How to generate power models for EAS and IPA

... without talking to a hardware engineer

Presented by
Leo Yan
Daniel Thompson

Date
BKK16-317 March 9, 2016

Event
Linaro Connect BKK16
Agenda

- Background
- Low power states
- Measurement on Hikey
- Generate power model
- Conclusions
Generating platform-specific parameters for power models are a prerequisite for deploying EAS and IPA:
  - Software engineers need these parameters to work on these features
  - Gaining assistance from hardware engineers is difficult and, for engineers who don’t work for SoC vendors, impossible
  - It is possible to generate an acceptable (though perhaps not optimal) model using simple tools

Today we will focus on CPU and cluster level power modeling:
  - Excludes SoC level power modeling (GPU, DDR, ...)

The role of power modeling

1. Generate power modeling
2. Enable EAS/IPA
3. Enable profiling environment
4. Check if use bad power modeling
5. Refine algorithm
6. Refine power modeling
Agenda

● Background
● Low power states
● Measurement on Hikey
● Generate power model
● Conclusions
## CPU power states

<table>
<thead>
<tr>
<th>CPU State</th>
<th>PD_CPUx</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU P-State</td>
<td>On</td>
</tr>
<tr>
<td>WFI</td>
<td>On, internal clock gating</td>
</tr>
<tr>
<td>CPU Off</td>
<td>Off</td>
</tr>
</tbody>
</table>
CPU P-State

Cluster 0

- ARM CPU0
- ACP
- ACE
- L2 RAM
- SCU

- CLKIN
- PDCPU0

Powered Off
Powered On
Clock Gating
CPU WFI State

Cluster 0

ARM CPU0: WFI

ACP
ACE
L2 RAM
SCU

Powered Off
Powered On
Clock Gating
CPU Off State

Cluster 0

- ARM CPU0
  - ACP
  - ACE
  - L2 RAM
  - SCU

CLKIN

PDCPU0

- Powered Off
- Powered On
- Clock Gating
## Cluster power states

<table>
<thead>
<tr>
<th>Cluster State</th>
<th>PD_CPUx</th>
<th>PDCORTEXA53</th>
<th>PD_L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster P-State</td>
<td>On or Off</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>Cluster L2 Retention</td>
<td>Off</td>
<td>Off</td>
<td>Retention</td>
</tr>
<tr>
<td>Cluster Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
</tbody>
</table>

Cannot keep cache coherent between two clusters.
Cluster P-State

Cluster 0

- ARM CPU0
- ACP
- ACE
- L2 RAM
- SCU

- CLKIN
- PDCPU<n>
- PDCORTEXTA53
- PDL2

States:
- Powered Off
- Powered On
- Clock Gating
Cluster L2 Retention State

Cluster 0
- ARM CPU0
  - ACP
  - ACE
  - L2 RAM
  - SCU

Connections:
- CLKIN
- PDCPU<n>
- PDCORTEXA53
- PDL2

Power States:
- Powered Off
- Powered On
- Clock Gating
## Link power states in Linux Kernel

<table>
<thead>
<tr>
<th>Power State</th>
<th>Linux Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU P-state</td>
<td>P-state</td>
</tr>
<tr>
<td>WFI</td>
<td>C-state</td>
</tr>
<tr>
<td>CPU Off</td>
<td>C-state</td>
</tr>
<tr>
<td>Cluster P-state</td>
<td>P-state</td>
</tr>
<tr>
<td>Cluster L2 Retention</td>
<td>C-state</td>
</tr>
<tr>
<td>Cluster Off</td>
<td>C-state</td>
</tr>
</tbody>
</table>

CPUFreq does not distinguish "CPU P-state" and "Cluster P-state" status as usually binding CPU and Cluster frequency from hardware design.

But EAS needs to model these two states separately.
Agenda

- Background
- Low power states
- Measurement on Hikey
- Generate power model
- Conclusions
Hikey PMIC Hierarchy

DC IN TO 4.2VDC, 4A → VDD_4V2 → PMIC: Hi6553

BUCKs & LDOs: LX0, LX1

Ideal measure point, but hardware design does not encourage measurement here

Select CPU's nearest measurement point

SoC: Hi6220

ACPU

Linaro Connect
Bangkok 2016
Connect with ARM energy probe

Hikey Board

- DC IN TO 4.2 VDC, 4A
- VDD_4V2: 0.033Ω
- PMIC: Hi6553

ARM energy probe

PC: Linux

- V+
- V−
- Ground
- USB

PC: Linux

Linaro Connect
Bangkok 2016
Methodology For Power Measurement

- Use single CPU to measure power
  - All other CPUs will be hot unplugged (CPU off state)
  - CPU0 is left connected
- Use ARM Trusted Firmware to perform system suspend
  - Force cluster and CPU0 to enter specific C-states and P-states
- Measure power data based on different operating points
  - Each OPP implies specific pair of voltage and frequency
  - Measure all C-states and P-states at given OPP
  - Use “dhrystone” to generate 100% CPU utilization for CPU P-state
Setup power state

1. `echo 0 > /sys/devices/system/cpu/cpu[1..7]/online`

   ARM CPU1
   \[\rightarrow\]
   psci_cpu_off
   \[\rightarrow\]
   hikey_afflvl_off

