УДК 004.427

С. В. Очкур

Эффективная обработка видеоинформации на встраиваемых системах с использованием современных программных интерфейсов

Очкур Сергей Владимирович, инженер, аспирант радиотехнического факультета Санкт-Петербургского электротехнического университета (СПБГЭТУ “ЛЭТИ”). Инженер-программист встраиваемых систем ОАО «Авансофт» (Vanguard Software Solutions), младший научный сотрудник кафедры телевидения СПБГЭТУ “ЛЭТИ”. Имеет публикации и занимается НИР в области разработки видеокодеков, кодирования мультиспектральных данных и высокопроизводительных вычислений на графических ускорителях (GPGPU). [e-mail: sergey.ochkur@rusvision.com].

Аннотация: Целью данной статьи является обзор наиболее популярных программных интерфейсов (API) и стандартов, лежащих в основе современных мультимедийных устройств, их применение при проектировании приложений, требующих эффективной работы с аппаратными ресурсами. Актуальность этой задачи связана с существующей высокой загрузкой терминального оборудования видеоконференцсвязи, транскодеров и систем цифрового телевидения, что требует разработки новых аппаратно-программных решений на фоне постоянного увеличения объёмов аудио/видеоинформации и требований к скорости их обработки. В заключительной части статьи производится описание архитектуры многопрофильного мультимедийного приложения на основе рассматриваемых стандартов, приводятся данные об особенностях реализации описываемых стандартов на операционных системах Linux и Android.

Ключевые слова: OpenMAX, OpenGL, VA API, DSP, видеоускорение, встраиваемые системы, видеоконференцсвязь, видеокодек, Linux, Android, OpenKODE, Bellagio, GStreamer

Реферат: В данной статье описываются программные интерфейсы VA API, OpenMAX и OpenGL. VA API позволяет обеспечить кодирование и декодирование видеосигнала через аппаратные кодеки при графическом процессоре (GPU). OpenGL в рамках решения поставленной задачи используется для обеспечения ускорения входных и выходных согласующих преобразований видеопотока и работы с дисплеем. Особое внимание уделяется стандарту OpenMAX, позволяющему создавать и управлять мультимедийными компонентами, которые в полной мере используют аппаратные возможности базовой платформы (SIMD-расширения, цифровые процессоры (DSP) и специализированные аппаратные ускорители) и обладающие возможностью работы на

различном оборудовании. Предлагаемая архитектура легко переносится на другие аппаратные платформы, существенно сократив производственные риски, связанные с возможностью замены оборудования.

S. V. Ochkur

Effective video information processing in embedded systems based on modern application programming interfaces

Ochkur Sergey Vladimirovich, engineer, post-graduate student of Saint Petersburg Electrotechnical University “LETI”. Engineer-programmer of “Avansoft” JSC (Vanguard Software Solutions), scientific associate in SPBETU “LETI”, Television Systems department. Author and leader of articles and research works in coding video and multispectral data, general purpose graphical processing unit (GPGPU) programming. E-mail: sergey.ochkur@rusvision.com

Annotation: The purpose of this paper is review of the most popular software interfaces and standards that underpin today's multimedia devices and their usage in hardware-enabled applications. The urgency of this problem is largely due to the high load of the video conferencing systems, digital television terminals, transcoders and new hardware and software solutions requires while processing information demands increase. The conclusion part presents multimedia application architecture based on these standards is provided with additional information about design and behavior of API`s in Linux and Android operating systems.

Keywords: OpenMAX, OpenGL, VA API, DSP, video acceleration, embedded systems, videoconferencing, video codec, Linux, Android, OpenKODE, Bellagio, GStreamer

Abstract: This article describes the VA API, OpenMAX and OpenGL programming interfaces. VA API enables fast encoding/decoding via hardware videocodecs based on graphics processor (GPU). OpenGL is provide acceleration of input/output matching transformations and work with video display in the context of problem solution. Particular attention is paid to OpenMAX standard that enable the multimedia components creation and control providing comprehensive usage of hardware resourses (SIMD-extensions, DSP, accelerators) and enabling application portability amount different platforms. Proposed architecture enable easy design migrate between different hardware platforms, significantly reducing the operational risks associated with the possibility of the equipment replace.

1. Обзор текущей ситуации на рынке мультимедийных встраиваемых систем

На данный момент на рынке встраиваемых решений можно отметить следующие особенности развития: с одной стороны, уровень достижений в области микроэлектроники

позволяет активно выводить на рынок новые устройства с характеристиками, удовлетворяющими требованиям пользователей. С другой стороны, в области программного обеспечения существующая продукция отличается не меньшим уровнем диверсификации используемых программных решений, нежели разнообразие решений аппаратных. Отсутствие общих стандартов разработки мультимедийной составляющей устройства вынуждает разработчиков уделять большое количество времени и сил адаптации функционала предыдущего семейства встраиваемых решений к новым разработкам.

С точки зрения выбора архитектуры разработчики перспективных телекоммуникационных служб в большинстве случаев делают свой выбор в пользу признанных лидеров рынка – большое количество коммерчески удачных решений выпускается на основе конфигурации, в которой в качестве основного процессора (CPU) используется процессор на основе архитектуры ARM / PowerPC и цифровой процессор, в качестве вспомогательного. Это, в первую очередь, связано с той гибкостью, которой обладает это решение: основную нагрузку в аудио/видеоустройствах создаёт обработка мультимедийного потока, эффективность обработки которого с использованием DSP по сравнению с CPU более высока. В большинстве случаев цифровому процессору необходимо в 2-3 раза меньше тактов, нежели процессору общего назначения, для осуществления кодирования/ декодирования мультимедиа-информации [1].

В общем случае, процессы, с которыми работает типичное клиент-серверное AV-приложение можно разделить по следующей классификации [2] :

- детерминированные процессы - захват и отображение аудио/видеоданных, (де)кодирование видео- и аудиоданных.
- недетерминированные процессы – процессы связанные с обработкой событий и работой с Ethernet, USB, FLASH, AV-(де)мультиплексором.

