OpenMAX™

Streaming Media Portability

Shiv Ramamurthi, Software Technology Marketing Manager, Texas Instruments

OpenMAX Chair

What is OpenMAX?

- A set of open royalty-free APIs for abstracting multimedia functionality on embedded devices
- Strong OpenMAX Participation

ARM AMD BROADCOM. BEATNIK emuzed

ERICSSON FhG freescale semiconductor Launched by Motorola Infineon technologies intel

NVIDIA. NOKIA CONNECTING PEOPLE MOTOROLA PHILIPS SAMSUNG ELECTRONICS sasken

SIEMENS SKY MOBILEMEDIA ST symbian TEXAS INSTRUMENTS

Why OpenMAX?

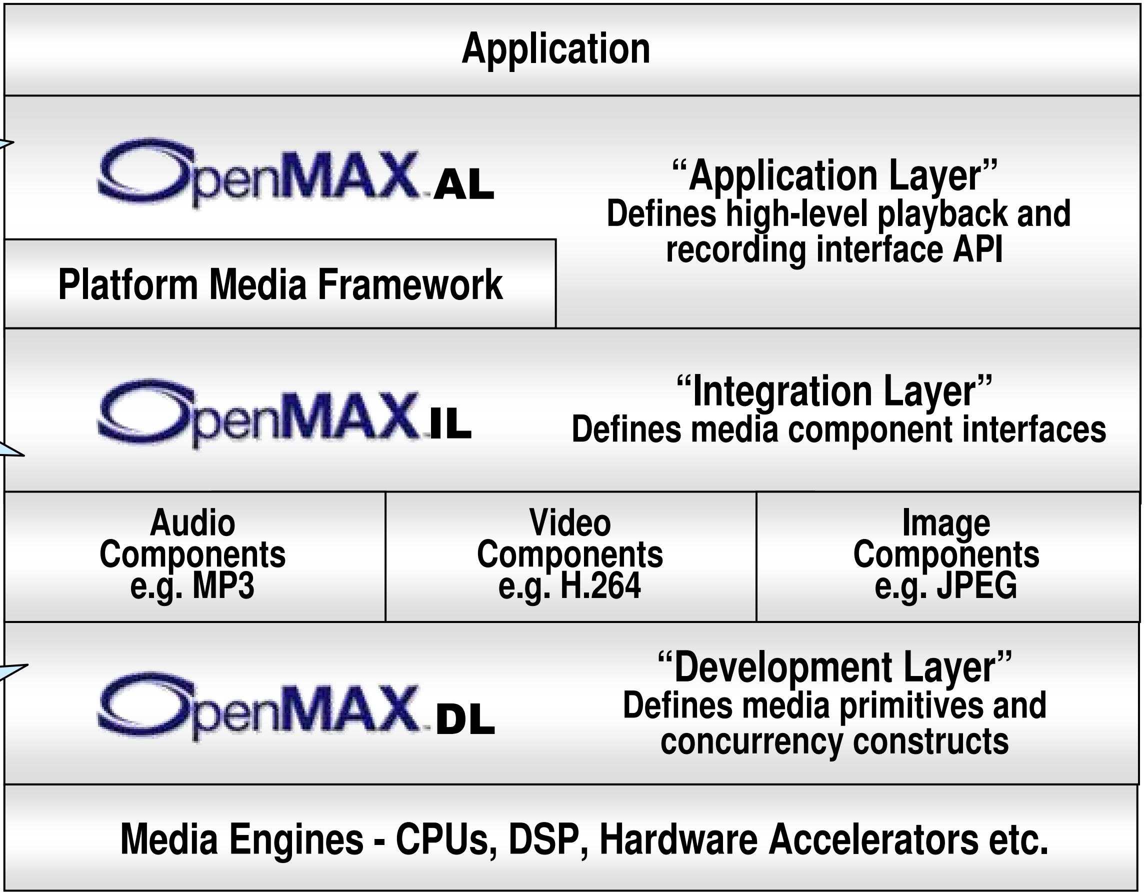
- **Classic reason for a standard: fragmentation compromises portability**
- **Multiple pieces to a multimedia ecosystem**
  - General purpose processors with varying ISA
  - Media co-processors, DSP, dedicated HW
  - Codecs, readers/parsers, renderers, sources, post processors
  - Multimedia frameworks/middleware
  - Applications
- **Multiple implementations of each piece multimedia processing**
- **Multiple (often proprietary) API solutions between any two pieces**
- **Portability problem:**
  - How do you write a single implementation of a codec, multimedia framework, or application across different platforms?

