Vinod Koul, Sanyog Kale
Intel Corp

MIPI SoundWire® Linux Subsystem: An introduction to Protocol and Linux Subsystem

Agenda

- MIPI SoundWire® Basics
- Linux Subsystem
- SoundWire Linux Bus
- SoundWire Master
- SoundWire Slave

SoundWire® PHY

- Support one of 1.2 or 1.8 V
- Clock up to 12.288MHz
- Dual data rate
- Modified NRZI encoding
  - Logic 0: physical signal inverted from value in preceding bitSlot
  - Logic 1: no change in physical signal, level maintained by bus-keeper
- TDM
  - Each device ‘owns’ bit slot
- Data
  - PCM
  - PDM
  - Bulk Transfers
    - Bulk Register Access (BRA)
    - Bulk Transfer Protocol (BTP)

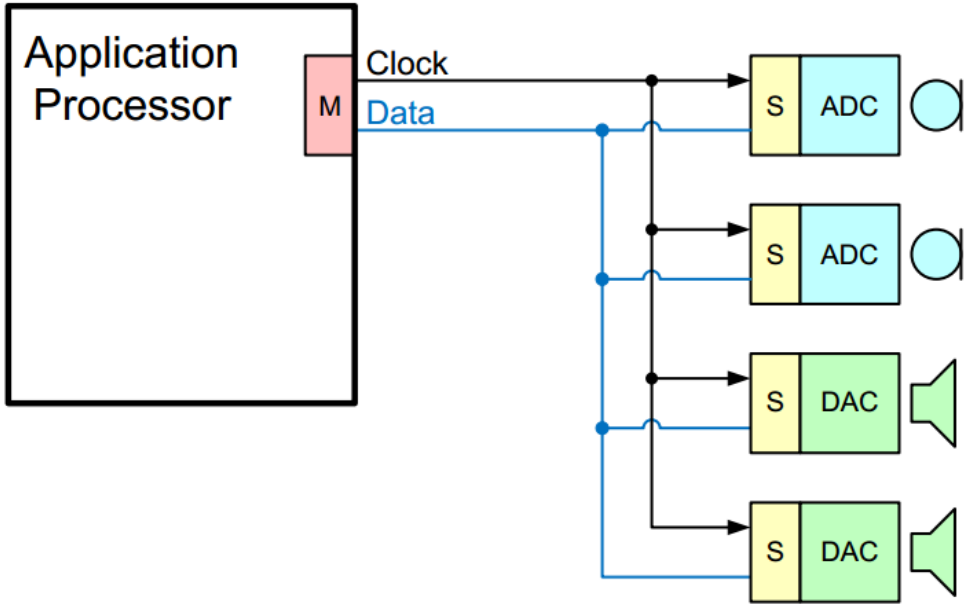
Intel Corp

Device Types

- Master
  - Provides Clock and Sync pattern on data line
  - Bus management, bit allocation
- Slave
  - Audio peripheral (MIC, Codec, Amps)
  - 1 to 11 Slaves connected, Multi-drop bus
  - In band Interrupt, System wake
- Monitor
  - Test equipment, in snoop analyze mode
  - Temporarily take over bus, issue Read/Write commands

