для реализации табличного алгоритма (1024 байт).

Моделирование вычисления CRC32 для микроконтроллеров

Для моделирования скорости вычисления алгоритмов для микроконтроллеров были исследованы различные варианты программных реализаций с установленными буферами 8 и 1 байт.

Буфер 8 байт. В данном случае размер буфера был установлен равным 8 байт, что соответствует доступному объёму для микроконтроллеров. Все этапы исследования проводились аналогично исследованиям с буфером 1 мегабайт. Временные затраты на вычисления контрольной суммы файлов разного объёма и доверительные интервалы полученных результатов приведены в табл. 5. Аналогично предыдущему этапу в табл. 6 и 7 приведены результаты ускорений от однобайтового сдвига и от табличного алгоритма.

Как видно из табл. 6 и 7, уменьшение размера буфера до 8 байт изменило картину, полученную ранее.

Таблица 6. *Ускорение относительно однобайтового сдвига*

Алгоритм	Ускорение относительно матричного алгоритма со сдвигом 1 байт, %						Среднее ускорение, %
	Объём использованных файлов, Мб						
	10	210	410	610	810	1010	
МС 2 байта	31,4	30,91	31,2	30,63	30,62	30,82	30,93
МС 4 байта	47,19	45,9	47	39,82	45,53	45,44	45,14

Таблица 7. *Ускорение матричного алгоритма относительно табличного*

Алгоритм	Ускорение относительно табличного алгоритма, %						Среднее ускорение, %
	Объём использованных файлов, Мб						
	10	210	410	610	810	1010	
МС 1 байт	−67,92	−58,543	−61,92	−59,82	−59,83	−63,45	−61,91
МС 2 байта	−15,09	−9,533	−11,28	−10,76	−10,88	−13,06	−11,76
МС 4 байта	11,32	14,242	12,29	12,58	12,93	10,78	12,35

Для двухбайтового матричного алгоритма ускорение относительно однобайтового увеличилось в среднем до 30,93 %, в то время как для четырехбайтового алгоритма – до 45,14 %. Также из табл. 7 видно, что для однобайтового и двухбайтового алгоритма уменьшилось отставание по скорости

Таблица 4. *Ускорение матричного алгоритма относительно табличного*

Алгоритм	Ускорение относительно табличного алгоритма, %						Среднее ускорение, %
	Объём использованных файлов, Мб						
	10	210	410	610	810	1010	
С 1 байт	−119,298	−135,343	−134,841	−135,575	−136,575	−135,398	−132,83
МС 2 байта	−61,403	−74,539	−74,858	−74,93	−74,573	−75,21	−72,58
МС 4 байта	−40,35	−44,807	−44,41	−45,46	−44,716	−44,97	−44,11

Таблица 5. *Средние значения времени расчёта CRC32 и доверительные интервалы результата*

Алгоритм	Средние значения времени расчёта CRC32, с (Доверительные интервалы, %)					
	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
Табличный	0,106 (±1,789)	2,183 (±1,966)	4,139 (±0,980)	6,166 (±1,537)	8,239 (±1,0067)	10,003 (±1,082)
Матричный (сдвиг 1 байт)	0,178 (±1,088)	3,461 (±0,521)	6,702 (±0,328)	9,855 (±0,474)	13,169 (±0,404)	16,350 (±0,371)
Матричный (сдвиг 2 байта)	0,122 (±1,447)	2,39112 (±0,516)	4,606 (±0,408)	6,83664 (±0,463)	9,13604 (±0,366)	11,31524 (±0,319)
Матричный (сдвиг 4 байта)	0,094 (±2,580)	1,87208 (±0,386)	3,6392 (±0,464)	5,39032 (±0,267)	7,1732 (±0,308)	8,92452 (±0,337)

Таблица 8. *Средние значения времени расчёта CRC32 и доверительные интервалы результата*

