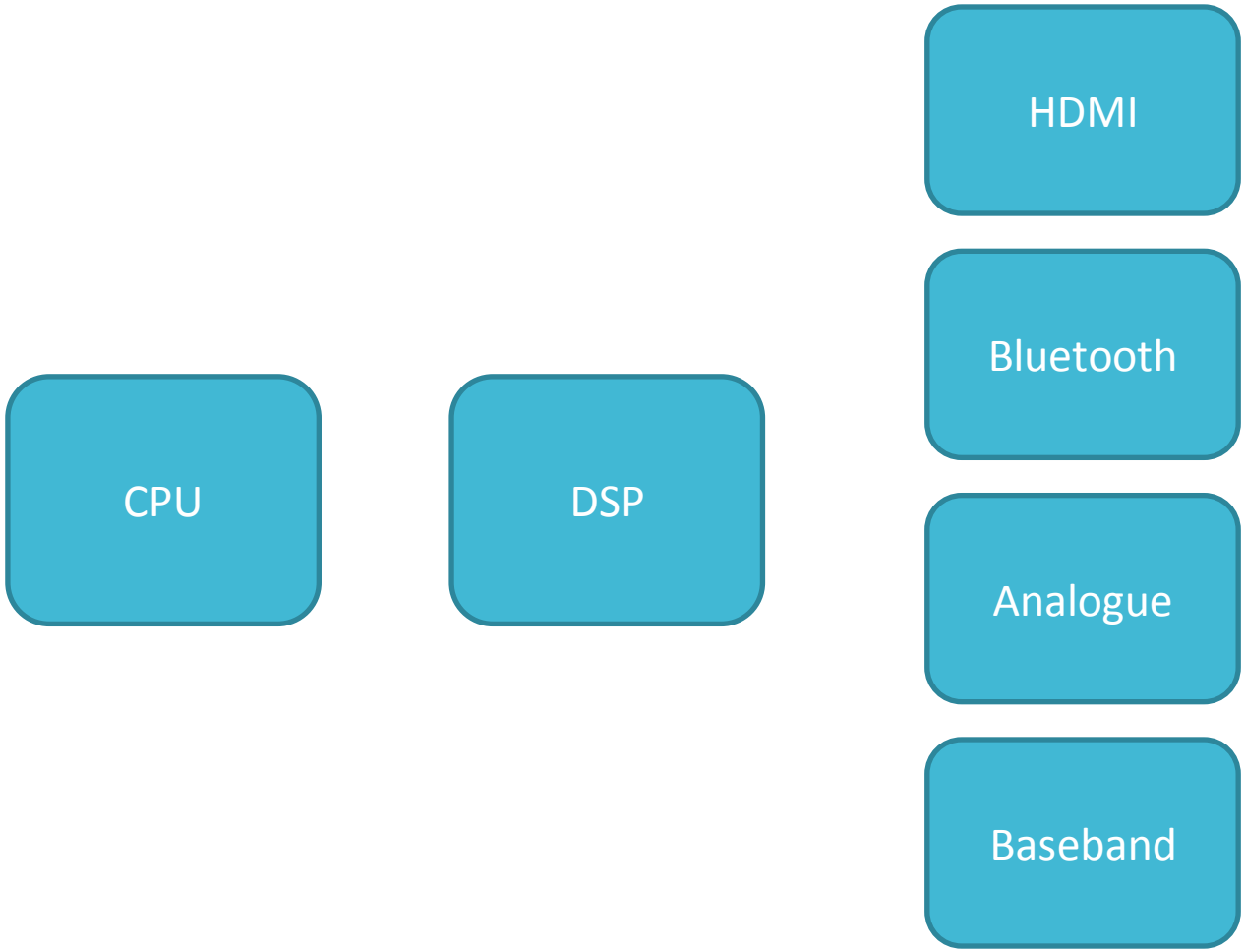
Towards a standard audio HAL for Linux

Introducing TinyHAL

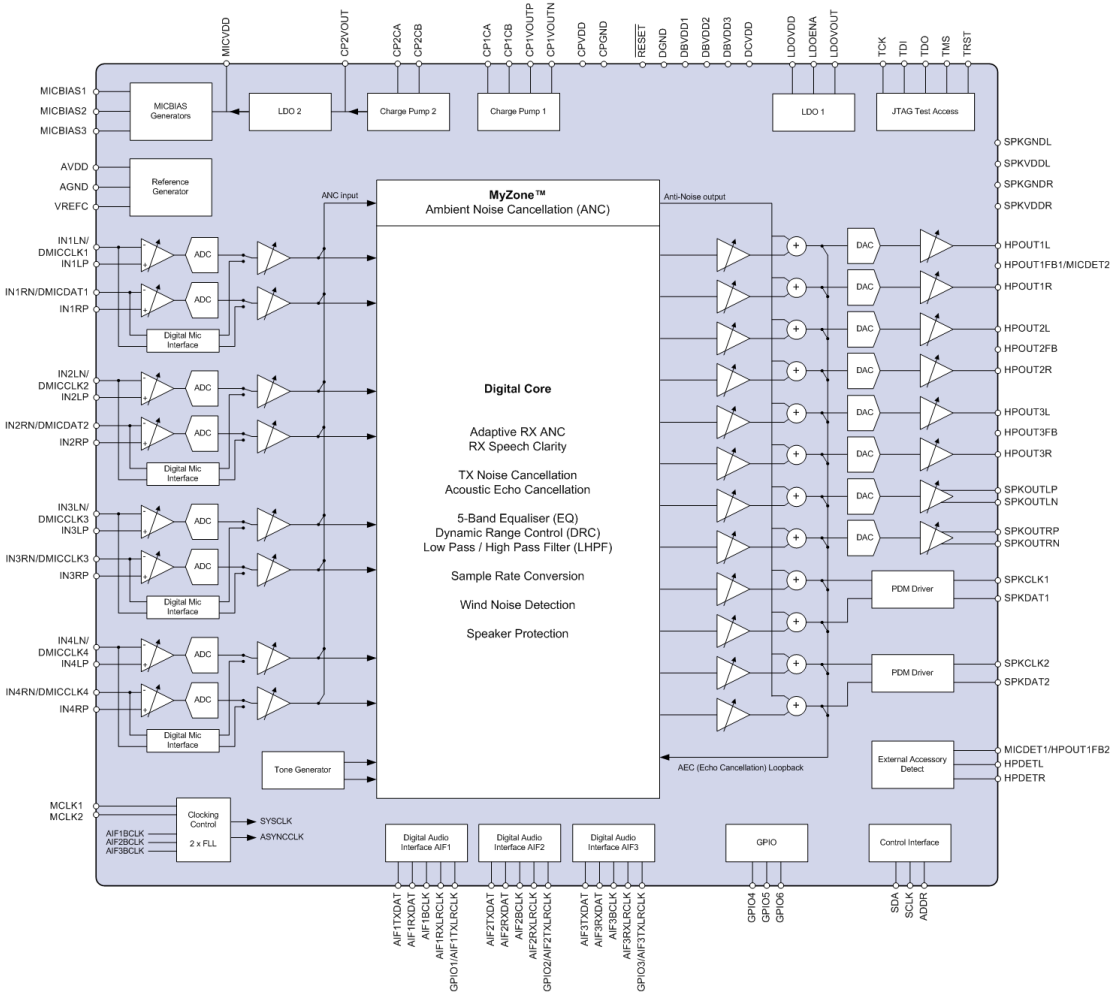
Overview

- Introduction to smartphone audio
- System integration in the Android audio stack
- Existing audio HALs
- Introducing TinyHAL

