Sound Open Firmware

Liam Girdwood - Intel
Sound Open Firmware is an open source audio digital signal processing firmware and driver infrastructure.
SOF Goals 1

- DSP architecture and platform agnostic.
  - Separate architecture and platform code like Linux.
  - Abstract interfaces for architecture APIs (again like Linux).
- Host AP architecture and host AP OS agnostic.
  - Drivers are generic and not coupled to any host architecture.
  - Nothing in SOF that couples firmware or low level driver to any OS.
- Toolchain freedom
  - Fully support open source tools – GCC, Octave, M4, Qemu
  - Also support proprietary tools - Cadence xcc and emulator, Matlab.
- Permissive licensing of all firmware and most SDK code.
**SOF Goals 2**

- Freedom for users to define new audio processing pipelines using opensource components and tools.
- Freedom for users to develop and integrate proprietary audio processing algorithms.
- All development done in public.
- Support for several CI systems to best match device and infrastructure under test.
**SDK Birds Eye View 1**

**SOF Source Code**
- Xtos OS
- Zephyr OS
- NuttX OS
- Crosstool-NG Tool Chain

**SOF Linux Driver**
- /lib/firmware/intel
- Topology Metadata
- m4 topology file (*.m4)
- ITT
- ALSA topology file (*.conf)
- alsaplg

**SOF Firmware**
- ELF Image
- Signing Tool
- rmbox / trace Tool
- debugfs

**Audio DSP**
- IPC-Mailbox
- ALSA SoC Topology

**Legend**
- SOF Source
- 3rd Party Tool
- SDK Binary Image
- Optional
- Config Data

*Note: Docker contains only the open source components*
SDK Birds Eye View 2

Algorithm Performance Criteria

Matlab → GNU Octave → Coefficients → SOF Linux Driver → SOF Firmware → Audio DSP

SOF Test Bench

Test Algorithm.

Legend:
- SOF Source
- 3rd Party Tool
- SDK Binary Image
- Optional
- Config Data

*Note: Docker contains only the open source components
Firmware Overview

- Architecture agnostic implementation in C with some assembler.
- Permissive BSD/MIT licensed code.
- Build time selection of features based on target (moving to use Kconfig).
- Scales down to ~40kB footprint (text, data and bss).
- Support for user defined audio processing pipeline topologies.
  - Dynamically loaded pipelines at runtime.
  - Statically stitched into image for host less operation.
- Allow general compute and signal processing. Not just an audio FW.
- Image generation tools provided to
  - Convert ELF to device specific firmware image formats.
  - Sign firmware images with PKCS#1.5
Firmware Architecture

X86 or ARM Host

Linux
ASoC Audio Driver

RTOS/HV Audio Driver

SOF HW/IPC driver

DSP or Discrete Chip

Control Data

SOF Audio Components

mixer
volume
ASRC
AEC/NS
mux
EQ
buffer queue
speaker protection
splitter
SRC
tone

Audio Tasks

IPC
Configurable pipelines

Generic Micro Kernel

(XTOS, Zephyr, FreeRTOS, NuttX options)

boot
msg queues
work queues
memory
IRQs
schd
mutex
exceptions
timer
threads
atomic services
semaphores

Platform Drivers

DMA
HDA
I2C
DMIC
GNA
SPI
SSP
USB
UART
GPIO

2 of 2 Firmware diagram – complete diagram
Driver Overview

- Architecture agnostic.
- Dual GPL/BSD licensed code.
- Hardware and physical IO abstraction, so can work equally well with local MMIO and remote SPI based DSPs.
- Supports virtualisation through VirtIO backend and frontends (using ACRN).
- Driver has no hard coded assumptions about DSP pipelines or components.
  - Topologies loaded from FS.
  - Component configurations can be runtime updated via ALSA kcontrols.
Driver Architecture

ASoC Machine Driver
- codec integration
- board integration
- HW config

ASoC PCM Driver
- topology
- PCMs
- Kcontrols
- DAPM

Generic IPC Driver
- mixer
- stream
- PM
- pipeline

DSP Platform Driver
- doorbell
- mailbox
- IRQ
- code loader
- IO
- PM

Physical IO layer can currently be either PCI, SPI, MMIO or VirtIO.
Pipeline Architecture

- A pipeline is a set of audio processing components and buffers that share common scheduling.
- Pipelines can currently be defined using M4 in a few lines of code. Plans for GUI.
Continuous Integration - Travis

*Note: Docker contains only the open source components*
Continuous Integration - 0day