Недетерминированные процессы являются апериодическими, детерминированные — периодическими. С точки зрения архитектуры, эти процессы не следует смешивать: хост-процессор должен обрабатывать детерминированные процессы с помощью цифрового сигнального процессора (DSP), действующего в качестве сопроцессора. В противном случае обеспечение достаточного уровня быстродействия при обработке видеопотока высокого пространственного разрешения возможно только при условии наращивании частоты основного процессора, что крайне негативно отражается на основных характеристиках мобильного устройства. Такое разделение даёт большой выигрыш в производительности, дополнительно обеспечивая высокую масштабируемость вычислительной конфигурации. В связи с этим обусловлена постановка задач по

удовлетворению нужд разработчиков различных уровней разработки мультимедийных решений [3]:

- Для программистов обеспечить кроссплатформенный API для высокоуровневого программирования мультимедийных приложений
- Для системных интеграторов – создать мультиплатформенные (кроссплатформенные) стандарты для облегчения процессов интегрирования
- Для поставщиков/разработчиков программных компонент – создать условия для ускорения создаваемых кодеков на различном оборудовании

На данный момент перечисленные задачи, в той или иной степени, решены (новыми или перенесёнными и/или адаптированными с настольных систем) системами стандартов и развитой поддержкой последних в современных устройствах. Тем не менее, предлагаемая инфраструктура, увеличивающая своё присутствие в продукции крупнейших мировых производителей, крайне редко находит своё применение в продукции малых компаний и научно-исследовательских лабораторий. В данной статье рассматриваются вопросы построения эффективной мультимедийной системы на базе современного оборудования путём максимального использования существующей надежной кодовой базы при сохранении модульности подсистем.

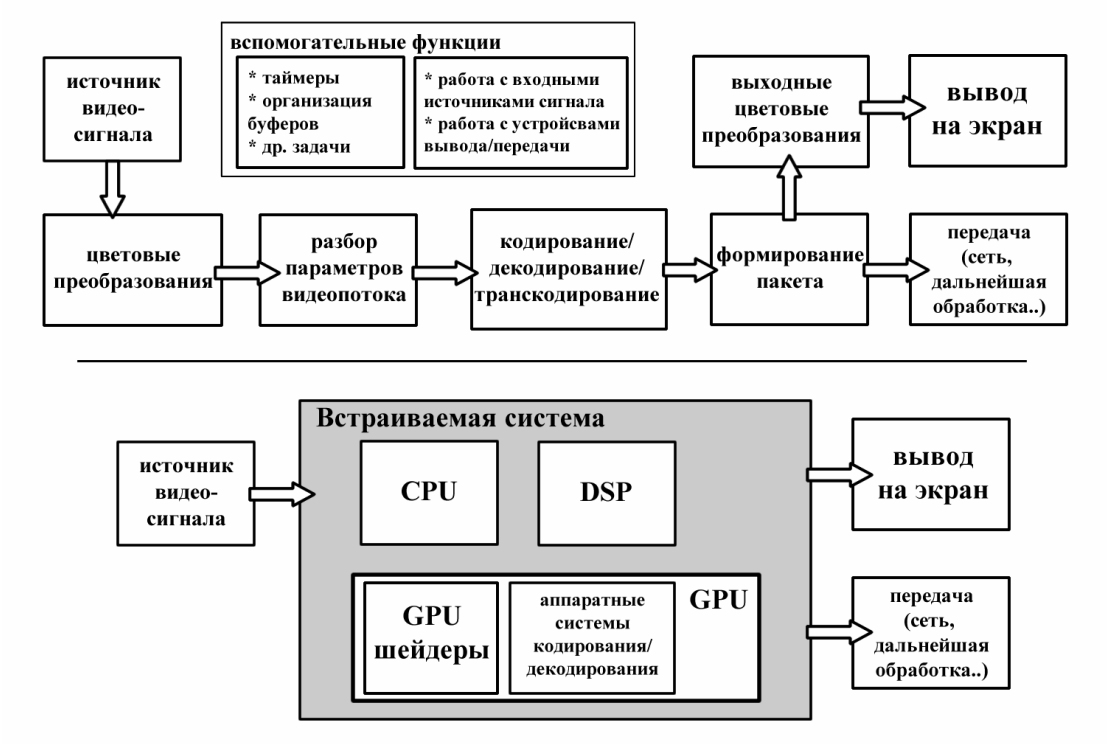

Рис. 1. Принципиальная схема видеопотока и структура современной мультимедийной встраиваемой системы

На рисунке 1 представлены основные модули современной встраиваемой системы (нижняя часть рисунка) и предлагаемый к реализации типичный видеотракт (верхняя часть рисунка). Все предлагаемые к рассмотрению стандарты далее будут рассмотрены в свете степени реализации предложенного тракта обработки. Исходя из степени распространённости в качестве базовых были выбраны операционные системы семейства GNU Linux и базирующаяся на ядре Linux популярная операционная система Android.

2. API видеоускорения

API видеоускорения (VA API) является программной системой, состоящей из библиотеки с открытым исходным кодом libVA и общей спецификации API. Данный программный интерфейс дает возможность получить доступ к чипам кодирования/декодирования видеосигнала различных стандартов на базе графического процессора (GPU). Изначально VA API предназначается для графической системы X Window операционных систем семейства Unix (включая Linux, FreeBSD и Solaris) и, впоследствии, был адаптирован и для графической системы ОС Android (Surfaceflinger).
Возможности, предоставляемые VA API:

- кодирование / декодирование видео
- добавление субтитров
- рендеринг

Использование VA API позволяет обеспечить полную разгрузку основного процессора для преобладающей группы стандартов кодирования (MPEG-2, MPEG-4 ASP/H.263, MPEG-4 AVC/H.264, а VC-1/WMV3).

В большинстве современных настольных и встраиваемых систем есть только аппаратные декодеры видеопотока, сжатого в соответствии с вышеперечисленными стандартами кодирования. Однако, в последнее время распространение получили и чипы с поддержкой функций кодирования, что позволяет создавать приложения кодирования и транскодирования видеосигнала с полным видеоускорением. В данном случае речь идёт о чипах семейства VXE компании PowerVR, новых процессоров Sandy Bridge (Intel) [4] и др.

