УДК 004. 658 DOI: 10.18698/2541-8009-2018-8-360

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ АППАРАТНОЙ И ПРОГРАММНОЙ РЕАЛИЗАЦИЙ АЛГОРИТМА СЖАТИЯ ВИДЕО

В.А. Кобецкий[1] rk62ck@mail.ru
SPIN-код: 3992-1846

Н.Д. Казаков[2] nkazakovd@gmail.com
SPIN-код: 1971-5303

[1] **МГТУ им. Н.Э. Баумана, Москва, Российская Федерация**
[2] **Московский авиационный институт, Москва, Российская Федерация**

Аннотация

Проанализирована эффективность работы различных реализаций алгоритма кодирования видео. Рассмотрен общий алгоритм работы видеокодека. Разработана структурная схема видеокодека и описание ее составных частей. Проведено тестирование программной реализации видеокодека на процессорной системе. Также проведено тестирование аппаратной реализации видеокодека на программируемой логической интегральной схеме (ПЛИС). В ходе исследований для обоих вариантов реализации видеокодека определена скорость сжатия данных видеопотока. Результатом статьи является сравнение эффективности сжатия видеокодека на ПЛИС и на процессорной системе. Сделаны выводы о сферах применения различных реализаций видеокодека и актуальности их развития в будущем.

Ключевые слова

Кодирование видео, ПЛИС, процессорная система, эффективность сжатия, схема видеокодека, реализация видеокодека, алгоритм кодирования, сжатие видеопотока

Поступила в редакцию 24.05.2018

Введение. Цифровая техника развивается очень быстро. В современных условиях устройства воспроизведения и записи видео поддерживают все более и более широкоформатные изображения, а следовательно, необходимо как-то хранить возрастающие объемы информации [1]. Так, при условии, что пиксели изображения кодируются в цветовой модели RGB (англ. *red green blue*), полное описание каждого из них потребует три байта информации в памяти, по одному на каждый компонент цветности [2]. Для записи видео в формате 1920×1080 пикселей (full HD) со скоростью воспроизведения 60 кадров в секунду необходимо выделять около 355 Мб памяти на секунду. Этим практически невозможно пользоваться, оптический диск на 50 Гб мог бы вмещать всего около 2 мин такого видео.

Встает вопрос о возможности сжатия видео, что облегчит его последующее использование и хранение. На сегодняшний день самым распространенным стандартом кодирования видео является стандарт сжатия H.264 [3, 4]. Существует большое количество вариантов его реализации. Однако в данной статье рассмотрим только два принципиально различающиеся варианта.

За время существования кодеков изменялись и улучшались алгоритмы сжатия, но принцип сокращения информации остался прежним. Самое главное, что нужно сделать, — это убрать как можно больше повторяющихся данных.

Несжатое *видео* [5] принято считать последовательность кадров, представляющих собой двумерный массив, элементами которого являются пиксели изображения. *Формат видео* определяют количеством пикселей, расположенных по высоте и ширине каждого кадра. Поскольку каждый пиксель состоит из трех цветовых компонент, значение каждой из них принято называть *коэффициентом*. Выбор коэффициентов зависит от цветовой модели. *Видеокодеком* называют алгоритм сжатия и восстановления сжатых данных видеопотока.

Поскольку кадры видеопотока связанны между собой и зачастую состоят из одних тех же предметов и действующих лиц, нам требуется передать только разницу кадров, а не все изображение. Также имеет смысл определять, как изменилось местоположение предметов на соседних кадрах, и передавать данные о том, куда они движутся, в виде векторов передвижений. Если мы готовы мириться с потерями качества [6], то с помощью квантования и обнуления малозначащих данных можно искусственно создавать большее количество одинаковых данных.

Анализ алгоритма работы видеокодека. Видеокодек состоит из части, которая кодирует и передает видеоданные (энкодер), и части, которая декодирует их обратно (декодер). Рассмотрим подробнее алгоритм работы составных частей реализованного кодека.

