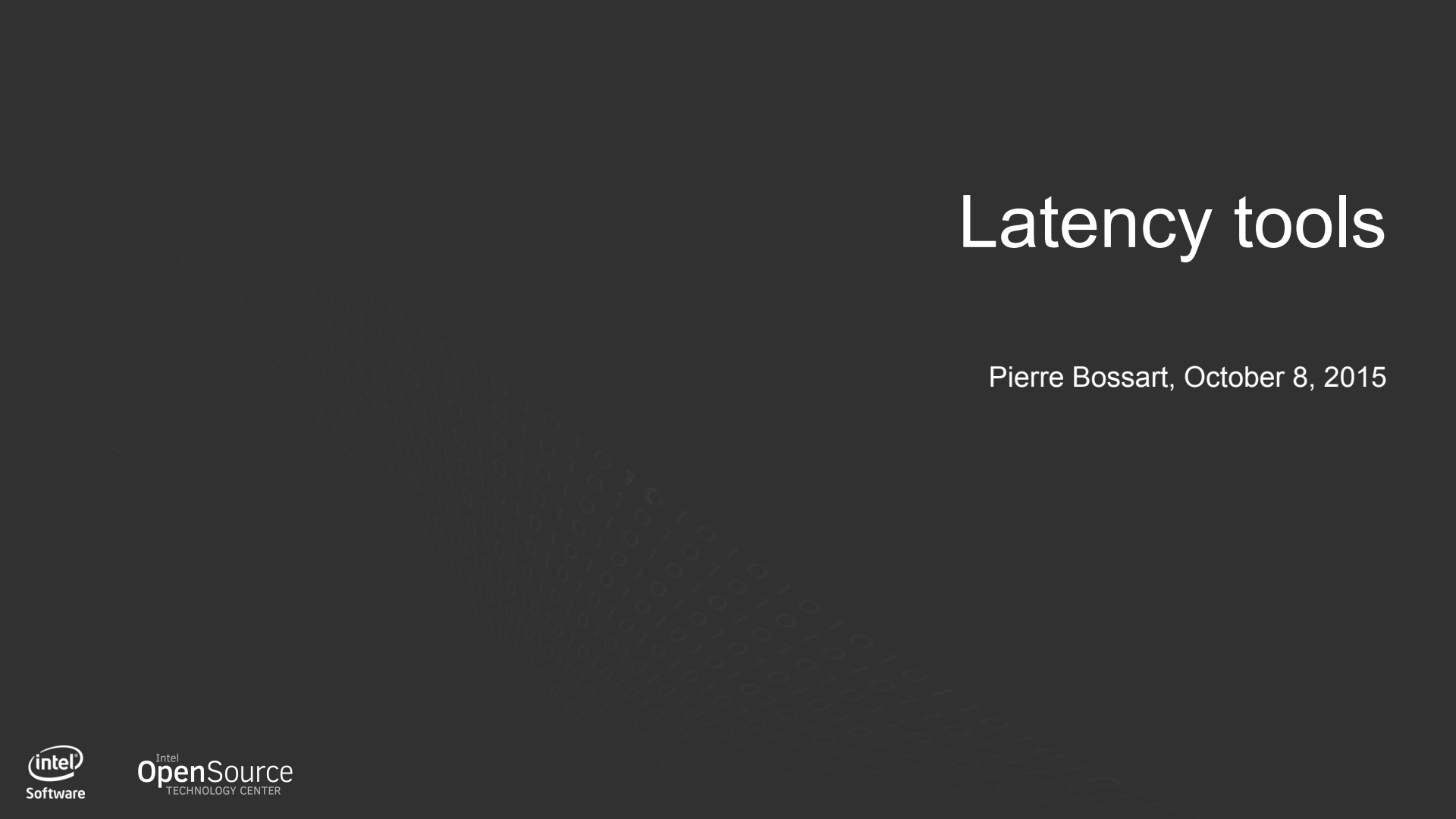
# Latency tools

Pierre Bossart, October 8, 2015



intel Software
Intel OpenSource TECHNOLOGY CENTER

# Latency tools

Code on https://github.com/01org/AudioLatencyTools

App to measure

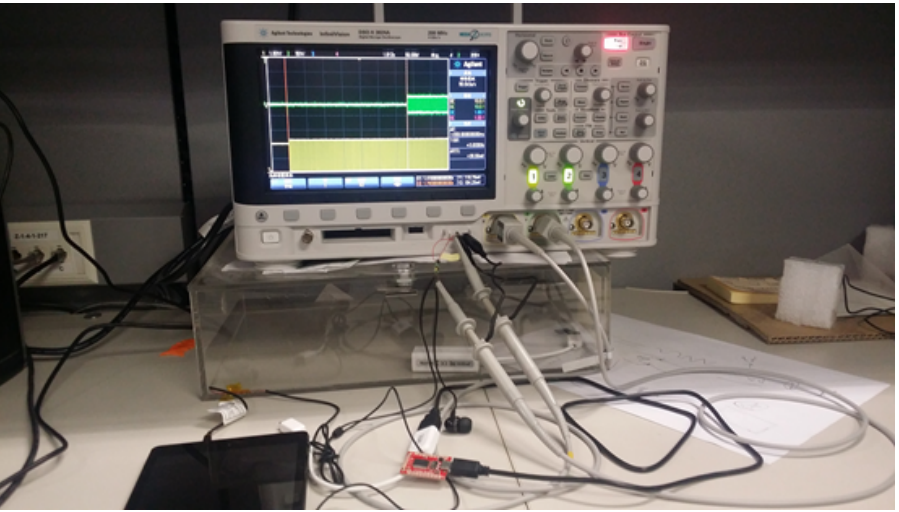
- Cold latency
- Continuous latency

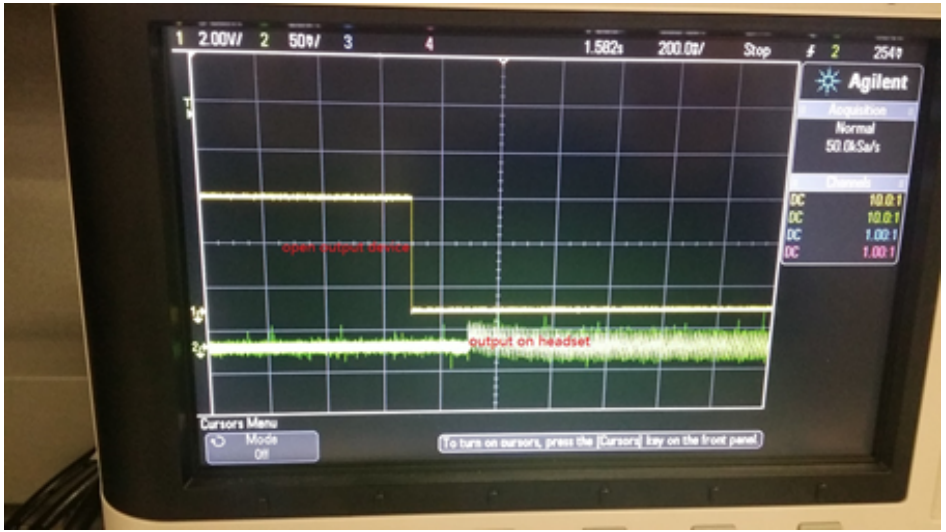
Based on FTDI USB-to-serial -> gpio available on all Android and Chrome platforms

App toggles GPIO before action, scope grabs gpio and analog headset output

~1ms resolution, limited jitter

# Latency tools





# Power consumption optimizations:

Pierre Bossart, October 8, 2015

intel Software
Intel OpenSource TECHNOLOGY CENTER

# Patches coming to mainline

- DeepBuffer (typically ~100ms or more)
  - Available on most Android Nexus devices
  - Will be available on all Intel platforms
  - Output mixed in hardware/firmware, used when latency does not matter.
  - Pulseaudio: need notion of sink groups
- Need to disable rewinds on deep-buffer
  - Help tell hardware it can fetch more data from ring buffer
- Provide DMA burst information for cases where rewinds are still used
  - Worst-case

