Sound Open Firmware

Liam Girdwood - Intel

Sound Open Firmware is an open source audio digital signal processing firmware and driver infrastructure.

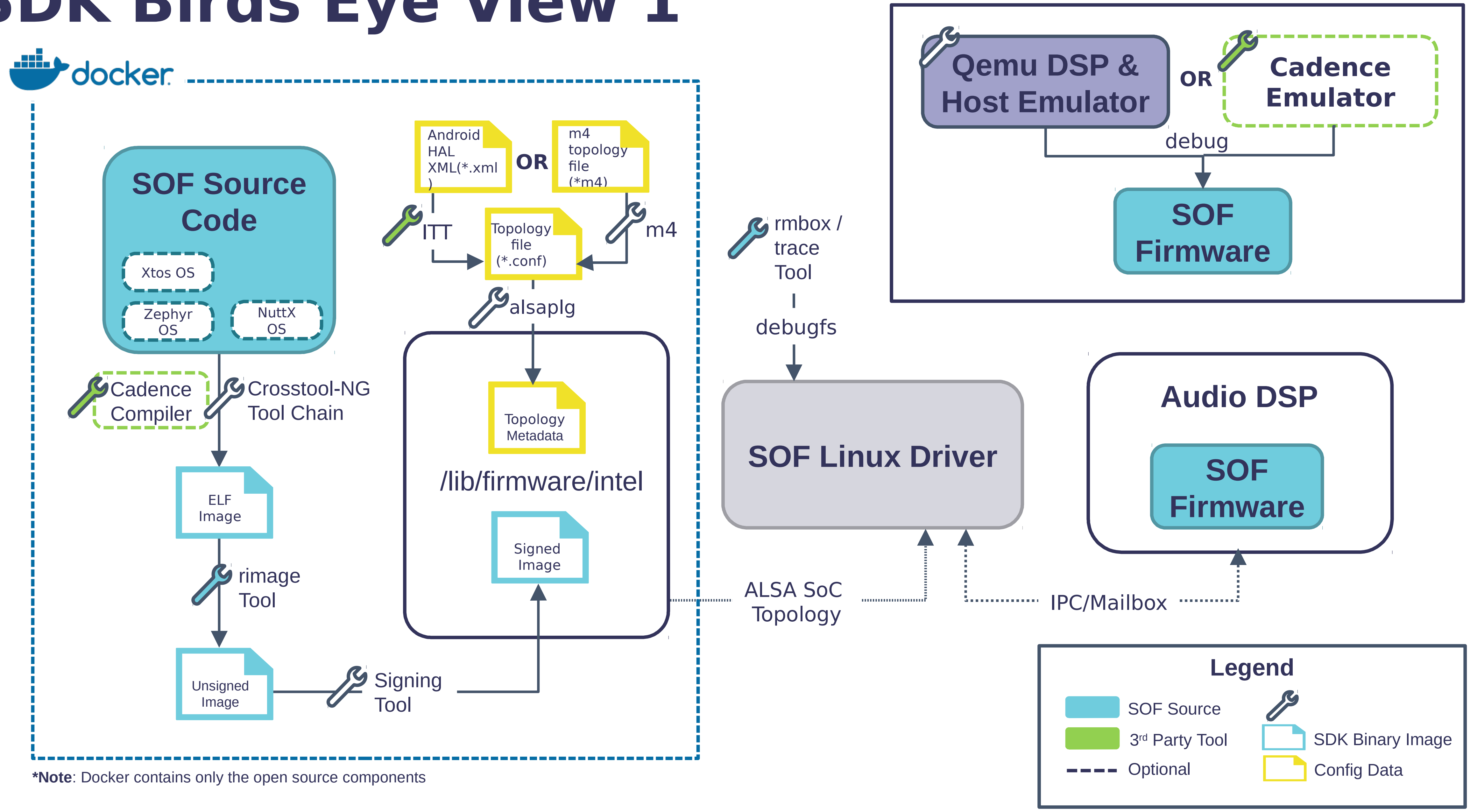
SOF Goals 1

- 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

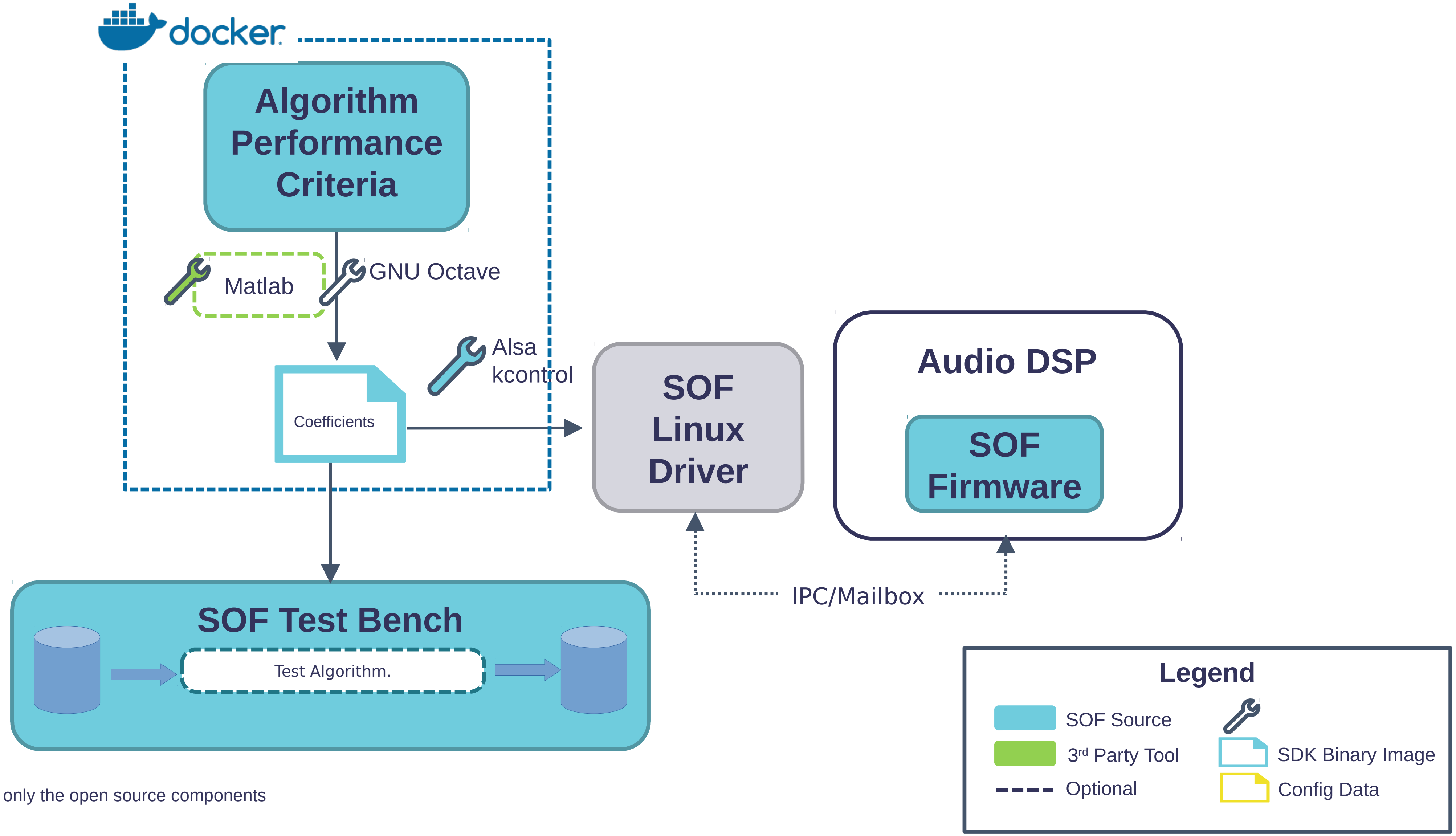
- Freedom for users to define new audio processing pipelines using opensource components and tools.
- Freedom for users to develop and integrate proprietary audio processing algorithms.
- All development done in public.
- Support for several CI systems to best match device and infrastructure under test.