Структурная схема энкодера представлена на рис. 1. На вход кодека поступает поток видео, которое необходимо сжать. Изначально он записывается в память — хранилище незакодированных кадров.

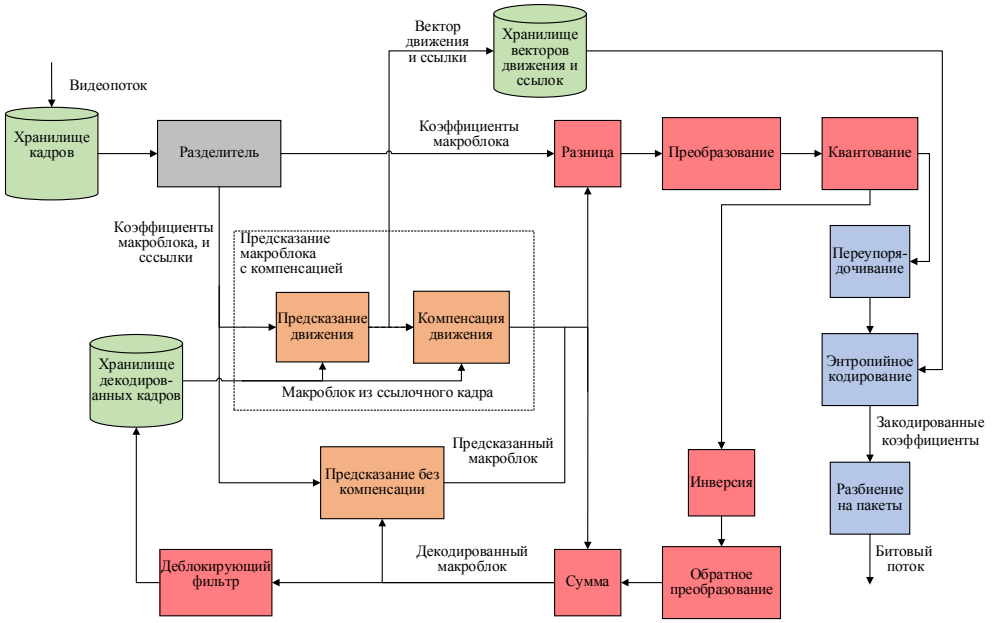


Рис. 1. Структурная схема энкодера

С помощью разделителя каждый кадр считывается по коэффициентам в макроблок слева направо, сверху вниз. Размер макроблока выбирают в зависимости от степени сжатия и точности кодирования, он может колебаться от 4×4 коэффициентов до 16×16. Также здесь каждый новый кадр имеет возможность получить одну или две ссылки на другие кадры.

Далее энкодер видео можно подразделить на три группы блоков, которые предсказывают текущий кадр, выполняют математические преобразования и непосредственно кодируют данные.

1. Часть предсказаний вычисляет коэффициенты предсказанного микроблока по ссылке или без нее на основе ранее декодированных кадров и, если возможно, вычисляет наиболее похожий макроблок по кадру из ссылки, а после определяет вектор передвижения текущего макроблока [2]. Отметим, что для точного предсказания необходимо использовать кадры не входного потока, а уже испорченные преобразованиями. Данный подход позволяет избавиться от накопления ошибки при декодировании.

2. Математическая часть энкодера вычисляет разницу между предсказанным макроблоком и исходным. Далее преобразует и квантует полученные коэффициенты разности, чтобы передать новые значения одновременно для энтропийного кодирования и обратного преобразования.

Преобразование выполняют для перемещения наиболее значащих коэффициентов в левую верхнюю часть макроблока. Квантование используют с определенным шагом. И в зависимости от него значения коэффициентов округляют до ближайшего целого, которое кратно значению выбранного шага.

Обратное преобразование и деквантование (инверсия), делают обратные операции. И отправляют обратно декодированные коэффициенты разности в блок суммирования. Результатом работы является сложение декодированных коэффициентов с ранее предсказанными для этого макроблока. Декодированный таким образом макроблок фильтруют (сглаживают) и записывают в хранилище для будущих предсказаний.

