Арзуманян Р.В., Сухинов А.И.

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТИ РЕАЛИЗАЦИИ ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ПРОГРАММНОГО ДЕКОДЕРА GOOGLE VP9

Аннотация: *Предметом данной работы является оптимизация и организация параллельного исполнения стадий декодирования видеосигнала, сжатого в соответствие со спецификацией Google VP9. Подробно рассматриваются наиболее затратные с точки зрения машинного времени стадии декодирования и восстановления сжатого видеосигнала, исследуются возможности оптимизации и параллельного исполнения алгоритмов, лежащих в основе таких стадий как на центральных процессорах, так и на видеокартах с поддержкой вычислений общего назначения. Даётся комплексная оценка характеристик стадий декодирования, включая требования к производительности процессора и подсистемы памяти. Метод проведения работы – проведение численного эксперимента со сбором интересующей информации и последующим анализом результатов. Сбор информации реализован при помощи модификации исходного кода эталонного кодека и последующей сборки его в программное приложение - кодек. Новизна работы заключается в том, что в ней проведён комплексный анализ возможности вычислительных методов, лежащих в основе кодека и даны оценки возможности реализации параллельных вычислений с учётом особенностей целевого аппаратного обеспечения – MCCPU и GPGPU, а также проведена оптимизация стадии арифметического декодирования с учётом статистических особенностей распределения длин литералов, декодируемых из сжатого битового потока.В данной статье были сделаны выводы относительно наиболее вычислительно сложных стадий декодирования и возможности их оптимизации и параллельной реализации, а также проанализированы отличия от конкурирующего кодека H265.*

Ключевые слова: *кодек, Google VP9, производительность, арифметическое кодирование, доступ к памяти, оптимизация, межкадровое предсказание, анализ алгоритма, параллельное программирование, GPGPU*

Abstract: *The article is devoted to optimization and execution of parallel decoding stages of the video signal compressed in accordance with specification Google VP9. The authors in detail discuss the most time consuming stages of decoding and restoring a compressed video and study possible optimization*

DOI: 10.7256/2305-6061.2016.2.18430

and parallel execution of algorithms underlying such steps using both CPUs and graphics cards with general-purpose computing support. The article gives a comprehensive assessment of the characteristics of the decoding stages, including the requirements for processor and memory subsystem. The main method of the study is in carrying out a numerical experiment with the collection of information of interest and then analyzing the results. Gathering of information is implemented by modifying the source code reference codec and subsequent assembly into a software codec application. The novelty of the work lies in the fact that it carried out a comprehensive analysis of the possibility of computational methods lying in the codec based. The research evaluates the feasibility of parallel calculations, taking into account peculiarities of the target hardware (MCCPU and GPGPU). The authors performed an optimization of arithmetic decoding step taking into account the statistical characteristics of the distribution of the lengths of literals, decoded from a compressed bit stream.In this article, the authors make conclusions regarding the most computationally complex decoding stages and the possibility of their optimization and parallel implementation, and analyze differences between the described codec a competing codec N265.

Keywords:*inter-frame prediction, optimization, memory access pattern, arithmetic coding, performance, Google VP9, codec, algorithm analysis, parallel programming, GPGPU*

Введение

В данный момент существует 3 основных видеокодека – ITU-T H264[1, 2], ITU-T H265[3, 4] и Google VP9[5]. H264 и H265 являются стандартами ITU-T, использование данных кодеков предполагает лицензионные отчисления комитету MPEG LA. Google VP9 стандартом не является, доступна спецификация и исходные коды эталонного кодека, разработанного самой компанией Google. Лицензионная плата за использование данного кодека не взымается.

Наиболее распространённым видеокодеком на данный момент является H264. Новый стандарт H265, известный так же как HEVC (High Efficiency Video Codec) был принят комитетом ITU-T в 2013г, и на данный момент ещё не нашёл достаточно широкого применения в индустрии в силу новизны, высокой вычислительной сложности и вопросов лицензирования. Продвигаемый компанией Google кодек VP9 в настоящее время в основном используется в веб-сервисе YouTube.

Исследованию производительности алгоритмов, лежащих в основе кодеков H264 и H265 посвящено большое количество публикаций. В тоже время, количество материалов, посвящённых кодеку Google VP9 значительно меньше. В данной статье в качестве предмета исследования рассмотрим задачу организации параллельных вычислений применительно к декодированию видеосигнала, сжатого в соответствие со спецификацией Google VP9.

Объектом исследования данной статьи является производительность алгоритмов декодирования и восстановления сжатого видеосигнала.

Актуальность данной работы обусловлена тем, что кодирование и обработка сжатого видеосигнала являются одними из самых вычислительно сложных задач, исполняемых на клиентских и серверных компьютерах и мобильных устройствах. Дополнительно задача усложняется тем, что программные реализации декодеров должны не только обеспечивать высокую производительность, но также и низкие задержки для поддержания частоты

DOI: 10.7256/2305-6061.2016.2.18430

смены кадров.

Цель данной работы заключается в том, чтобы выявить наиболее вычислительно затратные стадии декодирования и восстановления видеосигнала, сжатого кодеком Google VP9, а также исследовать возможность оптимизации и параллельной реализации таких стадий – как на центральном процессоре, так и на видеокартах с поддержкой вычислений общего назначения.

Общий профиль производительности эталонного кодека

Перед тем, как переходить к детальному анализу стадий декодирования и возможности их оптимизации, рассмотрим общий профиль производительности эталонного кодека Google VP9 на нескольких видеофайлах. Эти данные дадут представление о фактическом вкладе каждой стадии декодирования в общее время работы реализации кодека. Так же, экспериментальные данные покажут приоритетные направления оптимизации.

Для снятия профиля производительности эталонного декодера Google VP9, кодек был профилирован при помощи утилиты AMD CodeXL. Данная утилита в режиме «Time Based Profile» отслеживает адрес указателя на текущую исполняемую инструкцию с заданной наперёд частотой. В проведённых экспериментах был выбран период сбора 10мс. Эта величина была подобрана опытным путём, т. к. позволяла сохранить повторяемость результатов и не сильно сказывалась на общем времени исполнения программы.

Был снят профиль производительности для нескольких видеофайлов, сжатых с различными настройками качества. В качестве тестовых последовательностей использовались видеопотоки «Sintel» и «Tears of steel». На рис. 1-2 показано распределение суммарного времени декодирования по стадиям для различных настроек визуального качества кодирования. Наибольшее визуальное качество соответствует наименьшему значению коэффициента квантования остатков предсказания сигнала QP.

