# INTRODUCTION TO THE OPENCORE AUDIO COMPONENTS USED IN THE ANDROID PLATFORM

**JAVIER TAPIA, JIM KOSMACH, DUSAN VESELINOVIC, GREG SHERWOOD, RALPH NEFF**

*PacketVideo Corporation, San Diego CA, USA*
tapia@pv.com
kosmach@pv.com
veselinovic@pv.com
sherwood@pv.com
neff@pv.com

Audio and speech codecs such as MP3, AAC, and AMR are used extensively on mobile devices throughout the world. In the ideal case, such codecs rely on hardware acceleration. However, it is also very common to see software audio codecs running on the main application processor, which is often an ARM core processor. Such codecs must be memory efficient, processing cycle efficient, portable to multiple operating systems, robust to data loss, and must also have a modular interface. In this paper, we introduce the OpenCORE multimedia framework and associated optimized audio codecs which are a part of the Android platform. We show how these components meet the challenging requirements for use in mobile devices. The OpenCORE audio components are currently available from the Open Handset Alliance as part of the Android SDK, and the source code for these components is scheduled for release in late 2008. The components are thus freely available for use in mobile device projects, and for non-mobile projects as well.

## INTRODUCTION

The current paper provides an introduction to the OpenCORE multimedia framework and associated audio codecs which are a part of the Android platform from the Open Handset Alliance. The OpenCORE multimedia framework provides a modular interface which allows the associated audio codecs to be easily used and configured. The framework may be used purely for audio, or in conjunction with companion components such as video codecs or file format elements, to support many typical multimedia scenarios. OpenCORE audio codecs are highly optimized for speed, efficient use of memory, robustness and portability. The codecs are based on PacketVideo's commercial audio codecs, which have been deployed in over 200 million handsets to date.

Section 1 introduces the Android Platform and provides some relevant history. Section 2 describes the OpenCORE multimedia framework on which the Android multimedia subsystem is built. Section 3 describes the OpenCORE audio codecs, and provides details about the availability, performance, features, and usability of the codecs.

## 1 THE ANDROID PLATFORM

In fall of 2007, the mobile phone industry was abuzz with rumors that Google would soon release its long awaited "G-Phone." Nobody was quite sure what this phone was, but many believed or hoped that it would be Google's answer to the iPhone. Various web articles were guessing at what revolutionary features the phone might contain, and some suggested that the phone would be commercially available in early 2008. However, when the actual announcement came in November 2007, it was clear that there would be no single "G-Phone" coming from Google. Instead, there would be a revolutionary new mobile phone platform. The platform's name was Android, and it would come not only from Google but from an alliance of companies from around the mobile communications industry [1].

Android is the first complete platform for mobile devices which is truly free and open. By "open," we mean the operating system, middleware, UI, and an assortment of typical mobile applications will all be available as open-source which may be seen and modified, thus removing many of the development limitations present on traditional mobile platforms. Thus the development of mobile applications and services for Android will be open in the same way that the development of general internet applications and services has been for years. By "free," we mean the complete Android source code will be made available for use in commercial products under a free software license. This should enable manufacturers to produce inexpensive Android-based handsets, since software license fees are typically a significant part of a handset's Bill of Materials (BOM).

Although Android was initiated by Google, it has evolved into a collaborative project with membership from 34 companies, many of which are key players in the mobile phone industry. The resulting organization is called the Open Handset Alliance (OHA) [2].

Within a week of the initial November 2007 Android announcement, the OHA released an "early-look" version of the Android SDK. This SDK included development tools, a device emulator, documentation and sample projects, as well as the rich set of libraries that implement the Android system functionality. Many developers downloaded the SDK and began writing applications based on it. As a result, Google announced in April 2008 that nearly 1800 Android applications were submitted in response to the first phase of its Android Developer Challenge contest [3].

The first Android-powered handsets are expected to be available in the second half of 2008. As well, the Android source code is scheduled for release by the end of 2008. The Linux Kernel which powers Android will be available under the free GNU Public License v2.0 (GPL2.0) [4]. The rest of the Android code, including that of the audio components, will be made available under the equally free but less commercially restrictive Apache 2.0 license [5].

PacketVideo [6] was chosen to be the multimedia sub-system provider for Android. Thus PacketVideo provides the multimedia framework and associated components (e.g audio and video codecs, file format parsing and authoring, streaming components, etc) which power Android's multimedia experience and which will be freely available to the Android developer community. These components are collectively called OpenCORE, and include the multimedia framework and codecs which are the focus of the current paper.