3. В части, связанной с кодированием, коэффициенты квантованного макроблока разности переупорядочивают так, что в начале оказываются наиболее значащие из них. И теперь данная последовательность коэффициентов и ранее найденных векторов передвижений энтропийно кодируется в последовательность битов, выбор которой осуществляется с помощью кодов Голомба, метода CABAC или CAVLC, с использованием стандартных таблиц [3].

В итоге работы энкодера последовательность битов разбивается по пакетам, к которым добавляются заголовки на основе данных кадровой синхронизации. Полученные пакеты передаются по линии передачи данных или записываются для последующего хранения.

Структурная схема декодера представлена на рис. 2. Аналогично энкодеру, он содержит три группы блоков, и их задача — выполнить в нужном порядке операции, обратные кодированию [2, 3].

Таким образом мы расшифровываем входную последовательность битов в коэффициенты макроблока и далее складываем из них декодированный видеопоток.

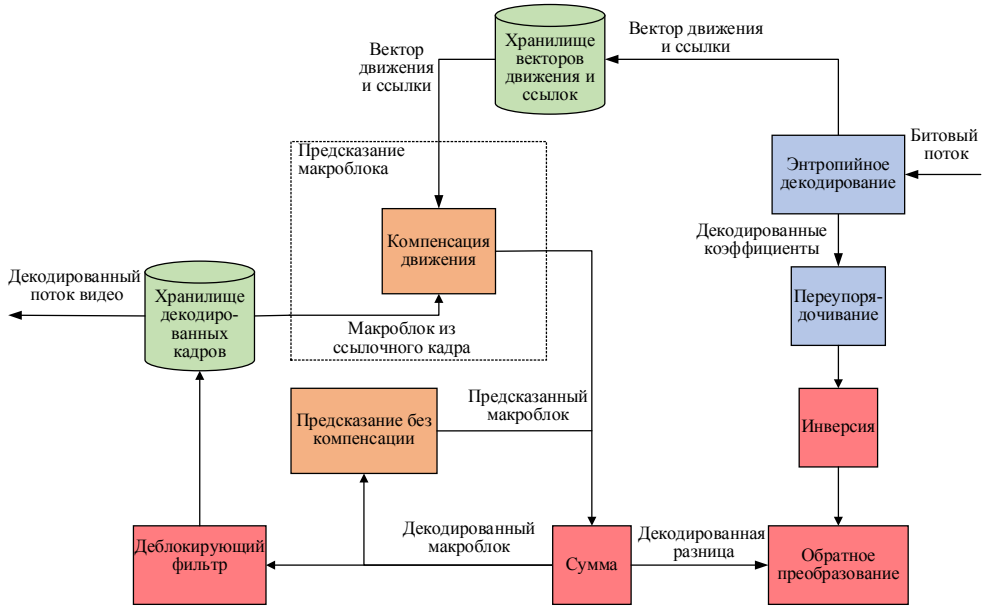


Рис. 2. Структурная схема декодера

Декодер содержит меньшее количество операций, чем энкодер, и декодирование последовательности битов проще, чем кодирование, поэтому суммарное время, затраченное на операции декодирования, меньше, чем на операции кодирования.

Анализ программной реализации видеокодека. Программная реализация кодека представляет собой прикладное программное обеспечение (ПО), которое обрабатывается центральным процессором (ЦП) системы без помощи специального оборудования.

Характеристики тестовой процессорной системы для тестирования видеокодека приведены ниже:

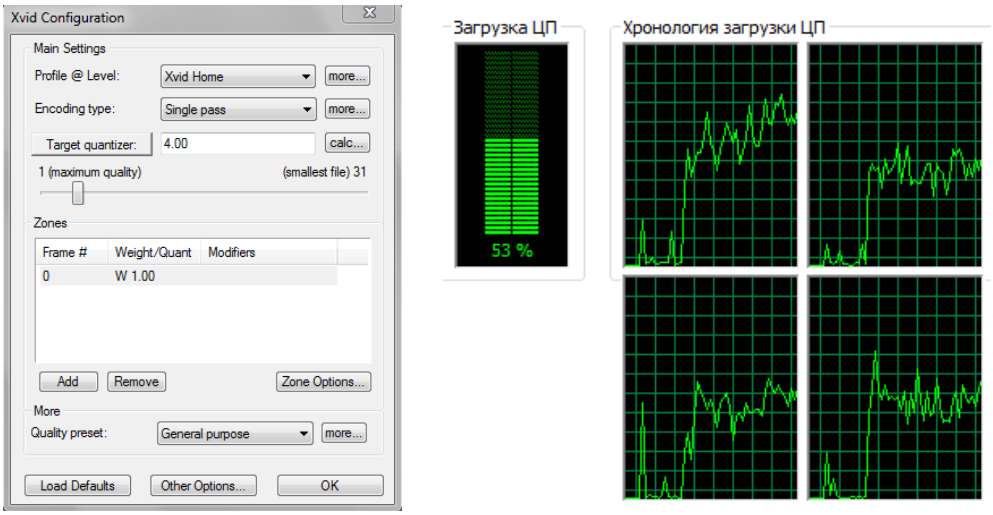
Наименование ОС ..... Windows 7 SP1
Тип ОС 64-разрядная
Процессор ..... Intel Core i5 760 2.80 GHz
Объем кэша L1, Кб 64
Объем кэша L2, Кб 1024
Объем кэша L3, Кб 8192
Тепловыделение процессора, Вт 95
Количество ядер 4
Количество потоков 4
ОЗУ, Гб 6,00
Видеоадаптер ..... NVIDIA GeForce GTX 670

Характеристики исходной видеопоследовательности для тестирования представлены ниже:
Продолжительность, с 33

Ширина кадра, пкс 1920
Высота кадра, пкс 1080
Общая скорость потока, кБ/с 1 492 992
Частота кадров, кадров/с 60
Количество кадров 2013
Размер, Гб 2,98

Обработку несжатого видео проводили с помощью программы VirtualDub 1.10.4 Rus. Это свободная утилита для захвата, монтажа и редактирования видеопотока. Ее функционал позволяет осуществлять также компрессию видео с использованием разных кодеков.

Также реализация программы обладает широким спектром настроек, позволяющим реализовывать компрессию (сжатие) видео под любые задачи и параметры (рис. 3, *а*).



а *б*

Рис. 3. Конфигурация программного кодека Xvid (*а*) и хронология загрузки ЦП во время сжатия видеопотока программной реализацией видеокодека (*б*)

Конфигурация *Profile @ Level* — это выбор стандартных профилей. Для нашего случая выбираем *unrestricted*. Атрибут *Encoding type* — тип кодирования в один или два прохода. Выбираем — однопроходный режим, он годится для кодирования в реальном времени или если важна скорость. Настройка *Target quantizer* / *Target size* — выбор шага квантования. Будем изменять его значение от 1,00 (фактически без потерь качества) до 31,00 (лучшее сжатие). Тест будет проводиться для пяти коэффициентов: 1,00; 4,00; 10,00; 20,00 и 31,00.

Для остальных настроек кодека (см. рис. 3, *а*) установлены значения «по умолчанию».

В ходе проведения тестов можно было убедиться в том, что программная компрессия видео значительно нагружает центральный процессор системы, потребляя его ресурсы (рис. 3, *б*). На рисунке виден резкий скачок графика загрузки ЦП в момент начала работы кодека.

Результаты тестов и полученные характеристики сжатого видео представлены в табл. 1.

Таблица 1

Результаты тестирования программного видеокодека

Параметр	Значение				
Коэффициент (шаг) квантования	1,00	4,00	10,00	20,00	31,00
Время обработки, с	194	205	172	153	147
Размер файла, Мб	993	265	87,1	34,2	20,3
Общая скорость потока, Кб/с	248 510	66 493	21 786	8 554	5 094
Частота обработки кадров, кадр/с	10,37	9,82	11,70	13,16	13,70

Таблица наглядно демонстрирует, насколько эффективно программа провела сжатие и за какое время. Например, сжатие с наилучшим качеством позволяет уменьшить размер файла в 3,07 раза, а максимально доступное сжатие уменьшит файл более чем в 150 раз. Однако время обработки по-прежнему остаётся больше чем исходный файл в несколько раз.