На рисунке 2 видно, что использование библиотеки VA API обеспечивает практически полную разгрузку основного тракта преобразования. В случае декодирования исключением является работа с источником сигнала, необходимость извлечения и синтаксического анализа (парсинга) данных видеопотока с последующей загрузкой в графический процессор вместе с данными для обеспечения заданных параметров декодирования.

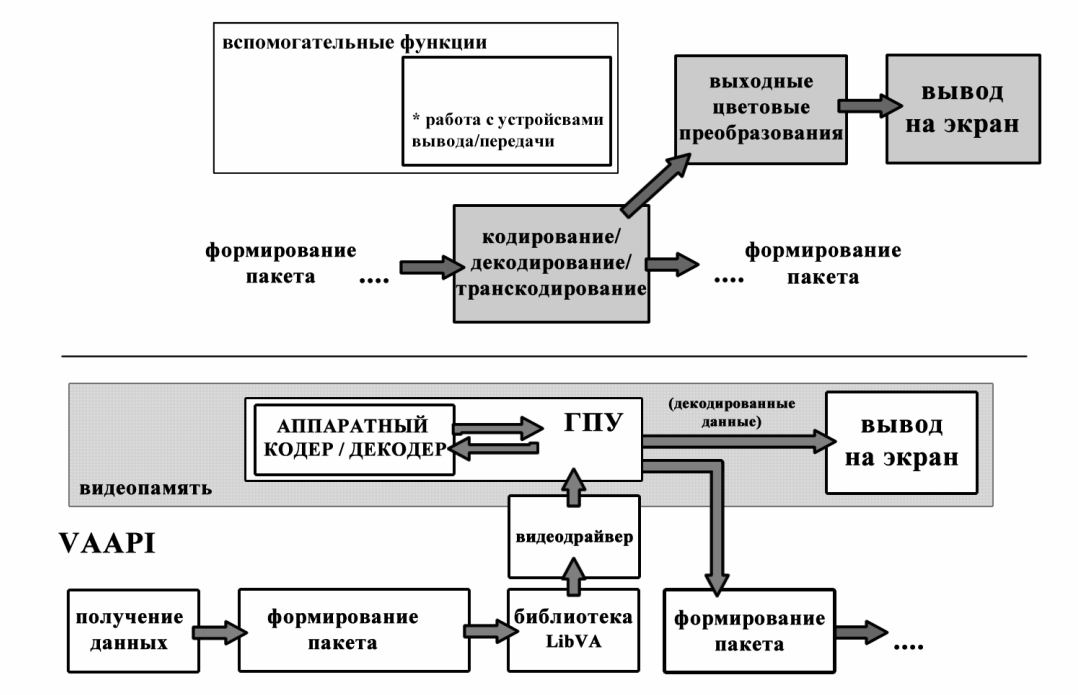

Рис. 2. Часть целевой принципиальной схемы видеообработки, задействованной использованием VA API и соответствующая ей структура обработки данных в библиотеке и устройстве

Общий план разработки приложения с видеоускорением через VA API:

1) Убедиться в поддержке устройством необходимых функций и работе драйвера, поддерживающего VA API. Возможны варианты, когда в системе уже существует фирменная альтернатива VA API, для использования которой необходимо изучение документации производителя. Однако в таких случаях, как правило, использование VA API возможно в режиме оболочки вокруг фирменного программного решения. В таком варианте VA API в англоязычной литературе называется frontend (“передний край”), а фирменное решение – backend (“задний край”).

2) Установка последней версии библиотеки libva, проверка её работоспособности через тестовый запуск одного из прилагающихся демо-приложений

3) Разработка приложения в соответствии с существующими примерами использования различных видеокодеков (пакет hwdecode-demos) [5].

К сожалению, нередко производители не приводят функции кодирования и декодирования к стандарту, тем самым лишая разработчиков возможности использования их напрямую (или же полностью закрывая программный интерфейс). Также нередки случаи, когда в устройстве присутствует аппаратный декодер, но декодирование производится с использованием цифрового процессора через OpenMAX API.

3. OpenMAX

3.1. История и экосистема проекта OpenMAX

OpenMAX — свободный кроссплатформенный API, предлагающий комплексное решение для создания мультимедийных кодеков, активно использующих аппаратные возможности оборудования в различных операционных системах и аппаратных платформах. OpenMAX позволяет существенно ускорить разработки в области кодирования/декодирования информации. Программный интерфейс создаёт жёсткую трёхуровневую структуру, которая скрывает в себе многие особенности мультимедийного окружения производителя (см. рисунок 3).

Стандарт разработан промышленным некоммерческим консорциумом Khronos Group, занимающимся разработкой открытых стандартов и интерфейсов программирования (API) в области обработки и воспроизведения динамической графики и звука на широком спектре платформ и устройств, с поддержкой аппаратного ускорения. На данный момент в консорциум входят более 100 компаний, таких как AMD, ARM, Intel, Apple, Google и NVIDIA [6]. Такой уровень сотрудничества серьёзно облегчает решение всех вопросов, связанных с распространением стандартов, повсеместным внедрением их поддержки в конечных устройствах.

OpenMAX API состоит из 3 уровней: разработки (DL), интеграции (IL), приложения (AL). Уровни являются полностью независимыми друг от друга, что позволяет последовательно реализовывать каждый из них в своих приложениях.

