САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

РЕАЛИЗАЦИЯ ВИДЕО ПОДСИСТЕМЫ ОС ANDROID С ИСПОЛЬЗОВАНИЕМ АППАРАТНЫХ ВОЗМОЖНОСТЕЙ СЕМЕЙСТВА МУЛЬТИМЕДИЙНЫХ ПРОЦЕССОРОВ TI OMAP35XX

Дипломная работа студента 544 группы

Елизарова Егора Алексеевича

Научный руководитель	..... /подпись/	главн.констр. ЗАО Ланит-Терком , к.ф.-м.н., доцент Фоминых Н.Ф.
Рецензент	..... /подпись/	ген.директор Embedded Alley Solutions Rus, входящей в группу компаний Mentor Graphics Иванов А.С.
"Допустить к защите" заведующий кафедрой,	..... /подпись/	д.ф.-м.н., профессор Терехов А.Н.

Санкт-Петербург
2010

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

IMPLEMENTATION OF ANDROID VIDEO SUBSYSTEM USING HARDWARE CAPABILITIES OF OMAP35XX MULTIMEDIA PROCESSORS FAMILY

by

Egor Elizarov

Master's thesis

Supervisor	.....	Chief Architect of Lanit-Tercom Corp., PhD, Ass. Prof. N.F. Fominykh
Reviewer	.....	Engineering Director of Embedded Alley Solutions Rus, owned Mentor Graphics company A.S. Ivanov
"Approved by" Head of Department	.....	PhD, Professor A.N. Terekhov

Saint Petersburg
2010

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня компьютерные магазины предлагают нам большой выбор разнообразных мобильных устройств от простейших монохромных телефонов до всевозможных нетбуков. Существует множество различных платформ для подобной техники. Но лидирующие роли сейчас занимают ОС от компаний Apple, Microsoft, Google, Nokia, Rim. В данной работе мы подробнее остановимся на операционной системе Android, являющейся частью одноименной платформы от компании Google.

Данная платформа была предложена компанией Google как универсальная платформа для смартфонов с открытым кодом. В настоящее время ее поддержкой занимается Open handset alliance[19] , в который помимо самого Google входят ведущие производители оборудования, операторы связи и другие крупные игроки на рынке мобильных устройств. К моменту написания данной работы в продаже уже появились смартфоны на базе ОС Android таких известных компаний, как HTC, Motorola, Samsung, Sony Ericsson, LG.

В наше время практически любое мультимедийное устройство должно уметь проигрывать видео. И если для просмотра любимого фильма или передачи, загруженной с компьютера, не потребуется дополнительная конвертация в другой формат, то устройство будет несомненно выигрывать у своих конкурентов, не обладающих такой возможностью. Для таких устройств, как телевизионные приставки и телевизоры, воспроизведение видео является основной функцией. Они должны уметь проигрывать видео высокой четкости, на высоких разрешениях. Некоторые устройства помимо основного маленького дисплея имеют выходы DVI или HDMI для подключения к внешнему монитору или телевизору. Задача проигрывания видео высокой четкости осложняется тем, что современные компьютеры и ноутбуки имеют довольно мощную аппаратную составляющую, в то время как мобильным устройствам не хватает ресурсов для ее решения. Компании, производящие аппаратные платформы,

предлагают для решения этих проблем разного рода аппаратные кодеки и ускорители. Большинство современных мобильных устройств на базе ОС Android оборудовано процессорами таких известных производителей, как Qualcomm, Samsung и Texas Instruments.

В работе речь пойдет о процессорах семейства OMAP 35xx от компании Texas Instruments[11]. Данные процессоры, построенные на базе ядра ARM Cortex A8, имеют специальное DSP ядро для ускоренной обработки видео и аудио файлов, а также изображений. Схожие DSP ядра компания Texas Instruments использует в семействе видео процессоров Da Vinci.

Разумно было бы использовать возможности этих процессоров в полной мере. Целью данной работы является внедрение поддержки аппаратного ускорения при проигрывании видео, в операционную систему Android. Хотелось бы, чтобы любое пользовательское приложение при попытке проигрывания видео в формате, поддерживаемом ускорителем, использовало ресурсы DSP ядра. Таким образом, задача не ограничивается лишь добавлением поддержки DSP в ядро Linux, необходимо изменение всей видео подсистемы ОС Android. Помимо этого, дополнительной трудностью является то, что платформа активно развивается и за последний год появились уже 3 новые версии, а значит при решении этой задачи необходимо помнить об упрощении дальнейшего обновления системы.

В качестве версии ОС Android выбрана версия с открытым кодом Rowboat[1], в разработке которой автор принимает участие. Она поддерживает два компьютера на базе процессора OMAP3530: OMAP3EVM и Beagleboard. ОС Rowboat Android построена на базе версии 1.6 («Donut») проекта Android Open Source Project от компании Google.

Работу над поставленной задачей можно разделить на четыре части.

Первая часть состоит в изучении способов работы с DSP ядром процессоров OMAP3530.

Вторая часть состоит в изучении платформы Android. Также в этой части необходимо разобраться в подсистеме ОС Android, отвечающей за воспроизведение видео.

В третьей части будут рассмотрены возможные варианты реализации аппаратного ускорения в ОС Android для данного семейства процессоров и более детально описан один из них.

И в качестве заключительного этапа будет рассмотрена реализация выбранного варианта в рамках проекта Rowboat.

ОБЗОР ОСНОВНЫХ ПОНЯТИЙ КОНТЕКСТА ЗАДАЧИ

Рассмотрим типовой случай воспроизведения видео файла. Обычно видео файл представляет из себя набор закодированных потоков, запакованный в некоторый контейнер. После чтения файла происходит распаковка контейнеров и выделение потоков (видео, аудио, субтитры), далее каждый из потоков декодируется и выводится на устройство вывода. Обычно каждый поток декодируется в отдельном процессе. Синхронизация между потоками происходит по специальным контрольным точкам, за основу берется аудио поток. Для упрощения написания программ существует множество различных библиотек, изолирующих от программиста такие моменты, как распаковка контейнеров, декодирование, вывод. В ОС Android для пользовательских приложений предоставляется высокоуровневый интерфейс Android API, а все детали реализации скрыты внутри самой ОС. Например, для работы с видео программисту предлагаются такие примитивы: задать путь до файла, приготовиться к воспроизведению, проиграть, приостановить, остановить, установить громкость звука, перемотать на позицию и т.д.

Изначально платформа Android была разработана для процессоров без дополнительных аппаратных возможностей по обработке видео. В существующей схеме декодирование происходит на основном ядре процессора, в то время как DSP ядро бездействует. На рисунке 1 сплошными линиями схематично изображен текущий процесс работы пользовательского видео плеера. Пунктиром выделены компоненты, которые необходимо добавить, а пунктирными стрелками показано, как должен проходить тот же процесс по окончании данной работы.

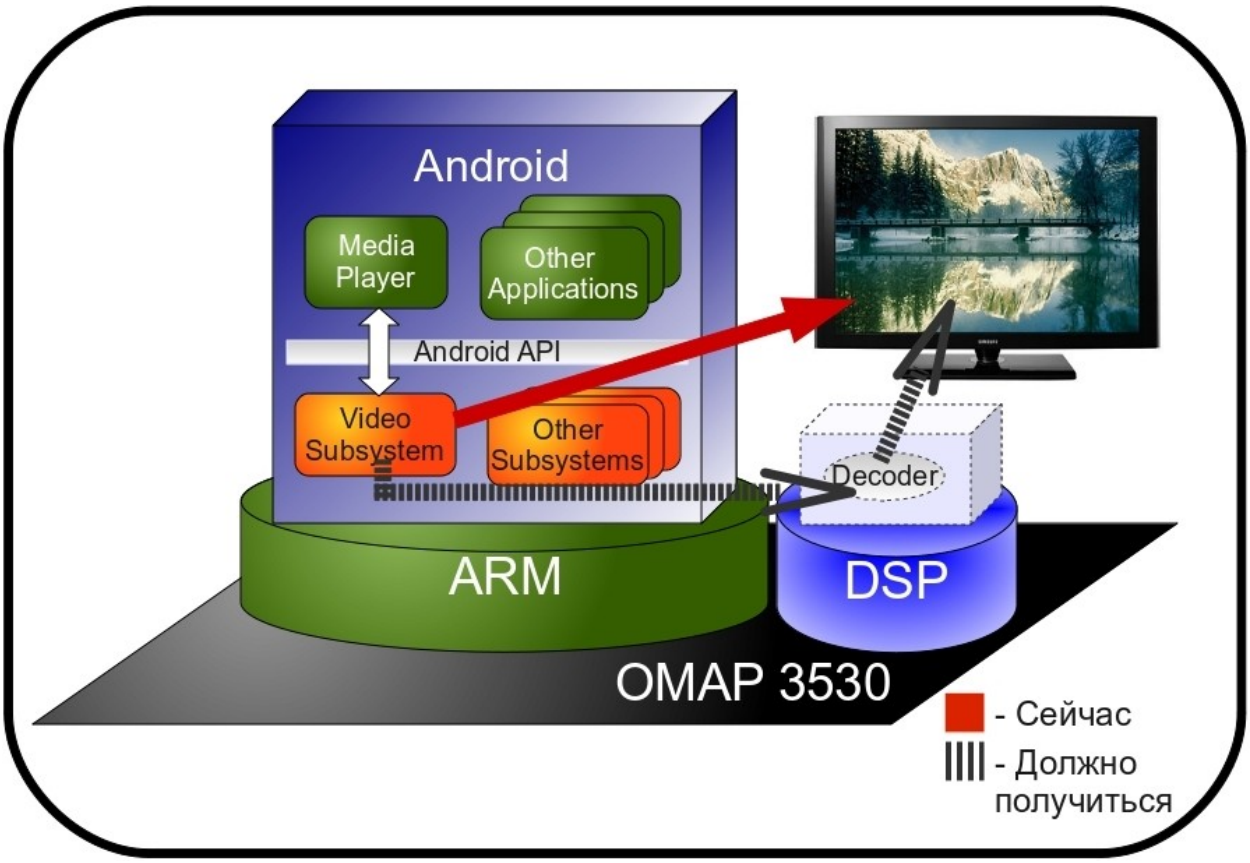

Рисунок 1: Работа видео плеера в ОС Android

Процессоры OMAP 35xx.