Multimedia Ecosystem

Media applications can be written portably, independent of the underlying media platform

Media components can be integrated into flexible media graphs for advanced streaming media processing

Media components can be written using primitives for portability across diverse parallel and serial silicon architectures



Application

OpenMAX AL

"Application Layer"
Defines high-level playback and recording interface API

Platform Media Framework

OpenMAX IL

"Integration Layer"
Defines media component interfaces

Audio Components e.g. MP3

Video Components e.g. H.264

Image Components e.g. JPEG

OpenMAX DL

"Development Layer"
Defines media primitives and concurrency constructs

Media Engines - CPUs, DSP, Hardware Accelerators etc.

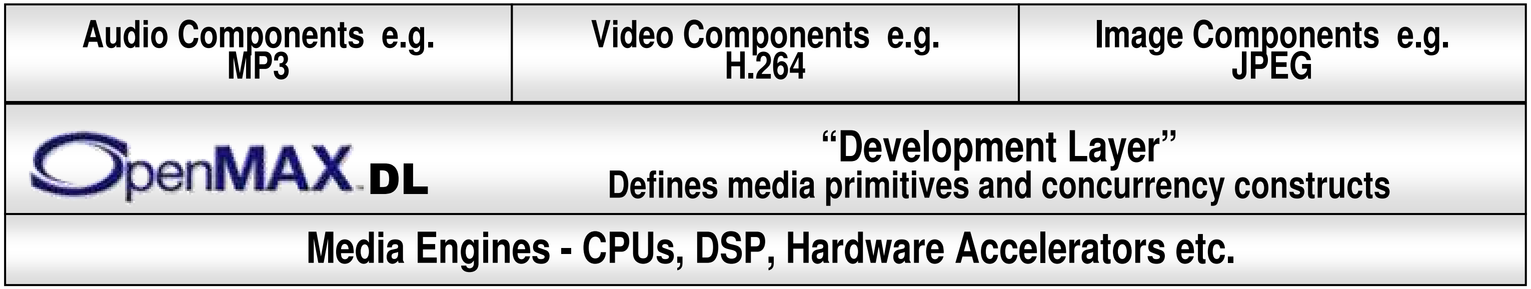
OpenMAX layers can be implemented together or independently from the other layers

OpenMAX DL – Overview

Problem: Porting media components to new platforms is costly and time consuming

- Software components are not portable across processors
  - Exacerbated by proliferation of media standards and increasing silicon complexity
- Software component & silicon vendors need a reliable way to accelerate diverse codecs on diverse silicon

Solution: OpenMAX DL – media components are written using primitives for rapid portability across diverse parallel and serial silicon architectures