## 2 OPENCORE MULTIMEDIA FRAMEWORK

OpenCORE is a modular, extensible framework for combining independent media processing components such as file formats, codecs, streaming protocol components, rendering components, and other elements in different ways to implement a wide variety of multimedia scenarios. Figure 1 shows a very high-level view of the major functional blocks of OpenCORE. A detailed description of the entire framework is beyond the scope of the current paper, but this section will point out some of the highlights.

The lowest layer provides the interface to operating system APIs and platform services needed by the framework including memory management, DNS lookup, network and file I/O, process and thread control, etc. Although not shown as a separate functional block, the multimedia framework defines methods and data structures for negotiating formats and parameters between components, and for passing media data between components. The design has provisions which minimize the copying of media data as it flows through the resulting graph.

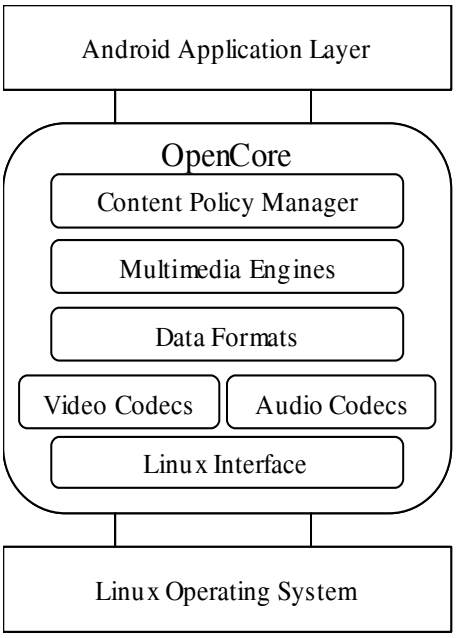


Figure 1 – High-level view of the major OpenCORE functional blocks.

The video codec block includes MPEG4, H.263, and H.264 video codecs along with interfaces to integrate other formats and hardware accelerated codecs. The audio codec block similarly provides interfaces for the integration of additional codecs along with the included audio codecs which are described in detail in Section 3.

The data formats block handles the reading and writing of file formats, as well as support for various streaming protocols. Supported file formats include .mp3, .mp4, .3gp, .aac, .amr, and .wav. Supported streaming protocols include RTSP and HTTP, the latter including both simple and "progressive" download functionality. The multimedia engines block provides the logic for selecting and connecting components into the appropriate graph structure as well as providing the high level controls to the application for a given multimedia use-case. Finally, the Content Policy Manager block provides a framework for content access and control, which can be used for integration of DRM functionality.

In a typical scenario, the Multimedia Engines block would be used to assemble a graph containing appropriate file formats, audio and video decoders, and rendering interfaces in order to achieve local playback of media files. Other graphs could be assembled to achieve other typical scenarios, such as media authoring/recording, or streaming media playback. The availability of standard components with modular graph assembly should allow Android application developers to easily access and make effective use of the available multimedia resources.

## 3 OPENCORE AUDIO AND SPEECH CODECS

The OpenCORE audio and speech codecs are listed in Table 1, along with the supported sampling frequencies (KHz), channel configurations (#Ch), and bit rates (kbps). Supported codecs include MP3, AAC, HE-AACv1, HE-AACv2, AMR-NB and AMR-WB decoders and an AMR-NB encoder.

| Codec Type | KHz | #Ch | kbps |
|---|---|---|---|
| MP3 1/2/2.5 | 8~48 | 2/1 | 8 to 320 |
| AAC LC/LTP | 8~48 | 2/1 | Up to 160 |
| HE-AACv1 | 8~48 | 2/1 | Up to 96 |
| HE-AACv2 | 8~48 | 2 | Up to 64 |
| AMR-NB | 8 | 1 | 4.75 to 12.2 |
| AMR-WB | 16 | 1 | 6.6 to 23.85 |

Table 1: OpenCORE audio codecs for Android

### 3.1 Modular Design

OpenCORE has adopted the OpenMAX Integration Layer (IL) interface [7]. OpenMAX IL is a well-known cross-platform API that allows multimedia components to be developed once and then easily integrated across multiple operating systems and hardware platforms. Figure 2 shows the modular OpenCORE design that takes advantage of the OpenMAX IL interface.