Далее представлен обзор семейства мультимедийных процессоров OMAP35xx, а так же рассматрены способы реализации аппаратного ускорения в ОС Linux. В конце данной части рассказывается о наборе кодеков с поддержкой аппаратного ускорения для мультимедийной библиотеки Gstreamer.

Аппаратные особенности

Линейка мультимедийных процессоров OMAP35xx построена на базе ARM ядра Cortex A8 с тактовой частотой до 720 мГц, работающего с набором инструкций ARMv7.[5] Кроме того для оптимизации работы с векторными операциями это ядро содержит сопроцессор NEON SIMD. На процессорах OMAP3530 и OMAP3515 для разгрузки основного ядра при работе с двухмерной и трехмерной графикой также установлены графические

ускорители POWERVR SGX, которые позволяют работать с такими программными интерфейсами, как OpenGLES 1.1 и 2.0.

Отличительной особенностью процессоров OMAP3525 и OMAP3530 является дополнительное подсистема для ускоренной обработки изображений, видео и аудио IVA. Система включает в себя DSP ядро TMS320C64x+, поддерживающее частоту до 520 мГц, расширенную систему прямого доступа к памяти (DMA), состоящую из 128 независимых каналов, а также набор аппаратных видео ускорителей. В дальнейшем в работе будут рассматриваться только процессоры с системой IVA, а все примеры будут приведены на базе процессоров OMAP3530.

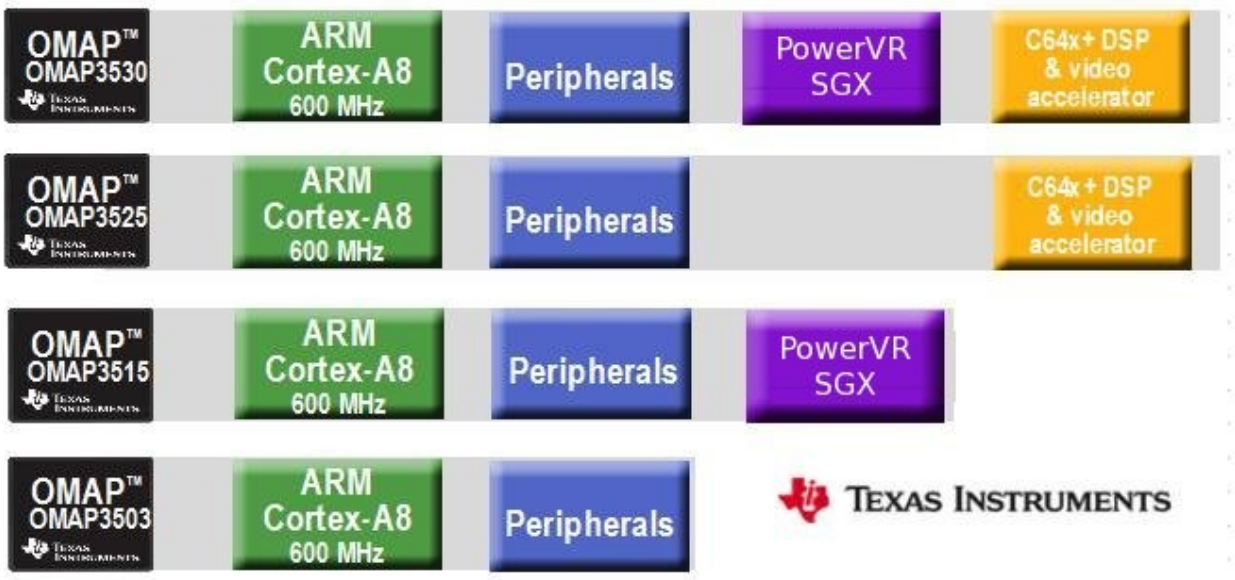

Рисунок 2: Линейка мультимедийных процессоров OMAP35xx

Видео подсистема этих процессоров позволяет производить различные операции с изображением, такие как повороты на 90/180/270 градусов и изменение размеров. Подсистема поддерживает интерфейсы для работы с различными типами дисплеев.

В рамках данной работы будут рассматриваться два компьютера на базе OMAP3530:

BeagleBoard[22] — компьютер, появившийся как часть открытого проекта по созданию мощных встраиваемых устройств. Он оснащен интерфейсом HDMI для подключения монитора.

OMAP3EVM[13] - компьютер разработанный компанией Mistral. В устройстве помимо встроенного сенсорного дисплея с разрешением 640x480 используется порт DVI для подключения внешнего дисплея.

Согласно документации на процессоры OMAP3530, они поддерживаются такими операционными системами, как Windows CE и Linux. Интерес для работы представляет поддержка аппаратного ускорения при воспроизведении видео на устройствах, оборудованных данным процессором в ОС Linux, так как ОС Android построена на базе этого ядра. Будет рассматриваться версия ОС Linux Angstrom[8], в которой поддержка DSP уже реализована. Кроме этого в сети можно найти множество проектов, в которых в ОС добавляется поддержка процессоров семейства OMAP 3. Подробнее остановимся на каждом из них.

Rowboat[1] Об этом проекте уже упоминалось выше. К моменту написания работы поддерживалась версия Android 1.6 (Donut) на двух компьютерах OMAP3EVM и BeagleBoard без поддержки аппаратного ускорения.

Embinux[16] Целью данного проекта является запуск ОС Android на компьютерах на базе процессоров OMAP3530. В настоящее время поддерживаются такие компьютеры, как BeagleBoard и OMAP3EVM. Здесь не реализована поддержка аппаратного ускорения при воспроизведении видео. Проект основан на версиях 1.5 (Cupcake), 1.6 (Donut) и 2.1 (Eclair) ОС Android.

0xdroid[15] Его целью является поддержка компьютеров Beagleboard в ОС Android, он построен на базе версии 1.6 (Donut) ОС Android. К началу

написания этой работы аппаратное ускорение при проигрывании видео не поддерживалось.

Ingenient[9] Компания Ingenient Technologies Inc специализируется на создании видео кодеков для процессоров с поддержкой DSP от Texas Instruments, поэтому в проекте поддерживается аппаратное ускорение при воспроизведении видео. В данном случае в ОС Android добавлена поддержка компьютеров Beagleboard. Однако проект является коммерческим с закрытым кодом.

Omapzoom[18] Его целью служит поддержка устройств на базе схожих процессоров OMAP3430 в ОС Android версии 1.6 (Donut) и 2.1 (Eclair). Здесь реализовано аппаратное ускорение при проигрывании видео. Однако ввиду особенностей реализации использовать это решение в полной мере на процессорах OMAP35xx не удается, о чем будет рассказано позже.

Среди перечисленных проектов только три поддерживают устройства на базе процессоров OMAP35xx и имеют открытый код. Но, к сожалению, ни в одном из них процессор не задействован в полном объеме: не используется DSP ядро при проигрывании видео файлов. Как обсуждалось ранее, возникающая проблема является актуальной для платформы Android.

Поддержка DSP в ОС Linux

Компания Texas Instruments предлагает три способа работы[4] с DSP ядром:

DSP-Gateway Решение предложенное компанией Nokia для мобильных планшетов на базе платформы MAEMO. Код для ядра Linux и операционная оболочка для DSP ядра распространяются в открытом виде. В настоящее время данное решение поддерживается для платформ OMAP1 и OMAP2 и реализовано в продуктах N800 и N810 от компании Nokia. Также заявлена

поддержка процессоров семейства OMAP3, но код нестабилен. В настоящее время разработки по данному проекту не ведутся.

DSP-Bridge Решение, используемое в проекте omapzoom. Появилось по инициативе Texas Instruments, в настоящее время данный проект поддерживается TI и Nokia. Код для ядра Linux распространяется в открытом виде, в то время как код оболочки DSP закрыт. В отличие от предыдущего, на базе данного решения построено множество пользовательских приложений, к примеру: gst-openmax и gst-goo. Недостатком же его является то, что оно поддерживается лишь на небольшом количестве аппаратных платформ.