Audio Components e.g. MP3

Video Components e.g. H.264

Image Components e.g. JPEG

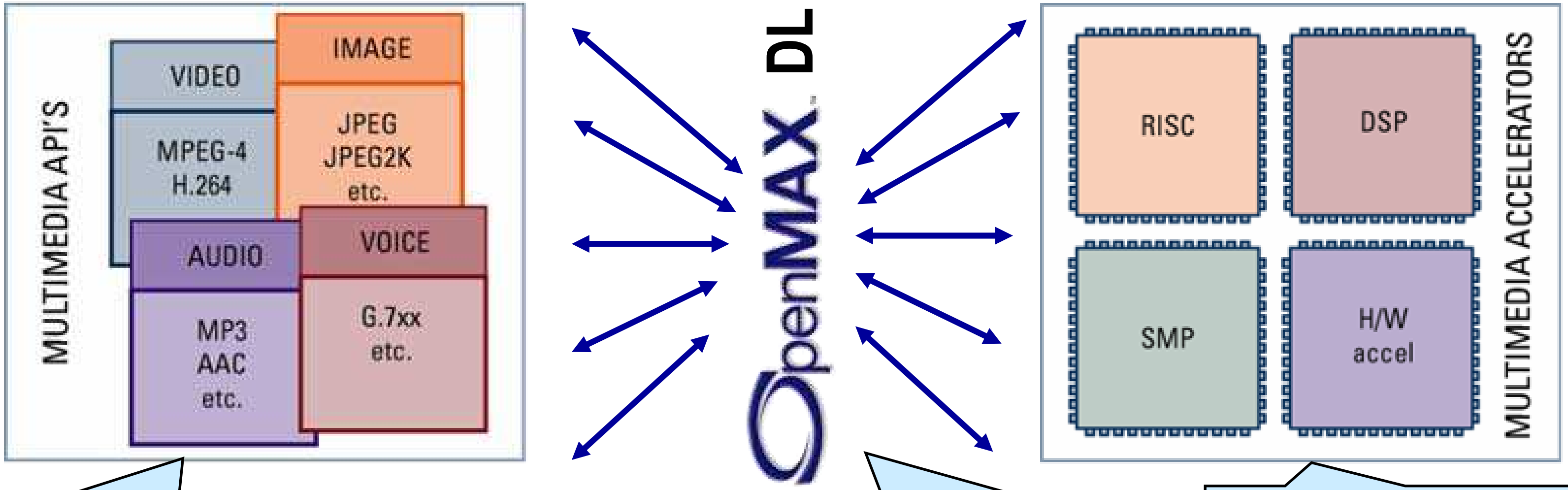
OpenMAX DL

"Development Layer"
Defines media primitives and concurrency constructs

Media Engines - CPUs, DSP, Hardware Accelerators etc.

- Focus on what one does best
  - Component vendors provide advanced functionality without worrying about hardware specific optimization
  - Silicon vendors provide optimized implementations of DL primitives to reduce cost to market [DL APIs are IP free]

OpenMAX DL – Overview



An increasing number of multimedia codecs for video, audio, graphics and images

Silicon vendors supply optimized OpenMAX DL library covering 80% of processing requirement

A wide range of media acceleration silicon using many diverse architectures

DL Domains

- **Video Domain**
  - MPEG-4 SP/H.263 BL (encode and decode)
  - H.264 (encode and decode)
- **Image Processing Domain**
  - Color space conversion
  - Pixel packing/unpacking
  - De-blocking / de-ringing
  - Rotation, scaling, compositing, etc.
- **Image Codec Domain**
  - JPEG (encode and decode)
- **Multimedia Audio Domain**
  - MP3
  - AAC
- **Signal Processing Domain**
  - FIR
  - IIR
  - FFT
  - Dot Product

OpenMAX DL – Overview

DL API Examples (Video)

- `OMXResult omxVCCOMM_Average_8x(...)`
- `OMXResult omxVCCOMM_Copy8x8(...)`
- `OMXResult omxVCCOMM_SAD_16x(...)`
- `OMXResult omxVCM4P2_IDCT8x8blk(...)`
- `OMXResult omxVCM4P2_DecodeVLCZigzag_Inter(...)`
- `OMXResult omxVCM4P10_InvTransformResidualAndAdd(...)`
- `OMXResult omxVCM4P10_DeblockLuma_I(...)`

Advanced Concurrency Mechanisms

- **aDL (Asynchronous DL)**
  - Chain together multiply DL primitives to create one API
  - Each building block could run aBetter optimization on some platforms
- **iDL (Integrated DL)**
  - Integration of DL APIs in OpenMAX Integration Layer (IL)
  - DL APIs are mapped into IL structures; execution is controlled by IL state machine

OpenMAX IL

- **Componentized architecture for abstracting blocks of multimedia functionality**
  - Could be implemented in software or hardware
- **Building blocks categorized by**
  - Domain: Audio, video, image, of some combination thereof
  - Function: Encode, decode, apply an effect, capture, render, split, mix, etc
- **Allows blocks from different sources to work together**
- **Allows clients to build arbitrary multimedia pipelines by plugging blocks together**

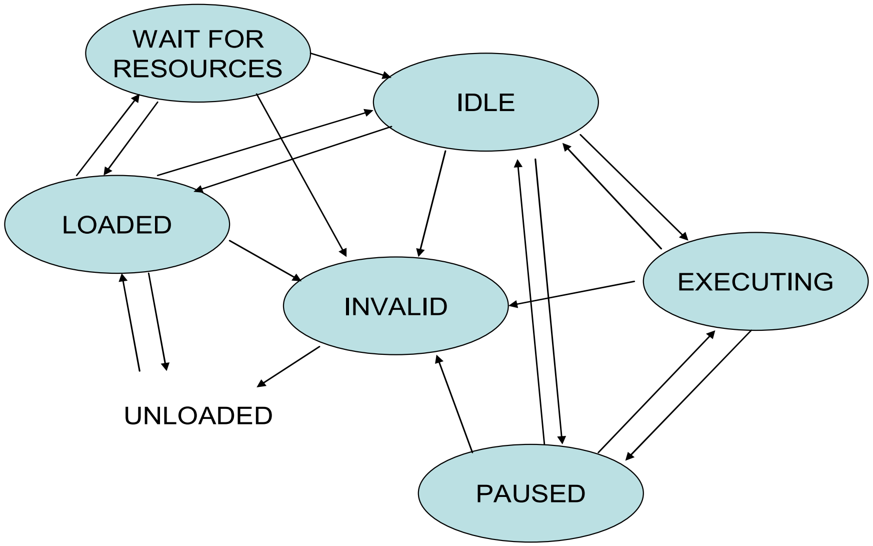
Portable & Re-usable ‘building blocks

OpenMAX IL “Component”