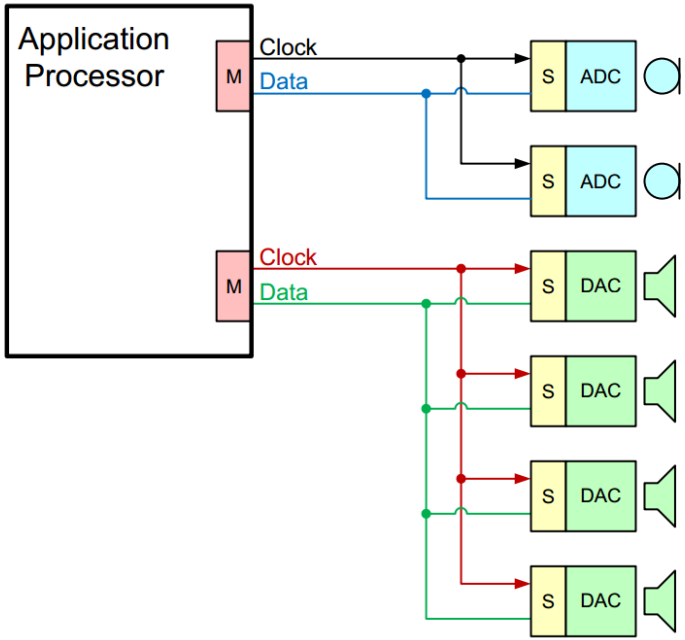
Intel Corp

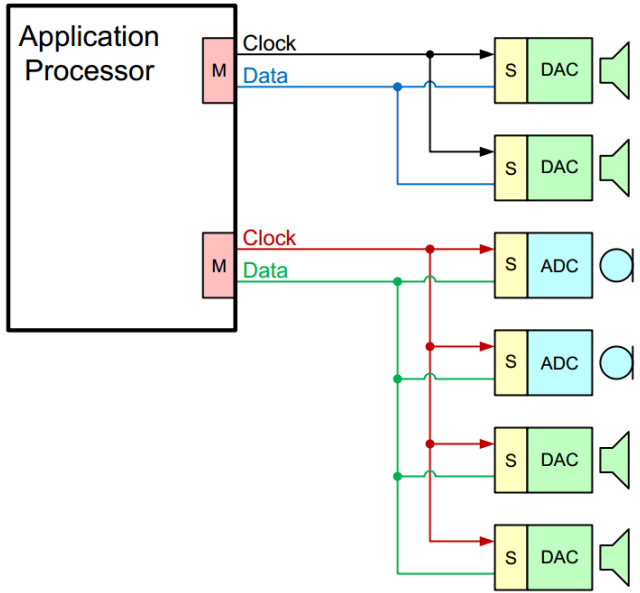
SoundWire® Topologies



Intel Corp

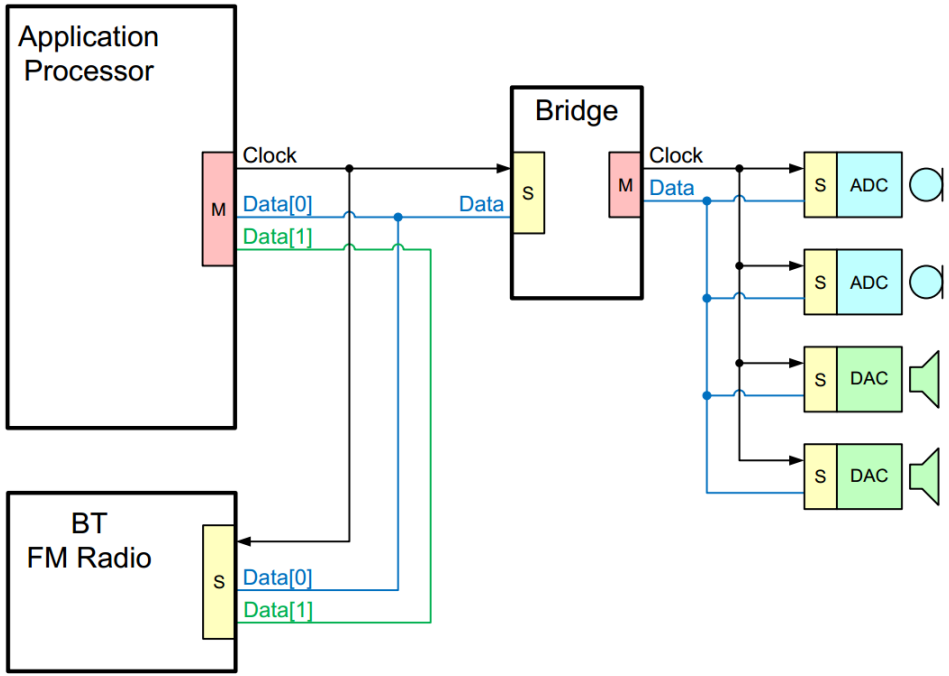
SoundWire® Topologies





Intel Corp

SoundWire® Topologies



Intel Corp

Enumeration

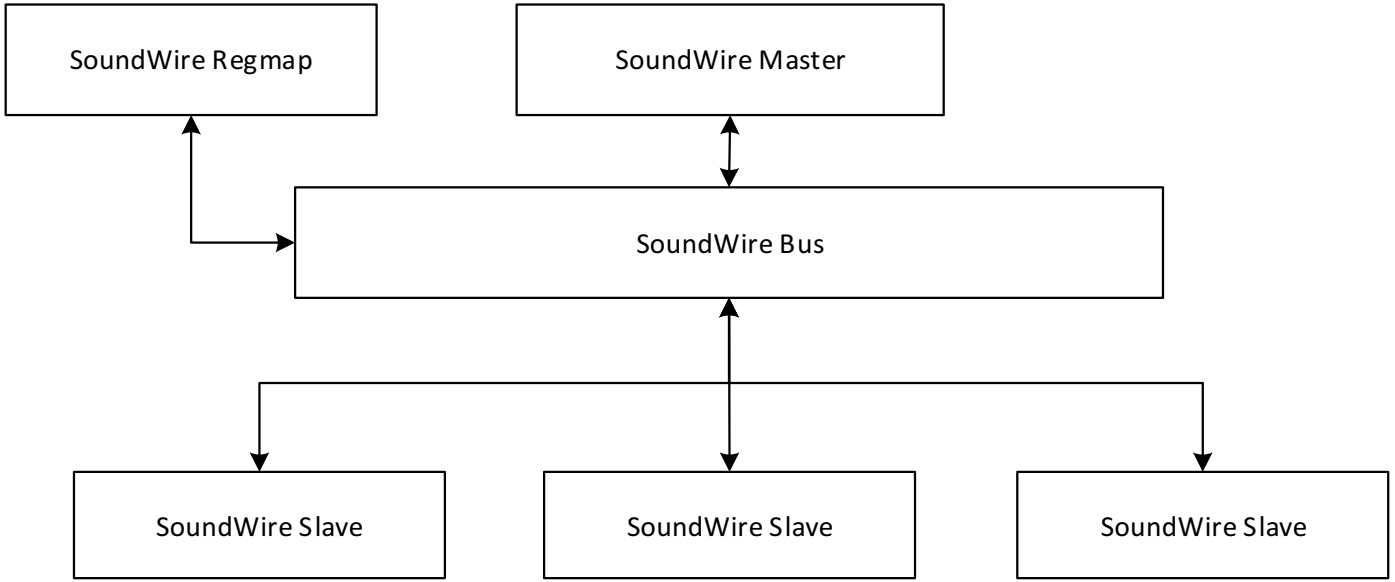
- Register DevId 0-5, 48-bit unique value
  - ManufacturerID, PartID
  - Class (not defined yet)
  - SoundWire spec version