Тестовый компьютер был оснащён процессором Intel Core I5 4200U (2 ядра, 2.6 ГГц, 3Мбайт кэша 3го уровня), 8 Гбайт оперативной памяти LP DDR3 1600 МГц в двухканальном режиме и работал под управлением 64х-битной операционной системы Linux Ubuntu 14.04.

Рисунок 1: Распределение времени работы декодера по стадиям для разрешения HD

DOI: 10.7256/2305-6061.2016.2.18430

Рисунок 2: Распределение времени работы декодера по стадиям для разрешения FullHD

Отметим, что наиболее затратными с токи зрения времени выполнения, являются стадии арифметического декодирования (CABAC) и межкадрового предсказания (Interpred). Поэтому в следующих разделах данной статьи сконцентрируемся на анализе данных стадий. Так же, необходимо отметить, что межкадровое предсказание занимает меньший процент времени по сравнению со стандартом кодирования H265 [6].

Анализ стадий декодирования

Арифметическое кодирование

Входом данной стадии является битовый поток, выходом – набор синтаксических элементов кодека, определённый спецификацией.

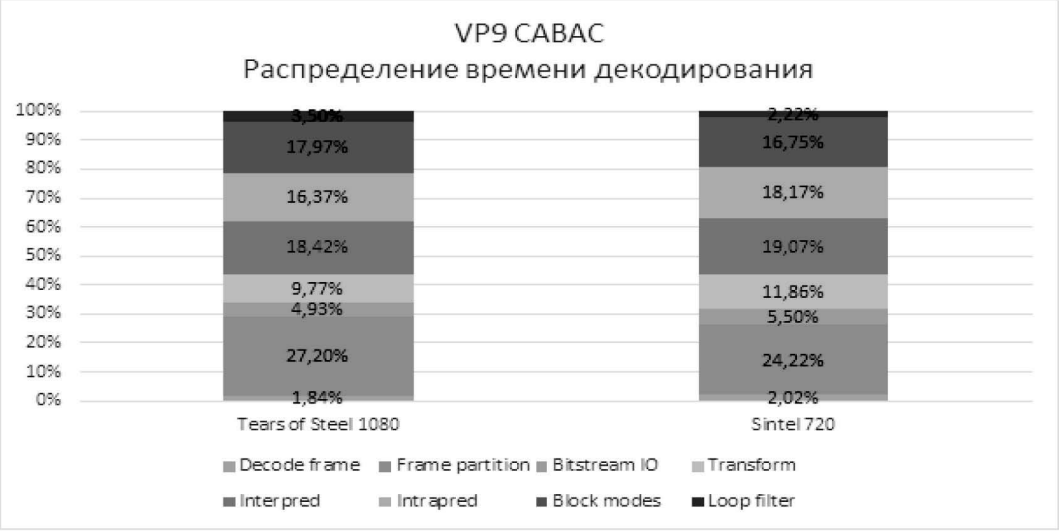

Рисунок 3: Распределение времени работы декодера на стадии декодирования сжатого битового потока

DOI: 10.7256/2305-6061.2016.2.18430

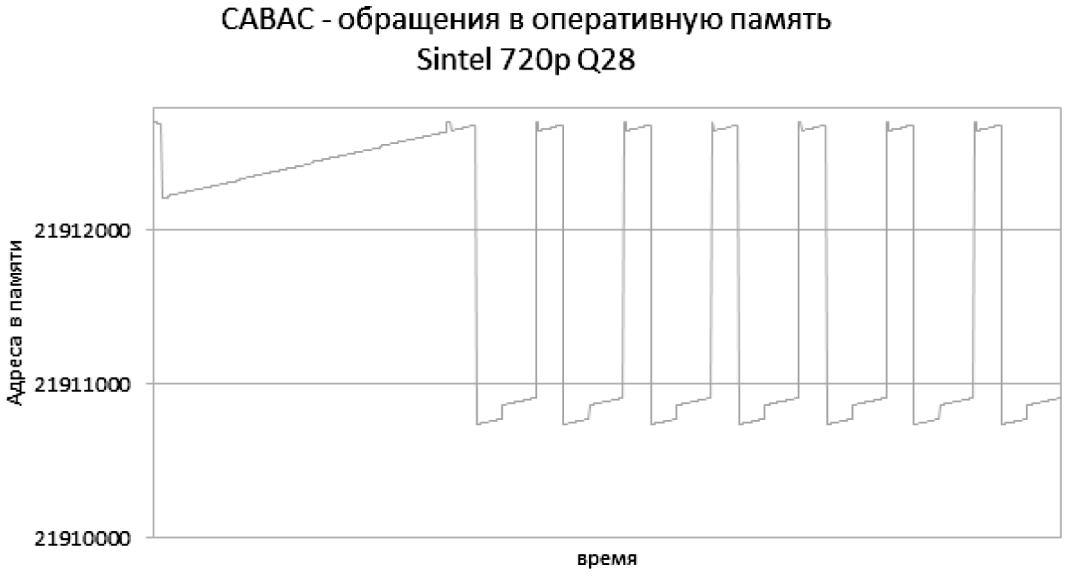

Рисунок 4: Статистика обращений в оперативную память для последовательности Sintel

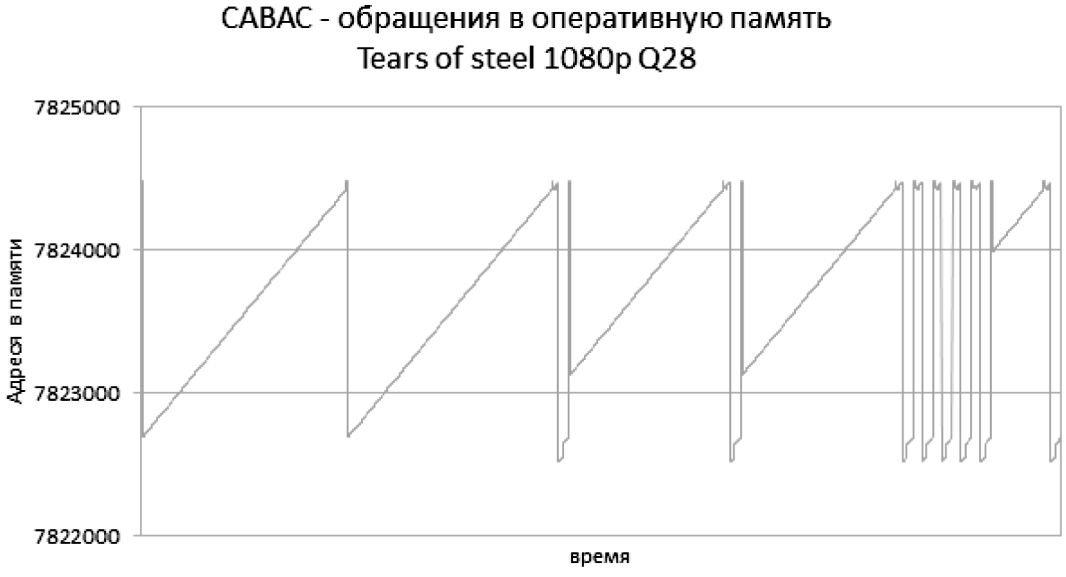

Рисунок 5: Статистика обращений в оперативную память для последовательности Tears of steel