- **A component is a building block encapsulating one function**
- **Plumbing**
  - Each component *port* is either the entrypoint or exitpoint for one particular stream of data of the component
  - A port may be connected to the client or to a port on another component
  - A client exchanges buffers with a port or two ports (from different components) exchange buffers with each other
- **Knobs**
  - A component *parameter* is a value that is set prior to component execution
  - A component *config* is a value that may be set during component execution
  - OpenMAX defines an independent set of “knobs”
  - Examples: rate, volume, resolution, scaling, bitrate, error correction, brightness, etc
- **Control**
  - Via a standard interface common to all components (i.e. a structure of function pointers)
  - The connection of ports
  - Data flow in and out of ports
  - State management
  - Query/set configs and parameters

Component State Machine

- **The client controls the operation of each component by manipulating its state.**
  - Loaded: alive without resources
  - Waiting for resources: deficient of resources and actively waiting for them to become available
  - Idle: has resources but not transferring buffers
  - Paused: has resources, transferring buffers, but not processing data
  - Executing: has resources, transferring buffers, and processing data

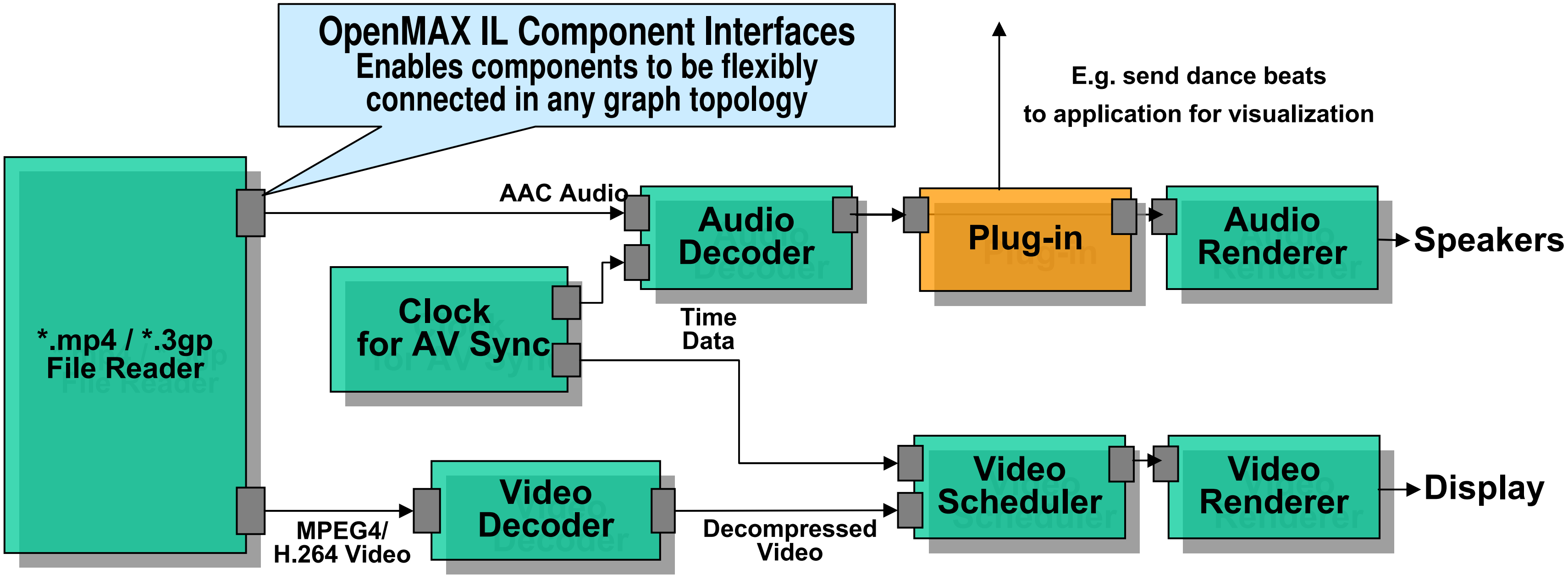


OpenMAX IL Standard Components

- **OpenMAX IL 1.1 (just released!) defines a set of standard components**
- **Examples**
  - Readers/writers: 3gp, asf, image, video, audio
  - Audio decoders/encoders: AAC, AMR, MP3, WMA, Real Video
  - Audio post-processor: stereo widening, equalizer, reverb
  - Video decoders/encoders: MPEG4, H.264, etc
  - Image decoders/encoders: JPEG
  - Input devices: camera, audio input
  - Output devices: audio renderer, video renderer
  - Synchronization: clock component, video scheduler
- **Each component definition is a “black box” recipe consisting of:**
  - Functional description
  - Which ports of which type are required
  - Which configs/parameters and which settings on those configs/parameters are required
- **Standard set of building blocks allows for more portable graphs**

OpenMAX IL Example Graph

