Dynamic Thermal Power Management framework

On the embedded world, the complexity of the SoC leads to an increasing number of hotspots which need to be monitored and mitigated as a whole in order to prevent the temperature to go above the normative and legally stated 'skin temperature'.

Another aspect is to sustain the performance for a given power budget, for example virtual reality where the user can feel dizziness if the performance is capped while a big CPU is processing something else. Or reduce the battery charging because the dissipated power is too high compared with the power consumed by other devices.

The user space is the most adequate place to dynamically act on the different devices by limiting their power given an application profile: it has the knowledge of the platform.

The Dynamic Thermal Power Management (DTPM) is a technique acting on the device power by limiting and/or balancing a power budget among different devices.

The DTPM framework provides an unified interface to act on the device power.

Overview

The DTPM framework relies on the powercap framework to create the powercap entries in the sysfs directory and implement the backend driver to do the connection with the power manageable device.

The DTPM is a tree representation describing the power constraints shared between devices, not their physical positions.

The nodes of the tree are a virtual description aggregating the power characteristics of the children nodes and their power limitations.

The leaves of the tree are the real power manageable devices.

For instance:

```
SoC
 |
 `-- pkg
      |
      |-- pd0 (cpu0-3)
      |
      `-- pd1 (cpu4-5)
```

The pkg power will be the sum of pd0 and pd1 power numbers:

```
SoC (400mW - 3100mW)
 |
 `-- pkg (400mW - 3100mW)
      |
      |-- pd0 (100mW - 700mW)
      |
      `-- pd1 (300mW - 2400mW)
```

When the nodes are inserted in the tree, their power characteristics are propagated to the parents:

```
SoC (600mW - 5900mW)
 |
 |-- pkg (400mW - 3100mW)
 |    |
 |    |-- pd0 (100mW - 700mW)
 |    |
 |    `-- pd1 (300mW - 2400mW)
 |
 `-- pd2 (200mW - 2800mW)
```

Each node have a weight on a 2^10 basis reflecting the percentage of power consumption along the siblings:

```
SoC (w=1024)
 |
 |-- pkg (w=538)
 |    |
 |    |-- pd0 (w=231)
 |    |
 |    `-- pd1 (w=794)
 |
 `-- pd2 (w=486)

  Note the sum of weights at the same level are equal to 1024.
```

When a power limitation is applied to a node, then it is distributed along the children given their weights. For example, if we set a power limitation of 3200mW at the 'SoC' root node, the resulting tree will be:

```
SoC (w=1024) <--- power_limit = 3200mW
```

```
|
|-- pkg (w=538) --> power_limit = 1681mW
|    |
|    |-- pd0 (w=231) --> power_limit = 378mW
|    |
|    `-- pd1 (w=794) --> power_limit = 1303mW
|
`-- pd2 (w=486) --> power_limit = 1519mW
```

Flat description

A root node is created and it is the parent of all the nodes. This description is the simplest one and it is supposed to give to user space a flat representation of all the devices supporting the power limitation without any power limitation distribution.

Hierarchical description

The different devices supporting the power limitation are represented hierarchically. There is one root node, all intermediate nodes are grouping the child nodes which can be intermediate nodes also or real devices.

The intermediate nodes aggregate the power information and allows to set the power limit given the weight of the nodes.

User space API

As stated in the overview, the DTPM framework is built on top of the powercap framework. Thus the sysfs interface is the same, please refer to the powercap documentation for further details.

- power_uw: Instantaneous power consumption. If the node is an intermediate node, then the power consumption will be the sum of all children power consumption.
- max_power_range_uw: The power range resulting of the maximum power minus the minimum power.
- name: The name of the node. This is implementation dependent. Even if it is not recommended for the user space, several nodes can have the same name.
- constraint_X_name: The name of the constraint.
- constraint_X_max_power_uw: The maximum power limit to be applicable to the node.
- constraint_X_power_limit_uw: The power limit to be applied to the node. If the value contained in constraint_X_max_power_uw is set, the constraint will be removed.
- constraint_X_time_window_us: The meaning of this file will depend on the constraint number.

Constraints

- Constraint 0: The power limitation is immediately applied, without limitation in time.

Kernel API

Overview

The DTPM framework has no power limiting backend support. It is generic and provides a set of API to let the different drivers to implement the backend part for the power limitation and create the power constraints tree.

It is up to the platform to provide the initialization function to allocate and link the different nodes of the tree.

A special macro has the role of declaring a node and the corresponding initialization function via a description structure. This one contains an optional parent field allowing to hook different devices to an already existing tree at boot time.

For instance:

```
struct dtpm_descr my_descr = {
        .name = "my_name",
        .init = my_init_func,
};

DTPM_DECLARE(my_descr);
```

The nodes of the DTPM tree are described with dtpm structure. The steps to add a new power limitable device is done in three steps:

- Allocate the dtpm node
- Set the power number of the dtpm node
- Register the dtpm node

The registration of the dtpm node is done with the powercap ops. Basically, it must implements the callbacks to get and set the power and the limit.

Alternatively, if the node to be inserted is an intermediate one, then a simple function to insert it as a future parent is available.

If a device has its power characteristics changing, then the tree must be updated with the new power numbers and weights.

Nomenclature

- dtpm_alloc() : Allocate and initialize a dtpm structure
- dtpm_register() : Add the dtpm node to the tree
- dtpm_unregister() : Remove the dtpm node from the tree
- dtpm_update_power() : Update the power characteristics of the dtpm node