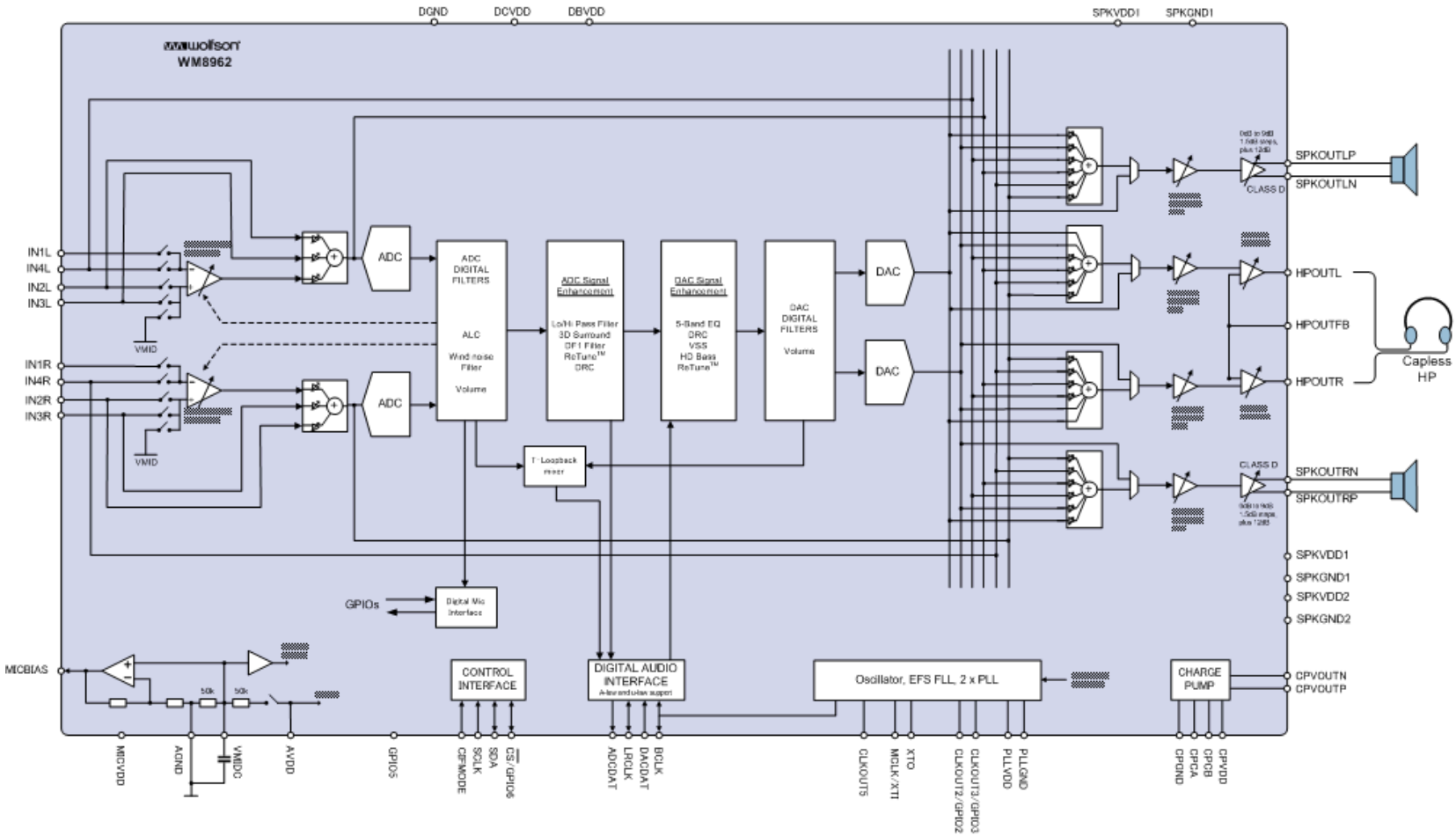
Smartphone audio hardware



A smartphone CODEC



A tablet CODEC



Audio configuration development

- **Product configuration**
  - Configuring audio paths
  - Acoustic engineering
  - DSP algorithm configuration
  - Many aspects require specialist measurement techniques
- **Use case management**
  - Transitions
  - Overlapping use cases
- **Many interdependencies**
- **May need different tunings for different markets**