- Standardized component interfaces enable flexible media graphs
- Includes multi-stream synchronization
- Allows for custom plug-ins


OpenMAX IL Component Interfaces
Enables components to be flexibly connected in any graph topology

E.g. send dance beats
to application for visualization

Example: MPEG-4 video synchronized with AAC audio decode

OpenMAX AL Motivation

- **OpenMAX IL has more complexity than most application developers require**
- **Instead of building a multi-component playback graph and coordinating the numerous pieces most app developers just want to**
  - Specify where the content comes from
  - Where the content should be rendered to
  - Manipulate a few playback controls
  - Have simple configurability
- **Likewise for recording use cases**
- **That's what OpenMAX AL provides…**
  - A simple high level multimedia API for playback and recording use cases

Use Cases

- **OpenMAX AL was designed around satisfying a few specific use case yet also to have a model *general enough to be extensible* to other use cases**
- **Targeted by 1.0:**
  - Audio player
  - Audio recorder
  - Image displayer
  - Image capture
  - Synchronized audio/video playback
  - Synchronized audio/video recording
  - Analog Radio

How to learn more

- On the web: http://www.khronos.org

- Availability
  - OpenMax DL 1.0 available now
  - OpenMax IL 1.1 available now
  - OpenMax AL available Q2 2007

- Download materials www.khronos.org
  - Specification
  - Headers
  - White papers

OpenSL|ES™

OpenSL ES – changing the face of mobile audio

The Embedded Audio Problem

- Lots of fragmentation!

- Many closed proprietary audio APIs of varying functionality
  - Playing a simple sound on different platform requires different code.

- No standard way to access audio hardware acceleration
  - Lots of work for developers to re-write code for every platform – no application/source level portability

- Newer multimedia devices incorporating more advanced audio functionality
  - Increases in audio quality & functionality further complicate the efforts of content developers aggravating the portability problem.

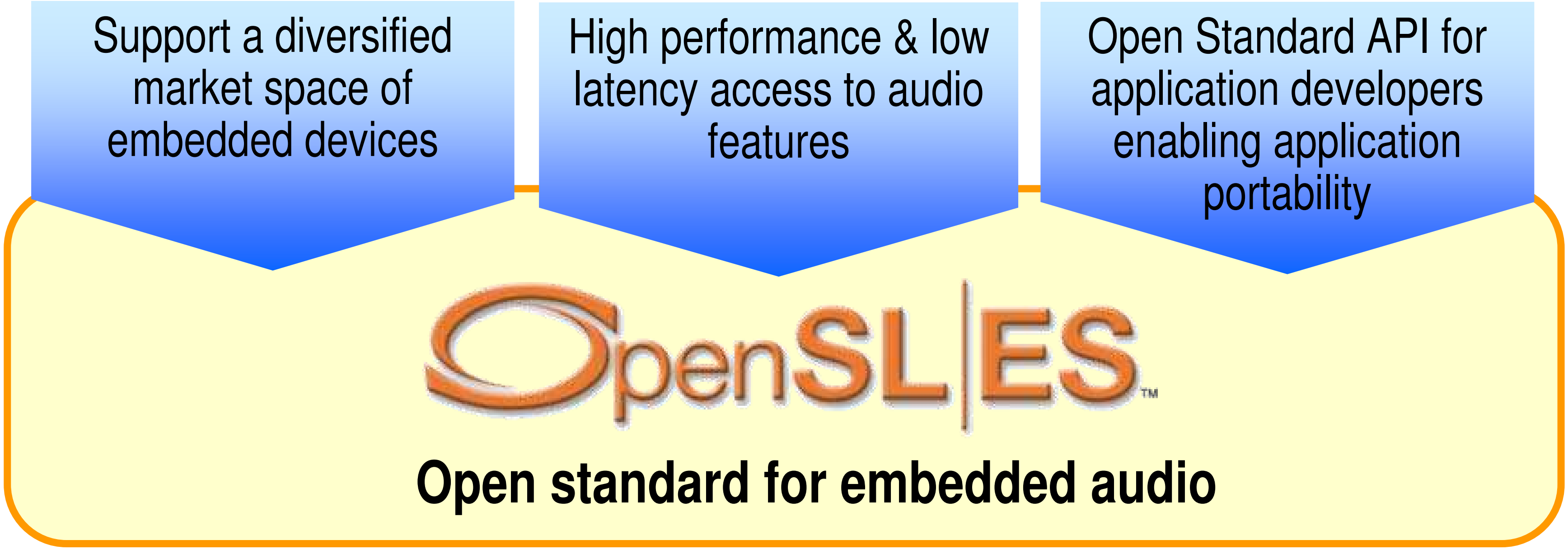
Solution - Open standard application level API for embedded audio !