SDK Birds Eye View 1



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

SDK Birds Eye View 2



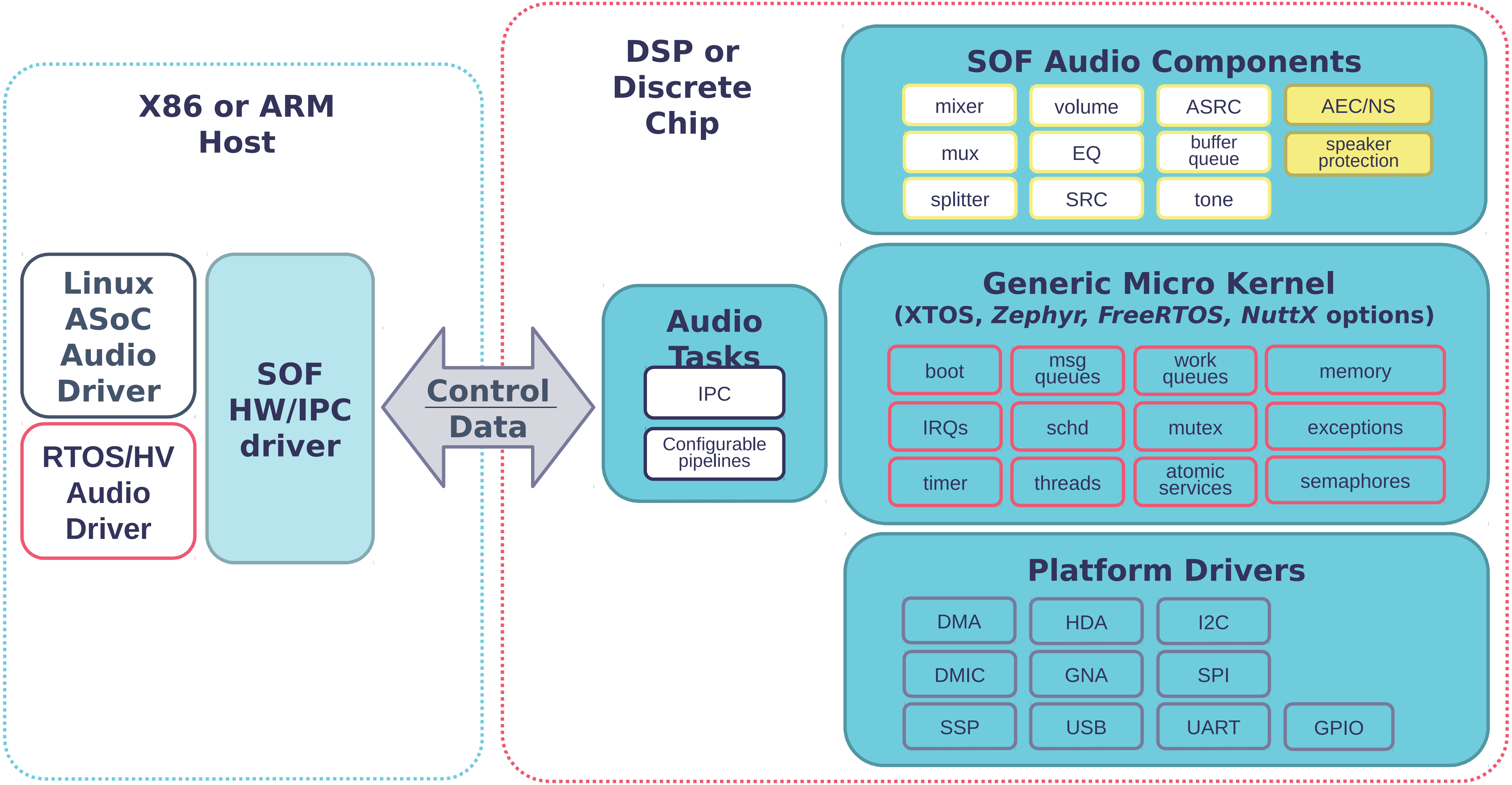
***Note**: Docker contains only the open source components