  - UniqueID
- Slave reports present on Dev Num 0
- Master reads DevId 0-5
- Assigns Dev Num X (1..11)
- Slave reports present on Dev Num X

Intel Corp

SoundWire® DisCo℠

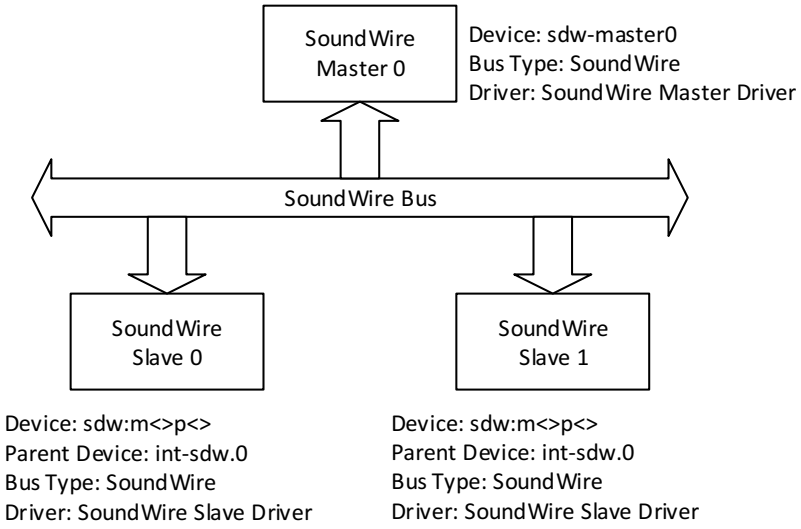
- MIPI Discovery and Configuration (DisCo℠) mechanism
- MIPI adopted Spec
- Optional, but values mandatory for Software
- Implemented as ACPI _DSD methods
- Describes SoundWire Master properties and Slave properties

