Simplified DVFS and OPP Framework

Viresh Kumar (PMWG)
Topics

- What is OPP?
- How we did DVFS earlier
- How we do DVFS now
- Clock with connection id
- Multiple regulators
- Multiple OPP tables
- Different voltage/current range for OPP
What is operating performance point (aka OPP)

- SoCs have become complex and power efficient.
- Range of frequency/voltages to work.
- Not only for CPU, but GPU, MMC controllers, etc.

- OPP is set of discrete tuples consisting of frequency and voltage pairs.
- Example:

<table>
<thead>
<tr>
<th>Hz</th>
<th>uV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000000 1000000</td>
<td>- OPP 1</td>
</tr>
<tr>
<td>1100000000 1100000</td>
<td>- OPP 2</td>
</tr>
<tr>
<td>1200000000 1300000</td>
<td>- OPP 3</td>
</tr>
</tbody>
</table>

- OPP tables in DT are a bit complex than a simple tuple.
How we did DVFS earlier (before v4.6)

Each driver would do something like:

```c
    do_dvfs(target)
    {
        target = clk_round_rate(target);

        /* scaling up? scale voltage before frequency */
        if (!IS_ERR(cpu_reg) && target > old_freq) {
            opp = dev_pm_opp_find_freq ceil(dev, &target);
            regulator_set_voltage(reg, opp->volt);
        }

        ret = clk_set_rate(clk, target);
        if (ret) {
            /* Restore regulator voltage constraint here */
            return;
        }
    }

    /* scaling down? scale voltage after frequency */
    if (IS_ERR(cpu_reg) && target < old_freq) {
        ret = regulator_set_voltage(reg, opp->volt);
        if (ret)
            clk_set_rate(clk, old_freq);
    }

    return;
```
How we do DVFS now (v4.6 and beyond)

- Each driver would do something like:

  ```
  do_dvfs(target)
  {
    dev_pm_ops_set_rate(dev, target);
  }
  ```

- Yeah, that’s it.
- And this will also take care of the upcoming performance states for PM domains.
Programming clk with connection id

- OPP core does this by default to get the clk pointer:
  \[ \text{clk} = \text{clk\_get}(\text{dev}, \text{NULL}); \]

- Wouldn’t work for device that have multiple clocks.
- APIs to set connection id
  ```c
  struct opp_table *dev_pmOpp_set_clkname(struct device *dev, const char * name);
  void dev_pm_oppput_clkname(struct opp_table *opp_table);
  ```
Programming (multiple) regulators

- Some platforms need to program (multiple) regulators along with clk.
- API to set regulator names:
  ```c
  struct opp_table *dev_pm_opp_set_regulators(struct device *dev, const char * const names[], unsigned int count);
  void dev_pm_opp_put_regulators(struct opp_table *opp_table);
  ```
- OPP core doesn’t have a general ordering policy for configuring multiple regulators.
- Driver needs to provide another API for that:
  ```c
  struct opp_table *dev_pm_opp_register_set_opp_helper(struct device *dev, int (*set_opp)(struct dev_pm_set_opp_data *data));
  void dev_pm_opp_register_put_opp_helper(struct opp_table *opp_table);
  ```
- OPP core will call that ->set_opp() callback instead from dev_pm_opp_set_rate().
Multiple OPP tables

- Need to have multiple OPP tables (per hardware version) per device.
- OPP core provides a way to select only a subset of the OPPs dynamically.
- Each OPP needs to have “opp-supported-hw” property.
- Can have multiple masks and each bit of mask represents a hardware version.
- APIs and Example:

```c
struct opp_table *dev_pm_opp_set_supported_hw(struct device *dev, const u32 *versions, unsigned int count);
void dev_pm_opp_put_supported_hw(struct opp_table *opp_table);

opp {
    opp-hz = /bits/ 64 <600000000>;
    opp-supported-hw = <0xF 0xFFFFFFFF>; /* Supports all substrate versions for 0xF cuts, i.e. only first four cuts */
    /* OR */
    opp-supported-hw = <0xFFFFFFFFF 0xFFFFFFFFF>; /* Supports all substrate versions for all cuts (except first four cuts) */
};
```

u32 version[2] = {0x00000008, 0x000F0000}; /* Selects 4th cut and 4 substrate versions (17-20) */
dev_pm_opp_set_supported_hw(dev, version, 2); /* Only the first opp-supported-hw will be accepted */
Different voltage/current range for OPP

- Need to select a different voltage and current range based on hardware chip.
- Can depend on manufacturing process, every chip is different.
- Some properties can have multiple values using the “-<name>” DT extension.
- APIs and Example:

  ```c
  struct opp_table *dev_pm_opp_set_prop_name(struct device *dev, const char *name);
  void dev_pm_opp_put_prop_name(struct opp_table *opp_table);

  opp {
    opp-hz = /bits/ 64 <1000000000>;
    opp-microvolt-slow = <900000 915000 925000>;
    opp-microvolt-fast = <970000 975000 985000>;
    opp-microamp-slow = <70000>;
    opp-microamp-fast = <71000>;
  };
  dev_pm_opp_set_prop_name(dev, "fast"); /* This will choose the properties with “-fast” */
```

- Driver using “-name” and “opp-supported-hw” property: sti-cpufreq.c.
Thank You

#SFO17

BUD17 keynotes and videos on: connect.linaro.org

For further information: www.linaro.org