DSP-Link Данное решение, разработанное TI, охватывает множество платформ, среди которых Da Vinci, OMAP2, OMAP3. Существует плагин для GStreamer , построенный на его базе. Именно оно поддерживается в Angstrom Linux.

В данной работе более детально будет рассматриваться последнее решение, так как именно в нем есть поддержка процессоров OMAP35xx. В случае Angstrom Linux оно используется как часть DVSDK.

DVSDK (Digital Video Software Development Kit)[21] представляет собой набор инструментальных средств для создания мультимедийных приложений с поддержкой аппаратного ускорения на процессорах семейства Da Vinci, а также OMAP35xx в ОС Linux. DVSDK избавляет программиста от необходимости непосредственно работать с DSP ядром предоставляя высокоуровневый интерфейс DMAI (DaVinci Multimedia Interface). Так же существует специальная библиотека Codec Engine Multimedia stack, которая предоставляет интерфейс для написания различного рода кодеков и парсеров, которые в последствии будут исполняться на DSP ядре. Для процессоров OMAP35xx на базе Codec Engine предоставляется набор кодеков Codec Server, в который входят декодеры H264, MPEG4, JPG, AAC.

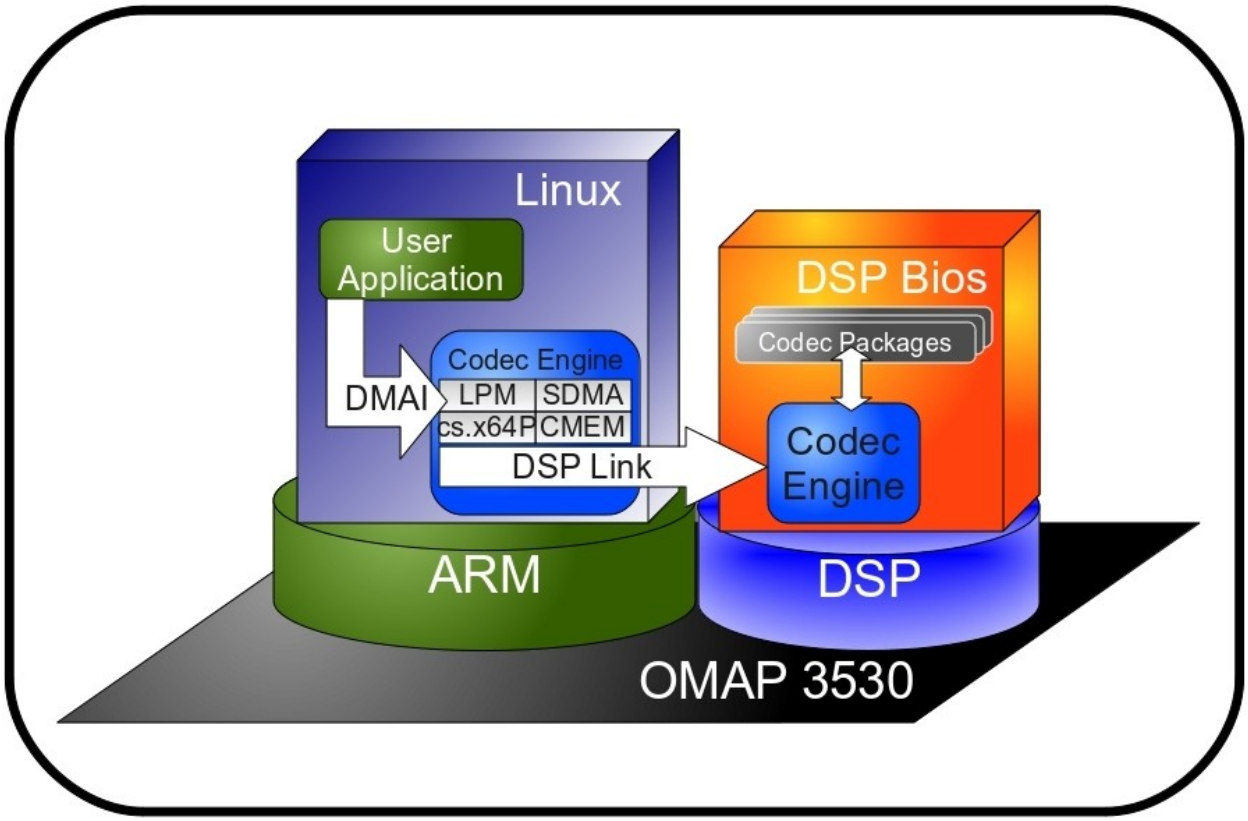

Рисунок 3: Взаимодействие ARM и DSP ядер

Архитектура решения на базе DSPLink представлена на рисунке 3 и состоит в следующем: на DSP ядре запущена операционная оболочка под называнием DSP BIOS, в то время как на ARM ядре запущен Linux с набором встроенных модулей ядра. Управление DSP происходит с помощью модуля ядра под названием dsplink. Данные передаются через разделяемую память, при помощи модулей CMEM и SDMA. Модуль LPM (Local Power Managment), позволяет ядру Linux управлять питанием DSP ядра. (При отсутствии этого модуля работа с DSP ядром не поддерживается.) Кодеки, входящие в набор Codec Server исполняются на DSP ядре при помощи Codec Engine. Также благодаря прямому доступу к памяти (DMA), Codec Engine позволяет осуществлять вывод изображения напрямую, без вмешательства ARM ядра.

Весь набор для разработки DVSDK собирается при помощи специальной билд системы от компании TI, построенной на базе XDC (eXpress DSP Components).

XDC[6] — стандарт для предоставления программных компонент, называемых пакетами, для встроенных операционных систем реального времени. Преимуществом использования XDC является то, что все компоненты платформенно-независимы, настройка под конкретную платформу осуществляется на скриптовом языке. Кроме того, поскольку все пакеты имеют одинаковый интерфейс, упрощается построение приложений на базе нескольких различных пакетов. Данная билд система отличается от билд системы Android.

Плагины для GStreamer

Как упоминалось выше, существуют построенные на базе DSP-Link плагины для GStreamer[20]. Данный плагин был разработан фирмой RidgeRun[10] по заказу Texas Instruments.

GStreamer[7] — кросс платформенная мультимедийная библиотека с открытым кодом, являющаяся базовой в оконном менеджере GNOME в ОС Linux. Она служит для построения пользовательских приложений, скрывает внутри себя кодеки, фильтры, платформенно-зависимый ввод и вывод, предоставляя унифицированный программный интерфейс. Данная библиотека представляет из себя набор элементов, соединяемый в специальный канал, каждый элемент которого выполняет некоторое действие с потоком данных. Элементы делятся на: источники (отвечают за получение данных), декодеры (отвечают за декодирование), демультиплексоры (отвечают за распаковку контейнеров), приемники (отвечают за вывод данных), конвертеры (отвечают за конвертирование данных). Кроме того, GStreamer предоставляет набор мета-элементов, таких как playbin (данные элементы автоматически собирают различные элементы в канал и

обрабатывают данные). Плагины для GStreamer представляют из себя наборы из подобных элементов.

В плагин TI Gstreamer[2,3] входят декодеры для аудио, видео и изображений, а так же элемент для вывода на экран. Одна из целей написания данного плагина состояла в том, чтобы максимально эффективно использовать DSP ядро. Для реализации аппаратного ускорения в данном проекте используется DMAI. На рисунке 4 приведена общая схема работы приложения с использованием TI GStreamer плагина.

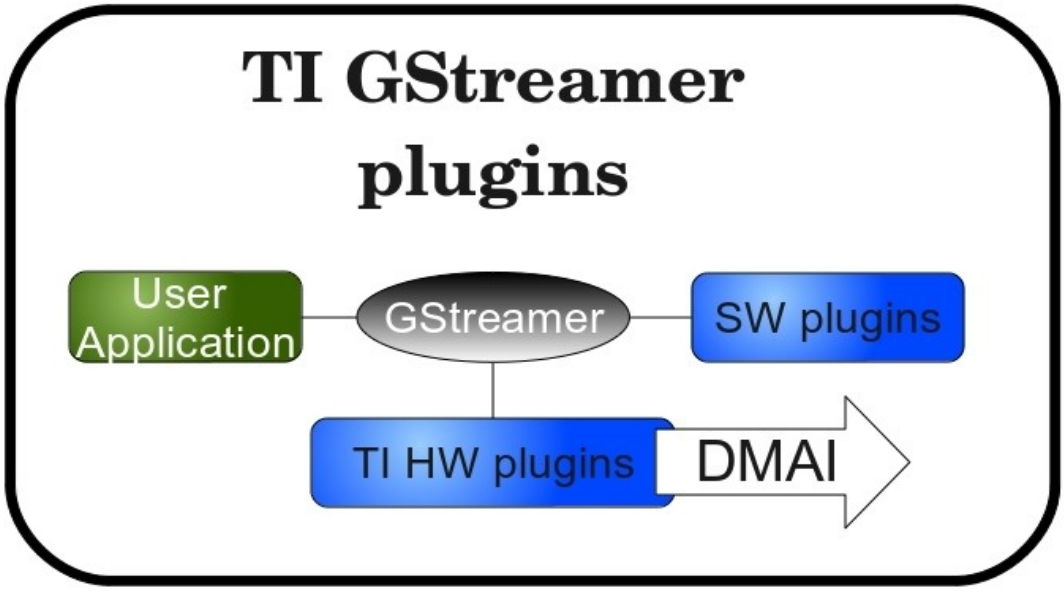

Рисунок 4: Аппаратно ускоренное проигрывание видео с использованием плагина TI GStreamer

Данный плагин предоставляет возможность работы с форматами, поддерживаемыми DVSDK.