OpenSL|ES

KHRONOS GROUP

The Embedded Audio Solution

- **Diversified collection of embedded devices**
  - Target devices include “*Mobile phones, personal media players & handheld gaming consoles*”.
- **High performance, low latency access to audio**
- **Application developer friendly**
  - Source-level portability of native code from platform to platform
  - Same interface for both hardware and software solutions
- **Royalty-free open standard**



Support a diversified market space of embedded devices

High performance & low latency access to audio features

Open Standard API for application developers enabling application portability

Open standard for embedded audio

Target Applications

- OpenSL ES API is designed with the application developers in mind

- Example applications include:

Simple 2D Games

Music Playback

User interface sounds

3D Games

Recording

Ring-tone Playback

Sequencer

Features Overview

- **Playback of audio files**
  - Playback PCM and encoded content
  - Good for sound effects; device UI sounds
- **SP-MIDI, Mobile DLS, Mobile XMF**
  - For interactive music and ring-tones
- **Effects & Controls**
  - Music and media player effects
  - Advanced environmental effects for gaming
- **3D Audio**
  - Provided for gaming as companion to OpenGL ES

3D position

Doppler

MIDI messages

Preset Reverb

Environmental Reverb

Volume

EQ

Pitch

Rate

Buffer Queues

Metadata extraction

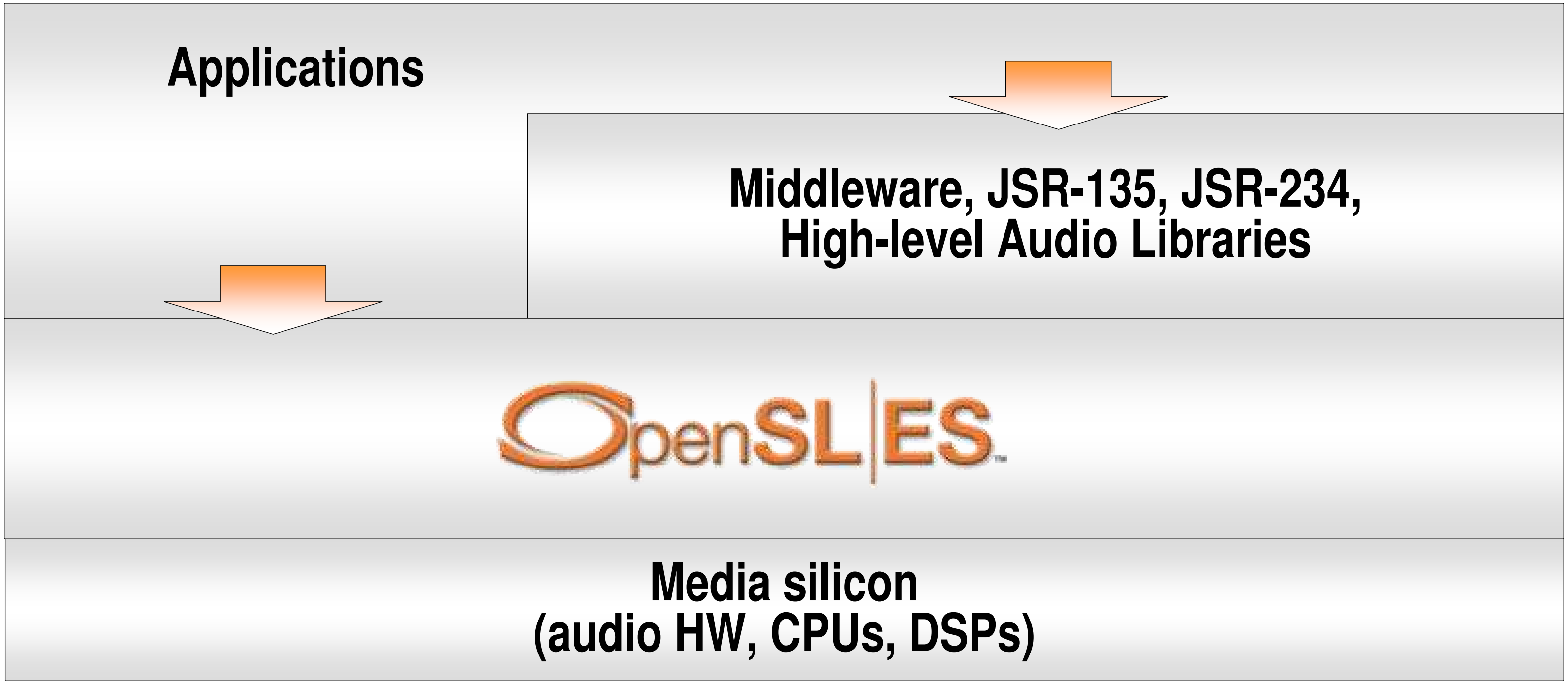
Virtualization

Stereo widening

LED & Vibra

Supporting Applications

- Native application support
- Cross-platform foundation for other APIs
  - Support foundation APIs (JSR-135, JSR-234)
- Same API for H/W and S/W solutions



Sneak preview of OpenSL 1.0 Architecture: Objects

- **Object and interface API architecture**
- **Engine**
  - An instantiation (or “context”) of the API
  - Created at startup; destroyed at shutdown
  - Used to instantiate other objects.
  - Affect the state of all objects created from that engine
- **Media Objects (Player, Recorder, MIDIPlayer)**
  - Implements an audio use case
  - Each Media Object type exposes a different set of interfaces
- **Listener**
  - Abstract object representing the listener of the 3D positioned players
- **3D Groups**

Disclaimer: OpenSL ES is not due for release until later this year. Details are still being finalized so some things might change!

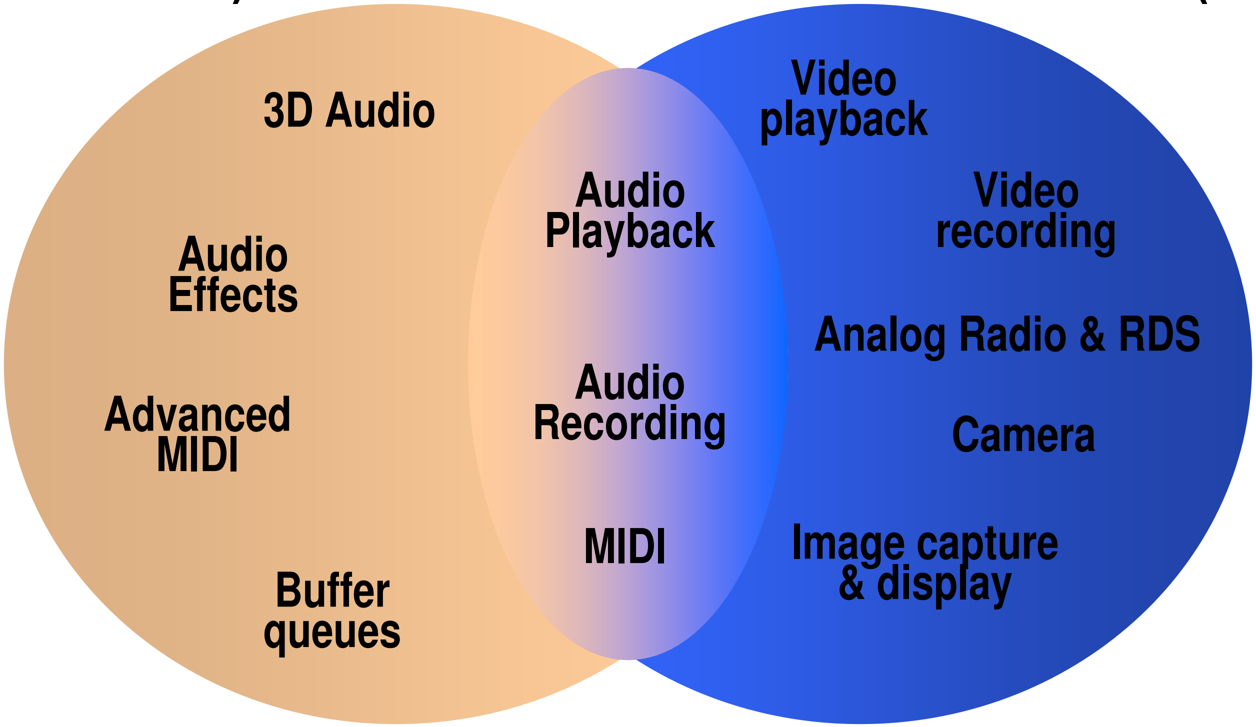
Relationship with OpenMAX-AL (1/2)