The OpenCORE multimedia framework provides OMX components which are used to integrate the audio and video codecs. An OMX component serves as a wrapper around the audio codec, and provides compatibility to the OpenMAX IL API set. Most of the APIs are asynchronous, meaning that the OMX component queues the commands and input data buffers which it receives via the OpenMAX Integration Layer, and processes them at a later time. When scheduled to run, the OMX component may process one or more queued input data buffers in order to decode the associated audio bitstream data using the connected software audio codec. The OMX component then returns the processed input buffer back to the OpenMAX Audio Decoder Node via the OpenMAX integration layer.

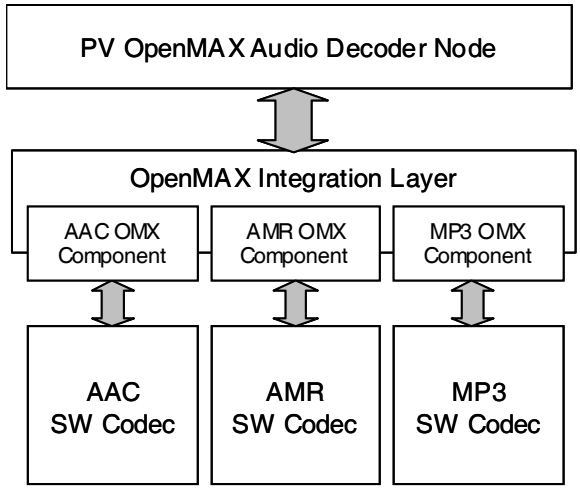


Figure 2: OpenCORE modular design for Android

Where applicable, the audio codec libraries have a configurable design which allows optimization of the memory footprint. For instance, the AAC library uses separate layers to accommodate baseline AAC, Spectral Band Replication (SBR) and Parametric Stereo (PS) tools. Thus when selecting among the AAC and HE-AAC codec options at build time, it is possible to configure the code to exclude the layers which are not needed. As another example, the MP3 decoder has an embedded equalizer that can be included or excluded using build-time configuration.

### 3.2 Optimizations

The OpenCORE audio codec libraries were originally based on floating point reference implementations of the various standard codecs [8][9][10][11]. Over time, the codecs have been translated to efficient fixed-point implementations, and then re-architected and optimized to make the best use of the available system resources.

The audio codec libraries are written mainly in C. Since many embedded devices utilize an ARM core processor, the code is supplemented in limited key areas using optimized inline assembly code. In many places there are multiple configurable inline assembly sets, which are chosen at build time depending on the hardware configuration. Such optimizations may be switched on or off at compile time, and are particularly useful for cases in which the hardware does not accommodate an advanced optimizing compiler such as ARM ADS/RVCT.

The OpenCORE audio codec libraries make use of an efficient internal memory management process. A pool of memory is allocated to the library upon instantiation, and this memory is recycled and reused throughout the decoding session. This prevents the decoding process from incurring delay due to dynamic memory allocation requests to the OS.

The OpenCORE audio codec libraries make intrinsic use of an OS portability layer called OSCL. Thus the libraries may be easily ported to new operating systems by simply porting the OSCL layer. OSCL ports exist for many operating systems, including Win32, Symbian, Linux, and ARM-Linux. However, the initial releases of Android will include only the Linux port of OSCL, since the Android platform is Linux-based.

The OpenCORE audio codecs are based on PacketVideo commercial codecs which have been deployed on nearly 200 different device models across a wide variety of operating systems and compilers. The environments which have been supported include:

- Linux (using gcc compiler)
- Symbian (using gcc and rvct compilers)
- Brew (using ads compiler)
- Windows CE (using Microsoft compiler)

### 3.3 Performance

The OpenCORE audio codecs have been optimized with ARM's ADS/RVCT compiler and profiled on the ARMulator (ARM emulator). Table 2 shows the zero-wait state performance of the codecs as profiled on the ARMulator using an ARM9E processor.

| Codec Type | KHz | Ch | kbps | MHz |
|---|---|---|---|---|
| MP3 | 44.1 | 2 | 128 | 21.8 |
| AAC | 44.1 | 2 | 96 | 13.7 |
| HE-AACv1 | 44.1 | 2 | 48 | 29.1 |
| HE-AACv2 | 44.1 | 2 | 32 | 40.2 |
| AMR-NB | 8 | 1 | 12.2 | 11.2 |
| AMR-WB | 16 | 1 | 23.85 | 20 |

Table 2: ARMulator performance results for OpenCORE audio codecs on zero-wait state ARM9E.