На рис. 4, *а–е* представлены фрагменты сжатого видео, они позволяют оценить качество сжатого кадра относительно исходного для разных коэффициентов квантования.

а *б* *в*

г *д* *е*

Рис. 4. Фрагмент исходного (*а*) и сжатых кадров (*б–е*) с разными коэффициентами квантования:

б — 1,00; *в* — 4,00; *г* — 10,00; *д* — 20,00; *е* — 31,00

На полученных результатов тестирования и визуального восприятия кадров квантование с коэффициентом 4,00 можно принять за оптимальное, поскольку оно обеспечивает значительное сжатие данных практически без потери качества картинки.

При наличии нескольких ядер у ЦП имеется возможность улучшить скорость кодирования за счет параллельного использования программных кодеков.

Анализ аппаратной реализации видеокодека. Для анализа аппаратной реализации было разработано описание логики кодека на языке VHDL, проведена симуляция ее работы, и оценено время прохождения сигналов в самой ПЛИС [7]. Максимальный формат кодируемого видео для реализуемого тестового модуля является видео формата full HD.

Характеристики тестового модуля представлены ниже:

Название	Zynq-7007S
Логических ячеек	23 000
Триггеров	28 800
Таблиц поиска (LUTs)	14 400
Задающая частота, МГц	33,33
Частота синхронизации, МГц	433,33
Процессорное ядро	ARM Cortex-A9
Максимальная частота, МГц	667
Мощность, Вт	10

Из-за особенностей реализации логики на ПЛИС все процессы выполняются параллельно (одновременно), следовательно, тормозить процесс кодирования будет тот блок структуры, на который данные поступают последовательно и не могут быть обработаны сразу.

Известно, что входной поток видео содержит следующие основные сигналы: тактовый или сигнал горизонтальной синхронизации, вертикальной синхронизации, кадровой синхронизации, а также шину данных, содержащую значения коэффициентов.

Отметим, что все блоки кодека связанны тактовым сигналом или сигналом горизонтальной синхронизации, по фронту которого, каждый такт происходит какое-то событие. Сигналы кадровой и горизонтальной синхронизации следуют по кодеку за макроблоком, содержащим значение коэффициента последнего пикселя в строке или кадре. Это необходимо для добавления верных заголовков пакету, в котором будут содержаться эти граничные коэффициенты, при передаче.

В результате модуляции тестовой реализации кодека и оценки времени работы всех его элементов (см. рис. 1 и 2) было выявлено, что блок энтропийного кодирования является одним из самых долгих. Следовательно, по времени его работы можно оценить время работы всего кодека.

Блок энтропийного кодирования использует для нахождения нужной последовательности кодов метод кодирования CAVLC (англ. *context adaptive variable length coding*). Бóльшую часть кодов выбирают из таблиц на основе дан-

ных макроблока. То есть для работы CAVLC необходимо знать все коэффициенты макроблока, чтобы обработать их значения и выбрать соответствующую им уже готовую к передаче последовательность битов.

Во время и по окончанию передачи коэффициентов необходимо их обработать и собрать данные о том: сколько в них значащих коэффициентов (не нулевых) и сколько нулей между ними, какое количество подряд идущих единиц с конца переупорядоченной последовательности до любого другого коэффициента. Также для каждого значащего коэффициента по очереди с конца последовательности потребуется рассчитать значения так называемых суффиксов и префиксов и после на основе всех этих контрольных значений выбрать (из таблиц CAVLC) выходные коды, которые, в свою очередь, передаются в следующий блок кодека для разбиения на пакеты и передачу в выходной битовый поток.