SDK diagram – complete SDK

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

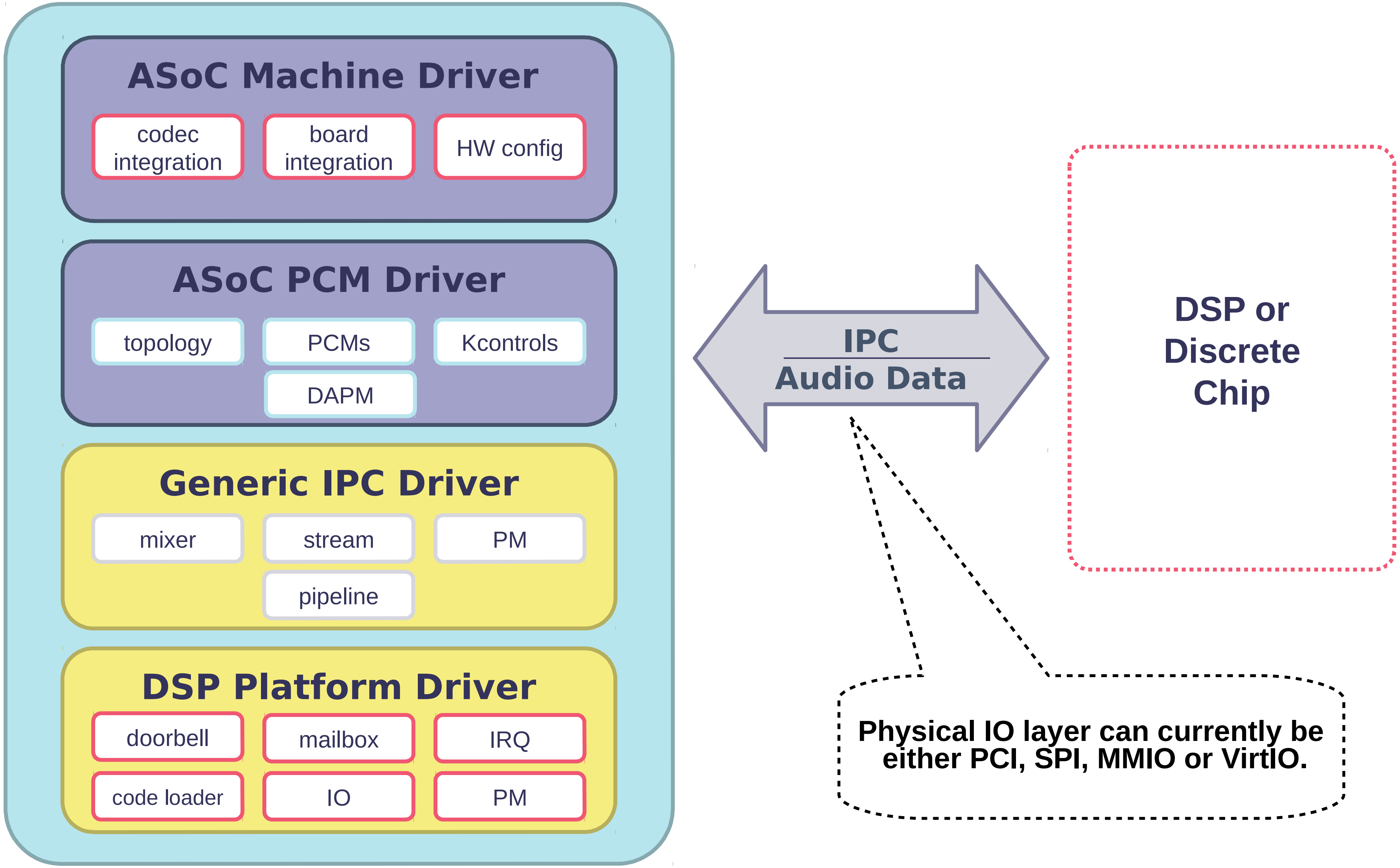


2 of 2 Firmware diagram – complete diagram

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.