OpenSL|ES

(Enhanced audio API)

OpenMAX

(Multimedia API)



- Working groups collaborating to define the common API functionality.

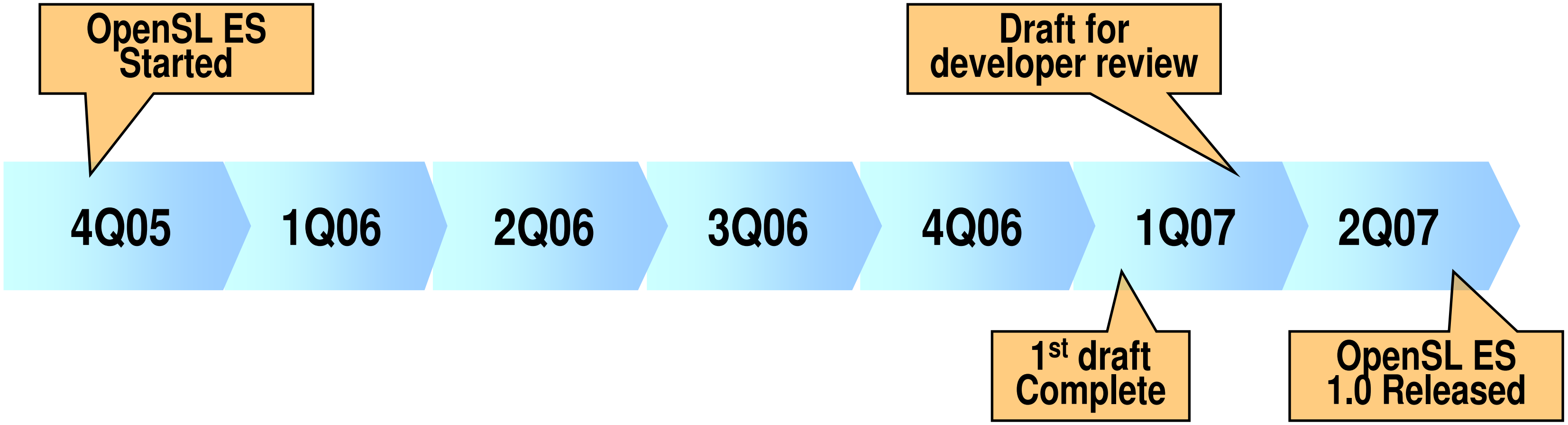
KHRONOS GROUP

Relationship with OpenMAX-AL (2/2)

- **Independent**
  - There is no dependency between the APIs.
  - A device can support a combination of the APIs that most suits the device:
    - OpenMAX AL + OpenSL ES (Music)
    - OpenMAX AL + OpenSL ES (Game)
    - OpenMAX AL only
    - OpenSL ES (Phone, Game, Music) only
- **Compatible**
  - Working groups collaborated to make sure the APIs work well together.
- **Consistent**
  - Identical API architecture.
  - Identical APIs for same functionality.

Schedule

- **Khronos working group**
  - Started in Q4 2005
  - Requirements for 1.0 release finalized
- **Draft Specification for developer review**
  - Requires signing Khronos reviewer agreement
    - Provides opportunity for developers to specification issues
- **OpenSL ES 1.0 scheduled for release 2Q07**
  - Release includes specification & headers
  - Conformance tests soon after



OpenMAX™

Any questions?