SoundWire® Linux Subsystem



Intel Corp

SoundWire® Device Model



Intel Corp

SoundWire® Linux Bus

- The Bus is instantiated by:
  - int sdw_add_bus_master(struct sdw_bus *bus);
  - Master allocates sdw_bus structres
  - Initialized by invoking this API
- Similarly exit is performed by:
  - void sdw_delete_bus_master(struct sdw_bus *bus);

```
struct sdw_bus {
	struct device *dev;
	bool acpi_enabled;
	unsigned int link_id;
	struct list_head slaves;
	bool assigned[SDW_MAX_DEVICES + 1];
	struct mutex bus_lock;
	struct mutex msg_lock;
	const struct sdw_bus_ops *bus_ops;
	const struct sdw_master_ops *ops;
	const struct sdw_master_port_ops *port_ops;
	struct sdw_bus_params params;
	struct sdw_master_prop prop;
	struct sdw_master_sysfs *sysfs;
	struct sdw_defer defer_msg;
	enum sdw_bus_ops_impl impl;
	unsigned int clk_stop_timeout;
};
```

Intel Corp

IO

- Slaves typically have Audio function
- Need IO access for implementation defined registers
  - Support read/write along with ‘n’ consecutive read/write

```
int sdw_read(struct sdw_slave *slave, u32 addr);

int sdw_write(struct sdw_slave *slave, u32 addr, u8 value);

int sdw_nread(struct sdw_slave *slave, u32 addr, size_t count, u8 *val);

int sdw_nwrite(struct sdw_slave *slave, u32 addr, size_t count, u8 *val);
```

Intel Corp

SoundWire® Enumeration

- SoundWire protocol enumerable but not discoverable
- Slaves maybe powered off on boot
  - Protocol enumeration doesn't work then
- Implementation (ACPI/DT) based
  - Create SoundWire Slaves based on firmware description
  - Mark Slaves PRESENT when they show up

Intel Corp

SoundWire® Linux Slave

- Slave Driver registers Slave to Bus
  - int sdw_register_driver(struct sdw_driver *drv);
  - void sdw_unregister_driver(struct sdw_driver *drv);
- Probed when device found in firmware description
  - Matching uses only Manufacturer ID, Part ID
  - Instance not used, load same driver for different instances

SoundWire® Slave Driver

- Typical Linux subsystem driver
- Provides probe, remove, shutdown methods
- ID table for SDW IDs
- Ops for Slave driver callbacks

```
struct sdw_driver {
	const char *name;

	int (*probe)(struct sdw_slave *sdw,
		     const struct sdw_device_id *id);
	int (*remove)(struct sdw_slave *sdw);
	void (*shutdown)(struct sdw_slave *sdw);

	const struct sdw_device_id *id_table;
	const struct sdw_slave_ops *ops;

	struct device_driver driver;
};

struct sdw_device_id {
	__u16 mfg_id;
	__u16 part_id;
	__u8 class_id;
	kernel_ulong_t driver_data;
};
```

Intel Corp

Q & A

Intel Corp

Links

- SoundWire® Brief http://www.aes.org/e-lib/browse.cfm?elib=17407
- SoundWire® Spec v1.1 https://members.mipi.org/wg/All-Members/document/70290
- SoundWire® DisCo℠ Spec v1.0 https://members.mipi.org/wg/All-Members/document/71260
- SoundWire® Source Tree https://git.kernel.org/pub/scm/linux/kernel/git/vkoul/soundwire.git/

mipi®
DEVCON

THANK YOU

2017
MIPI ALLIANCE
DEVELOPERS
CONFERENCE

BANGALORE, INDIA

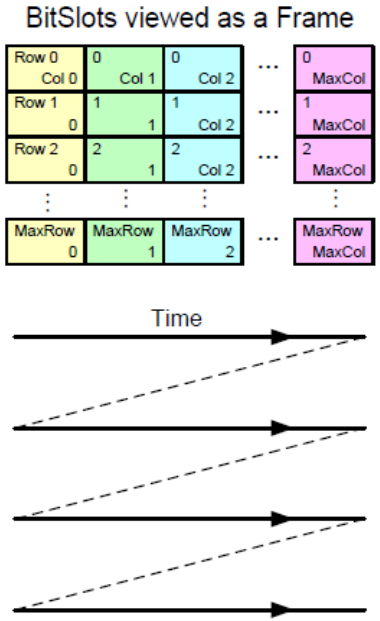
MIPI.ORG/DEVCON

Slaves

- Status
  - Not Attached (Not present/operational)
  - Attached (synchronized w/ Master and able to handle commands)
  - Alert (synchronized, at least one Interrupt condition raised)
- Up to 11 Slaves per link, each w/ Unique Device Number
  - Dev Num 0: Attached, but not enumerated
  - Dev Num 1 -11: Enumerated Device
  - Dev Num 12-13: Group
  - Dev Num 14: Reserved for Master
  - Dev Num 15: Broadcast
- Data Ports
  - DP0: Bulk command protocol (BRA, BTP)
  - DP1-DP14: Data ports (for audio streaming)
  - DP15: Alias port (program all ports with single command)