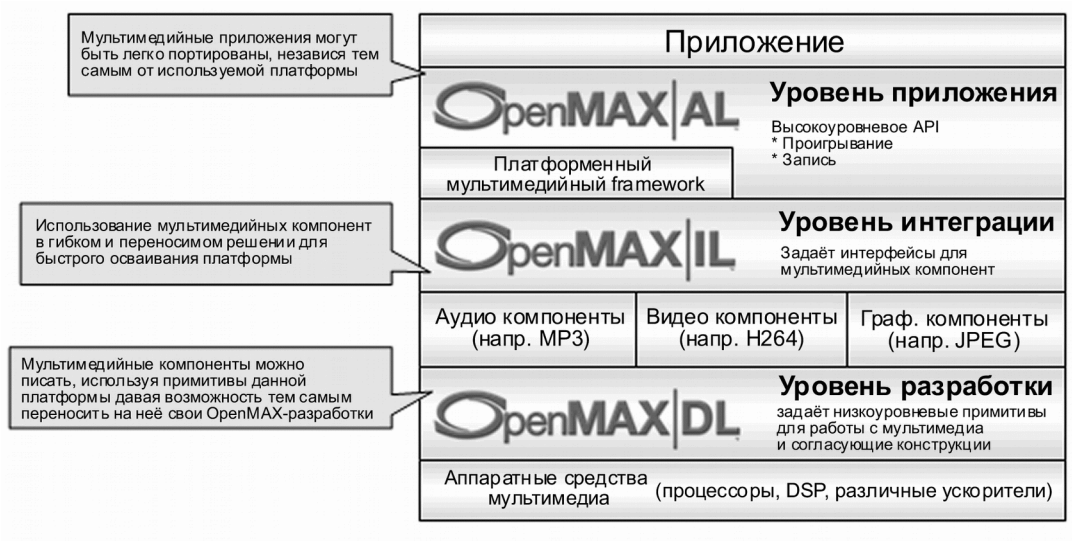

Рис. 3. Три уровня OpenMAX API [3].

Кроме широких возможностей по программной интеграции, важным достоинством работы с OpenMAX API является её интеграция в общую программную платформу

OpenKODE, которая, обеспечивая глубокие связи с другими стандартами: OpenGL | ES, OpenVG и OpenSL | ES, является арбитром их взаимодействия.

Из рисунка 4 видно, что OpenMAX практически полностью обеспечивает реализацию всех необходимых модулей. Все базовые модули видеотракта предоставляются в той или иной степени (в засимости от платформы) и управляются с помощью OpenMAX IL. OpenMAX DL позволяет создавать свои модули, впоследствии интегрируя их в общий контекст OpenMAX IL-элементов (см. нижнюю часть рисунка 4). OpenMAX AL выполняет роль высокоуровневой оболочки.

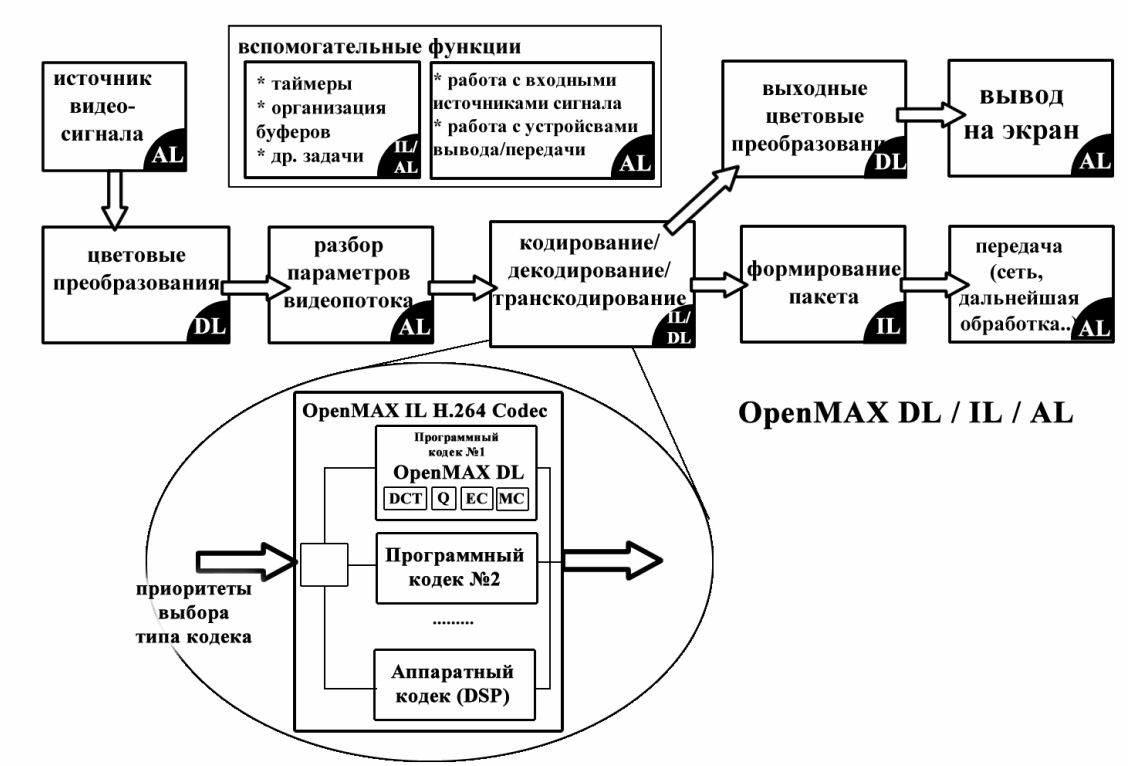

Рис. 4. Принципиальная схема видеообработки, рассмотренная с точки зрения реализаций её модулей через стандарт OpenMAX

3.2. OpenMAX DL: уровень разработки

OpenMAX DL — самый нижний, аппаратно-ориентированный уровень OpenMAX. Он определяет те мультимедийные функции, которые могут быть реализованы на этом оборудовании и то, как они будут реализованы с учётом имеющихся аппаратных возможностей. Среди оборудования могут быть процессоры, аппаратные ускорители и DSP, активно используемые при реализации функций, используемых кодеками MPEG-4, H.264, MP3, AAC и JPEG.