С учётом того, что данная стадия является самой вычислительно сложной для тестовой машины, был проведён более подробный анализ производительности. Для этого была произведена инструментация эталонного кодека с сохранением количества вызовов функций и времени их исполнения на этапе исполнения. На рис. 3 представлены результаты замеров распределения времени декодирования для тестовых последовательно-

DOI: 10.7256/2305-6061.2016.2.18430

стей с коэффициентом квантования QP28. Данное значение коэффициента соответствует хорошему компромиссу между качеством кодирования и вычислительной сложностью.

Энтропийное кодирование, применяемое в кодеке VP9, представляет собой субэкспоненциальное кодирование (в отличие от экспоненциального кодирования Голобма для стандарта H264). В общем виде алгоритм можно представить в следующем виде [7]:

На первом шаге вычисляются значения переменных:

$$b = \begin{cases} k, \text{если } n < 2^k \\ [log_2 n], \text{если } n \geq 2^k \end{cases} u = \begin{cases} 0, \text{если } n < 2^k \\ b - k + 1 \text{если } n \geq 2^k \end{cases}$$

где k - параметрическое значение (для кодека Google VP9 k=4)

На втором шаге унарный код u (u+1 бит) дополняется младшими битами n. Длина кода равна:

$$u + 1 + b = \begin{cases} k + 1, \text{если } n < 2^k \\ 2[log_2 n] - k + 2 \text{если } n \geq 2^k \end{cases}$$

Таким образом, декодирование литерала сводится к декодированию составляющих его бит в цикле. Для оптимизации производительности данной стадии декодирования важно знать распределение вероятности для длин литералов. Т. к. литералы, занимающие в сжатом битовом потоке наибольшее количество бит (такие как коэффициенты обратного преобразования и вектора движения) кодируются сериями, то высока вероятность того, что в сжатом битовом потоке распределение длин литералов будет иметь вырожденный вид с большим количеством повторяющихся элементов с одинаковым значением. Для проверки этой гипотезы были собраны экспериментальные данные о распределении длин литералов в последовательности Tears of steel Q18, поскольку для данной последовательности арифметическое декодирование занимает больше всего времени.

Таблица 1: распределение вероятности длин литералов для последовательности Tears of Steel Q18

Длина литерала, бит	1	2	3	4	5	6
Вероятность	3,42%	0,00%	65,55%	19,50%	0,00%	11,53%

Выдвинутая гипотеза получила подтверждение – наиболее вероятными являются литералы длиной 3, 4 и 6 бит. Ещё одним важным фактом является то, что максимальная возможная длина литерала для данной последовательности составляет всего 6 бит. Этот факт важен для программной оптимизации функции субэкспоненциального декодирования литерала. На основе полученных данных были применены следующие подходы для оптимизации:

1. Сохранение результатов вычисления длины литерала для декодирования серий литералов одинаковой длины.
2. Размотка цикла субэкспоненциального декодирования литерала.
3. Более эффективный алгоритм расчёт числа бит в литерале.
4. Более эффективное использование регистров процессора непосредственно внутри функции арифметического декодирования.

DOI: 10.7256/2305-6061.2016.2.18430

Реализация пунктов 1 и 4 достаточно очевидна, поэтому рассмотрим более подробно пункты 2 и 3. Внутри функции субэкспоненциального декодирования происходит составление декодированного литерала по битам, декодированным из сжатого битового потока. В данном случае узким местом является цикл с переменным количество итераций. Его можно заменить на набор switch-case без конструкции break в конце. Данная техника известна под названием «устройство Даффа» (Duff's Device). Она позволяет заменить выполнение нескольких итераций цикла на последовательное выполнение инструкций без необходимости условных переходов. Так же, битовый сдвиг происходит на константную величину, которую не нужно считывать из регистра-счетчика цикла.

```
static int vp9_read_literal(vp9_reader *br, int bits) {
  int z = 0, bit;

  for (bit = bits - 1; bit >= 0; bit--)
    z |= vp9_read_bit(br) << bit;

  return z;
}
```

```
static int vp9_read_literal(vp9_reader *br, int bits) {
  register int z = 0;

  switch(bits-1){
  case 6:
       z |= vp9_read(br, 128) << 6;
  case 5:
       z |= vp9_read(br, 128) << 5;
  case 4:
       z |= vp9_read(br, 128) << 4;
  case 3:
       z |= vp9_read(br, 128) << 3;
  case 2:
       z |= vp9_read(br, 128) << 2;
  case 1:
       z |= vp9_read(br, 128) << 1;
  case 0:
       z |= vp9_read(br, 128);
       break;
  }
  return z;
}
```

Ещё одним узким местом является расчёт числа бит литерала в цикле while. Данное решение плохо тем, что число итераций цикла заранее непредсказуемо. Вместо него был использован быстрый алгоритм подсчёта числа бит, который выполняет расчёт за фиксированное число шагов без использования условных переходов [8].

DOI: 10.7256/2305-6061.2016.2.18430

```
unsigned int v;              // 32-bit value to find the log2 of
register unsigned int r;     // result of log2(v) will go here
register unsigned int shift;

r = (v > 0xFFFF) << 4; v >>= r;
shift = (v > 0xFF) << 3; v >>= shift; r |= shift;
shift = (v > 0xF) << 2; v >>= shift; r |= shift;
shift = (v > 0x3) << 1; v >>= shift; r |= shift;
r |= (v >> 1);
```

Для измерения времени выполнения были произведено большое количество 7 запусков эталонного и модифицированного кодеков и сравнено среднее время исполнения. Модифицированный кодек показал производительность, на 4.67% большую по сравнению с эталонным. Таким образом, прирост общей производительности (снижение времени исполнения) составил 4.67% для декодирования в целом и 6.86% для арифметического декодирования в отдельности.