*Note: Docker contains only the open source components.*
## SOF Platforms

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Platform</th>
<th>DSP</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Host PC</td>
<td>Host PC</td>
<td>Testbench</td>
</tr>
<tr>
<td>Qemu</td>
<td>Host PC</td>
<td>Qemu targets</td>
<td>Currently only has support for xtensa based devices.</td>
</tr>
<tr>
<td>Intel</td>
<td>Baytrail, Cherrytrail, Brasswell</td>
<td>1 * Xtensa HiFi2EP</td>
<td>Upstream, support for upto 6 I2S DAIs.</td>
</tr>
<tr>
<td>Intel</td>
<td>Haswell, Broadwell</td>
<td>1 * Xtensa HiFi2EP</td>
<td>Upstream, need I2S integration for some codecs.</td>
</tr>
<tr>
<td>Intel</td>
<td>Apollolake, Geminilake</td>
<td>2 * Xtensa HiFi3</td>
<td>Upstream, support for I2S, DMIC, HDMI and HDA.</td>
</tr>
<tr>
<td>Intel</td>
<td>Cannonlake, Whiskylake</td>
<td>4 * Xtensa Hifi4</td>
<td>Upstream, support for I2S, DMIC, HDMI and HDA.</td>
</tr>
<tr>
<td>Intel</td>
<td>Suecreek HAT on Rasberry PI</td>
<td>2 * Xtensa HiFi4</td>
<td>Upstream core, boot over SPI WiP</td>
</tr>
<tr>
<td>Xiaomi</td>
<td>ARM M4</td>
<td>CEVA Teaklite DSP</td>
<td>Upstreaming WiP.</td>
</tr>
</tbody>
</table>
# SOF Processing Components (opensource)

<table>
<thead>
<tr>
<th>Name</th>
<th>Generic C available</th>
<th>SIMD support</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>Yes</td>
<td>Xtensa HiFiEP, HiFi3</td>
<td>Upstream, can process 1 – 8 channels, 16, 24 and 32bit formats.</td>
</tr>
</tbody>
</table>
| SRC | Yes | Xtensa HiFiEP, HiFi3 | Upstream. 24 and 32bit formats. FIR polyphase optimized M/N and M/N x O/P fractional rate conversions.  
Configuration tools upstream. |
| FIR | Yes | Xtensa HiFiEP, HiFi3 | 16/24/32 bit formats, up to 192 taps.  
Configuration tools upstream. |
| IIR | Yes | Planned | 16/24/32 bit formats, up to 11 biquads or 22th order.  
Configuration tools upstream. |
History

Year ending

2015

Inception

Platforms: Bay Trail
Features: Qemu Xtensa

2016

Firmware open sourced
Cherry Trail and Braswell
Features: Qemu host

SOF 1.0 public release

Driver open sourced
Lake, Haswell, Broadwell
Features: BSD/GPL driver, topologies

Became Linux Foundation project

2017

Driver upstream
Platforms: Cannon Lake, Ice Lake,
Sue Creek, TeakLite
Features: Docker build, SOF on host

2018
# SOF Future Plans

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDB stub and tunnel.</td>
<td>Initial firmware and kernel code in developer repository. Upstream in 2019.</td>
</tr>
<tr>
<td>RPMSG</td>
<td>Code complete, now upstreaming to SOF and Linux</td>
</tr>
<tr>
<td>CMSIS RTOS Abstraction layer</td>
<td>Work in progress.</td>
</tr>
</tbody>
</table>
Supporting Organizations

- Intel
- ALSA
- Google
- Pinecone
- Xiaomi
Firmware Preference?

- Linux already has a preference for open source drivers over closed source drivers.
- Should this preference also be extended to open source firmware in the future?
- Selection criteria would also have to consider:
  - Features.
  - Stability.
  - Portability.
- Some open source firmware could still contain binary blobs.
  - Regulatory reasons (e.g. comply with FCC regulations).
  - 3rd party IP (e.g. some audio processing algorithms).
Q & A

www.sofproject.com
https://github.com/thesofproject

Visit the booth for demo.
Qemu Heterogeneous Virtualization
Baytrail DSP Architecture

- Xtensa HiFi 2EP core.
- 96kB Instruction RAM
- 168kB Data RAM
- 2 * DMACs
- 3 * I2S ports
- PCI device from host OS
- Firmware and host share
  - SHIM registers
  - Mailbox memory.

Must be shared with IA and Xtensa VMs.
Heterogeneous Virtualisation

- Firmware must be debugged alongside driver.
- Qemu used to virtualise drivers and firmware together.
- Host side almost real time.
- DSP side emulated.
Heterogeneous IO bridging

- X86 Qemu/KVM
- Guest OS *
- VM #0

- PCI BAR 0
- /dev/shm

- X86 GDB
  - On host

- Xtensa Qemu
- SOF Firmware
- VM #1

- PCI BAR 1
- /dev/shm

- IRQ
- /dev/mqueue

- xtensa GDB
  - On host
Zero Copy Virtual Audio
Zero Copy Audio Virtualisation

SOS buffer PHY pages shared with guests.

SOS allocates SG PHY pages for guest audio buffers. Audio DMA copies directly from/to these buffers.