Платформа Android

В этой части будет рассмотрена платформа Android[14], а так же более подробно будет разобрана видео подсистема ОС Android. В завершении будет рассказано о проекте в котором вместо мультимедиа библиотеки OpenCore используется GStreamer.

Общие сведения о платформе

Android - это программный стек для мобильных устройств, который включает в себя операционную систему, связующее ПО и ключевые приложения. При разработке приложений для данной платформы на языке Java используется специальное набор программ для разработки (SDK). Android API , являющийся его частью, предоставляет программный интерфейс для работы с ОС и программной инфраструктурой.

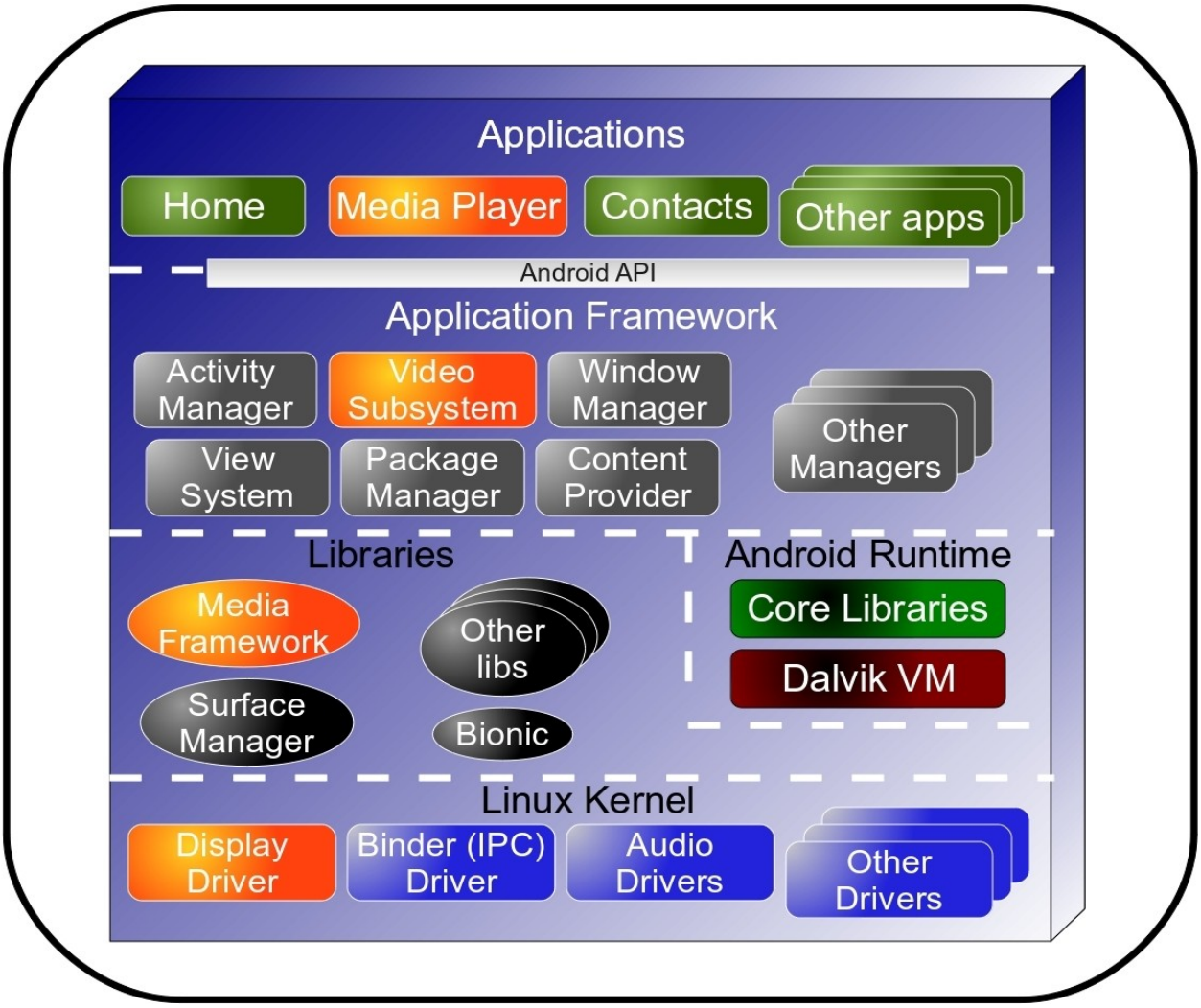

Рисунок 5: Архитектура ОС Android

В данном стеке (рис 5), мы можем выделить несколько основных частей: в основе лежит ядро Linux, далее расположены низкоуровневые библиотеки и виртуальная машина Dalvik. Далее можно выделить различные

библиотеки (OpenCore, Surface manager, WebKit, OpenGL и т.д.). На уровень выше стоят компоненты, которые реализуют фрэймворк для приложений (Media system, View System и т.д.). И последняя группа - это непосредственно приложения(Phone, Browser, Video Player).

Пользовательские приложения исполняются на виртуальной машине Dalvik, при этом некоторые вызовы при помощи интерфейса JNI (Java Native Intarface) вызывают библиотеки, написанные на C/C++. При помощи таких механизмов получается добиться переносимости на различные платформы, а также модульности, при которой некоторые компоненты могут заменяться. Вместо стандартной для Linux систем библиотеки libc в ОС Android используется библиотека Bionic, оптимизированная для данной платформы.

Билд система Android основана на GNU Make. В корневом каталоге дерева исходных фалов находятся директории содержащие исходные коды bionic (bionic), виртуальной машины (dalvik) и др. Особо интересными для данной работы являются директории external и frameworks. В первой хранятся внешние библиотеки, написанные на C/C++, обращение к ним происходит со стороны фрэймворков, код которых хранится во второй директории.

Видео подсистема

Для проигрывания видео в ОС Android используется высокоуровневый программный интерфейс для языка Java. В пользовательское приложение должен быть импортирован класс MediaPlayer, который при помощи JNI вызывает методы одноименного класса, написанного на C++, который в своей реализации вызывает методы одного из классов наследников виртуального класса MediaPlayerInterface. Классы наследники представляют из себя медиа плееры, реализующие высокоуровневый программный интерфейс (начать проигрывание, остановить, перемотать). Выбор плеера осуществляется в библиотеке libmediaplayservice. В версии ОС Android от

компании Google для проигрывания видео используется медиа плеер PVPlayer на базе библиотеки Opencore от компании Packet Video. Код данной библиотеки в дереве исходных файлов находится в каталоге external. Помимо PVPlayer для некоторых типов аудио файлов используются Sonivox player и Vorbis player. На рисунке 6 представленна упрощенная модель видео подсистемы ОС Android.