Одной из важных особенностей реализации арифметического кодека в кодеке Google VP9 является то, что таблицы распределения вероятностей символов могут быть обновлены только на уровне отдельных кадров, а не адаптивно, как это реализовано в стандартах ITU-T H264 и H265 (контекстно-адаптивное арифметическое кодирование CABAC). Сами авторы VP9 обуславливают данный выбор тем, что фиксация таблиц распределения вероятностей на уровне кадров повышает устойчивость кодирования к ошибкам передачи данных и способствует облегчению исправления ошибок (error resilience) при достаточной гибкости и эффективности кодирования [8, 9].

На рис. 4, 5 показана статистика записи в оперативную память при выполнении арифметического декодирования первых 8 кадров последовательностей. Такая техника применяется для определения характеристик локальности алгоритма[10]. Полученные результаты показывают, что шаблон обращения к памяти для данной стадии декодирования дружелюбен к кэшу – обращения происходят по повторным адресам, отражающим структуру группы кадров (GOP). Разброс значений адресов в обоих случаях мал, и составляет около 2 Кбайт вне зависимости от разрешения. Такая ситуация характерна для поочередного порядка обхода блоков в кадре.

Заметим, что спецификация Google VP9 не позволяет проводить параллельную реконструкцию векторов движения для соседних блоков, т. к. присутствует зависимость по данным. Возможности организации декодирования «волновым фронтом» (Wave front Parallel Processing[11]) так же нет. Всё это приводит к тому, что стадия арифметического декодирования является «бутылочным горлышком» производительности.

Таким образом, по результатам анализа данной стадии можно сделать следующие выводы:

- Алгоритм имеет линейную сложность - $O(N)$.
- Алгоритм оперирует целыми беззнаковыми числами на входе. Выходом алгоритма является набор данных в тех форматах, которые имеют синтаксические элемен-

DOI: 10.7256/2305-6061.2016.2.18430

ты, описанные в спецификации кодека. Над входными данными совершаются простые операции целочисленной арифметики – сложение, вычитание, умножение, деление, битовый сдвиг.

- Процесс арифметического декодирования по своей сути последователен и не может быть векторизован.
- Количество входных данных алгоритма равно размеру сжатого битового потока. Из-за применения различного типа ссылочных кадров и реорганизации порядка следования кадров в битовом потоке по сравнению с порядком воспроизведения, количество входных данных может значительно меняться. По этой же причине, количество выходных данных так же непостоянно.
- Шаблон обращения к памяти на входе алгоритма хорошо подходит для кэширования, т. к. алгоритм принимает малое количество данных, исчисляемое килобайтами для видеосигналов малого разрешения / качества и сотнями килобайт для видеосигналов высокого разрешения и качества. Таким образом, считанные данные легко помещаются в кэши данных 2-го и 3-го уровня современных процессоров, которые, как правило имеют объём около 64 - 256 КБ и 1 - 8 МБ соответственно. Шаблон обращения к памяти на выходе может сильно варьироваться в зависимости от организации структур данных декодера, но, как правило, он достаточно однороден для декодеров, которые осуществляют поблочное декодирование кадра.
- Спецификация кодека Google VP9 не предусматривает возможности выполнения параллельного декодирования блоков в отличие от стандарта ITU-T H265, в котором такая возможность присутствует [4].

Межкадровое предсказание

Межкадровое предсказание, или компенсация движения, является стадией декодирования, которая выполняется вслед за внутрикадровым предсказанием. В кодеке реализована двухсторонняя компенсация движения с точностью до $^{1}/_{8}$ пиксела. Для обхода патентов на двухстороннюю компенсацию движения [12-14], двухсторонняя компенсация движения применяется только для тех кадров, которые не будут показаны (помечены как не предназначенные для показа), но могут быть использованы в дальнейшем для осуществления предсказания. Фактически, для этого достаточно закодировать кадр, который будет ссылаться на помеченный кадр, использующий двухстороннюю компенсацию движения с минимальными поправками. Ценой обхода патентов в данном случае является необходимость кодирования фиктивных кадров, и дополнительное копирование для осуществления предсказания. Расчёты, приведённые далее были проведены для яркостного канала изображений.

В процессе декодирования, поддерживается буфер из 8 ссылочных кадров. Один кадр может использовать для предсказания максимум 3 кадра из буфера. Особенностью кодека Google VP9 является высокая точность предсказания – для осуществления интерполяции используется свёрточная фильтрация фильтром размером 8x8. Блоки межкадрового предсказания могут иметь различный размер – от 4x4 до 64x64. Приведём иллюстрацию того,

DOI: 10.7256/2305-6061.2016.2.18430

какие пикселы понадобятся для осуществления фильтрации яркостного канала (рис. 6)

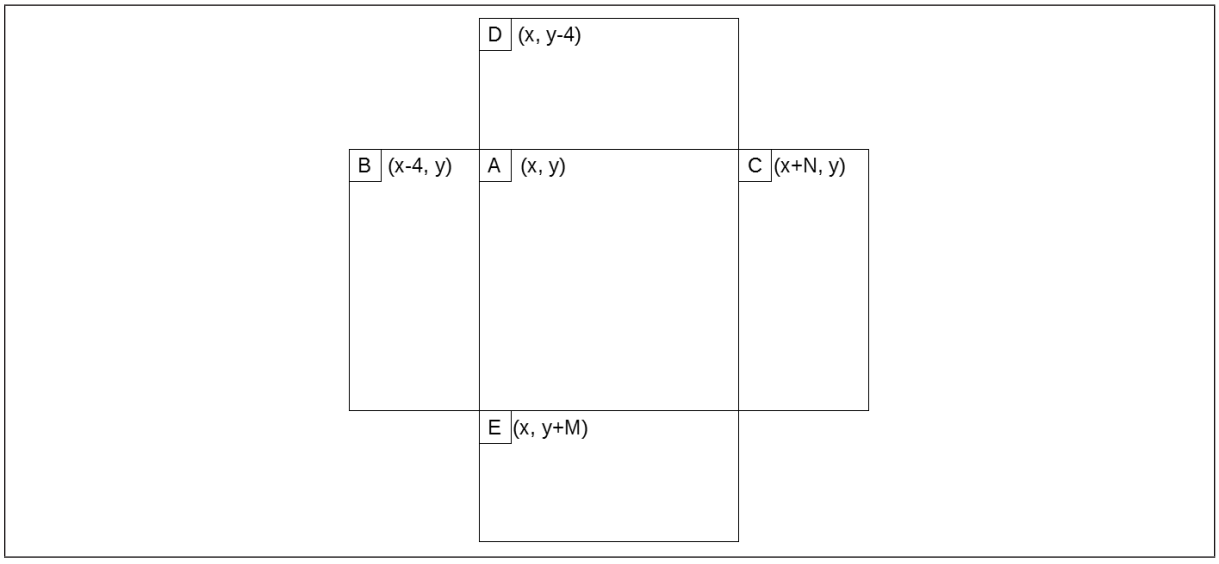

Рисунок 6: блок межкадрового предсказания и его окрестность

Количество пикселов, необходимое для фильтрации блока размером N x M легко рассчитать по формуле

$$Q = (N + 8) \times (M + 8) - 64 \quad (1)$$

Такое межкадровое предсказание благотворно сказывается на качестве декодированного видеосигнала, однако, предъявляет высокие требования к пропускной способности системы памяти. Приведём таблицу с оценкой количества пикселов для блоков различного размера:

Таблица 2: количество пикселов в блоке и его окрестности

	4	8	16	32	64
4	80				
8	128	192			
16		320	512		
32			896	1536	
64				2816	5120

Так как количество пикселов в самом блоке равно квадрату его стороны, сразу приведём таблицу с оценкой того, во сколько раз нам нужно больше пикселов для фильтрации по сравнению с самим блоком межкадрового предсказания для яркостного канала:

Таблица 3: Соотношение чтение/запись для блоков межкадрового предсказания различных размеров.

	4	8	16	32	64
4	5				
8	4	3			
16		2,5	2		
32			1,75	1,5	
64				1,375	1,25

DOI: 10.7256/2305-6061.2016.2.18430

Для получение примерной оценки того, каким количеством данных необходимо оперировать для осуществления компенсации движения, приведём статистику распределения блоков предсказания по их размерам и типам. Полученные данные представлены на рис. 7-10 и иллюстрируют стратегию разбиения ссылочных кадров на блоки на примере рассмотренных видеофайлов.

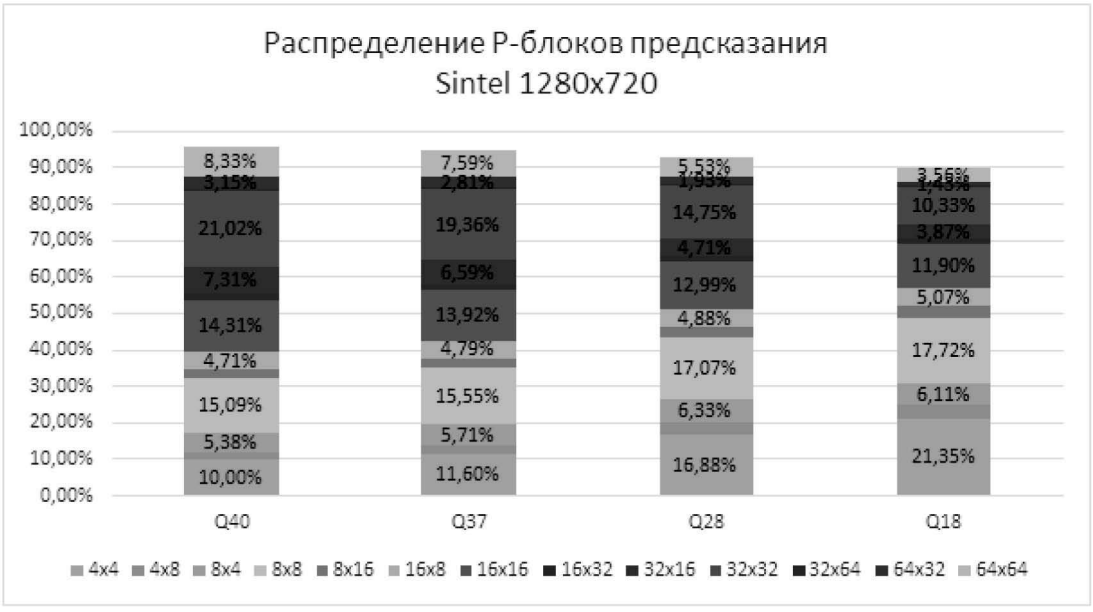

Рисунок 7: Распределение блоков одностороннего предсказания по размерам – последовательность Sintel

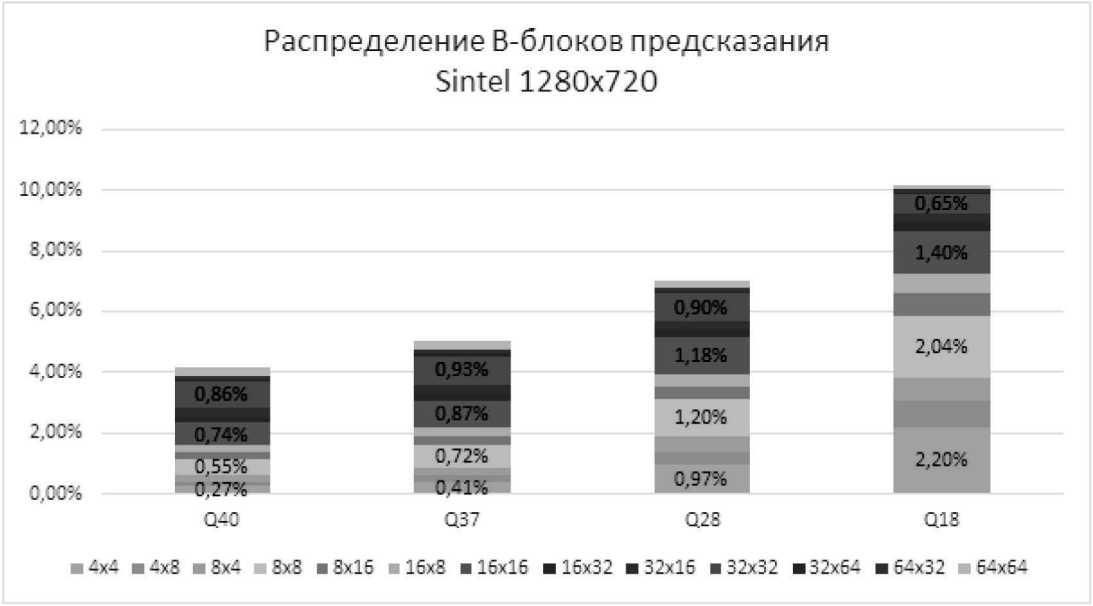

Рисунок 8: Распределение блоков двустороннего предсказания по размерам – последовательность Sintel

