ASOC TOPOLOGY

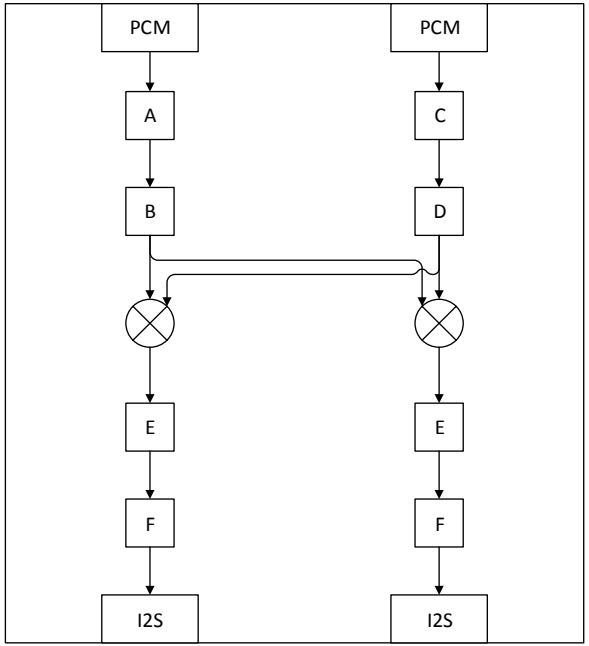
Vinod Koul vkoul@kernel.org

Intel Corp

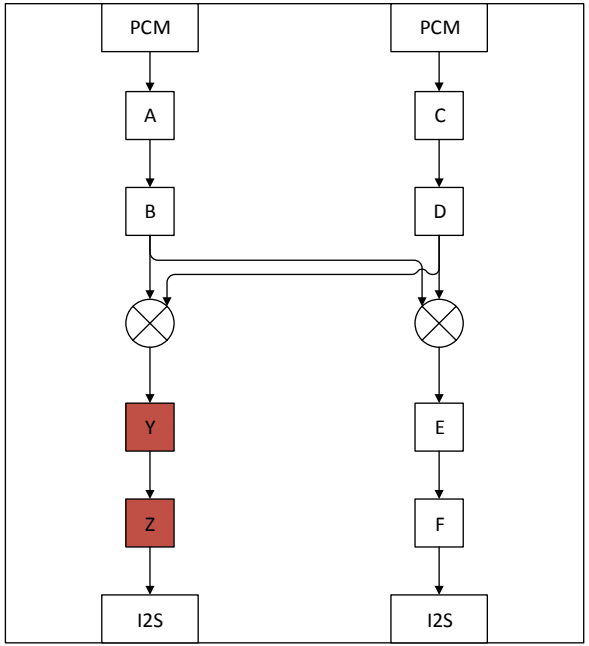
HISTORY

- Credits to Liam Girdwood, Started as **D**ynamic **F**irm**W**are (DFW)
- Simple DAPM description in user mode
- Implemented for TI OMAP
- Rewrote & Upstreamed by Liam while working for Intel
- Merged in v4.2
- ABI deemed stable in v4.9
- ALSA Lib support in v1.1.0