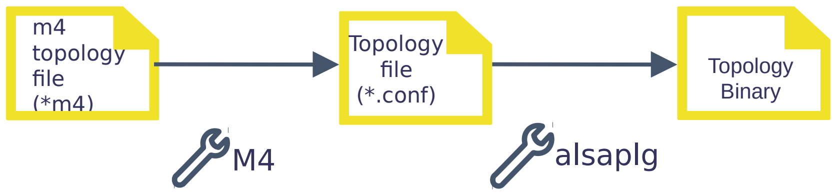
Driver Architecture

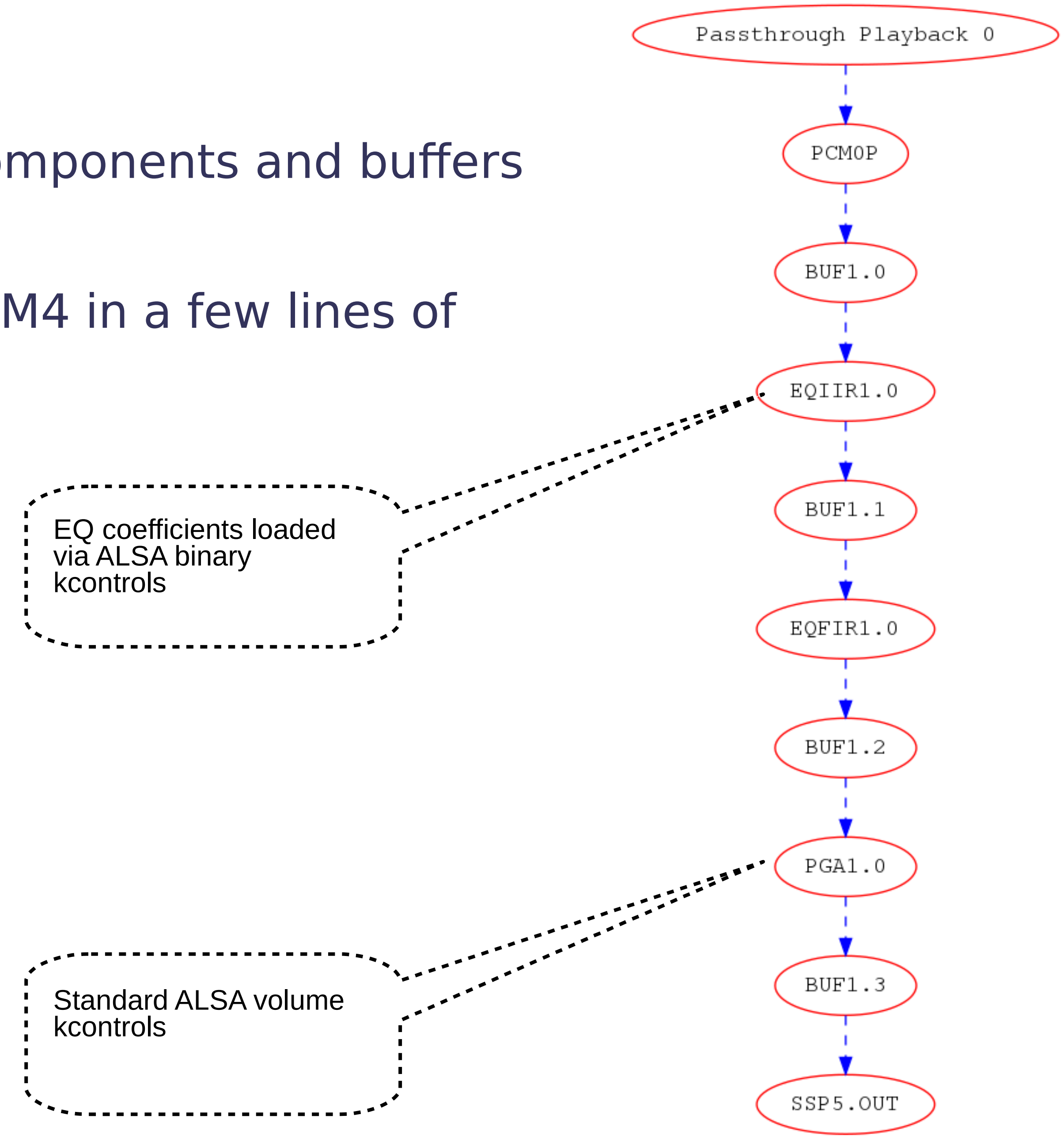


2 of 2 Driver diagram – complete diagram