DOI: 10.7256/2305-6061.2016.2.18430

Рисунок 9: Распределение блоков одностороннего предсказания по размерам – последовательность Tears of steel

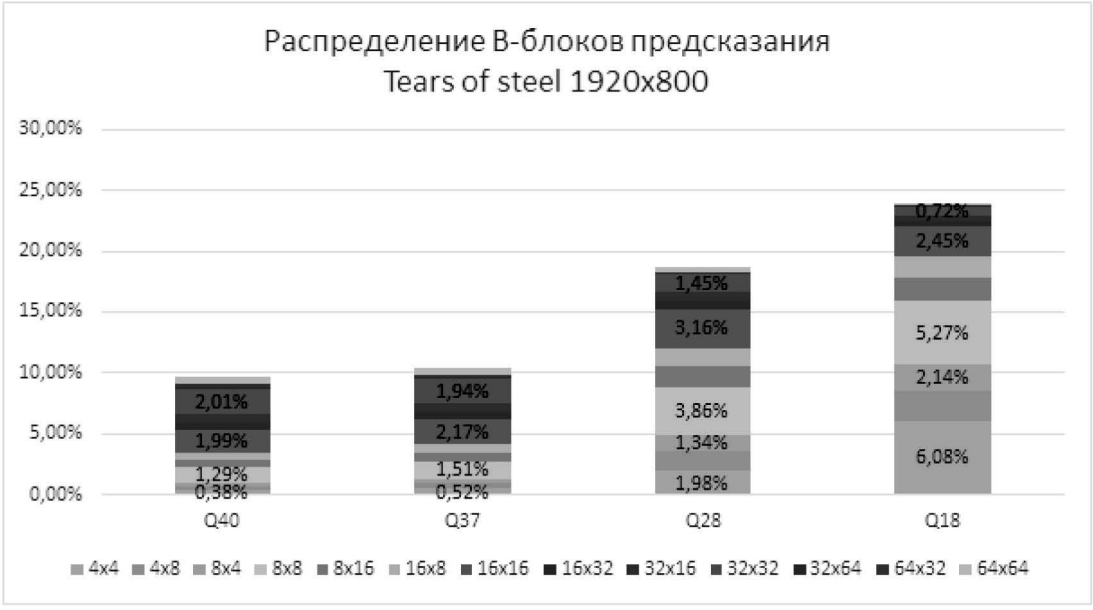

Рисунок 10: Распределение блоков двустороннего предсказания по размерам – последовательность Tears of steel

Таблица 4: Соотношение чтение/запись для межкадрового предсказания в последовательности Sintel:

Коэффициент QP	P блоки	B блоки	Итого
40	234%	19%	253%
37	241%	25%	265%
28	262%	39%	301%
18	275%	64%	339%

DOI: 10.7256/2305-6061.2016.2.18430

Таблица 5: Соотношение чтение/запись для межкадрового предсказания в последовательности Tears of steel:

Коэффициент QP	P блоки	B блоки	Итого
Q40	233%	43%	277%
Q37	241%	48%	289%
Q28	249%	106%	355%
Q18	263%	163%	426%

На основании полученных данных можно сделать вывод о том, что межкадровое предсказание в Google VP9 организовано проще, чем в стандарте H265. Так, кодек использует малое количество блоков двустороннего предсказания. Необходимо отметить, что значительная часть обращений к памяти производится по повторным адресам – таким образом, эти обращения могут быть кэшированы. Для иллюстрации этого утверждения приведём рисунки 11, 12, на которых представлена статистика чтений из оперативной памяти по адресам при выполнении компенсации движения первых 8 кадров последовательностей. Можно отметить то, что доступ происходит по схожим адресам с периодичностью, совпадающей со структурой GOP. Для видеопоследовательности Sintel результаты экспериментов показывают высокую регулярность чтения. Запись происходит по фиксированным заранее известным адресам, поэтому статистика по ним не собиралась.

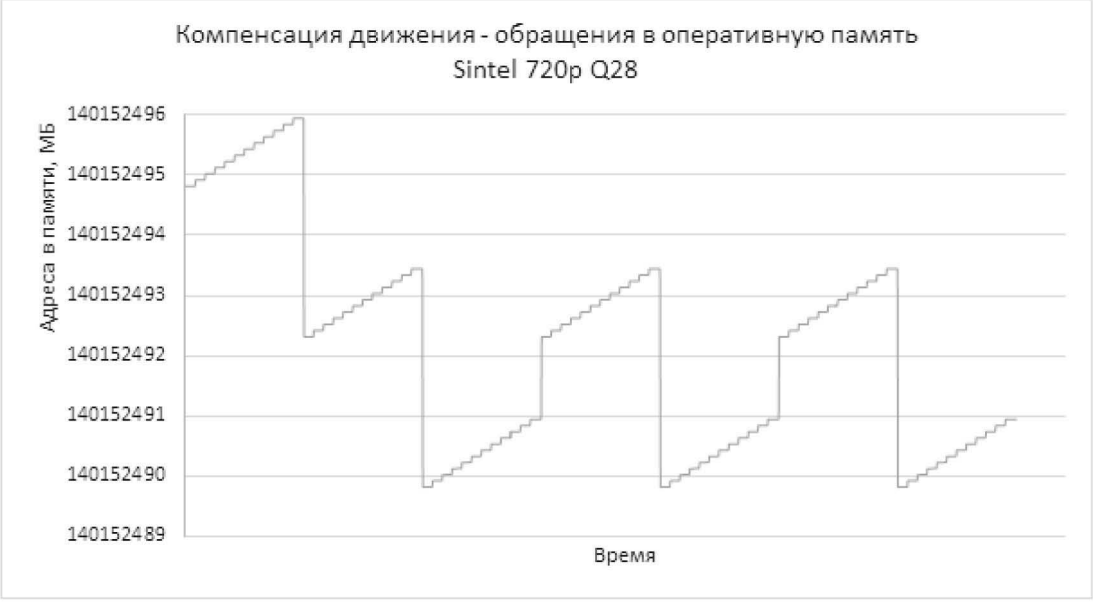

Рисунок 11: Статистика обращений в оперативную память при компенсации движения для последовательности Sintel

Результаты экспериментов для последовательности Tears of steel показывают значительно больший разброс адресов, по которым происходят обращения. Это обусловлено тем, что при большем количестве пикселов в кадре, дискретизация при выборе предиктора меньше на этапе оценки движения. Разброс в обращениях к памяти при проведении компенсации движения для одного кадра в разрешении 1280x710 точек примерно равен

DOI: 10.7256/2305-6061.2016.2.18430

7 Мбайт, в разрешении 1920x1080 точек – около 14 Мбайт.