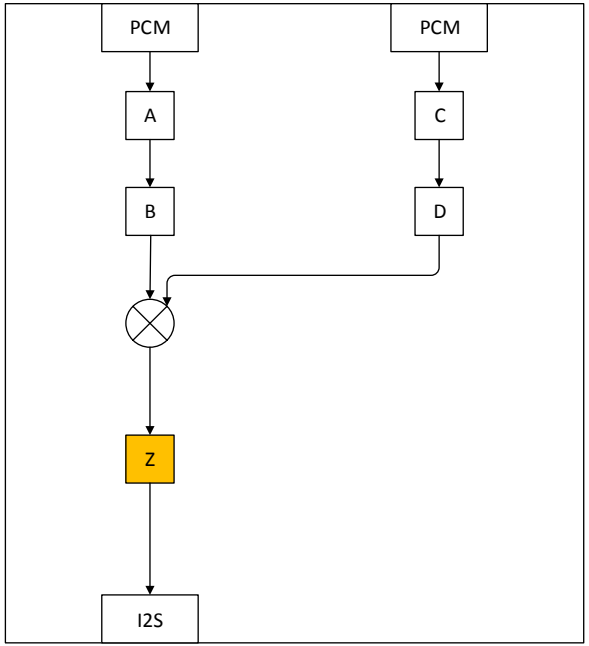
WHY TOPOLOGY



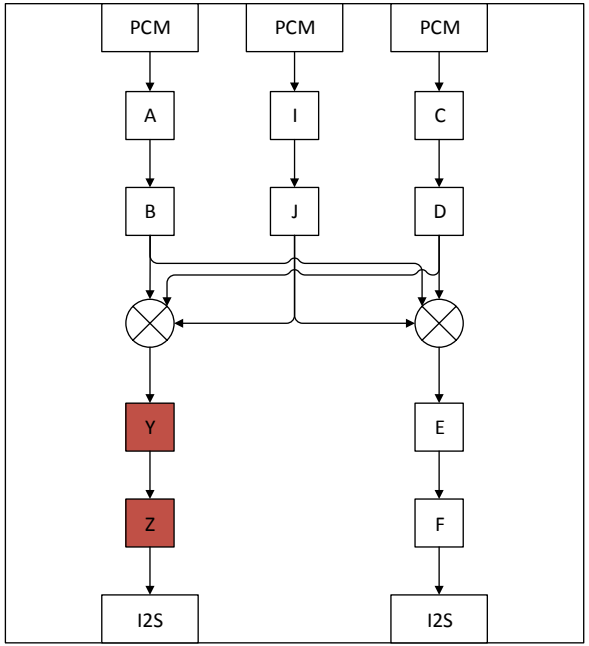
WHY TOPOLOGY



WHY TOPOLOGY



WHY TOPOLOGY



SYSTEM VIEW - USER

SYSTEM VIEW - KERNEL

HOW

- Describe the topology in a configuration file, topology.conf
- APIs to parse the conf file
- In “UCM” style syntax, reuse UCM parser
- Built using ALSA –Utils Topology tool
- Manifest for book keeping
- Allows “vendor pvt data” for elements

TOPOLOGY CONF

- Controls
  - Mixer
  - Enums
  - Bytes
- DAPM Widget
- DAPM Graph
- DAIs
  - Front End aka PCMs
  - DAI Links

MIXER CONTROL

```
SectionControlMixer."mixer name" {
    comment "optional comments"
    index "1"                       # Index number
    channel."name" {                # Channel maps
      ....
    }
    ops."ctl" {                     # Ops callback functions
      ....
    }
    access [                        # Control Access
      read
      write
    ]
    max "32"                        # Max control value
    invert "0"                      # Whether control values are inverted
    tlv "tld_data"                  # optional TLV data
    data "pdata for mixer1"         # optional private data
}
```

BYTE CONTROL

```
SectionControlBytes."name" {
    comment "optional comments"
    index "1"                       # Index number
    channel."name" {                # Channel maps
      ....
    }
    ops."ctl" {                     # Ops callback functions
      ....
    }
    base "0"                        # Register base
    num_regs "16"                   # Number of registers
    mask "0xff"                     # Mask
    max "255"                       # Maximum value
    tlv "tld_data"                  # optional TLV data
    data "pdata for mixer1"         # optional private data
}
```

ENUMERATED CONTROL

```
SectionText."name" {
        Values [
            "value1"
            "value2"
            "value3"
        ]
}
```

```
SectionControlEnum."name" {
    comment "optional comments"
    index "1"                   # Index number
    texts "EQU1"                # Enumerated text items
    channel."name" {            # Channel maps
      ....
    }
    ops."ctl" {                 # Ops callback functions
      ....
    }
    data "pdata for mixer1“ # optional pdata
}
```

DAPM WIDGET

```
SectionWidget."name" {

        index "1"                   # Index number

        type "aif_in"               # Widget type
        stream_name "name"          # Stream name

        no_pm "true"                # No PM control bit.
        reg "20"                    # PM bit register offset
        shift "0"                   # PM bit register shift
        invert "1                   # PM bit is inverted
        subseq "8"                  # subsequence number

        event_type "1"              # DAPM widget event type
        event_flags "1"             # DAPM widget event flags

        mixer "name"                # Optional Mixer Control
        enum "name"                 # Optional Enum Control

        data "name"                 # optional private data

}
```

DAPM GRAPH

```
SectionGraph."dsp" {
        index "1"                   # Index number
        lines [
                "sink1, control, source1"
                "sink2, , source2"
        ]
}
```

DAI

- PCM Capabilities

```
SectionPCMCapabilities."name" {
    formats "S24_LE,S16_LE"
    rate_min "48000"
    rate_max "48000
    channels_min "2
    channels_max "2”
 }
```

DAI

- PCM Capabilities

```
SectionPCMCapabilities."name" {
    formats "S24_LE,S16_LE"
    rate_min "48000"
    rate_max "48000
    channels_min "2
    channels_max "2”
}
```

- PCM Configuration

```
SectionPCMConfig."name" {
    config."playback"
        format "S16_LE
        rate "48000"
        channels "2"
        tdm_slot "0xf”
    }
    config."capture“{
        format "S16_LE"
        rate "48000"
        channels "2"
        tdm_slot "0xf"
    }
}
```