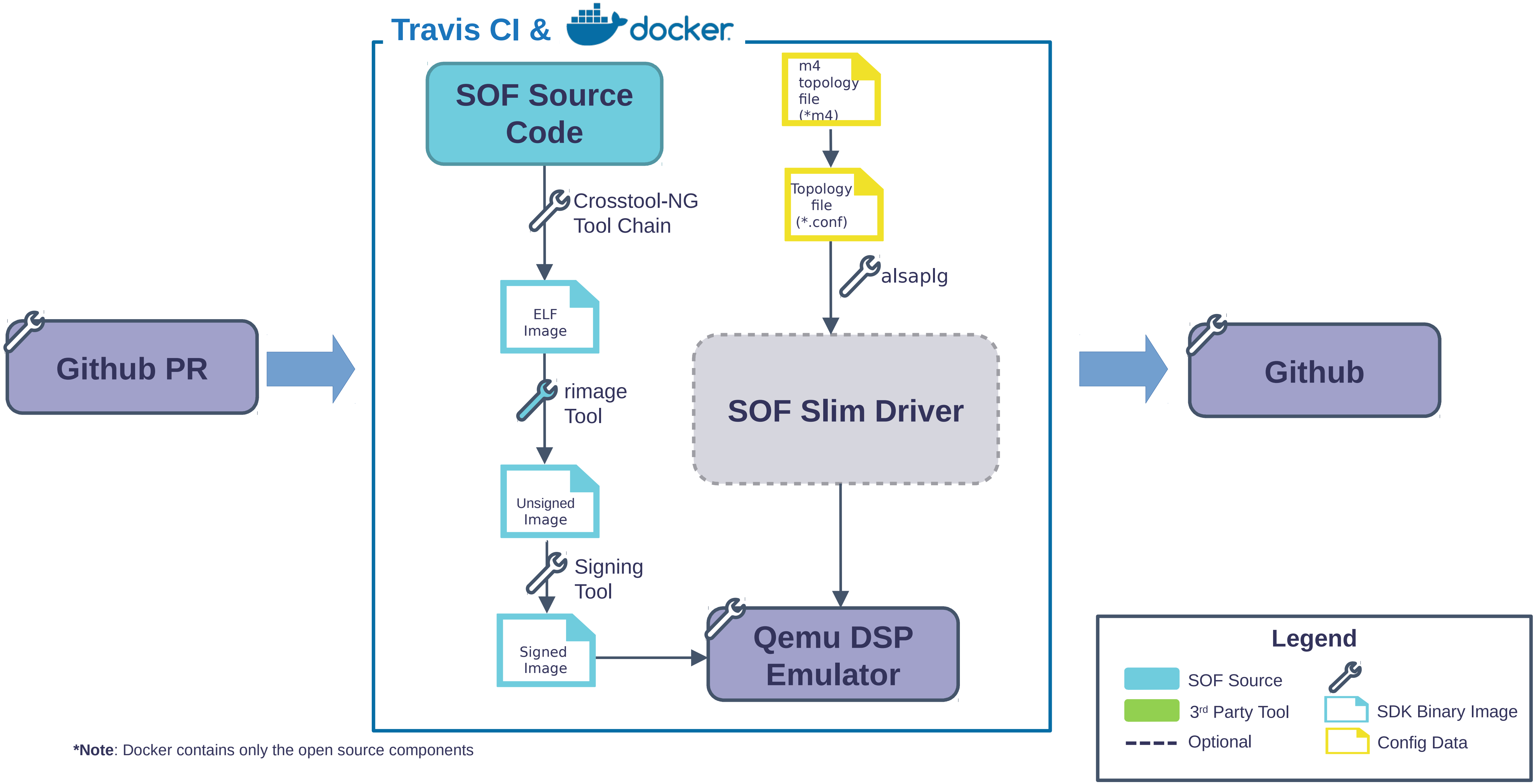
Pipeline Architecture

- 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.