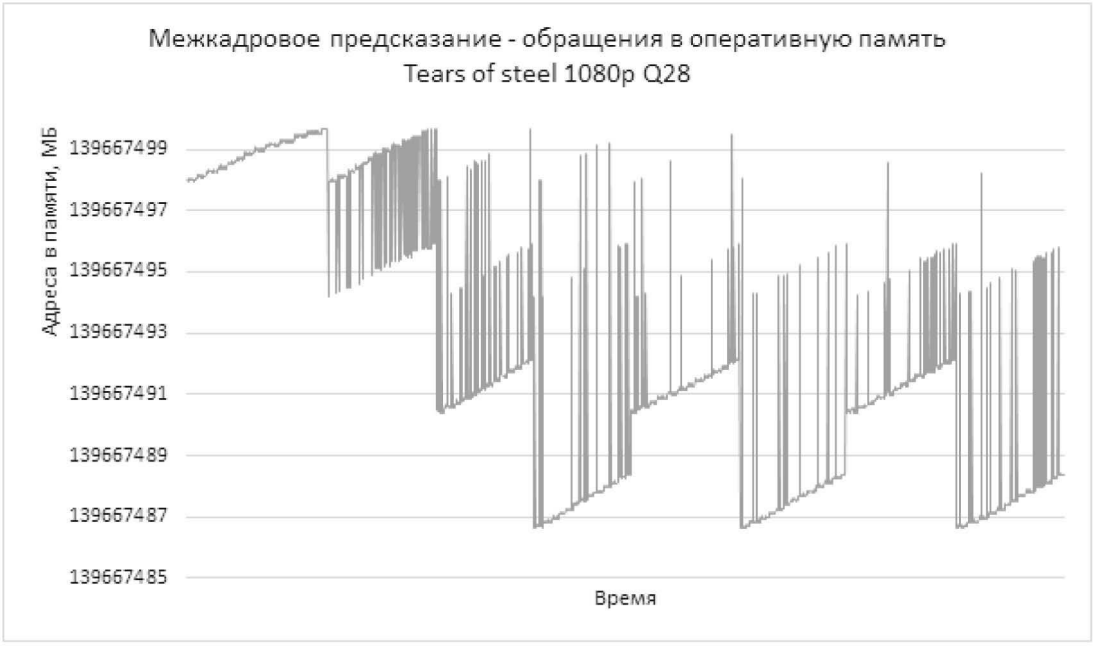

Рисунок 12: Статистика обращений в оперативную память при компенсации движения для последовательности Tears of steel

Полученные сведения дают хорошее объяснение тому, почему межкадровое предсказание является одной из самых сложных стадий декодирования видео. Дадим характеристику данной стадии декодирования в соответствии с критериями анализа:

- Алгоритм имеет линейную сложность – для каждого пиксела блока межкадрового предсказания выполняется постоянное количество операций, не зависящее от размера блока.
- Алгоритм оперирует целыми беззнаковыми числами на входе и выходе. В ходе осуществления свёрточного преобразования, промежуточные результаты могут выходить за пределы разрядной сетки входного типа данных. Такая ситуация является типичной, для её решения можно применять хранение результата в типе данных с большей разрядностью и/или использовать регистры-аккумуляторы.
- Реализация алгоритма может использовать параллелизм на уровне инструкций для осуществления свёрточного преобразования. Так, например, окрестность точки может быть загружена в векторный регистр.
- Количество данных на входе алгоритма рассчитывается по формуле (1), количество данных на выходе равно размеру блока межкадрового предсказания.
- Шаблон обращения к памяти на уровне блока предсказания хорошо подходит для кэширования, т. к. применяется свёрточное преобразование, которое повторно использует значения соседних пикселов при фильтрации скользящим окном. Шаблон обращения на уровне кадра слабо предсказуем, и, как правило, проис-

DOI: 10.7256/2305-6061.2016.2.18430

ходит по невыровненным адресам, т. к. значения векторов слабо предсказуемы заранее. Кроме того, один блок предсказания может использовать несколько участков из разных кадров, а соседние блоки предсказания могут использовать предикторы, удалённые друг от друга. В силу этого, данная стадия предсказания предъявляет высокие требования к подсистеме памяти компьютера.

- Процесс компенсации движения может быть легко распараллелен, т. к. все блоки обрабатываются независимо друг от друга.

Выводы

В данной статье был проведён общий анализ затрат вычислительного времени центрального процессора применительно к задаче декодирования видеосигнала, сжатого кодеком Google VP9 и проведён подробный анализ наиболее вычислительно затратных стадий декодирования и восстановления сжатого видеосигнала – контекстно-адаптивного бинарного арифметического кодирования и межкадрового предсказания.

Была проведена оценка алгоритмической и вычислительной сложности указанных стадий декодирования, требования, предъявляемые данными стадиями к подсистеме памяти. Так же, была оценена возможность распараллеливания данных стадий декодирования на центральном процессоре и видеокартах с поддержкой вычислений общего назначения.

По результатам проделанной работы были сделаны выводы о невозможности параллельной реализации контекстно-адаптивного арифметического кодирования, а также о том, что стадия межкадрового предсказания может быть реализована параллельно, в том числе на видеокартах с поддержкой вычислений общего назначения. Была проведена работа по оптимизации с учётом особенностей распределения вероятностей длин литералов, в результате которой удалось сократить время декодирования на 6.8% в наиболее благоприятном случае за счёт использования статистических особенностей внутри алгоритма.

Был сделан вывод о меньшей фактической сложности стадии межкадрового предсказания по сравнению с конкурирующим кодеком H265 несмотря на схожесть реализации данной стадии в стандарте кодека H265 и спецификации кодека Google VP9 и большую точность суб-пиксельной интерполяции.

Библиография :

1. The H.264/MPEG4 Advanced Video Coding Standard and its Applications // http://iphome.hhi.de URL: http://iphome.hhi.de/wiegand/assets/pdfs/h264-AVC-Standard.pdf (дата обращения: 09.12.2015).
2. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC) // hlevkin.com URL: http://www.hlevkin.com/Standards/h264.pdf (дата обращения: 09.12.2015).