Алгоритм	Средние значения времени расчёта CRC32, с (Доверительные интервалы, %)					
	Объём использованных файлов, Мб					
	10	210	410	610	810	1010
Табличный	0,315 (±2,526)	6,044 (±0,598)	11,797 (±1,36)	17,452 (±1,082)	23,152 (±0,985)	28,767 (±0,896)
Матричный (сдвиг 1 байт)	0,361 (±0,637)	7,152 (±0,630)	13,958 (±0,621)	20,610 (±0,6483)	27,558 (±0,5331)	34,129 (±0,523)
Матричный (сдвиг 2 байта)	0,216 (±0,786)	4,269 (±0,611)	8,263 (±0,614)	12,243 (±0,374)	16,389 (±1,091)	20,221 (±0,432)
Матричный (сдвиг 4 байта)	0,140 (±1,092)	2,744 (±0,664)	5,292 (±0,464)	7,858 (±0,423)	10,546 (±0,864)	13,090 (±0,552)

вычисления до 61,91 и 11,76 % соответственно; в то время как матричный четырехбайтовый алгоритм уже имеет опережение по скорости в среднем на 12,35 % относительно табличного.

Буфер 1 байт. Для буфера размером 1 байт результаты исследования приведены в табл. 8–10.

Таблица 9. *Ускорение относительно однобайтового сдвига*

Алгоритм	Ускорение относительно матричного алгоритма со сдвигом 1 байт, %						Среднее ускорение, %
	Объём использованных файлов, Мб						
	10	210	410	610	810	1010	
МС 2 байта	40,166	40,31	40,8	40,596	40,529	40,751	40,52
МС 4 байта	61,218	61,633	62,086	61,872	61,731	61,645	61,69

При исследовании реализации с буфером 1 байт, что по сути можно приравнять к вычислениям без буфера при побайтовой передаче данных, было установлено, что ускорение матричных алгоритмов относительно однобайтового матричного алгоритма увеличилось до 40,52 и 61,69 % соответственно. Исследование же ускорений матричных алгоритмов относительно табличного показало, что уже двухбайтовый матричный алгоритм опережает табличный на 29,91 %, в то время как четырехбайтовый – на 54,86 %, что является значительным приростом быстродействия относительно табличного алгоритма.

Полученные результаты характеризует тот факт, что при уменьшении буфера до 1 байта потери скорости вычислений для табличного алгоритма намного значительней, чем для матричных алгоритмов, в то время как быстродействие обоих алгоритмов с сокращением размера буфера уменьшается.

Заключение

В работе приведены результаты исследования быстродействия различных реализаций вычисления контрольной суммы CRC32. Поставленный компьютерный эксперимент по определению быстродействия программных реализаций алгоритмов вычисления CRC32 показал, что матричный алгоритм хорошо подходит для применения в микроконтроллерах, где доступный объем памяти не превышает 8 байт, в то время как табличный алгоритм следует применять в системах с большим доступным объемом памяти, так как в реализациях для микроконтроллеров быстродействие данного алгоритма существенно падает.

В результате было установлено, что самый быстродействующий из матричных алгоритмов – четырёхбайтовый, следует применять во встраиваемых системах, промышленных системах передачи данных, где используются микроконтроллеры. Рекомендация использования матричного алгоритма обусловлена простотой реализацией и значительной экономией памяти, требуемой для управляющей программы микроконтроллера, а также хорошими показателями скорости по сравнению с табличной реализацией для применения в данной сфере.

Исследование выполнено при поддержке Министерства образования и науки Российской Федерации, соглашение 14.B37.21.0457.

Таблица 10. *Ускорение матричного алгоритма относительно табличного*

Алгоритм	Ускорение относительно табличного алгоритма, %						Среднее ускорение, %
	Объём использованных файлов, Мб						
	10	210	410	610	810	1010	
МС 1 байт	−14,603	−18,332	−18,318	−18,095	−19,030	−18,639	−17,83
МС 2 байта	31,428	29,367	29,956	29,847	29,211	29,707	29,91
МС 4 байта	55,555	54,599	55,141	54,973	54,448	54,496	54,86