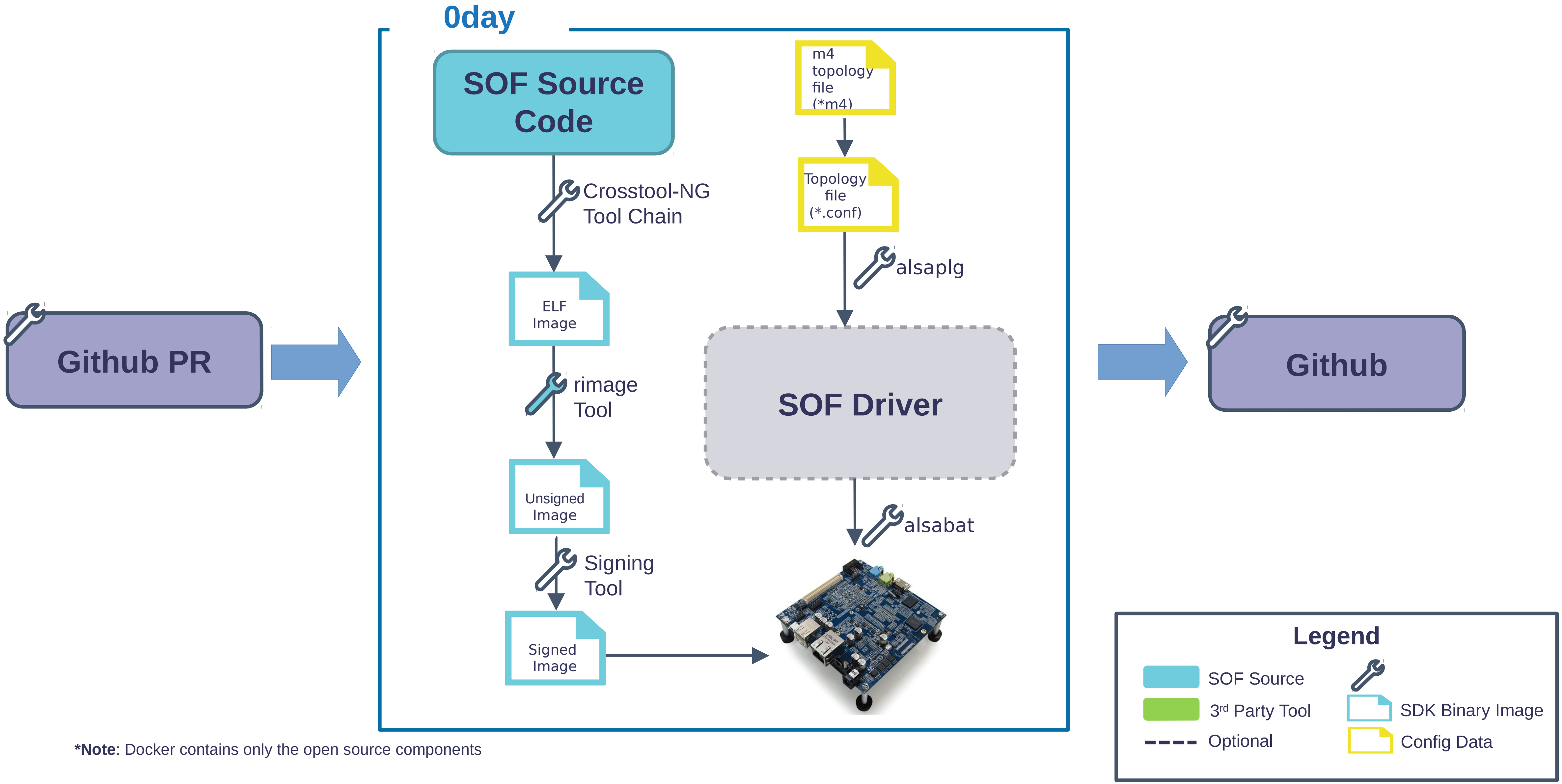


Continuous Integration - Travis



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

Continuous Integration - 0day



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

SDK diagram – complete SDK

SOF Platforms

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

SOF Processing Components (opensource)

Name	Generic C available	SIMD support	Status
Volume	Yes	Xtensa HiFiEP, HiFi3	Upstream, can process 1 – 8 channels, 16, 24 and 32bit formats.
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 2016 2017 2018

Inception

Platforms: Bay Trail
Features: Qemu Xtensa

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

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

SOF Future Plans

Feature	Status
GDB stub and tunnel.	Initial firmware and kernel code in developer repository. Upstream in 2019.
RPMSG	Code complete, now upstreaming to SOF and Linux
Loadable firmware modules	Initial code in github, still WiP. Upstream in 2019.
CMSIS RTOS Abstraction layer	Work in progress.

Supporting Organizations

intel

ALSA

Google

pinecone

Xiaomi

Firmware Preference ?