DAI

- FE

```
SectionPCM."name" {
    index "1"                   # Index number
    id "0"                      # used for binding to the PCM
    dai."name of front-end DAI" {
        id "0"                  # used for binding to the front-end DAI
    }
    pcm."playback" {
        capabilities "capabilities1"    # capabilities for playback
        configs [           # supported configs for playback
            "config1"
            "config2"
        ]
    }
    pcm."capture" {
        capabilities "capabilities2"
        configs [           # supported configs for capture
            "config1"
            "config2"
            "config3"
        ]
    }
    # Optional boolean flags
    symmetric_rates             "true"
    symmetric_channels          "true"
    symmetric_sample_bits       "false"
    data "name"                 # optional private data
}
```

DAI

- DAI Link Configuration

```
SectionLink."name" {
    index "1"
    id "0"                          # binding id for the link
    stream_name "name"              # used for binding to the link
    hw_configs [                    # runtime HW configs, optional
        "config1"
        "config2"
        ...
    ]
    default_hw_conf_id "1"        #default HW config ID for init
    # Optional boolean flags
    symmetric_rates               "true"
    symmetric_channels            "false"
    symmetric_sample_bits         "true"
    data "name"                 # optional private data
}
```

DAI

- Physical DAI Configuration

```
SectionDAI."name" {
        index "1"                       # Index number
        id "0"                          # used for binding to the Backend DAI
        pcm."playback" {
                capabilities "capabilities1"    # capabilities for playback
        }
        pcm."capture" {
                capabilities "capabilities2"    # capabilities for capture
        }
        symmetric_rates "true"          # optional flags
        symmetric_channels "true"
        symmetric_sample_bits "false"
        data "name                      # optional private data
}
```

PRIVATE DATA

```
SectionData."pdata for EQU1" {
      file "/path/to/file"
      bytes "0x12,0x34,0x56,0x78"
      shorts "0x1122,0x3344,0x5566,0x7788"
      words "0xaabbccdd,0x11223344,0x66aa77bb,0xefef1234"
      tuples "section id of the vendor tuples"
  };

 Sectionxxx."element name" {
      ...
      data [                              # optional private data
            "name of 1st data section"
            "name of 2nd data section"
            ...
      ]
  }
```

TUPLES

```
SectionVendorTokens."id of the vendor tokens" {
        comment "optional comments"
        VENDOR_TOKEN_ID1 "1"
        VENDOR_TOKEN_ID2 "2"
        VENDOR_TOKEN_ID3 "3"
        ...
}

SectionVendorTuples."id of the vendor tuples" {
        tokens "id of the vendor tokens"
        tuples."string" {
                VENDOR_TOKEN_ID1 "character string"
                ...
}
tuples."uuid" {             # 16 characters csv
                VENDOR_TOKEN_ID2 "0x01,0x02,...,0x0f"
                ...
}
        tuples."bool" {
                VENDOR_TOKEN_ID3 "true/false"
                ...
        }
        tuples."byte" {
                VENDOR_TOKEN_ID4 "0x11"
                VENDOR_TOKEN_ID5 "0x22"
                ...
        }
        tuples."short" {
                VENDOR_TOKEN_ID6 "0x1122"
                VENDOR_TOKEN_ID7 "0x3344"
                ...
        }
        tuples."word" {
                VENDOR_TOKEN_ID8 "0x11223344"
                VENDOR_TOKEN_ID9 "0x55667788"
                ...
        }
}
```

BUILDING CONF

- ALSA – Utils alsatplg
- $ ./alsatplg –c skl_i2s.conf -o dfw_sst.bin
- Uses alsa-lib topology APIs
  - snd_tplg_build_file()
  - Additional C apis available in ALSA-lib for parsing