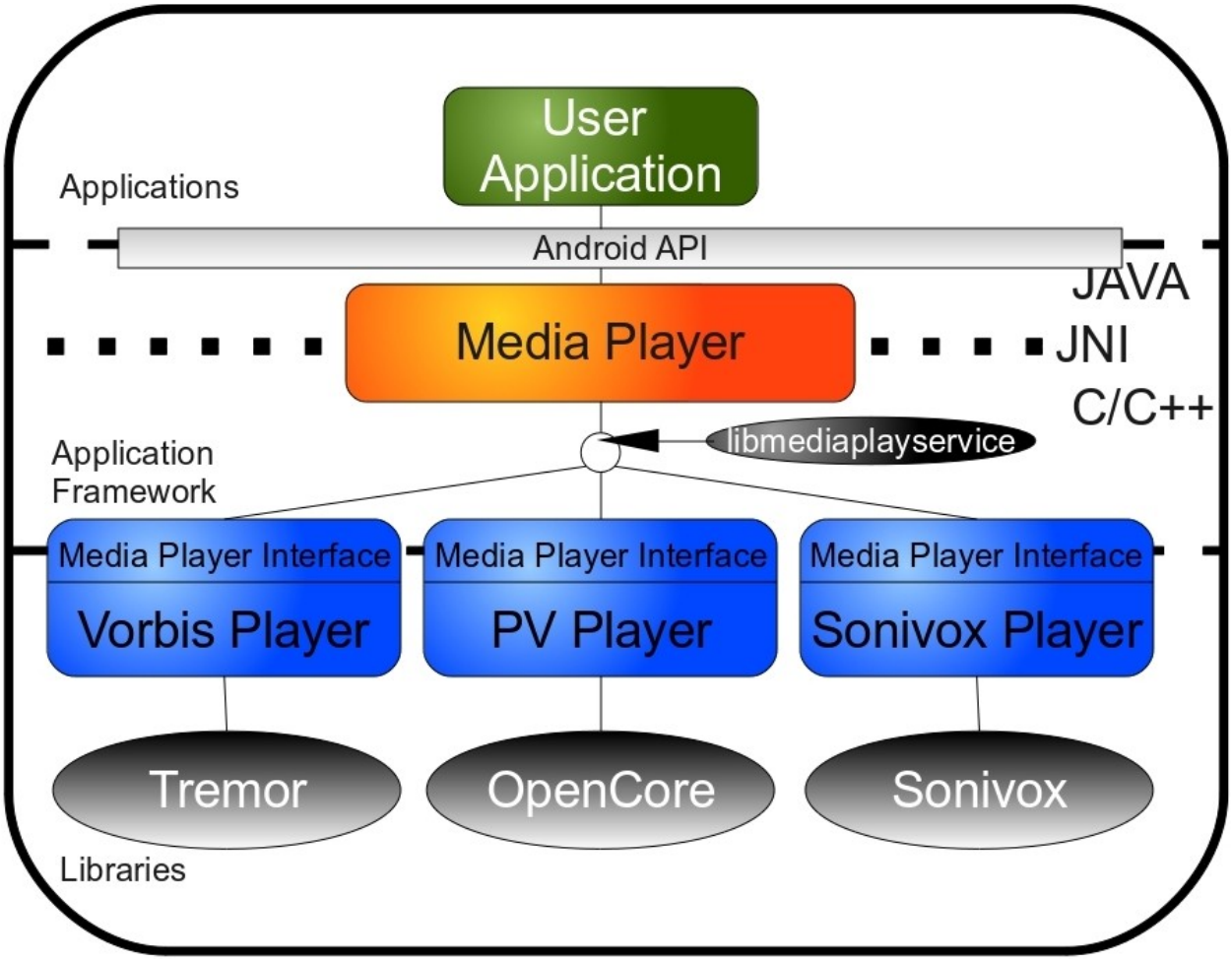

Рисунок 6: Видео подсистема ОС Android

Архитектура видео подсистемы ОС Android позволяет довольно просто заменять библиотеку, которая отвечает за проигрывание видео.

Android и GStreamer

Существует проект GstDroid[17], в котором в ОС Android вместо плеера на базе библиотеки OpenCore используется плеер, написанный на

основе библиотеки GStreamer. Для этого в билд систему Android в external были добавлены библиотека GStreamer и набор плагинов к ней. Кроме того, был создан дополнительный плагин, в котором были реализованы два элемента, отвечающие за вывод видео и звука в ОС Android, а также медиа плеер GstPlayer, реализующий интерфейс класса MediaPlayerInterface. Для построения канала в данном случае используется мета элемент playbin2. MediaPlayerService был изменен таким образом, чтобы в случае наличия конфигурационного файла для GstPlayer на SD карте использовался плеер на базе библиотеки GStreamer. В данном проекте нет поддержки OMAP35xx и ОС Android запускается на ARM эмуляторе.

Существующие проекты

Подводя итог, выделим наиболее интересные части описанных выше проектов. Рис 7.

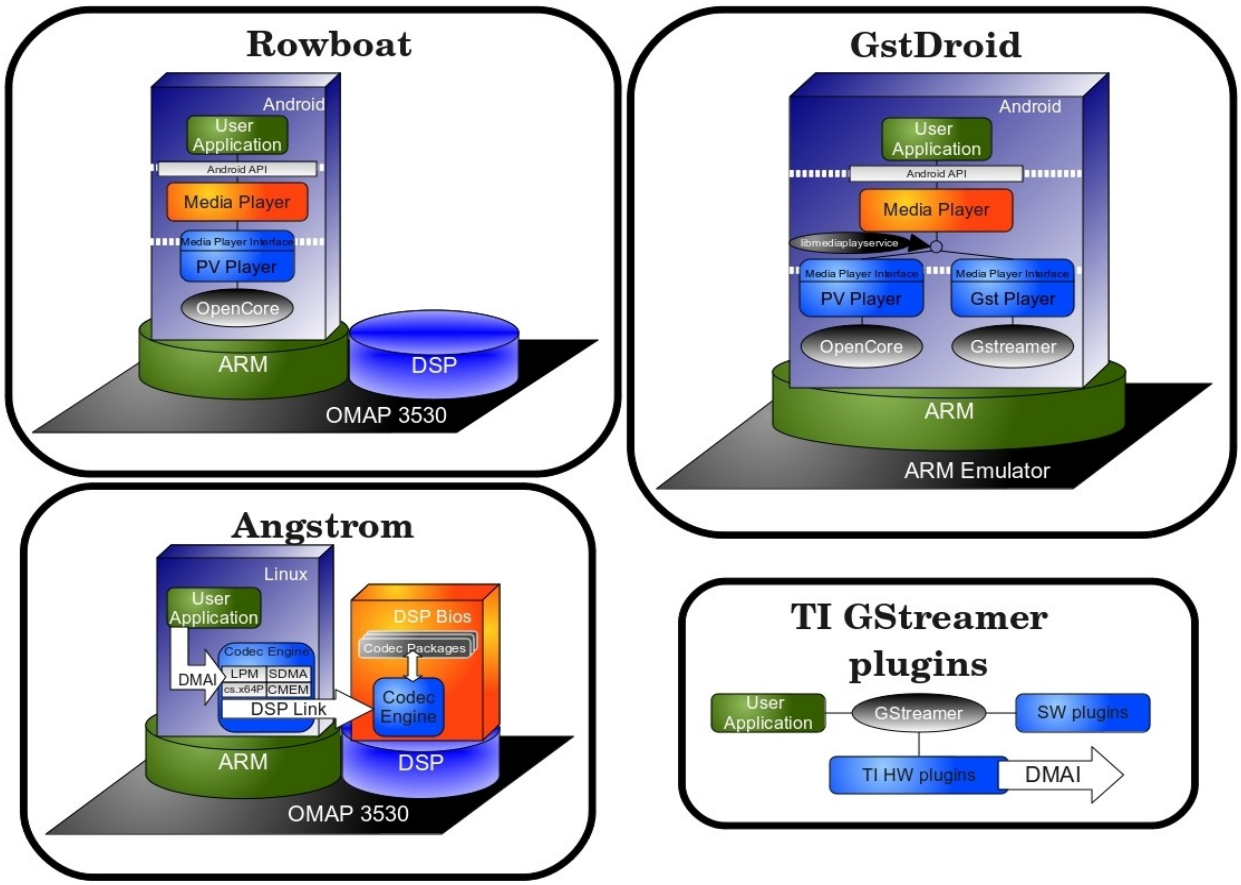

Рисунок 7: Существующие проекты

ПОСТАНОВКА ЗАДАЧИ

Как обсуждалось выше, для работы с DSP ядром будет использоваться DSP Link, работа с которым осуществляется через программный интерфейс DMAI. Что касается ОС, то необходимо сохранить без изменения программный интерфейс Android API. Самым оптимальным подходом здесь будет использование интерфейса класса Media Player Interface. В таком случае классы, реализующие Android API, останутся без изменений. Таким образом необходимо реализовать интерфейс класса Media Player Interface с использованием DMAI (рисунок 8) . При работе с форматами, не поддерживаемыми Codec Server, для декодирования должны использоваться ресурсы основного ядра.