Android audio - AudioFlinger

- **AudioFlinger manages all audio in the system**
  - Standard Android code
- **High level policy decisions**
  - “Output to headphones and speaker”
  - “Record from headset microphone”
  - “Output to HDMI”
- **Common behaviour between Android devices**
  - Can be overridden, but usually done by editing code
  - Not really anything to do with the tuning

Android audio - HAL

- **Relies on audio HAL plugins to implement policy**
  - Totally system specific code
  - Tell AudioFlinger which devices are available
  - Implement audio streams to and from hardware
- **Linux kernel provides standard interfaces below HAL**
  - ASoC – ALSA subsystem for embedded devices
  - Accessory detection
- **Not adopted by key vendors when Android was architected**
  - ...but are now, even by out of tree vendors

Current implementations

- **Google AOSP code for Nexus phones**
- **alsa_sound**
- **System integrator implementations**
  - SoC vendor code
  - Product vendor code
- **Much parallel development**

Google AOSP code for Nexus phones

- **Nexus S and Galaxy Nexus**
- **Based on TinyALSA, Apache licensed**
  - Requirement for core Android/AOSP code
  - Desirable for many system integrators
- **Device specific**
  - Difficult to reuse directly on other products
- **Configuration in code**
  - Only software engineers need apply!

Google AOSP code for Nexus phones

```
struct route_setting vx_ul_bt[] = {
    {
        .ctl_name = MIXER_MUX_VX0,
        .strval = MIXER_BT_LEFT,
    },

return strcmp(property,
  PRODUCT_DEVICE_TORO) == 0;
```

alsa_sound

- **In AOSP as an external project**
  - Contributed by Windriver early on
- **Based on standard ALSA library**
  - LGPL, unsuitable for standard AOSP usage
- **Use cases configured in asound.conf**
  - Good for maintainability but...
  - ...not designed for transitions
  - UCM not yet supported
- **Not yet updated to ICS HAL API**

System integrator HALs

- **Often based on alsa_sound**
  - Working around limitations in the configuration files
  - Sometimes adding features like DSP integration
- **Typically proprietary**
  - Device and system assumptions
  - Licensing

- **Need a license suitable for AOSP**
  - Use TinyALSA
- **Configuration moved out to files**
  - UCM style
  - XML parsed using expat
- **Prototype done last year for Gingerbread**
  - In active use by some users
- **Still in early development**
- **Sample configuration for Nexus S**

TinyHAL configuration

- **System defaults**
- **Top level use cases**
  - Media/default
  - Telephony
- **Per device routes**
- **Modifiers**
  - Notification tone in call

Configuration example

```
<!-- We are able to have most of our routing static so do that -->
    <path>
      <!-- AIF1->DAC1 -->
      <ctl name="DAC1 Switch" val="1" />
      <ctl name="DAC1L Mixer AIF1.1 Switch" val="1" />
      <ctl name="DAC1R Mixer AIF1.1 Switch" val="1" />

        <!-- DAC1->Headphone -->
        <ctl name="Left Headphone Mux" val="DAC" />
        <ctl name="Right Headphone Mux" val="DAC" />
```

Configuration example

```
<device name="headphone">
  <path name="on">
    <ctl name="HP Switch" val="1" />
  </path>
  <path name="off">
    <ctl name="HP Switch" val="0" />
  </path>
</device>
```

Future plans

- Record support
- Baseband support
- Dynamic power optimisations
- Algorithm plugins
- Support for explicit use case transition sequences
- Support for new ALSA features

- Contributions welcome!

More info & any questions

- **http://opensource.wolfsonmicro.com/content/tinyhal**