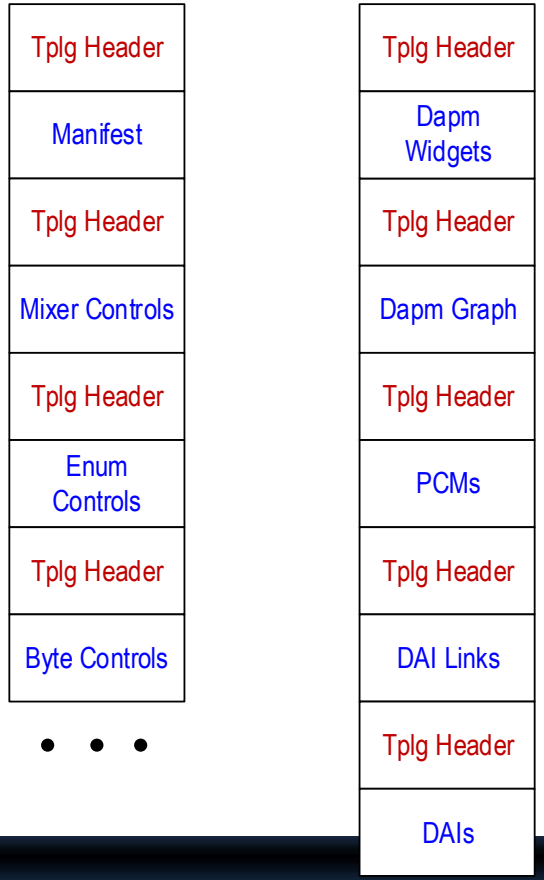
TOPOLOGY C APIS

- snd_tplg_t *snd_tplg_new(void);
- int snd_tplg_add_object(snd_tplg_t *tplg,
  snd_tplg_obj_template_t *t);
- int snd_tplg_set_manifest_data(snd_tplg_t *tplg,
  const void *data, int len);
- int snd_tplg_build(snd_tplg_t *tplg, const char *outfile);
- void snd_tplg_free(snd_tplg_t *tplg);

BINARY FORMAT

ASoC	ABI: 5	Vendor Ver	TPLG_TYPE
Hdr Size	Vendor Type	Payload Size	Index
Count			

BINARY FORMAT



KERNEL

Initialize ASoC with topology

```
request_firmware();

snd_soc_tplg_component_load(&platform->component,
                            &skl_tplg_ops, fw, 0);
```

TOPOLOGY CORE

- snd_soc_tplg_component_load()
  - soc_tplg_process_headers()
    - soc_valid_header()
      - Size, Magic, ABI, ...
    - soc_tplg_load_header()
      - Based on type invoke soc_tplg_XXX_elems_load()
    - soc_tplg_dapm_complete()
      - snd_soc_dapm_new_widgets()
        - dapm_new_xxx()
  - soc_tplg_complete()

TOPOLOGY CORE

- Based on type, process
  - Mixer, enum, bytes: soc_tplg_kcontrol_elems_load()
    - soc_tplg_add_kcontrol()
      - snd_soc_cnew()
  - Graph: soc_tplg_dapm_graph_elems_load()
    - snd_soc_dapm_add_routes()
  - Widgets: soc_tplg_dapm_widget_elems_load()
    - snd_soc_dapm_new_control()
  - PCM: soc_tplg_pcm_elems_load()
    - snd_soc_register_dai()
    - snd_soc_add_dai_link()

TOPOLOGY CORE

- Based on type, process
  - DAI: soc_tplg_dai_elems_load()
    - snd_soc_find_dai()
    - set_stream_info()
    - set_dai_flags()
  - DAI Link, BE: soc_tplg_link_elems_load()
    - snd_soc_find_dai_link()
    - set_link_hw_format()
    - set_link_flags()
  - Manifest: soc_tplg_manifest_load()
  - Bespoke: soc_tplg_vendor_load()
    - Default handler, vendor load

TOPOLOGY OPS

- Notification for driver on object load/unload
  - Control
  - Widget
  - DAI
  - Link
  - Manifest
  - Vendor
  - Complete

TOPOLOGY OPS

- IO ops driver handler
  - Control ops in topology-conf specifies driver handler
  - Driver declares IO ops
  - Core matches and sets

```
struct snd_soc_tplg_kcontrol_ops {
	u32 id;
	int (*get)(struct snd_kcontrol *kcontrol,
			struct snd_ctl_elem_value *ucontrol);
	int (*put)(struct snd_kcontrol *kcontrol,
			struct snd_ctl_elem_value *ucontrol);
	int (*info)(struct snd_kcontrol *kcontrol,
		struct snd_ctl_elem_info *uinfo);
};
```

MANIFEST

- MANIFEST BLOCK

```
struct snd_soc_tplg_manifest {
        __le32 size;                    /* in bytes of this structure */
        __le32 control_elems;           /* number of control elements */
        __le32 widget_elems;            /* number of widget elements */
        __le32 graph_elems;             /* number of graph elements */
        __le32 pcm_elems;               /* number of PCM elements */
        __le32 dai_link_elems;          /* number of DAI link elements */
        __le32 dai_elems;               /* number of physical DAI elements */
        __le32 reserved[20];            /* reserved for new ABI element types */
        struct snd_soc_tplg_private priv;
} __attribute__((packed));
```

FUTURE WORK

- Removing configuration files from alsa-lib
- Tinyalsa support...
- Loading sub graphs using index


QUESTIONS?


intel®