# ACPI & audio drivers:

Pierre Bossart, October 8, 2015

intel Software
Intel OpenSource TECHNOLOGY CENTER

# ACPI and DeviceTree: why can't we be friends?

- ARM platforms use DeviceTree bindings
- New solution introduced in 3.19 to provide similar functionality
- Typically straightforward code change
  - device_property_xxx() instead of of_property_xxx()
  - Representation in BIOS uses _DSD properties
    - Introduction from Rafael Wysocki at http://events.linuxfoundation.org/sites/events/files/slides/ACPI_vs_DT.pdf

# ACPI support and _DSD properties

- May require upgrade to gpiod framework
- Less straightforward as DeviceTree for hierarchical information, additional information adopted by ASWG
  - http://www.uefi.org/sites/default/files/resources/_DSD-hierarchical-data-extension-UUID-v1.pdf
  - Only indirect nesting supported.

- To test codec in ACPI environment
  - Option1: be selected as a lead partner
  - Option2: use the MinnowBoard Max
    - Open hardware, open firmware, works with your distro of choice and Android.
    - Minor modifications to set _HID and _DSD properties in firmware
    - Use low-speed connector to wire-up I2C and I2C

# ACPI and machine drivers

- No direct way to use DeviceTree/simple-card bindings
  - Phandle not modeled
  - No top-level mode for sound card, not really compatible with Windows-oriented BIOS.
- Current solution:
  - Detect board based on combination of SOC+codec pair and load relevant machine driver
  - Some DMI-based quirks
- Desired solution being worked on
  - Use audio configuration passed by BIOS
  - _DSD properties to set quirks or board changes.

# SoundWire Linux support:

Pierre Bossart, October 8, 2015

intel Software
Intel OpenSource TECHNOLOGY CENTER

# SoundWire in a nutshell

- MIPI standard currently accessible to MIPI members only
- Public information released: AES paper and webminar
  - Mipi.org . Learning Center . MIPI Alliance SoundWire Overview Webinar

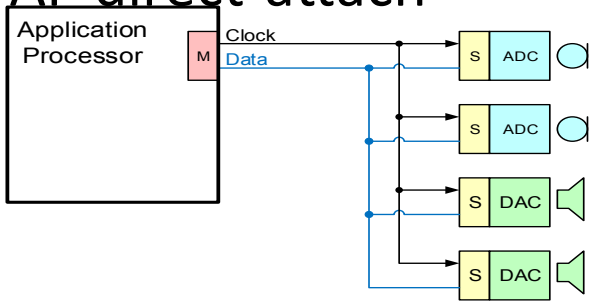
# SoundWire introduction

- Timelines
  - Initial discussions in June 2012, ratified in February 2015
  - Contributions from ~16 MIPI contributor companies
- Key features
  - Two-pin dual-data rate multi-drop bus for audio applications (1.2 or 1.8V)
  - Robustness and Scalability (clock and multiple lanes)
  - Low power, low latency, well-bounded (PHY and transport)
  - Support for multiple streams, formats (PCM/PDM/DATA), modes (isoc/async/block)
  - Embedded control/commands
  - In-band interrupts/wakes, support for low-power jack detection
- Benefits:
  - New use cases not possible with existing interfaces ($I^2S$, SLIMbus®, HDAudio)
  - New system topologies across mobile and mobile-influenced industries
  - Lower gate count allows for integration in cost-sensitive devices