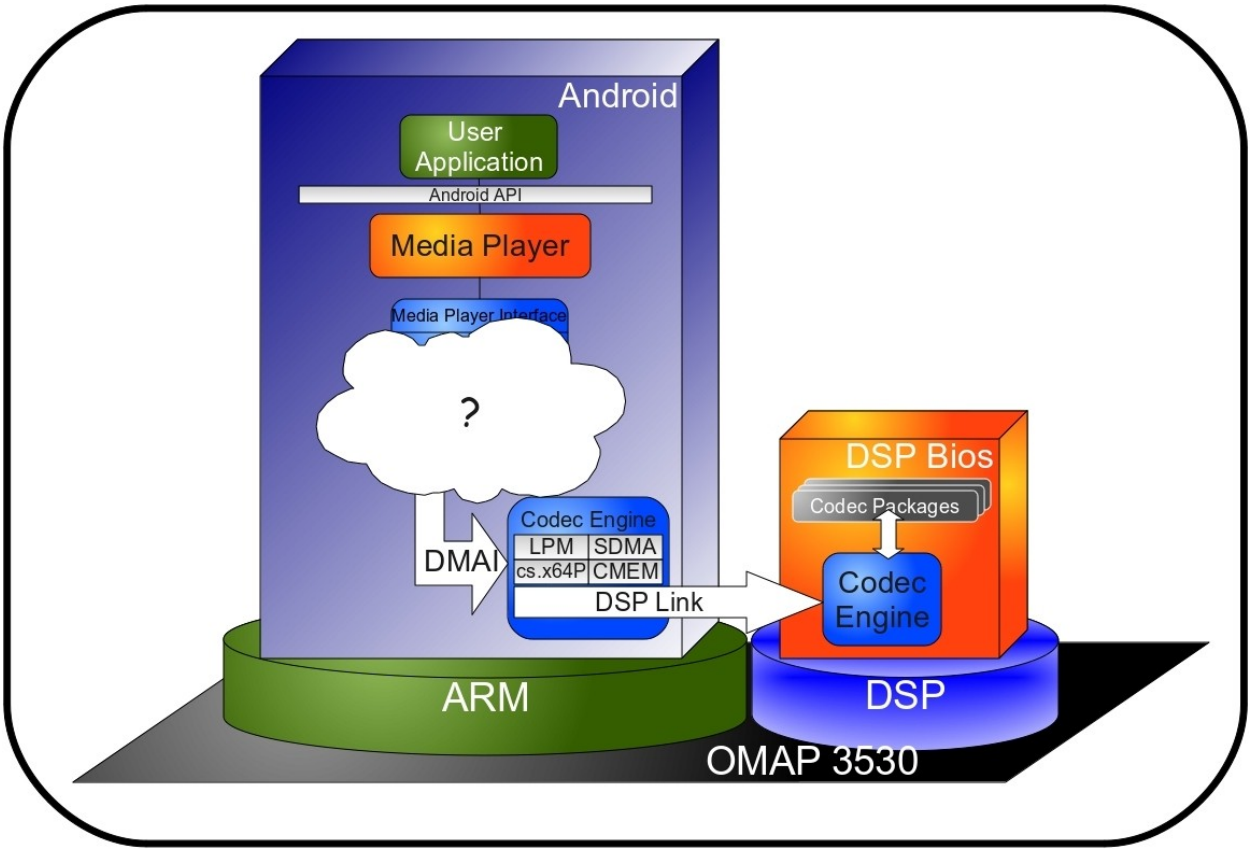

Рисунок 8: Место встраивания компонент новой видео подсистемы

ВОЗМОЖНЫЕ СПОСОБЫ РЕШЕНИЯ ЗАДАЧИ

Рассмотрим подходы к решению данной задачи:

1) Написать плеер, реализующий интерфейс класса MediaPlayerInterface, который в своей реализации будет использовать DMAI для работы с DSP ядром.
2) Добавить поддержку DSP ядра в OpenCore, реализовав набор кодеков, использующих DMAI.
3) Заменить PV Player на Gst Player и добавить аппаратные кодеки TIGStreamer, которые будут работать с DSP ядром через интерфейс DMAI.

Первый вариант является весьма объемным, так как в данном случае необходимо будет самостоятельно реализовывать распаковку контейнеров, выделение отдельных потоков, их синхронизацию и т.д. А также, для форматов, не поддерживаемых Codec Server, придется реализовывать программные декодеры. Кроме того, в отличие от других решений, не будет использовать уже существующие протестированные и поддерживаемые плееры.

Преимуществом второго метода является тот факт, что библиотека OpenCore была специально оптимизирована для работы с мобильными устройствами.

В отличии от OpenCore, на данный момент библиотека GStreamer позволяет использовать большее число форматов. Проект GStreamer является развивающимся, поэтому поддержка новых кодеков и форматов появляется довольно быстро, а добавление их в Android становится довольно простой задачей. Кроме того, как упоминалось выше, уже существует набор кодеков от TI, поддерживающих аппаратное ускорение с использование DSPLink. Для

OpenCore в данном случае необходимо будет написать новый набор кодеков на базе DMAI.

Итак, преимущество последних подходов над первым очевидно. Что касается выбора между вторым и третьим, то в данной работе было решено использовать подход на базе GStreamer, так как он является более гибким и использует большее число уже готовых протестированных и поддерживаемых компонент. Таким образом, помимо решения задачи с аппаратным ускорением проигрывания видео, удастся несколько расширить возможности видео подсистемы ОС Android, добавив множество декодеров и демультиплексоров не поддерживающихся существующей видео подсистемой.

АРХИТЕКТУРА РЕАЛИЗОВАННОГО РЕШЕНИЯ

В данной части будет более детально описана архитектура выбранного решения (рисунок 9).

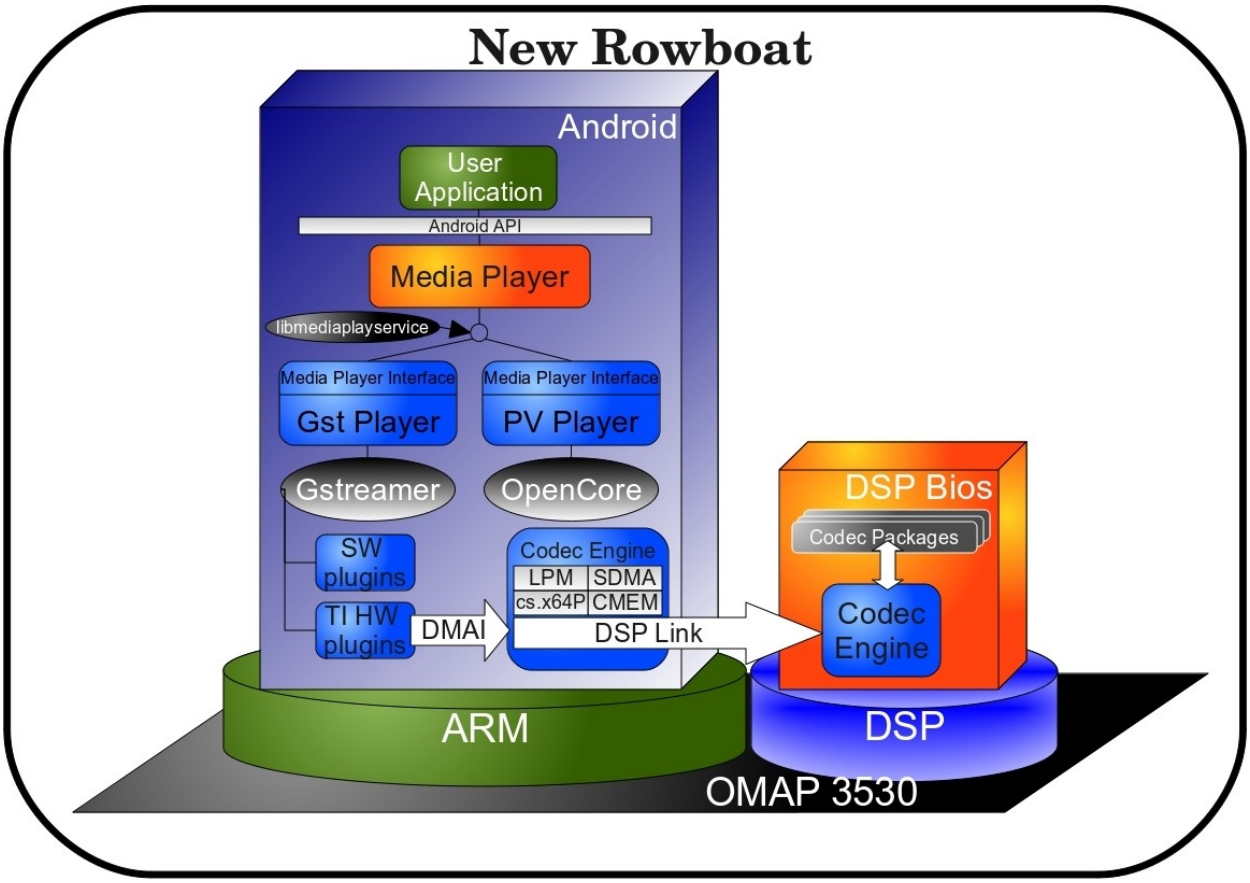

Рисунок 9: Архитектура предложенного решения

Пользовательское приложение, написанное на языке Java с использованием Android API, обращается к методам класса MediaPlayer. Данный класс при помощи JNI и некоторых вспомогательных классов обращается к GstPlayer. Этот плеер, построенный на базе библиотеки GStreamer и реализующий программный интерфейс класса Media Player Interface, распознает формат входного файла, и в случае обнаружения потоков, декодирование которых поддерживается Codec Server, он использует TI GStreamer кодеки, которые через интерфейс DMAI взаимодействуют с Codec Engine. Таким образом, декодирование происходит на DSP ядре. Вывод видео потока на экран производится через V4L2

устройство при помощи DMA, для этого используется элемент вывода для GStreamer, входящий в состав плагина от TI. В случае потоков, аппаратное декодирование которых не поддерживается GstPlayer должен использовать обычные плагины GStreamer. Синхронизация потоков обеспечивается с задействованием стандартных механизмов библиотеки GStreamer.

При реализации данного решения будут использоваться различные части проектов описанных выше. На рисунке 7 выделены наиболее важные из них. Как говорилось раньше, основная работа будет вестись на базе проекта Rowboat, в который уже добавлена поддержка двух компьютеров на базе процессора OMAP3530. Добавление библиотеки GStreamer в ОС Android, а также образец реализации GstPlayer предлагается взять из проекта Gstdroid. Далее, согласно предложенной архитектуре, в ОС Android будут внедрены кодеки с поддержкой аппаратного ускорения из проекта TI GStreamer Plugins. Низкоуровневая часть, включающая в себя оболочку для DSP ядра, DSP Link, ядерные модули, а также DMAI будет взята из проекта Angstrom.

РЕАЛИЗАЦИЯ В РАМКАХ ПРОЕКТА ROWBOAT

Реализация данного решения распадается на несколько задач:

1) Добавить DVSDK в билд систему Android.
2) Добавить GStreamer и основные плагины в билд систему Android.
3) Добавить аппаратные плагины TI GStreamer plugins в билд систему Android.
4) Написать Gst Player (на базе решения Gstdroid), поддерживающий правильный выбор плагинов при воспроизведении видео/аудио
5) Изменить медиаплеер, используемый по умолчанию в ОС
6) Добавить автоматическую загрузку модулей ядра и настроить окружение ОС

Далее каждая из поставленных задач будет раскрыта более подробно.