Table 3 shows an alternate result in which the OpenCORE audio codecs were compiled using arm-linux-g++ compiler (version 3.4.0) and were run on a PacketVideo reference platform based on OMAP 2420. Note that the OMAP 2420 contained an ARM 11 processor running at 266 MHz, and also a C55x DSP running at 220 Mhz. The OpenCORE audio codecs were in all cases running entirely on the ARM 11.

| Codec Type | KHz | Ch | kbps | MHz |
|---|---|---|---|---|
| MP3 | 44.1 | 2 | 128 | 27 |
| AAC | 44.1 | 2 | 96 | 21 |
| HE-AACv1 | 44.1 | 2 | 48 | 44 |
| HE-AACv2 | 44.1 | 2 | 32 | 64 |
| AMR-NB | 8 | 1 | 12.2 | 12 |
| AMR-WB | 16 | 1 | 23.85 | 26 |

Table 3: OpenCORE ARM11 performance data on an OMAP 2420 platform

The performance of the OpenCORE audio codec libraries, written in C code with limited inline assembly optimization, closely matches and often exceeds the performance of assembly-only implementations that are typical of other third party codecs. Maintaining most of the code in C has the advantage that the code remains easily portable, and that it remains easy to work on the algorithmic and architecture elements of the code.

### 3.4 Error Concealment

Error resilience and concealment are extremely important for audio and speech codecs which target a mobile environment. For speech codecs, the standard codec specifications provide simple mechanisms to compensate for the loss of speech frames. For example, AMR-NB specifies repetition of the last valid frame, together with an exponential decay. For audio codecs, the standard codec specifications may propose complex concealment methods which may not be efficient in a mobile environment. For example, AAC specifies a concealment method based on storing several AAC frames to mask a possible loss of data. This may increase code size, memory usage, and computational complexity.

In the speech case, we believe the non-continuous nature of the speech signal makes the decaying exponential mechanism sufficient. For general audio, the information tends to be more continuous and so any loss of data is more easily detectable. In those cases, the OpenCORE audio framework provides a simplified error concealment method. Loss of audio frames may be detected outside of the codecs and compensated through the insertion of pre-generated synthetic silence frames. The use of silence frames maintains the decoder flow and so maintains A/V synchronization. The end result is that the packet loss is concealed using less memory and processing cycles than the standard method, thereby saving memory footprint and battery life. This method can also be easily extended to include other audio codecs that may be added to the Android platform without requiring direct modifications to the codec libraries themselves.

## 4 CONCLUSION

Android is the first truly complete, free, and open platform for mobile devices. PacketVideo's OpenCORE components provide the multimedia functionality for the Android platform. OpenCORE provides a flexible and modular framework for assembling useful configurations of the provided audio and video codecs and related multimedia elements. The audio codecs included in OpenCORE are highly optimized, configurable, and easily adaptable to many combinations of OS, processor, and compiler.

We expect to see many interesting devices based on the Android platform in the near future. Thanks to the OpenCORE audio components, we expect to hear many interesting devices as well.

## REFERENCES


[1] "Industry Leaders Announce Open Platform for Mobile Devices," Press Release from the Open Handset Alliance, November 5, 2007.

[2] Open Handset Alliance web site, http://www.openhandsetalliance.com

[3] Announced on the Android Developer's Blog site, April 17, 2008. http://android-developers.blogspot.com/

[4] GNU Public License v2.0, June 1991. Available at http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[5] Apache Software License, Version 2.0, January 2004. Available at: http://www.apache.org/licenses /LICENSE-2.0.txt

[6] PacketVideo web site, http://www.pv.com/

[7] OpenMAX IL specification and tests, available at http://www.khronos.org/openmax/

[8] ISO/IEC 14496-3 "Information Technology -- Coding of Audio visual objects -- Part-3:Audio."

[9] ISO/IEC 11172-3 "Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbits/s -- Part-3:Audio."

[10] 3GPP TS 26.071: "Mandatory Speech CODEC speech processing functions; AMR Speech CODEC; General description".

[11] 3GPP TS 26.171: "AMR Wideband Speech Codec; General Description"