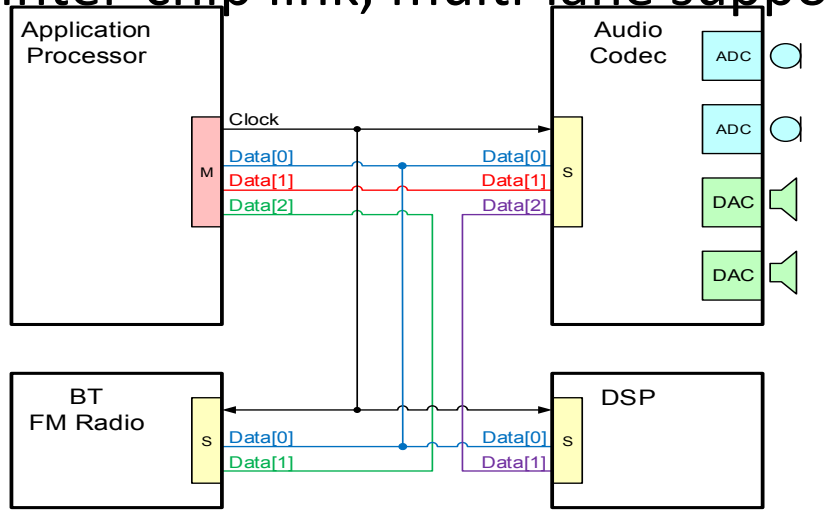

© 2015 MIPI Alliance, Inc. All rights reserved.