Для решения первой задачи необходимо встроить исходные коды DVSDK, в билд систему Android. Дополнительным условием в данном пункте является простота обновления до новой версии DVSDK. Для этого изменения, вносимые в билд систему TI XDC, должны быть минимальны.

Исходные коды Android хранятся в git репозиториях и загружаются при помощи специальной утилиты repo. При этом для хранения промежуточных файлов, появляющихся при сборке, а также уже готовых исполняемых файлов и библиотек используется выделенный католог out. Таким образом, в билд системе Android исходные коды хранятся отдельно от объектных файлов.

Билд система XDC состоит из пакетов, которые загружаются с сайта TI. Некоторые из них требуют регистрации. Сборка при этом происходит в каталогах, содержащих исходный код. Кроме того, на этапе

конфигурирования пакета, билд системе необходимо предоставить полные пути до ядра и других пакетов.

Таким образом, с целью минимизировать изменения DVSDK решено было поступиться принципом раздельного хранения кода и исполняемых файлов, а сам процесс сборки DSP стека выделить в отдельный этап, следующий после сборки файловой системы ОС и ядра. В директории для хранения внешних библиотек и приложений external билд системы Android должен быть заведен отельный каталог ti-dsp для хранения и сборки пакетов. При этом перед общей сборкой системы пользователь должен положить набор пакетов от TI в данных каталог. Если же это условие не будет выполнено, то система должна загрузить пакеты, предоставляемые в отрытом доступе, и сообщить о необходимости добавления оставшихся. После загрузки пакетов система должна установить их и применить к ним набор патчей. Далее должен запускаться стандартный процесс конфигурирования и сборки билд системой XDC. По завершению работы билд системы XDC, полученные ядерные модули и библиотеки должны быть установлены в собранный на начальных этапах образ файловой системы Android. Набор патчей включает в себя некоторые изменения, связанные с особенностями реализации, о которых будет сказано ниже, а также те изменения, которые отвечают за получение правильных путей до ядра и других пакетов из переменных окружения.

При решении задачи добавления библиотеки GStreamer необходимо расположить исходный код самой библиотеки, а также ее плагинов в каталоге для хранения внешних библиотек external. Кроме того, для каждой библиотеки необходимо добавить Android.mk файл, в котором описан процесс сборки данного модуля. Помимо уже существующих плагинов, нам понадобятся еще два, отвечающих за вывод аудио и видео в ОС Andriod. На этом этапе в основном будут использоваться результаты, уже полученные в

проекте GstDroid. Для проверки работы GStreamer в данной точке будет использоваться консольный плеер gst-launch. Эта утилита позволяет создавать каналы из элементов GStreamer путем последовательного указания их имен. Перед её использованием необходимо будет задать переменную окружения, указывающую на место хранения плагинов GStreamer.

При работе над третьей задачей плагин от TI для GStreamer должен быть добавлен в католог для хранения внешних библиотек. Но поскольку данный плагин при сборке требует наличие установленной версии DVSDK, то компиляцию плагина решено было производить на этапе сборки DVSDK. Исходный код, лежащий в svn репозитории, будет загружаться и устанавливаться при помощи скриптов, реализованных на первом этапе. Проверка результатов данного этапа, а также проверка того, что DVSDK установлена корректно, также будет производиться при помощи gst-launch. При этом GStreamer элементы, отвечающие за аппаратное декодирование видео, подключаются к специальному выходному элементу, входящему в комплект TI GStreamer plugins. Благодаря этому вывод на экран происходит при помощи DMA, без участия ARM ядра. Для проверки работоспособности DSP стека перед его использованием необходимо загрузить модули ядра, собранные на первом этапе.

Решение первыч трех задач, по сути, завершает низкоуровневые работы по внедрению аппаратного ускорения в ОС Andriod. На данной точке можно проигрывать видео с использованием DSP ядра при помощи консольного плеера gst-launch. Следующие два этапа отвечают за связывание Android API и GStreamer.

Четвертая задача подразумевает добавление плеера, построенного на основе GStreamer и использующий Gst TI плагины. Данный плеер должен реализовывать программный интерфейс MediaPlayerInterface. За основу берется GstPlayer из проекта Gstdroid. Далее плеер должен быть изменен

таким образом, чтобы при обнаружении потоков, аппаратное декодирование которых поддерживается Codec Server-ом, в качестве декодирующих элементов выбирались бы элементы из набора GST TI. В оригинальной версии GstPlayer используется метаэлемент playbin2, который самостоятельно распознает потоки и конструирует канал для проигрывания файла. При выборе элементов он пользуется информацией о типе входных и выходных данных каждого из них. Предпочтения при выборе элемента отдаются тем модулям, приоритет которых выше (данный приоритет жестко задан в коде). Таким образом, одним из решений является изменение приоритетов всех программных кодеков на более низкий, однако в данном случае не удается напрямую соединить декодирующий элемент с выводящим ввиду особенностей playbin2 (между выводящим элементом и декодирующим всегда встраивается конвертирующий, отвечающий за приведение цветовой схемы). Решено было воспользоваться другим метаэлементом: decodebin2, который автоматически подбирает нужные распаковщики контейнеров и декодеры и выдает набор декодированных потоков. При работе данного элемента используется динамическое связывание: в момент, когда появляется новый декодированный аудио или видео поток, порождается событие, в обработчике которого должны быть созданы и подключены к каналу соответствующие элементы. Благодаря этому удается добиться прямого соединения элемента, отвечающего за аппаратное декодирование, с выводящим элементом из набора GST TI. Для корректного определения декодера в данном решении используется следующий механизм: в момент определения нового элемента decodebin2 формирует очередь согласно приоритетам, описанным выше и вызывает некоторую функцию, которая решает, включать ли данный элемент в текущий канал. Как только какой-нибудь элемент из очереди выбран для включения в канал, обработка очереди прекращается. По умолчанию эта функция предлагает включать любой элемент, но ее можно переопределить.

Таким образом, получив в текущей точке программы тип входных данных и название элемента, из очереди может быть выбран необходимый декодер. Помимо описанной вспомогательной функции, а также обработчиков для выходных потоков, при реализации с помощью decodebin2 должны быть описаны элементы, отвечающие за входные потоки. (В случае реализации на базе playbin2 данное описание скрыто внутри самого элемента.)

Пятая задача сводится к изменению кода библиотеки MediaPlayService, отвечающей за выбор медиаплеера. На этом этапе необходимо снова воспользоваться результатами проекта GstDroid. Для того чтобы в качестве основного использовался GstPlayer, файл с настройками gst.conf должен быть помещен на sd карту. Если он не будет найден, то в качестве основного плеера предлагается использовать PVPlayer.

Работа над последней задачей заключается в том, чтобы автоматизировать загрузку модулей ядра, а также установить правильные переменные окружения для GStreamer во время загрузки системы. Для этого необходимо добавить соответствующий набор инструкций в загрузочные скрипты ОС Android.

ОСОБЕННОСТИ РЕАЛИЗАЦИИ

В ОС Android вместо стандартной библиотеки libc используется библиотека bionic, которая несколько отличается от нее. К примеру, в качестве способа межпроцессного взаимодействия IPC в bionic используется механизм под названием binder. Во время реализации первого этапа работы было обнаружено, что компоненты DVSDK используют IPC. Из данной ситуации было предложено два выхода: в коде DVSDK заменить механизм IPC механизмом binder либо добавить поддержку IPC в Bionic. Ввиду поставленного требования на внесение минимальных изменений в код DVSDK, решено было добавить IPC в bionic. В ранних версиях ОС Android bionic обладал таким механизмом. Оттуда были взяты некоторые файлы и добавлены в bionic, после чего при помощи стандартной утилиты был сгенерирован код, отвечающий за системные вызовы.

Далее, в момент сборки DVSDK было обнаружено, что переключение режима планировщика задач также не поддерживается в ОС Android и так как это действие носит скорее оптимизационный характер, соответствующий код решено было отключить.

Во время подгрузки модулей стало понятно, что для корректной работы DSP стека необходима память, не контролируемая ядром. Для этого при загрузке ядра в аргументах должен быть указан размер памяти, меньший, чем доступен на самом деле. А уже при загрузке модуля CMEM указываются адреса, находящиеся в освобожденной памяти.

Для проверки работоспособности системы после первого этапа использовался небольшой консольный плеер, поставляемый в качестве примера в DVSDK. Благодаря данному плееру удалось проиграть тестовые видео файлы при помощи кодеков h264 и mpeg4.

На втором этапе при попытке воспроизведения видео при помощи консольного плеера gst-launch была найдена ошибка при подгрузке динамических библиотек. После изучения данной проблемы выяснилось, что линковщик в Android умеет обрабатывать одновременно всего порядка 80 библиотек. В то время как при инициализации GStreamer требует подгрузки сразу же всех плагинов, каждый из которых представляет из себя набор динамических библиотек. Для решения данной задачи в коде линковщика было изменено значение максимального числа одновременно обрабатываемых библиотек.