Frame

- Serial transmission but frame defined by 2D pattern (MaxRow, MaxCol)
- Command/Control:
  - First 48 Rows of Col0
- PCM
  - Can use multiple columns/multiple rows
- Low latency streams (1-bit PDM):
  - Samples evenly distributed in time
  - ‘vertical stripes’ in bit allocation
  - No conflicts with command/control or PCM



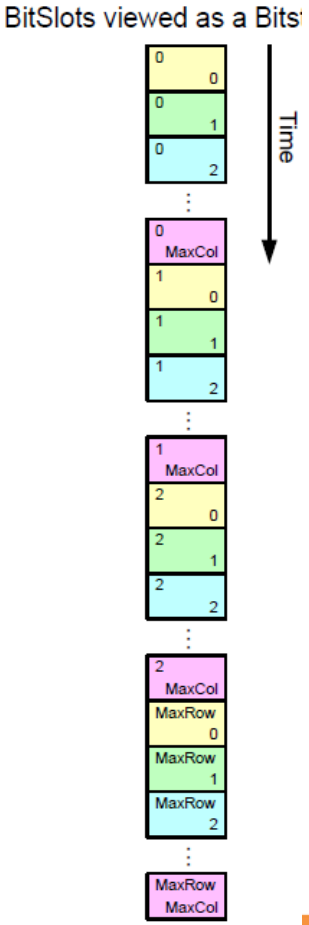


Figure 14 Frame Shape

Intel Corp

Master Operations

- Spec doesn't define Master implementation
- Bus provides ops for Master Driver, Mandatory
  - DisCo property read
  - Transfer messages (IO)
  - Register Bank Switches
- Optional
  - Setting SSP interval
  - Bus Parameters computation

```
struct sdw_master_ops {
    int (*read_prop)(struct sdw_bus *bus);

    enum sdw_command_response (*xfer_msg)
            (struct sdw_bus *bus,
            struct sdw_msg *msg, int page);
    enum sdw_command_response (*xfer_msg_defer)
            (struct sdw_bus *bus, struct sdw_msg *msg,
            int page, struct sdw_defer *defer);
    enum sdw_command_response (*reset_page_addr)
            (struct sdw_bus *bus, unsigned int dev_num);
    int (*set_ssp_interval)(struct sdw_bus *bus,
            unsigned int ssp_interval,
            unsigned int bank);
    int (*set_bus_conf)(struct sdw_bus *bus,
            struct sdw_bus_conf *conf);
    int (*pre_bank_switch)(struct sdw_bus *bus);
    int (*post_bank_switch)(struct sdw_bus *bus);

};
```

Intel Corp

Slave Driver Ops

- DisCo property read
- Interrupt update (implementation defined interrupts)
- Clock stop query and apply
- Bus configuration change update
- Data port prepare (pre/post)

```
struct sdw_slave_ops {
    int (*read_prop)(struct sdw_slave *sdw);
    int (*interrupt_callback)(struct sdw_slave *slave,
                struct sdw_slave_intr_status *status);
    int (*update_status)(struct sdw_slave *slave,
                enum sdw_slave_status status);
    int (*get_clk_stop_mode)(struct sdw_slave *slave);
    int (*clk_stop)(struct sdw_slave *slave,
                enum sdw_clk_stop_mode mode,
                enum sdw_clok_stop_type type);
    int (*pre_bus_config)(struct sdw_slave *slave,
                struct sdw_bus_conf *conf);
    int (*port_prep)(struct sdw_slave *slave,
                struct sdw_prepare_ch *prepare_ch,
                enum sdw_port_prep_ops pre_ops);
};
```

Intel Corp

Frame Shape

- Columns: 2-4-6-8-10-12-14-16
- Rows: 48 to 256, not all combinations are valid
- Frame shape determined by
  - Bus clock
  - Frame rate typically 8-48 kHz
  - Audio sample rate
  - Oversampling ratio for PDM
- Slaves are required to handle pairwise combinations of Rows, Cols
- Master will typically only use few combinations based on 2^n factors