Использование уровня DL в своих приложениях на конкретной конфигурации оборудования возможно при наличии аппаратных возможностей и реализации функций OpenMAX DL через инструкции вычислителя. Одним из наиболее ярких примеров

является реализация уровня в наиболее популярных процессорах высокопроизводительного сегмента рынка от компании ARM – Cortex-A8. Встроенный в процессор 10-стадийный конвейер обработки мультимедийных данных NEON позволяет серьёзно улучшить производительность в мультимедийных приложениях [8], обеспечивая:

- Возможность 60-150% выигрыша производительности в последних видеокодеках;
- В отдельных простых алгоритмах ЦОС, достижение 4-8 кратного ускорения;
- Уменьшение загрузки центрального процессора в пользу специализированных расширений, позволяет снизить общее энергопотребление.

Ручная оптимизация через векторные инструкции (NEON) и ARM assembler является задачей, требующей достаточно высокой квалификации. В контексте того, что данная работа уже была выполнена инженерами компании-производителя процессоров, OpenMAX DL представляет собой готовое высокопроизводительное решение для создания аудио и видеокодеков на базе готовых блоков, реализованных через высокоптимизированные инструкции. Такая работа была проделана автором в [9], в результате чего при создании кодека на основе 3D-DCT в кратчайшие сроки был получен средний выигрыш в произовадительности 28% относительно существующего оптимизированного аналога на языке Си.

Внутри стандарта выделено несколько доменов:

- Кодеки видео (MPEG-4 SP/H.263 BL, H.264)
- Кодеки изображения (JPEG)
- Обработка изображений ((де)блокинг, преобразования цветов и др.)
- Кодеки аудио (MP3, AAC)
- Специальная обработка сигналов (FIR, IIR, FFT и др.)

К каждому домену относится определённое количество функций, которые можно использовать в своих программах, написанных на языке C/C++.

Несмотря на все преимущества использования готовых блоков DL-слоя для существующих кодеков, необходимо признать, что процесс интеграции может вызвать дополнительные трудности ввиду особенностей программных решений различных реализаций слоя разработки: в случае использования OpenMAX DL требуется работать с теми примитивами, которые представлены в конкретной реализации, что нередко приводит к необходимости переработки формата потока.

3.3. OpenMAX IL: уровень интеграции

OpenMAX IL — интерфейсный слой для мультимедийных кодеков, реализованных аппаратно или программно. В этом слое не представлены средства для проигрывания аудио и видео, синхронизации и пр. В общем случае выступает как низкоуровневый интерфейс для аудио/видео и графических кодеков используемых для встраиваемых и мобильных устройств [10]. Этот уровень даёт возможность приложениям и мультимедийным средам разработки взаимодействовать с кодеками и поддерживаемыми компонентами в унифицированной форме (рисунок 5).

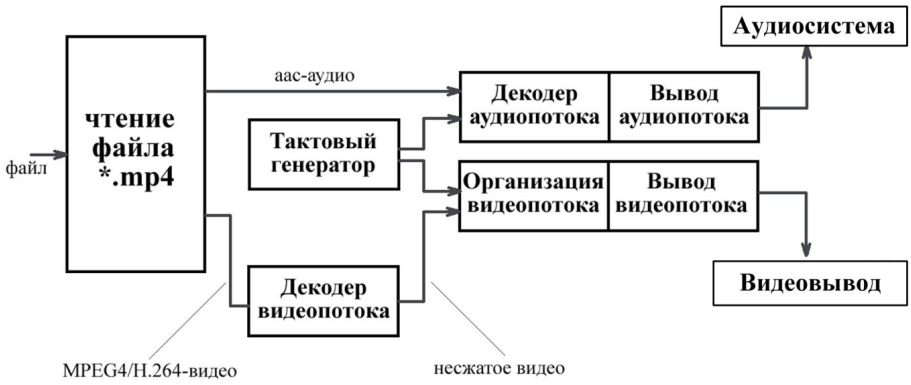

Рис. 5. Воспроизведение H.264-видео с помощью OpenMAX IL

Все компоненты IL-слоя работают по принципу “чёрного ящика”, который получает асинхронные команды и осуществляет приём/передачу информационного потока через свои порты.

На практике написание видеосистемы на основе OpenMAX IL “с нуля” чаще всего практикуется лишь крупнейшими корпорациями для своих продуктовых линеек. Средние и малые производители зачастую используют открытые реализации: Bellagio или, реже, LIM [11].

Система Bellagio, разработанная ST Microelectronics, является одной из наиболее популярных реализаций OpenMAX IL API для ОС Linux. Наиболее документированная и сопровождённая примерами использования она включает (в версии 0.9.2.1) [12]:

- Разделяемую библиотеку с IL-ядром и OpenMAX компонентом связи (загрузчиком статических библиотек)
- OpenMAX-компоненты, прошедшие тесты на соответствие стандарту

Bellagio позволяет разрабатывать собственные OpenMAX мультимедийные и потоковые компоненты, среди которых кодеки и аудио/видеомультиплексоры. Включённые в поставку примеры компонент (работа с камерой, JPEG-кодирование и др.) согласованы с ядром Bellagio и могут быть использоваться сразу после установки.

3.4. OpenMAX AL: уровень приложений

OpenMAX AL — интерфейсный слой для захвата и вывода/проигрывания аудио и видео. AL предназначен в первую очередь для удобного доступа к функциям нижележащих слоёв, защищая от лишних ошибок и нерационального использования, стандартизируя тем самым основные функции синхронизации и работы с потоками. AL обеспечивает переносимость мультимедийной части приложения на другие платформы.

Предпосылками к созданию надстройки в виде уровня AL послужила достаточно высокая сложность IL-уровня для большинства прикладных программистов, особенно в условиях совместного использования с другими API. Уровень приложения оперирует такими понятиями как “объект”, “объект-обработчик”, “объект-видеопроигрыватель”, “объект-микшер” и “интерфейс”. Единственной свободной реализацией OpenMAX AL является реализация LIM [12], сочетающая в себе уровни IL и AL. В качестве примеров использования документация приводит [13] схемы реализации управления потоками, сигналами и таймерами видеокамеры, фотокамеры, проигрывания радиотрансляции, записи аудио и др. Кроме того, на данный момент активно разрабатываются приложения верхнего уровня работы с AL для реализации систем цифрового телевидения. На момент написания статьи использование OpenMAX AL слабо распространено.

