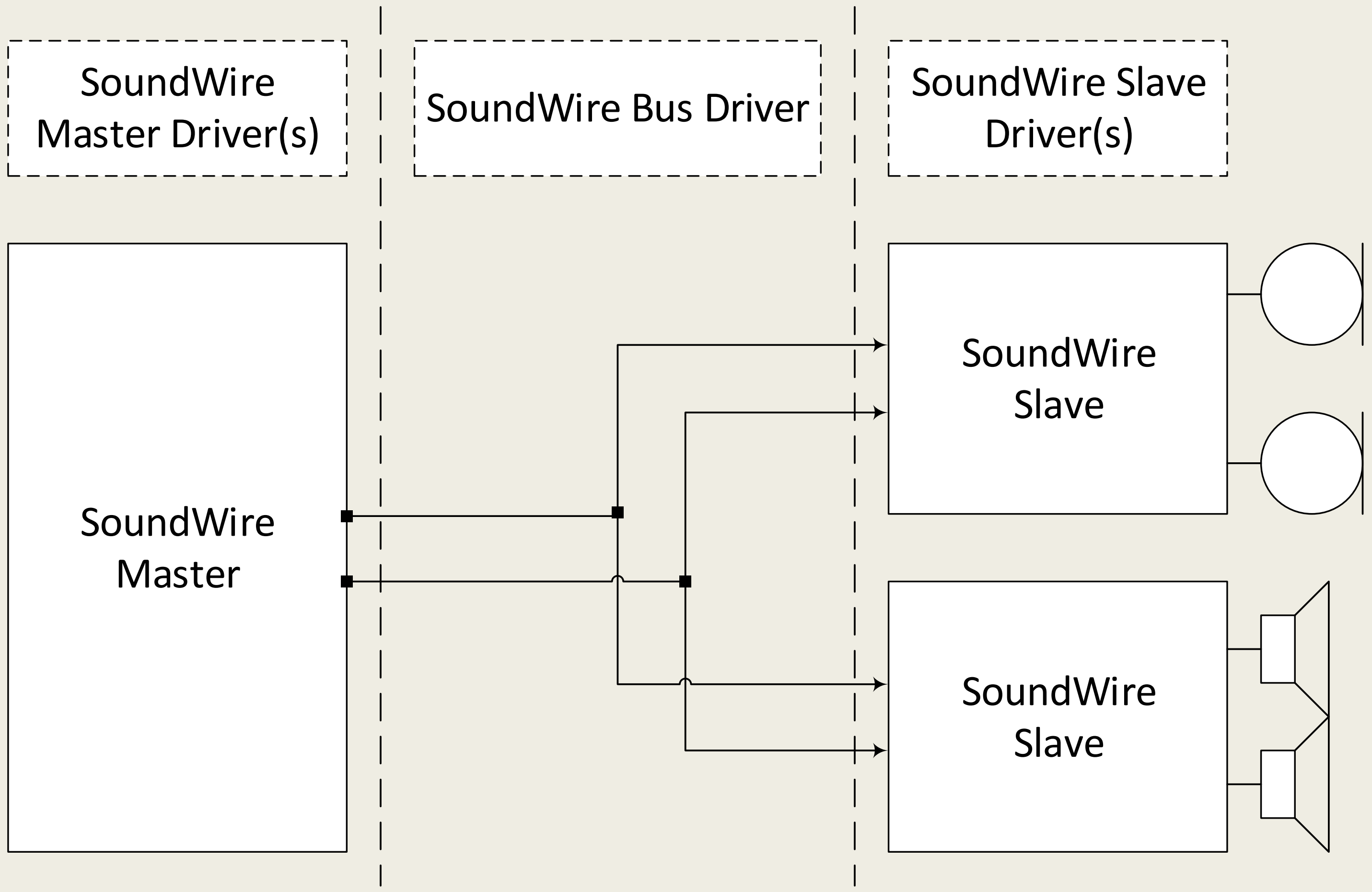
LINUX SOUNDWIRE DRIVERS

Rakesh Ughreja, Pierre Bossart

Acknowledgements: Hardik Shah, Sanyog Kale, Vinod Koul

Typical SoundWire System



SoundWire Bus driver

- SoundWire bus management and synchronization for all the bus instances
- Provides APIs for SoundWire Master and Slave drivers for registration
  - *Standard _DSD properties to be released in Q1'17*
- Enumerates Slaves present on the bus
- Manages bandwidth and clock based on the audio stream configurations
- Power management
- Handles standard events coming from Slaves
  - *Routes vendor specific events to respective Slave driver*
- Configures standard Slave registers
- Note: streams are defined by machine driver DAILinks, not hard-coded in bus driver

SoundWire Bus driver API

- SoundWire bus driver API are classified into three categories
- APIs which are called only by Master driver
  - *include/sound/sdw_master.h*
- APIs which are called only by Slave driver
  - *include/sound/sdw_slave.h*
- APIs which can be called by both Master and Slave drivers
  - *include/sound/sdw_bus.h*

SoundWire Master Driver

- Configuration and control of SoundWire Master interface
- Register Master interface with Bus driver
  - *Bus driver creates bus instance and enumerates Slave instances*
- Master registers are not defined by SoundWire specification
- Expose SoundWire Master capabilities to SoundWire bus driver
- Provide callback functions for Master interface and data port management to bus driver
  - *Bus driver needs to call these APIs as Master registers are not standard defined*
- Process interrupts from SoundWire Master interface
- Note: no assumption that Master is part of AP/Chipset, could be in codec accessed over HDA/I2C/SLIMbus

SoundWire Slave Driver

- Configuration and control of vendor specific registers on SoundWire Slave interface
  - *Standard defines Slave registers needed for transport/errors*
  - *Standard defined registers are handled by bus driver*
- Handles vendor-specific Slave interrupts e.g. Jack detection
  - *Standard interrupts are handled by bus driver*
- Vendor-specific registers can be accessed with regmap
  - *Delta from I2C/I2S ASoC driver: bus registration, regmap init, interrupts (if applicable)*

RFC Information

- RFC posted on 21st October by Hardik Shah (hardik.t.shah@intel.com)
- Documentation
  - */Documentation/sound/alsa/sdw/*
- SoundWire Bus Driver
  - */sound/sdw/*
- Regmap
  - */drivers/base/regmap/*

THANK YOU