# Example topologies

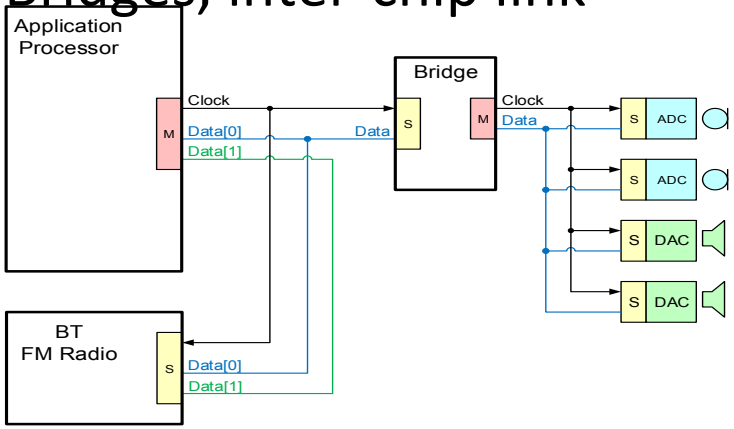
## AP direct-attach




## Inter-chip link, multi-lane support




## Bridges, inter-chip link

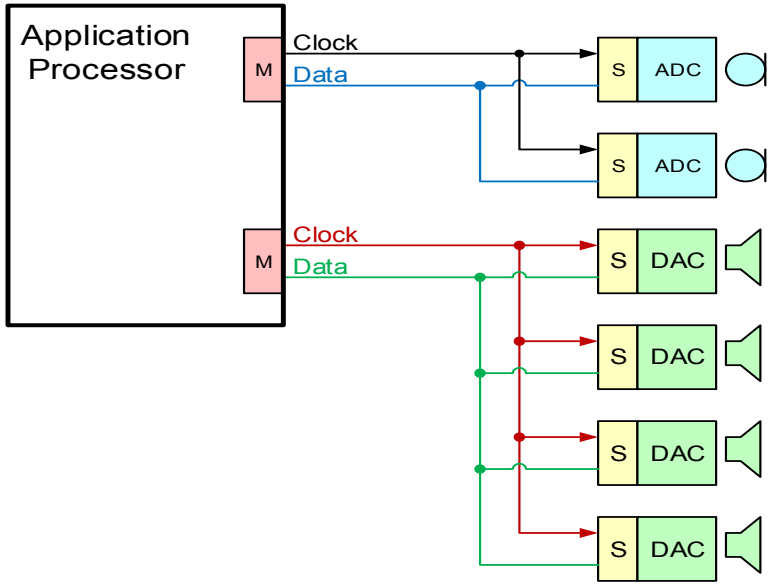




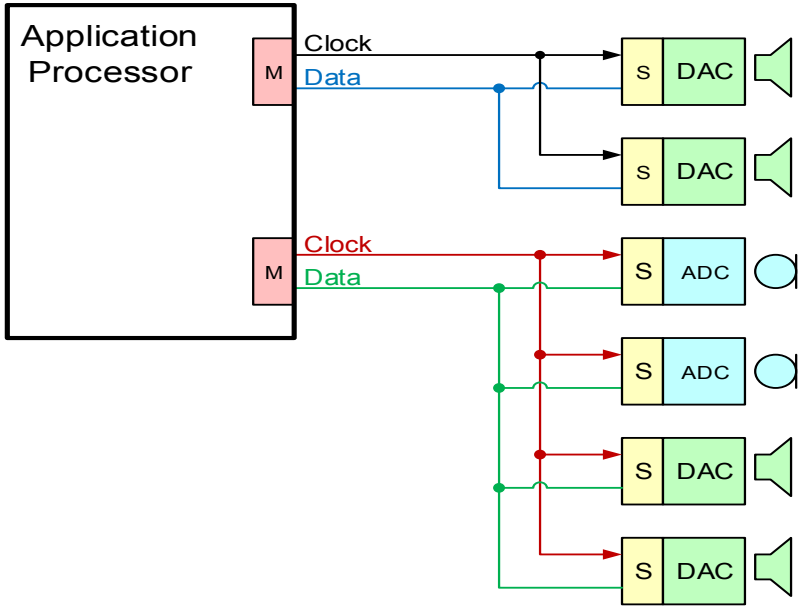

© 2015 MIPI Alliance, Inc. All rights reserved.

# Example topologies (2)

## Functional partitioning




## Routing/use case partitioning





© 2015 MIPI Alliance, Inc. All rights reserved.

# Comparison with other interfaces

| Other bus | Pro SoundWire | Con SoundWire |
|---|---|---|
| **I²S/TDM** | Lower pin count, clock scaling, dynamic slot mapping, burst mode, command embedded with data (No need for I²C/SPI), in-band interrupt capability (no need for GPIO), support for PDM | Slight command overhead, no ability to switch Master/Slave roles for clock |
| **PDM** | Clock scaling, embedded command and control, interrupt capability | Overhead is 70% for dual-mic, less than 5% for single mic-mode. In multilane mode power consumption is lower than PDM. |
| **HDAudio** | Clock scaling, lower pin count, support for PDM, scales to simple devices | Lower bandwidth device class functionality not yet standardized |
| **SLIMbus** | Lower gate count allows for integration in cost-sensitive devices, simpler protocol, low latency PDM support, lower power with adjustable Frame size and double-data rate | No clock and manager hand-over capabilities. Only Master and Monitor can send messages. |

# SoundWire impact on Linux

- Need for generic bus driver to handle link configurations
  - Device enumeration and assignment of DeviceNumber (logical address)
  - Clock changes, bandwidth allocation (dynamic or presets)
  - Stream start/stop sequences
- Codec driver handles all audio/non-standard features
  - Similar to I2C, abstracted through regmap
- SoundWire device capabilities
  - Extracted from firmware – MIPI document with list of properties to be released
    - Can help share code between implementations, e.g. for enabling transport
  - Or hard-coded in codec driver

# SoundWire/Linux: known issues to sort out

- Different topologies allowed by spec
  - SoundWire Master can be in SOC/Chipset
  - SoundWire Master can be in audio codec (bridged over Hdaudio or I2S/I2C)
- No mandatory master registers
  - Need to abstract out an interface that caters to different implementations.
- ‘Atomic’ reconfigurations on Master(s) and Slave(s)
- Mapping ‘Ports’ to DAIs
  - DAI concept was straightforward with point-to-point connections
  - Was extended with TDM links but was still 1:N
  - Gets more complex with SoundWire M(p):N(q)
    - Same ‘logical stream’ can originate from one or more Source ports in one or more devices and be consumed by one or more Sink ports in one or more devices
    - Same ports can be repurposed based on use case
    - Ports can be reconfigured as sink or source