4. OpenGL

Если в разделе, посвящённом библиотеке VA API речь шла об использовании расширений видеопроцессора – сопроцессоров с жёсткой логикой – то в случае с OpenGL работа ведётся с одним из основных вычислителей GPU – шейдерным процессором.

OpenGL (Open Graphics Library — открытая графическая библиотека, графическое API) — спецификация, определяющая независимый от языка программирования платформо-независимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику [14]. Для встраиваемых систем существует подмножество OpenGL ES 2.0, поведение которого, в рамках поставленных задач, практически не отличается от полной реализации.

GLSL (OpenGL Shading Language) — язык высокого уровня для программирования шейдеров. Само понятие “шейдер” означает спецпрограмму для одной из ступеней графического конвейера. Синтаксис языка базируется на языке программирования ANSI C, однако, из-за его специфической направленности, из него были исключены многие возможности, для упрощения языка и повышения производительности. В язык включены дополнительные функции и типы данных, например для работы с векторами и матрицами [14].

На рисунке 6 видно, что из всего тракта для шейдерных программ и OpenGL-функций отведены блоки работы с источником и дисплеем, а также цветовые преобразования. Начиная с OpenGL (ES) 2.0, GLSL включен в ядро, благодаря чему стало возможным прямое использование шейдерного процессора для целей входного, выходного цветового и геометрического (в том числе масштабирования) преобразования. Для этого на языке GLSL пишутся программы, подгружаемые вместе в GPU вместе с данными.

Как и в случае с использованием аппаратных видеокодеков, такой подход полностью разгружает основной процессор за счёт вывода масштабирования и конвертации цветовых пространств, выводит из основного тракта некоторые объёмные перемещения в основной памяти, а также позволяет более гибко подходить к процедуре вывода на экран, получая некоторые возможности (наложение, картинка-в-картинке) более простым способом и без увеличения нагрузки.

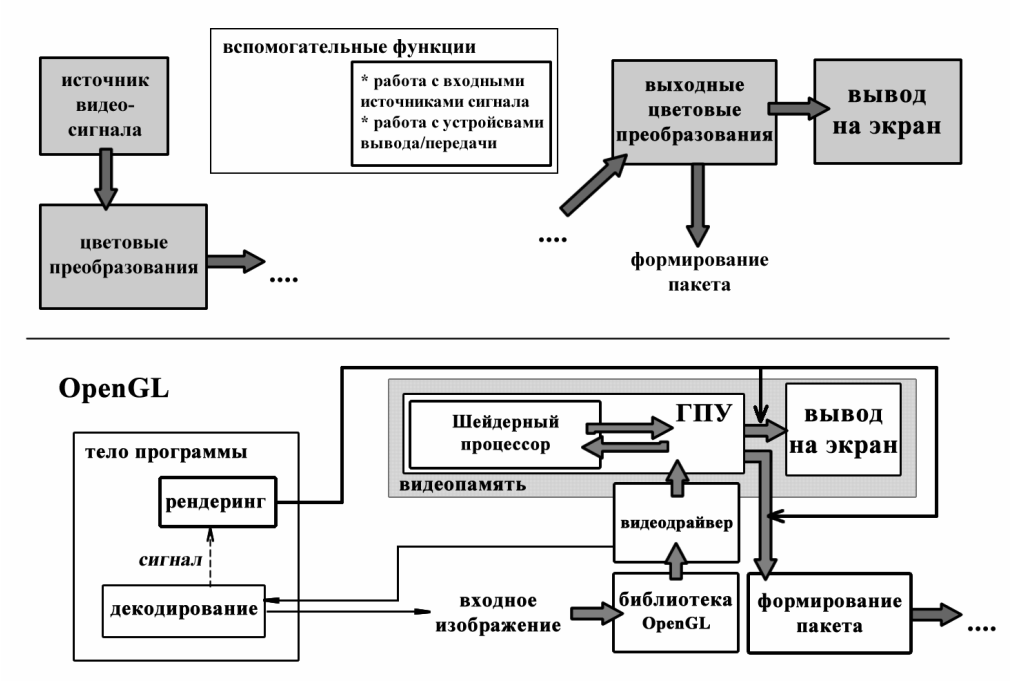

Рис. 6. Часть целевой принципиальной схемы видеообработки, задействованной использованием OpenGL и соответствующая ей структура обработки данных в библиотеке и устройстве

Возможности использования OpenGL могут варьироваться в зависимости от версии операционной системы, драйвера видеоустройства и версии самой библиотеки. Так, отмеченный на рисунке 6 как поддерживаемый модуль “источник видеосигнала”, в системе Android обрабатывается только начиная с версии 3. На устройствах с этой версией OpenGL получает возможность принимать видеоинформацию, например, с web-камеры

напрямую в видеобуфер, что не только позволяет сэкономить процессорное время основного вычислителя, но и иметь потенциально очень высокие скоростные характеристики вскупе с использованием VA API при условии, что изображение не будет передаваться в основную память и весь тракт будет находиться в памяти видеоускорителя.

5. Практика использования

Анализируя особенности применения рассмотренных стандартов, можно сказать, что VA API и OpenGL хорошо подходят для проектов низкой и средней сложности, без необходимости дальнейшей интеграции в сложные телекоммуникационные службы. В обоих случаях разработка приложений существенно ускоряется, быстродействие выходит на максимальный уровень, а сложность, в случае VA API зависит только подготовки программиста, а в случае с OpenGL дополнительно характеризуется сложностью GLSL-программ. В качестве примера можно привести задачи технического зрения, где кодирование сигнала может быть осуществлено аппаратным кодеком через VA API, а цветовые преобразования и масштабирование кадра для целей распознавания – через OpenGL. Похожим образом обстоят дела с OpenMAX DL – здесь процесс интеграции предлагаемых производителем DL-функций достаточно прост и не требует внимания к внешним деталям.