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

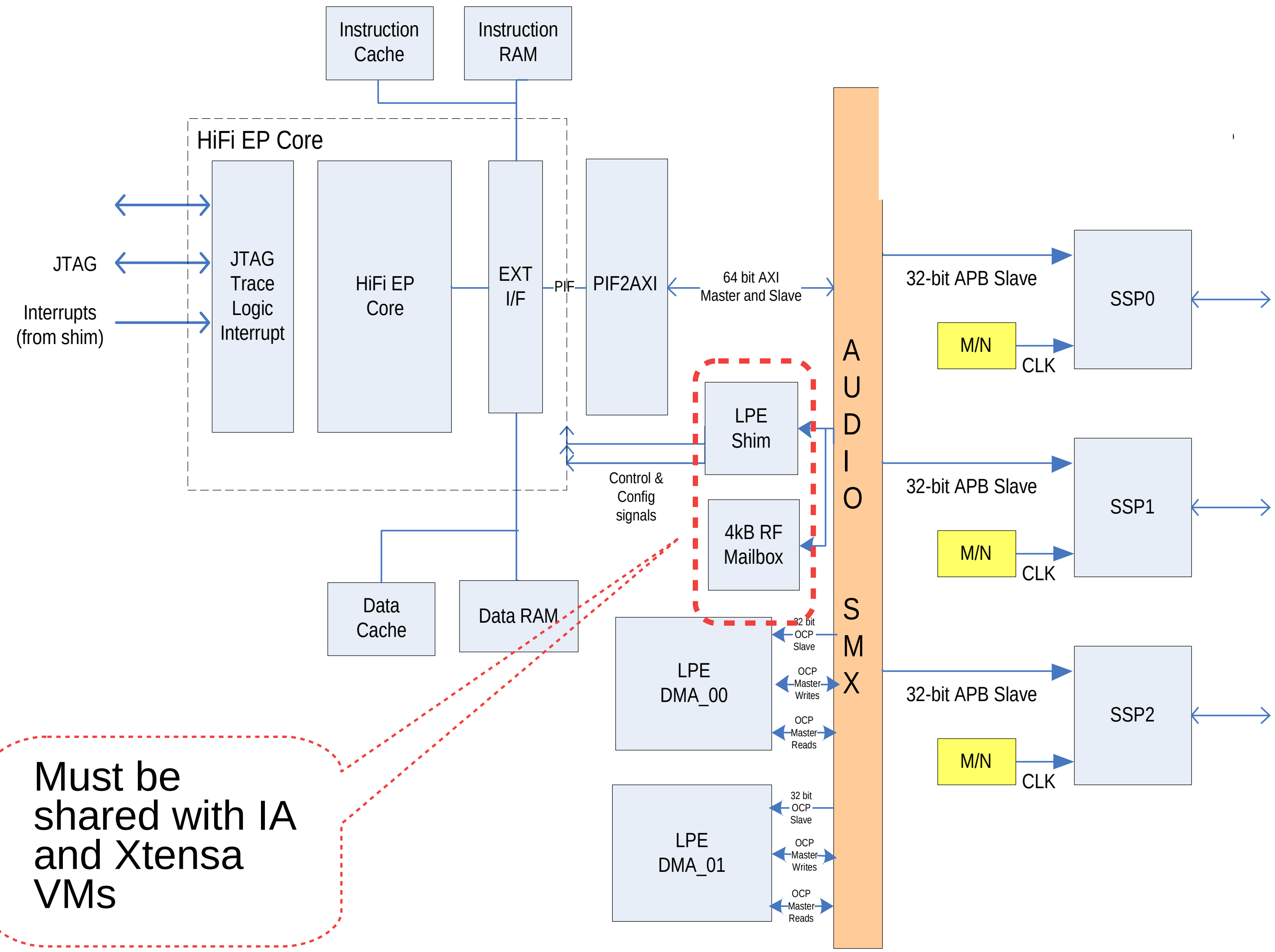
Q & A

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

Visit the booth for demo.