DOI: 10.7256/2305-6061.2016.2.18430

3. Overview of the High Efficiency Video Coding (HEVC) Standard // http://iphome.hhi.de URL: http://iphome.hhi.de/wiegand/assets/pdfs/2012_12_IEEE-HEVC-Overview.pdf (дата обращения: 09.12.2015).
4. High efficiency video coding // sist.sysu.edu.cn URL: http://sist.sysu.edu.cn/~isscwli/ref/h265.pdf (дата обращения: 09.12.2015).
5. Web M and the New VP9 Open Video Codec // http://commondatastorage.googleapis.com URL: http://commondatastorage.googleapis.com/io-2013/presentations/258%20-%20VP9%20Preso%20for%20IO%20%28FINAL%29.pdf (дата обращения: 09.12.2015).
6. Grois, D.; Marpe, D.; Mulayoff, A.; Itzhaky, B.; Hadar, O. Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders // Picture Coding Symposium. 2013.-С. 394-397.
7. David Solomon, Giovanni Motta. Handbook of data compression // Springer/-2010. – С. 170.
8. Bit Twiddling Hacks // Stanford.edu URL: // https://graphics.stanford.edu/~seander/bithacks.html (дата обращения 20.03.2016).
9. A VP9 Bitstream Overview // ietf.org URL: http://tools.ietf.org/html/draft-grange-vp9-bitstream-00 (дата обращения: 13.12.2015).
10. TECHNICAL OVERVIEW O F VP8, AN OPEN SOURCE VIDEO CODEC F OR THE WEB // research.google.com URL: http://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/37073.pdf (дата обращения: 15.12.2015).
11. Воеводин В.В., Воеводин Вл. В. Спасительная локальность суперкомпьютеров // Открытые системы. -2013. – №9. -С. 12-15.
12. Asaduzzaman, A.; Suryanarayana, V.R.; Rahman, M. Performance-power analysis of H.265/HEVC and H.264/AVC running on multicore cache systems // Intelligent Signal Processing and Communications Systems. 2013. -С. 174-179.
13. Пат. 7982641 B1 США, US 12/613,830. Context-based adaptive binary arithmetic coding engine/ [Способ контекстно-адаптивного арифметического кодирования] Guan-Ming Su, Leung Chung Lai, Wenchi Hsu, Qian Tang, Li Sha, Ching-Han Tsai; заявитель и патентообладатель Marvell International Ltd (США).
14. Пат. 7808406 B2 США, US7808406 B2. Method and apparatus for realizing arithmetic coding/decoding / [Способ и устройство для осуществления арифметического кодирования] Yun He, Wei Yu, Ping Yang, Xinjian MENG; заявитель и патентообладатель Huawei Technologies Co., Ltd., Tsinghua University.
15. Пат. 13/801,350 США, US20130243093 A1. Motion vector coding and bi-prediction in hevc and its extensions / [Кодирование векторов движения и двухсторонняя компенсация движения для HEVC и его расширений] Ying Chen, Ye-Kui Wang, Li Zhang; заявитель и патентообладатель Qualcomm Incorporated.

References:

1. The H.264/MPEG4 Advanced Video Coding Standard and its Applications // http://iphome.hhi.de URL: http://iphome.hhi.de/wiegand/assets/pdfs/h264-AVC-Standard.pdf (data obrashcheniya: 09.12.2015).
2. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC) // hlevkin.com URL: http://www.hlevkin.com/Standards/h264.pdf (data obrashcheniya: 09.12.2015).

DOI: 10.7256/2305-6061.2016.2.18430

3. Overview of the High Efficiency Video Coding (HEVC) Standard // http://iphome.hhi.de URL: http://iphome.hhi.de/wiegand/assets/pdfs/2012_12_IEEE-HEVC-Overview.pdf (data obrashcheniya: 09.12.2015).
4. High efficiency video coding // sist.sysu.edu.cn URL: http://sist.sysu.edu.cn/~isscwli/ref/h265.pdf (data obrashcheniya: 09.12.2015).
5. Web M and the New VP9 Open Video Codec // http://commondatastorage.googleapis.com URL: http://commondatastorage.googleapis.com/io-2013/presentations/258%20-%20VP9%20Preso%20for%20IO%20%28FINAL%29.pdf (data obrashcheniya: 09.12.2015).
6. Grois, D.; Marpe, D.; Mulayoff, A.; Itzhaky, B.; Hadar, O. Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders // Picture Coding Symposium. 2013.-S. 394-397.
7. David Solomon, Giovanni Motta. Handbook of data compression // Springer/-2010. – S. 170.
8. Bit Twiddling Hacks // Stanford.edu URL: // https://graphics.stanford.edu/~seander/bithacks.html (data obrashcheniya 20.03.2016).
9. A VP9 Bitstream Overview // ietf.org URL: http://tools.ietf.org/html/draft-grange-vp9-bitstream-00 (data obrashcheniya: 13.12.2015).
10. TECHNICAL OVERVIEW O F VP8, AN OPEN SOURCE VIDEO CODEC F OR THE WEB // research.google.com URL: http://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/37073.pdf (data obrashcheniya: 15.12.2015).
11. Voevodin V.V., Voevodin Vl. V. Spasitel'naya lokal'nost' superkomp'yuterov // Otkrytye sistemy.-2013. – №9.-S. 12-15.
12. Asaduzzaman, A.; Suryanarayana, V.R.; Rahman, M. Performance-power analysis of H.265/HEVC and H.264/AVC running on multicore cache systems // Intelligent Signal Processing and Communications Systems. 2013.-S. 174-179.
13. Pat. 7982641 B1 SShA, US 12/613,830. Context-based adaptive binary arithmetic coding engine/ [Sposob kontekstno-adaptivnogo arifmeticheskogo kodirovaniya] Guan-Ming Su, Leung Chung Lai, Wenchi Hsu, Qian Tang, Li Sha, Ching-Han Tsai; zayavitel' i patentoobladatel' Marvell International Ltd (SShA).
14. Pat. 7808406 B2 SShA, US7808406 B2. Method and apparatus for realizing arithmetic coding/decoding / [Sposob i ustroistvo dlya osushchestvleniya arifmeticheskogo kodirovaniya] Yun He, Wei Yu, Ping Yang, Xinjian MENG; zayavitel' i patentoobladatel' Huawei Technologies Co., Ltd., Tsinghua University.
15. Pat. 13/801,350 SShA, US20130243093 A1. Motion vector coding and bi-prediction in hevc and its extensions / [Kodirovanie vektorov dvizheniya i dvukhstoronnyaya kompensatsiya dvizheniya dlya HEVC i ego rasshirenii] Ying Chen, Ye-Kui Wang, Li Zhang; zayavitel' i patentoobladatel' Qualcomm Incorporated.

DOI: 10.7256/2305-6061.2016.2.18430