Проекты на основе OpenMAX IL, в большинстве своём, ориентированы на использование в качестве серьёзных компонент существующих программных сред, в том числе системных, как, например, это происходит сейчас в операционной системе Android, где вся видеоподсистема построена на основе OpenMAX IL. Системой предоставляется высокоуровневый интерфейс использования возможностей аппаратного ускорения (в данном контексте цифрового процессора), однако возможностей этого интерфейса достаточно лишь для построение простого медиапроигрывателя на его основе. В случае же более сложного проекта с необходимостью низкоуровневого контроля входных и выходных данных видеопотока программист так или иначе сталкивается с необходимостью взаимодействия с OpenMAX IL – элементами.

Как уже упоминалось выше, интеграция OpenMAX IL в приложение – задача нетривиальная и требует серьёзного понимания многих аспектов работы видеотракта. Учитывая также сложности в реализации вышестоящего высокоуровневого OpenMAX AL, автором предлагается, в качестве наиболее удобного и перспективного варианта, использование существующей программной реализации подобной системы, с возможностью модульного дополнения основного функционала. Одной из наиболее

подходящих каркас-структур программной системы (фреймворком) на встраиваемых системах является GStreamer.

GStreamer — мультимедийный фреймворк, написанный на языке программирования Си и использующий систему модулей. GStreamer является «ядром» ряда таких мультимедийных приложений, как видеоредакторы, потоковые серверы и медиаплееры. В его дизайн изначально была заложена кроссплатформенность, что позволяет использовать фреймворк, в том числе и на встраиваемых системах. В сотрудничестве с авторами GStreamer консорциумом был создан модуль расширения (плагин) gst-openmax [7], задействующий низкоуровневые модули для эффективной работы с DSP и реализующий поддержку компонентов OpenMAX IL. Фрейворк работает на большинстве встречаемых сегодня операционных систем, включая рассматриваемые Linux/Android. Фактически, GStreamer на данный момент используется во встраиваемых системах более часто, нежели OpenMAX AL, обладает несравнимо большими возможностями и, в то же время, способен интегрировать в свою инфраструктуру все вышеперечисленные стандарты и API. Ввиду постановки задачи о проектировании архитектуры мультимедийного тракта с максимально развитыми возможностями, замена OpenMAX AL на GStreamer видится обоснованным решением.

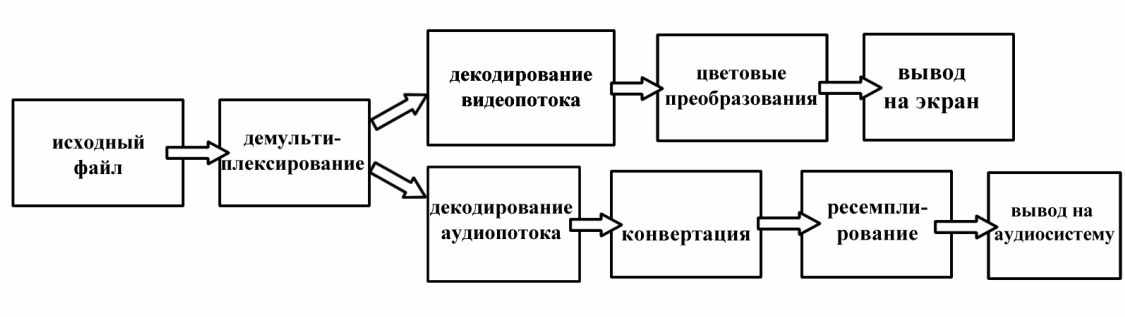

Рис. 7. Пример GStreamer-конвейера.

Приведённому на рисунке 7 видеопроигрывателю соответствует следующая строка запуска программы построения динамического конвейера GStreamer: gst-launch [15]:

gst-launch filesrc location="videofile.ogg" ! oggdemux name=d d. ! queue ! theoradec ! ffmpegcolorspace ! ximagesink d. ! queue ! vorbisdec ! audioconvert ! audioresample ! osssink

Кроме существующих базовых расширений, в GStreamer возможно добавление всех необходимых нам модулей:

VA API: модуль gst-vaapi [16]. Элементы: vaapidecode (декодирование данных), vaapidownload (возврат данных в основную память), vaapiupload (загрузка данных в видеопамять), vaapiencode (кодирование данных), vaapisink (вывод)

Пример использования: *gst-launch-0.10 -v filesrc location=/home/grid/Videos/serenity.mp4 ! qtdemux ! vaapidecode ! vaapisink fullscreen=true*

В данном примере происходит декодирование видеофайла через аппаратный видеокодек (в данном случае h.264), с последующим выводом на экран прямым отображением (блиттингом) из видеобуфера на экран.

OpenGL: элемент glupload (загрузка данных в видеопамять), glfilter (выполнение фильтрации), gldownload (возврат данных в основную память), Glimagesink (вывод)

Пример использования: *gst-launch videotestsrc ! glupload ! glfilter ! gldownload ! ffmpegcolorspace ! ximagesink*

Сгенерированный тестовый видеосигнал загружается в видеопамять, происходит его фильтрация (настроена заранее), возврат данных в основную память, цветовое преобразование и вывод на экран.

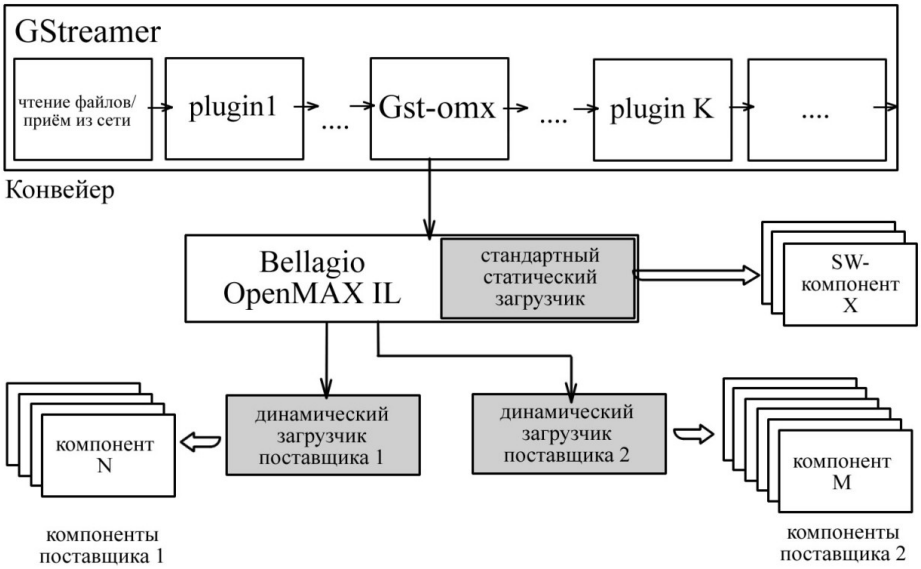

Рис. 8. Использование реализации OpenMAX IL (Bellagio) для обработки потоков в конвейере GStreamer