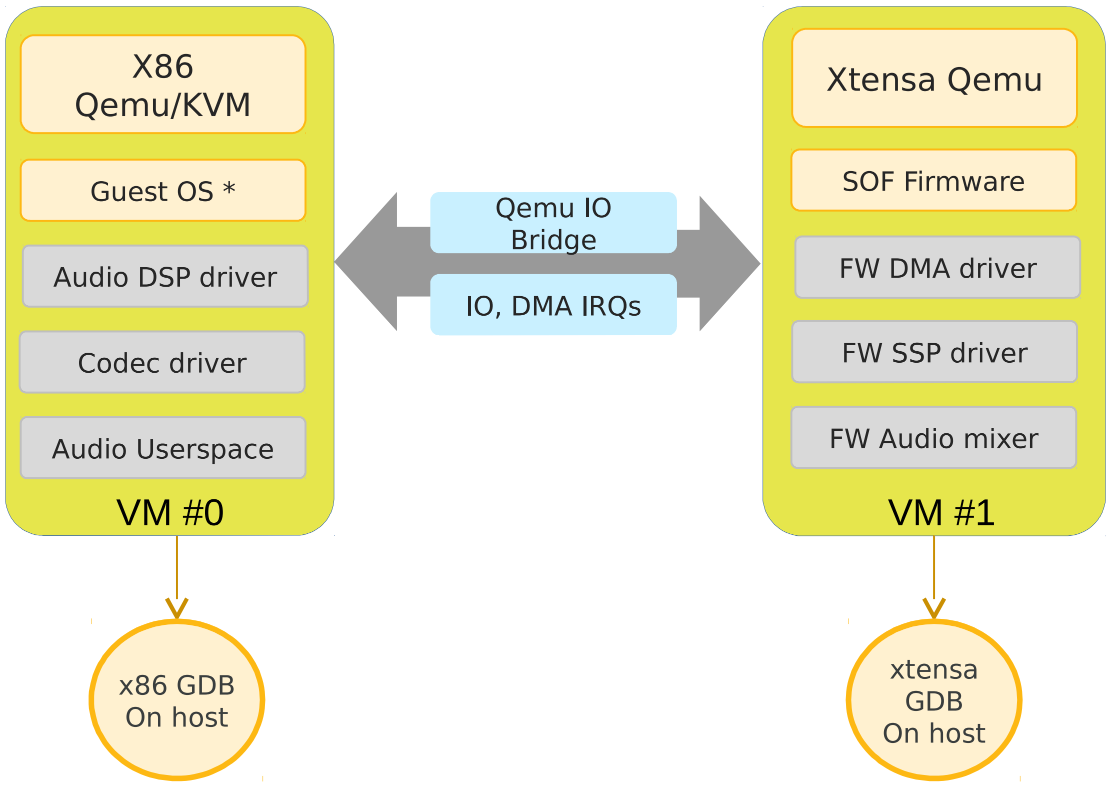
Qemu Heterogeneous Virtualization

Baytrail DSP Architecture



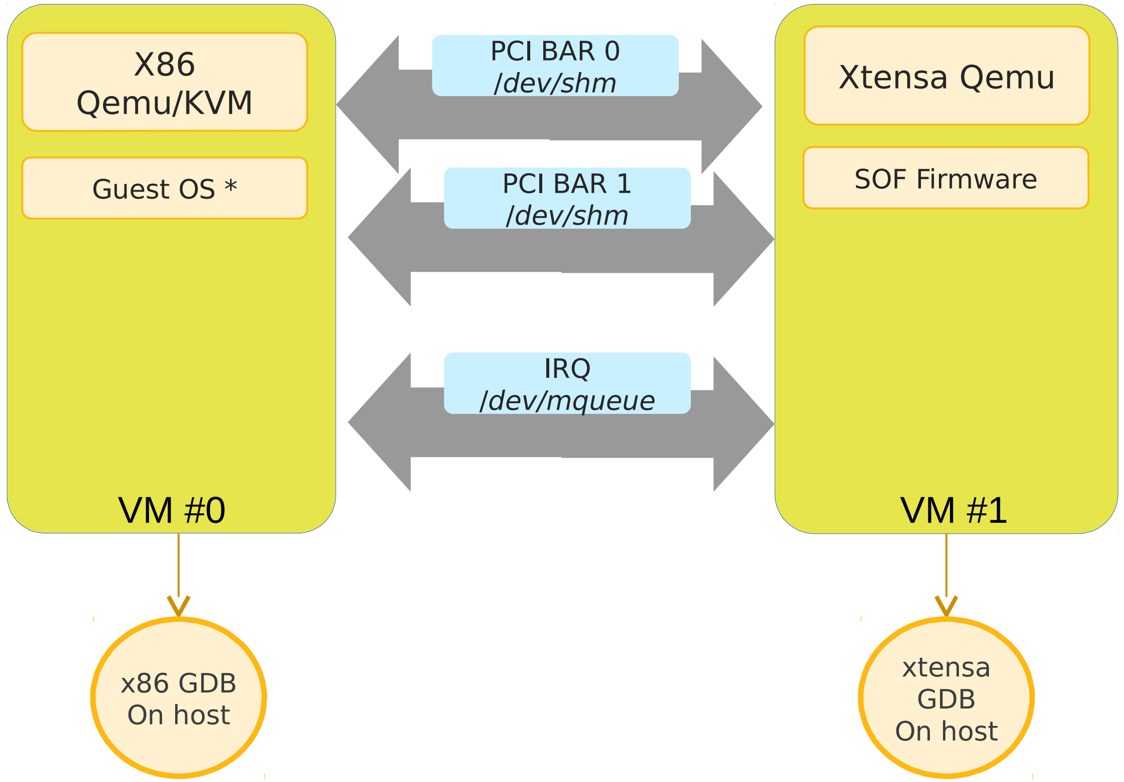
- Xtensa HiFi 2EP core.
- 96kB Instruction RAM
- 168kB Data RAM
- 2 * DMACs
- 3 * I2S ports
- PCI device from host OS
- Firmware and host share
  - SHIM registers
  - Mailbox memory.

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



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.