В худшем случае, когда все 16 коэффициентов являются значащими (не нулевые значения), для их кодирования потребуется 32 такта и 1 такт для передачи полученного кода дальше. В лучшем случае после 16 тактов получения коэффициентов потребуется только передать полученные данные — это займет 1 такт.

В итоге имеем худший вариант — 33 такта. При удачном стечении обстоятельств — только 17 тактов. Здесь следует заметить, что коэффициенты, поступающие в блок энтропийного кодирования, после всех преобразований среди своих значений имеют больше всего нулей, иногда встречаются единицы, значения же, отличные от них, встречаются значительно реже. Учитывая данный факт, можем смело использовать среднее значение обработки микроблока, равное 25 тактам.

Зная самое долгое время обработки макроблока кодеком, можно подсчитать требуемую тактовую частоту для обработки данных всей системой, которая при скорости видеопотока 60 кадров в секунду и формате видео full HD, вычисляют по формуле

$$f_{\text{треб}} = \frac{ab}{16} vt = \frac{1920 \cdot 1080}{16} \cdot 60 \cdot 25 = 194\ 400\ 000\ \text{Гц} = 194{,}4\ \text{МГц},$$

где a — количество пикселей по ширине кадра видеопотока; b — количество пикселей по высоте кадра видеопотока; v — скорость входного видеопотока, кадров в секунду; t — среднее количество затрачиваемых тактов на один макроблок; 16 — количество коэффициентов в макроблоке.

ПЛИС, под которую был написан тестовый модуль, имеет тактовую частоту шины 433 МГц, следовательно, можем посчитать максимальную пропускную способность или максимально возможное число обработанных в секунду кадров, которое рассчитывают следующим образом:

$$v = \frac{f_{\text{шины}}}{abt} \cdot 16 = \frac{433 \cdot 10^6}{1920 \cdot 1080 \cdot 25} \cdot 16 = 133{,}6\ \text{кадров/с}.$$

При необходимости передавать более широкие форматы видео (выше full HD) или необходимости увеличить количество кадров в секунду, можно улучшить систему. В этом случае, воспользовавшись особенностями проектирования систем для ПЛИС и используя каскадирование самых медленных элементов кодека, можно увеличить скорость кодирования до требуемого уровня, не изменяя логики работы.

Здесь каскадированием называют подключение в логическом описании еще больше дополнительных, уже имеющихся блоков. Как уже говорилось раньше, все процессы выполняются параллельно, а следовательно, добавление в систему одинаковых элементов увеличивает скорость выполняемой ими функции фактически кратно количеству этих блоков. Таким образом, легко увеличить быстродействие системы до требуемого уровня. Однако потребляемые ресурсы ПЛИС и сложность структуры возрастут пропорционально такой оптимизации, ведь потребуется проводить еще больше операций одновременно.

Сравнительная оценка реализаций видеокодека. Главное отличие программной реализации от аппаратной состоит в том, что кодек реализован в виде программы. И при выполнении исходного кода, программа кодирования использует ресурсы ЦП, который кроме этого поддерживает работу всей операционной системы и ее компонент. Кроме того, центральный процессор компьютера по своей архитектуре не ориентирован на требуемые для кодирования операции.

По-другому обстоят дела у видеокодека на ПЛИС. Он реализован в виде в виде отдельного устройства со своим процессором, предназначенным только для кодирования и декодирования. То есть на основе описанной логики создается требуемая архитектура, которая нацелена на решение задач кодирования видео.

Сравним общую эффективность работы видеокодека у программной реализации на процессорной системе и аппаратной реализация на ПЛИС.

Поскольку алгоритм видеокодека один, величина сжатия останется одинаковой, при использовании одинаковых коэффициентах. Однако скорость сжатия данных, производительность и энергоэффективность различаются.

Производительность оценивает возможность кодирования видео в режиме реального времени и считается, по формуле $D = v_{сж}/v$, где v — скорость входного видеопотока, $v_{сж}$ — скорость кодирования видеопотока.

Энергоэффективность показывает затраченную энергию на обработку одного и того же количества кадров. Чем она больше, тем более экономичная система, ее находят по формуле $E = D/P$, где D — производительность, P — выделяемая мощность.