СПИСОК ЛИТЕРАТУРЫ

1. Ross N.W. A Painless Guide to CRC Error Detection Algorithms // Dr Ross Williams. 1993. URL: http://www.ross.net/crc/download/crc_v3.txt (дата обращения: 23.10.2012).
2. Буркатовская Ю.Б., Мальчуков А.Н., Осокин А.Н. Быстродействующие алгоритмы деления полиномов в арифметике по модулю два // Известия Томского политехнического университета. – 2006. – Т. 309. – № 1. – С. 19–24.
3. Темников Ф.Е., Афонин В.А., Дмитриев В.И. Теоретические основы информационной техники. 2-е изд., испр. и доп. – М.: Энергия, 1979. – 512 с.
4. Koopman P. 32-Bit Cyclic Redundancy Codes for Internet Applications // Intern. Conf. on Dependable Systems and Networks (DSN). – Washington, July 2002. – P. 459–468.
5. Мальчуков А.Н., Осокин А.Н. Быстрое вычисление контрольной суммы CRC: таблица против матрицы // Прикладная информатика. – 2010. – № 2 (26). – С. 58–63.
6. Центр коллективного пользования «Суперкомпьютерный кластер «СКИФ-политех» // Сайт ЦКП СКК «СКИФ-политех» ТПУ. 2008. URL: http://cluster.tpu.ru/ (дата обращения: 23.10.2012).
7. 32 bit Cyclic Redundancy Check Source Code for C++ // Create Window Website. 2011. URL: http://www.createwindow.com/programming/crc32/ (дата обращения: 01.05.2011).

Поступила 25.12.2012 г.

УДК 004.62

ЭФФЕКТИВНОСТЬ РЕЖИМОВ ВНУТРЕННЕГО ПРЕДСКАЗАНИЯ БЛОКОВ В СОВРЕМЕННЫХ СТАНДАРТАХ СЖАТИЯ ВИДЕО

М.П. Шарабайко, Н.Г. Марков

Томский политехнический университет
E-mail: sme_box@tpu.ru

В 2013 г. ожидается финальная версия стандарта видеокомпрессии H.265/HEVC. Предполагается, что с его внедрением начнется массовый переход на системы сверхвысокого разрешения. В статье приведены результаты анализа эффективности методов внутреннего предсказания блоков стандарта H.265/HEVC с точки зрения их влияния на степень сжатия опорных кадров цифрового видео. Кроме того, приводятся результаты сравнения эффективности сжатия опорных кадров в стандартах H.265/HEVC и H.264/AVC.

Ключевые слова:
Сжатие видео, стандарт H265/HEVC, стандарт H.264/AVC, режимы внутреннего предсказания блоков, опорные кадры.

Key words:
Video compression, standard H.265/HEVC, standard H.264/AVC, intra prediction modes, key frames.

Актуальность задачи цифрового сжатия видеопоследовательностей растет вместе с развитием сетей передачи данных, в том числе беспроводных, а также возможностей и мощностей персональных и мобильных компьютерных устройств. При этом требования к степени сжатия видео непрерывно возрастают, стимулируя, в свою очередь, развитие стандартов видеокодирования.

Первый общепринятый стандарт видеокомпрессии *MPEG*2 [1] был окончательно принят в 1996 г., после чего началось быстрое развитие цифрового спутникового телевещания. Следующим стандартом стал *MPEG*4 *part* 10 (*H*.264/*AVC*) [2], принятый в 2003 г., что стало толчком к развитию систем *DVB-T/C*, Интернет ТВ и привело к появлению разнообразных сервисов видеообмена и видеосвязи.

В 2013 г. выходит стандарт цифрового сжатия видеопоследовательностей *H*.265/*HEVC* [3], являющийся стандартом нового поколения и обеспечивающий повышение степени сжатия видео до двух раз по сравнению со степенью сжатия видео с помощью текущего индустриального стандарта *H*.264/*AVC*.

В статье исследуется эффективность методов, положенных в основу стандартов *H*.265/*HEVC* и *H*.264/*AVC*, касающихся предсказания и сжатия опорных видеокадров.

Задача сжатия видеопоследовательностей

В цифровом представлении видеопоследовательность является потоком видеоданных, состоящим из последовательности видеокадров. Каждый кадр видеопоследовательности представляет определенный дискретный момент видеосигнала. Физически же кадр является матрицей пикселей изображения.

Большинство современных стандартов сжатия видеопоследовательностей реализуют блочную модель кодирования, суть которой заключается в разбиении кадра видеопоследовательности на множество непересекающихся блоков и выполнении алгоритма сжатия на каждом полученном блоке. В стандарте H.265/HEVC структура разбиения на блоки представляет собой квадродерево, подобное применяемому при фрактальном сжатии [4].