OpenMAX IL: модуль gst-openmax [7]. Элементы: используются элементы(кодеры/декодеры), присутствующие в системе. Список можно посмотреть командой gst-inspect.

Пример использования: *gst-launch filesrc location="kreml.avi" ! avidemux ! omx_mpeg4dec ! omapfbsink*

Последовательно происходят следующие действия: считывание видеофайла, разбор параметров, декодирование через DSP, вывод на экран (фреймбуфер *omapfbsink*).

В случае если задача требует введения собственного OpenMAX IL – элемента, целесообразно использовать OpenMAX IL – реализацию Bellagio, в которой возможна динамическая загрузка собственных разработок (рисунок 8).

Выводы:

1. На данный момент существуют достаточно развитые программные системы, ориентированные на создание тракта обработки, позволяющие повысить степень повторного использования кода, существенно сократить время разработки и обеспечить интеграцию модулей в большое число существующих приложений

2. Области действия стандартов и библиотек пересекаются. Таким образом необходимо тщательно выбирать вариант, подходящий для выбранных целей по временным и качественным характеристикам, а также по возможностям дальнейшего развития ПО. Использование VA API + OpenGL и OpenMAX направлено на создание приложений различной степени сложности и различного уровня интеграции с сервисами операционной системы

3. С точки зрения решения наиболее высокоуровневых задач мультимедиа-приложения целесообразным является использования модульных мультимедийных систем с возможностями расширения функционала и облегчения разработки в виде добавления и модификации отдельных программных модулей соответственно. В данной статье была выбрана система GStreamer, полностью удовлетворяющая данным требованиям, предоставляющая удобный инструментарий работы с видеопотоком и позволившая в минимальные сроки обеспечить аппаратное ускорение обработки данных задействовав все целевые ресурсы платформы.

Список литературы

1. Bhattacharya R. Open-source APIs up multimedia performance // EE Times Asia: In-depth analysis of industry issues. [Электронный ресурс]. URL: http://www.eetasia.com/ART_8800453622_499489_NT_65f881d8.htm (дата обращения 16.01.2012).

2. Саймерли Т. Транскодирование аудио- и видеоданных для бытовой электроники // Электронные компоненты. 2009. № 2. С. 53–55.

3. OpenMAX Streaming Media Portability // Khronos Group – 2006. [Электронный ресурс]. URL: http://3dshaders.com/s2006/OpenMAX.pdf (дата обращения 10.03.2012).

4. Larabel M. Intel Sandy Bridge VA-API Video Acceleration Performance. Дата обновления 07.03.2011. URL: http://www.phoronix.com/scan.php?page=article&item=intel_snb_video&num=1 (дата обращения: 25.01.2011).

5. Hwdecode-demos. URL: http://gitorious.org/hwdecode-demos/hwdecode-demos (дата обращения: 10.04.2012).

6. OpenMAX Overview // OpenMAX - The Standard for Media Library Portability. Beaverton, Oregon, 2000. [Электронный ресурс]. URL: http://www.khronos.org/openmax/ (дата обращения: 13.01.12).

7. GstOpenMAX. freedesktop.org: open source / open discussion software projects. 2007. [Электронный ресурс]. Дата обновления 14.07.2010. URL: http://www.freedesktop.org/wiki/GstOpenMAX (дата обращения: 21.01.2012).

8. NEON - ARM. URL: http://www.arm.com/products/processors/technologies/neon.php (дата обращения: 04.04.2012).

9. Очкур С. В. Оптимизация ДКП-3D кодека с применением программного интерфейса OpenMAX DL на примере процессора ARM Cortex-A8 // 8-я международная конференция "Телевидение: передача и обработка изображений" 30–31 мая 2011 г., Санкт-Петербург. Мат-лы конференции. СПб.: Изд-во СПбГЭТУ "ЛЭТИ", 2011. С. ?–?.

10. Trevett N. Introduction to OpenMAX. Khronos Group Overview // SIGGRAPH 2004 – 2004. [Электронный ресурс]. Систем, требования: Power Point. URL: http://www.khronos.org/news/articles/siggraph_openmax_aug04.ppt (дата обращения: 13.01.12).

11. OpenMAX Implementation. URL: http://limoa.sourceforge.net/ (дата обращения: 04.04.2012).

12. Urlini G. The OpenMAX Integration Layer standard – 2006. [Электронный ресурс]. Дата обновления 27.02.2008. URL: http://elinux.org/images/e/e0/The_OpenMAX_Integration_Layer_standard.pdf (дата обращения: 05.02.2012).

13. OpenMAX AL Quick Reference 1.1. URL: http://www.khronos.org/files/openmax-al-1-1-quick-reference.pdf (дата обращения: 04.04.2012).

14. OpenGL - Wikipedia. URL: http://ru.wikipedia.org/wiki/OpenGL (дата обращения: 04.04.2012).

15. Katafiasz M. Multipurpose multimedia processing with GStreamer // developerWorks: IBM's resource for developers and IT professionals. URL: http://www.ibm.com/developerworks/aix/library/au-gstreamer.html (дата обращения: 25.01.2012).

16. GStreamer-vaapi. URL: http://gitorious.org/vaapi/gstreamer-vaapi (дата обращения: 04.04.2012).