Результаты сравнения представлены в табл. 2.

Результаты тестирования показали, что потребление энергии у видеокодека на ПЛИС более чем в 100 раз меньше программной реализации, а производительность в 14 раз выше при одном и том же исходом видеофайле.

Таблица 2

Сравнение скорости кодирования, реализации на процессорной системе и ПЛИС

Параметр сравнения	Вид реализации	
	Программная	Аппаратная
Процессор	Intel Core i5 760	ARM Cortex-A9
Операционная система	Windows 7 SP1 (x64)	-
Тактовая частота, МГц	2800 (ЦП)	433
Ширина кадра	1920	1920
Высота кадра	1080	1080
Скорость входного потока, кадров/с	60	60
Мощность, Вт	95	10
Лучшая скорость обработки, кадров/с	13,70	200,4
Производительность сжатия	0,228	3,33
Энергоэффективность	0,0024	0,33

В итоге, огромные преимущества аппаратной реализации кодека, да и любого алгоритма, реализованного на ПЛИС, заключается в том, что все описанные логикой операции выполняются параллельно, и все процессы происходят в режиме реального времени. В программной реализации процесс кодирования занимает гораздо больше времени даже на значительно больших мощностях процессорной системы при использовании нескольких ядер.

Заключение. Программные кодеки более доступны, чем аппаратные. Это связано с их большой разновидностью и возможностью выполнить исходный код программы практически на любом компьютере. Более того, программная реализация используется повсеместно при просмотре видео на компьютере и в сети Интернет, в режиме реального времени. Однако там применяется лишь декодирующая часть видеокодека, поскольку мы находимся на принимающей стороне и нам не нужно ничего сжимать. А мощности любого современного компьютера и скорости линий передач достаточно для декодирования в комфортном для просмотра режиме.

Если же необходимо закодировать видео, то актуальность использования аппаратной реализации выходит на передний план. Особенно это важно, при проектировании линий связи для камер наружного наблюдения, или линий передач, в авиационной технике, где необходимо сразу получать и записывать данные.

Сегодня многие все еще недооценивают возможностей ПЛИС, считают, что это не эффективно или боятся ее использовать. Проведенный анализ и сравнение реализаций видеокодека показывают несправедливость такого суждения. ПЛИС находит применение во многих сферах человеческой жизнедеятельности, от медицинской техники до самообучающихся нейросетей [8]. И в будущем у программируемой логики остаются сильные перспективы.

Литература

[1] Аминев Д.А. Видеорегистраторы. Тенденции и перспективы. *Техника средств связи. Сер. Техника телевидения*, 2013, № 1, с. 84–92.
[2] Richardson I.E. *The H.264 advanced video compression standard*. Wiley, 2010, 346 p.
[3] Recommendation ITU-T H.264. Series H: audiovisual and multimedia systems. Infrastructure of audiovisual services – coding of moving video. ITU, 2017, 804 p.
[4] Аминев Д.А. Реализация системы встраивания дополнительной информации при кодировании видеопотока MPEG-2 с использованием ПЛИС. *Техника средств связи. Сер. Техника телевидения*, 2011, № 1, с. 98–103.
[5] Аминев Д.А., Фокин А.Н. Методы селекции кадрового синхроимпульса для ввода несжатого видеопотока от однонаправленной одноразрядной цифровой линии и их реализация на ПЛИС. *Цифровая обработка сигналов*, 2014, № 1, с. 52–55.
[6] Аминев Д.А, Гранкин Д.С. Испытательные таблицы для измерения качества изображения. *Системы и средства связи, телевидения и радиовещания*, 2013, № 1-2, с. 57–59.
[7] Аминев Д.А. Опыт применения САПР при проектировании аппаратуры на основе ПЛИС. *Техника средств связи. Сер. Техника телевидения*, 2009, № 1, с. 25–30.
[8] Матюшкин И.В, Соловьев Р.А. Модель адаптивного нейрона и его аппаратная реализация на ПЛИС. *Электронная техника. Сер. 3: Микроэлектроника*, 2017, № 3, с. 53–61.