Далее было обнаружено, что некоторые плагины не могли быть подгружены. Все они использовали внутри себя libm — библиотеку, содержащую различные математические функции. Позднее выяснилось, что проблема была именно в ней. Компилятор некорректно собирал libm с максимальной оптимизацией. Решением данной проблемы явилось изменение уровня оптимизации, используемого при сборке libm.

После решения данной проблемы удалось воспроизвести видео поток программными кодеками при помощи плеера gst-launch. Однако при попытке воспроизведения аудио появилась ошибка. Позднее выяснилось, что GStreamer неправильно получал параметры звука от устройства. Данная проблема на самом деле не является ошибкой, так как элемент для вывода звука устроен таким образом, что для своей работы он должен получить информацию от Audio flinger, который отвечает за вывод аудио в ОС Android. В проекте Gstdroid данный элемент получал данную информацию от GstPlayer, а в gst-launch нет возможности получить ее. Однако для проверки работоспособности библиотеки GStreamer был предложен небольшой патч, определявший значения переменных, которые не могли быть получены. Этот патч использовался только при разработке, поэтому данные изменения не

должны попасть в открытый доступ. После наложения патча удалось проиграть видео, аудио файлы, а также те, которые содержат оба этих потока.

После добавления плагина GST TI удалось успешно проигрывать аудио и видео файлы при помощи gst-launch с аппаратным ускорением, однако при попытке проигрывания файлов, содержащих оба потока, gst-launch зависал на этапе подготовки к проигрыванию видео. Исследование данной проблемы показало, что такое поведение системы является особенностью реализации плагина, и для решения данной проблемы необходимо указать корректные параметры очередей, используемых для синхронизации видео и аудио потоков.

После завершения первых пяти этапов было обнаружено, что видео не проигрывается через пользовательский интерфейс. Позднее оказалось, что проблема заключалась в том, что для корректной работы DSP стека необходимы права суперпользователя, в то время как сервис, отвечающий за воспроизведение видео запускался от пользователя media. В качестве решения этой проблемы была предложена замена пользователя, от лица которого запускался процесс.

На последнем этапе было обнаружено, что после проигрывания какого-либо видео файла, другой файл мог быть проигран лишь с некоторой задержкой. После консультации с инженерами из TI выяснилось, что данная проблема возникала из-за плагина GST TI. После остановки проигрывания система не успевала освободить место, необходимое для воспроизведения следующего видео. Данная проблема уже известна инженерам из TI и будет исправлена в следующих версиях плагина.

На компьютерах с памятью меньшей чем 256 Мб данное решение показало себя не совсем стабильно, т.к. ОС не хватает памяти для корректной работы. Поэтому описанный подход рекомендуется использовать на OMAP3EVM2, а также BeagleBoarg rev C3 и выше.

РЕЗУЛЬТАТЫ

В результате проделанной работы была изучена ОС Android и, в частности, ее видео подсистема, были рассмотрены возможности использования дополнительного ядра на процессорах OMAP35xx с системой IVA. Были предложены некоторые способы решения поставленной задачи.

Основным результатом проделанной работы является версия ОС Android с открытым кодом, собирающаяся из исходных кодов, которая поддерживает два компьютера на базе процессора OMAP3530: Beagleboard и OMAP3EVM. В данной версии ОС любое пользовательское приложение, построенное на базе Android API, для проигрывания видео использует DSP ядро в случае поддерживаемых Codec Server форматов. Для проверки применялось стандартное приложение проигрывания видео в ОС Android под названием Gallery.

При воспроизведении видео файла в поддерживаемом Codec Server формате с разрешением 480p при помощи стандартной видео подсистемы ОС Android без аппаратного ускорения средняя скорость отображения на экране составляла около 1 кадра/сек и основное ARM ядро было загружено почти на 100%. При использовании реализованной подсистемы на том же компьютере с тем же видео фалом результат составил 22-26 кадра/сек, а загрузка основного ядра снизилась более чем в 5 раз. Таким образом удалось добиться более чем двадцатикратного увеличения производительности видео подсистемы ОС Android. В таблице 1 приведены результаты, полученные при проигрывании видео файла в формате MP4 с AAC аудио потоком и H.264 видео потоком. Для получения результатов таблицы 2 был использован файл в формате AVI с MPEG4 видео потоком и MP3 аудио потоком. В обоих случаях разрешение видео составляет 640x480.

Видео подсистема	**оригинальная**	**модифицированная**
Загрузка ARM ядра (%)	95-100	5-15
Загрузка DSP ядра (%)	0	20-25
Скорость вывода на экран (кадр/сек)	< 1	22-25

Таблица 1: Сравнение производительности видео-подсистем при проигрывании видео файла в формате H.264/480p

Видео подсистема	**оригинальная**	**модифицированная**
Загрузка ARM ядра (%)	92-98	5-15
Загрузка DSP ядра (%)	0	17-23
Скорость вывода на экран (кадр/сек)	1-2	23-26

Таблица 2: Сравнение производительности видео-подсистем при проигрывании видео файла в формате MPEG4/480p

Исходные коды, полученные в результате проделанной работы, являются частью проекта Rowboat и могут быть найдены в сети интернет по адресу: http://code.google.com/p/rowboat/ (манифест: *rowboat-donut-dsp.xml).*

После появления результатов дипломной работы в открытом доступе, решения были так же использованы в проекте 0xdroid.

ЗАКЛЮЧЕНИЕ

Данный проект вызвал большой интерес в кругах разработчиков мобильных устройств, о чем говорят активные обсуждения в google группе проекта Rowboat. Он развивается, и на данный момент произведено обновление до версии eclair, недавно произошел переход на новое ядро, в настоящее время ведется работа по внедрению кодеков на основе DMAI для OpenCore, добавлена поддержка новых компьютеров, таких как IGEPv2[12], в скором будущем будет произведено обновление до новой версии DVSDK, а также добавление поддержки процессоров OMAP37xx. До настоящего времени не существовало версии Android с поддержкой DSP стека. И ее появление, без сомнения является ключевой особенностью проекта Rowboat. Некоторые большие компании интересуются им, и в частности, DSP стеком, который стал поддерживаться в результате произведенной работы.

Компьютерная техника довольно быстро стареет, и, возможно, код, написанный в этой работе довольно скоро перестанет быть актуальным, но идеи, предложенные в ней могут быть использованы при разработке подобных систем и в дальнейшем.

СПИСОК ЛИТЕРАТУРЫ

1. Официальный сайт проекта Rowboat: http://code.google.com/p/rowboat/
2. *Don Darling, Chase Maupin* GStreamer on TI DaVinci and OMAP Platforms. Texas Instruments Incorporated, 2009. - 24c. - URL https://GStreamer.ti.com/gf/download/docmanfileversion/68/1271/GStreameronTIDaVinciandOMAPPlatforms.pdf
3. *Don Darling, Chase Maupin, Brijesh Singh* GStreamer on Texas Instruments OMAP35x Processors. Texas Instruments Incorporated, 2009. - 9c. - URL https://GStreamer.ti.com/gf/download/docmanfileversion/85/2202/ols2009_darling.pdf
4. *Embedded Linux Wiki* BeagleBoard/DSP Clarification., 2010. - URL http://elinux.org/BeagleBoard/DSP_Clarification
5. *Texas Instruments Incorporated* OMAP3530/25 Applications Processor. - Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, October 2009. - 263c.
6. *Texas Instruments Incorporated* XDC Consumer User's Guide. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, July 2007. - 116c. - URL http://focus.ti.com/lit/ug/spruex4/spruex4.pdf
7. *Wim Taymans, Steve Baker, Andy Wingo, Ronald S. Bultje, Stefan Kost* GStreamer Application Development Manual (0.10.23.1). , 2009. - 96c.
8. Официальный сайт дистрибутива Angstrom Linux: http://www.angstrom-distribution.org/
9. Официальный сайт компании Ingenient Technologes Inc.: http://www.ingenient.com/

10. Официальный сайт компании RidgeRun: http://ridgerun.com/

11. Официальный сайт компании Texas Instruments: http://www.ti.com/

12. Официальный сайт компьютера IGEPv2:

http://www.igep-platform.com/index.php?option=com_content&view=article&id=46&Itemid=55

13. Официальный сайт компьютера OMAP3EVM:

http://www.mistralsolutions.com/products/omap_3evm.php

14. Официальный сайт платформы Android: http://www.android.com/

15. Официальный сайт проекта 0xDroid: http://code.google.com/p/0xdroid/

16. Официальный сайт проекта Embinux: http://labs.embinux.org/

17. Официальный сайт проекта GstDroid :

http://groups.google.com/group/prajnashi

18. Официальный сайт проекта OmapZoom: http://omapzoom.org/

19. Официальный сайт проекта Open Handset Alliance:

http://www.openhandsetalliance.com/

20. Официальный сайт проекта TI GStreamer Plugins:

https://GStreamer.ti.com/gf/project/GStreamer_ti/

21. Linux Digital Video Software Development Kits (DVSDK) for DaVinci Devices: http://focus.ti.com/docs/toolsw/folders/print/linuxdvsdk-dv.html

22. Официальный сайт компьютера BeagleBoard: http://beagleboard.org/