Кобецкий Владимир Андреевич — студент кафедры «Проектирование и технология производства электронной аппаратуры», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Казаков Никита Дмитриевич — студент кафедры «Вычислительные машины, системы и сети», Московский авиационный институт, Москва, Российская Федерация.

Научный руководитель — Волков Дмитрий Александрович, инженер научно-исследовательской лаборатории, Раменское приборостроительное конструкторское бюро, Московская область, Раменское, Российская Федерация.

COMPARATIVE ANALYSIS OF EFFICIENCY OF HARDWARE AND PROGRAM IMPLEMENTATION OF COMPRESSION ALGORITHM OF VIDEO

V.A. Kobetskiy[1] rk62ck@mail.ru
SPIN-code: 3992-1846

N.D. Kazakov[2] nkazakovd@gmail.com
SPIN-code: 1971-5303

[1] **Bauman Moscow State Technical University, Moscow, Russian Federation**
[2] **Moscow Aviation Institute, Moscow, Russian Federation**

Abstract

The article analyzes the running efficiency of various implementations of the video encryption algorithm. The general algorithm of video codec operation is considered. The structural scheme of the video codec and a description of its components are developed. The software implementation of the video codec on the processor system is tested. Testing of the hardware implementation of the video codec on a programmable logic integrated circuit (FPGA) is carried out. In the course of studies, the compression rate of the video stream data for both versions of implementation of the video codec is determined. Comparison of the compression efficiency of the video codec on the FPGA and on the processor system is the result of the article Conclusions about the scope of various implementations of the video codec and the relevance of their development in the future are made.

Keywords

Video coding, FPGA, processor system, compression efficiency, video codec scheme, implementation of the video codec, encryption algorithm, video stream compression

Received 24.05.2018

References

[1] Aminev D.A. Videoregistrators. Trends and prospects. *Tekhnika sredstv svyazi. Ser. Tekhnika televideniya*, 2013, no. 1, pp. 84–92.
[2] Richardson I.E. The H.264 advanced video compression standard. Wiley, 2010, 346 p.
[3] Recommendation ITU-T H.264. Series H: audiovisual and multimedia systems. Infrastructure of audiovisual services – coding of moving video. ITU, 2017, 804 p.
[4] Aminev D.A. Realization of FPGA based system for embedding of additional information in MPEG-2 encoding. *Tekhnika sredstv svyazi. Ser. Tekhnika televideniya*, 2011, no. 1, pp. 98–103.
[5] Aminev D.A., Fokin A.N. Methods of selection of personnel sync pulse input in uncompressed video stream from a unidirectional single-bit digital lines and their implementation on FPGA. *Tsifrovaya obrabotka signalov* [Digital Signal Processing], 2014, no. 1, pp. 52–55.
[6] Aminev D.A, Grankin D.S. Test charts for image quality measurement. *Sistemy i sredstva svyazi, televideniya i radioveshchaniya*, 2013, no. 1-2, pp. 57–59.
[7] Aminev D.A.Experience of SAPR application in development of FPGA-based equipment. *Tekhnika sredstv svyazi. Ser. Tekhnika televideniya*, 2009, no. 1, pp. 25–30.

[8] Matyushkin I.V, Colov'yev R.A. A model of adaptive neuron and its hardware implementation on FPGA. *Elektronnaya tekhnika. Ser. 3: Mikroelektronika* [Electronic engineering. Series 3. Microelectronics], 2017, no. 3, pp. 53–61.

Kobetskiy V.A. — Bachelor's Degree student, Department of Electronic Equipment Design and Technology, Bauman Moscow State Technical University, Moscow, Russian Federation.

Kazakov N.D. — Bachelor's Degree student, Department of Computers, systems and networks, Moscow Aviation Institute, Moscow, Russian Federation.

Scientific advisor — D.A. Volkov, Engineer of Research Laboratory, Ramenskoye Design Company, Ramenskoye, Moscow region, Russian Federation.