2. `echo mem > /sys/power/state`

   ARM CPU0
   \[\rightarrow\]
   psci_system_suspend_enter
   \[\rightarrow\]
   hikey_affinst_suspend
   \[\rightarrow\]
   hisi_ipc_cpu_suspend
   \[\rightarrow\]
   hisi_ipc_cluster_suspend
   \[\rightarrow\]
   hisi_ipc_psci_system_off

Hack low level code to only power off CPU but leave cluster on

Linux kernel

ARM TF

SoC
- Cluster 0
  - ARM CPU0
  - ARM CPU4
- Cluster 1
  - ARM CPU0
  - ARM CPU4
## Cluster and CPU’s Power Data

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Voltage (v)</th>
<th>Cluster Power Off State (mW)</th>
<th>Cluster P-State (mW)</th>
<th>CPU WFI State (mW)</th>
<th>CPU P-State (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>208</td>
<td>1.04</td>
<td>344</td>
<td>360</td>
<td>379</td>
<td>429</td>
</tr>
<tr>
<td>432</td>
<td>1.04</td>
<td>345</td>
<td>374</td>
<td>387</td>
<td>498</td>
</tr>
<tr>
<td>729</td>
<td>1.09</td>
<td>346</td>
<td>393</td>
<td>408</td>
<td>617</td>
</tr>
<tr>
<td>960</td>
<td>1.18</td>
<td>352</td>
<td>427</td>
<td>443</td>
<td>794</td>
</tr>
<tr>
<td>1200</td>
<td>1.33</td>
<td>367</td>
<td>479</td>
<td>508</td>
<td>1149</td>
</tr>
</tbody>
</table>

Difference is caused by other components in SoC that share the same regulator.

“Cluster P-state” means all CPUs in cluster have been powered off but cluster is powered on.
Agenda

- Background
- Low power states
- Measurement on Hikey
- Generate power model
- Conclusions
EAS energy model

- Formula: \( \text{energy} = \text{energy\_cluster} + \sum \text{energy\_CPU in cluster} \)
- C-states: Idle energy, include cpu level and cluster level
- P-states: Busy energy, include cpu level and cluster level
Normalized CPU capacity

- Normalized CPU capacity is a ratio value
  - Ratio between actual performance of CPU and the maximum performance of CPU in the system
  - Normalized into range [0..1024]

- Normally use “dhrystone” to measure CPU capacity
  - Safe to derive from CPU frequency if all CPUs are homogeneous

```c
static struct capacity_state cap_states_core_a53[] = {
    /* Power per cpu */
    { .cap = 178, .power = 69, }, /* 208MHz */
    { .cap = 369, .power = 124, }, /* 432MHz */
    { .cap = 622, .power = 224, }, /* 729MHz */
    { .cap = 819, .power = 367, }, /* 960MHz */
    { .cap = 1024, .power = 670, }, /* 1.2GHz */
};
```
Basic method to calculate for power data

Cluster 0

- ARM CPU0
  - OPP: 729 MHz
- ACP
- ACE
- L2 RAM
- SCU

Remove cluster level power consumption

Cluster 0

- ARM CPU0
- ACP
- ACE
- L2 RAM
- SCU

CPU level power consumption

ARM CPU0

OPP: 729 MHz

Remove cluster level power consumption
# Power of CPU C-State

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Cluster Power On State (mW)</th>
<th>CPU WFI (mW)</th>
<th>Power of CPU WFI (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>208</td>
<td>360</td>
<td>379</td>
<td>19</td>
</tr>
<tr>
<td>432</td>
<td>374</td>
<td>387</td>
<td>13</td>
</tr>
<tr>
<td>729</td>
<td>393</td>
<td>408</td>
<td>15</td>
</tr>
<tr>
<td>960</td>
<td>427</td>
<td>443</td>
<td>16</td>
</tr>
<tr>
<td>1200</td>
<td>479</td>
<td>508</td>
<td>31</td>
</tr>
</tbody>
</table>

Select 15mW as middle value for WFI state’s power data

```c
static struct idle_state idle_states_core_a53[] = {
    { .power = 15 },  /* active idle state */
    { .power = 15 },  /* WFI state */
    { .power = 0 },   /* CPU Off state */
    { .power = 0 },   /* Cluster Off state */
};
```
### Power of CPU P-State

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Cluster P-State (mW)</th>
<th>CPU P-State (mW)</th>
<th>Power of CPU P-state (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>208</td>
<td>360</td>
<td>429</td>
<td>69</td>
</tr>
<tr>
<td>432</td>
<td>374</td>
<td>498</td>
<td>124</td>
</tr>
<tr>
<td>729</td>
<td>393</td>
<td>617</td>
<td>224</td>
</tr>
<tr>
<td>960</td>
<td>427</td>
<td>794</td>
<td>367</td>
</tr>
<tr>
<td>1200</td>
<td>479</td>
<td>1149</td>
<td>670</td>
</tr>
</tbody>
</table>

CPU P-state is calculated with static leakage and dynamic power, so should use Power (CPU P-state) - Power (Cluster P-state)

```c
static struct capacity_state cap_states_core_a53[] = {
    /* Power per cpu */
    { .cap = 178, .power = 69, }, /* 208MHz */
    { .cap = 369, .power = 124, }, /* 432MHz */
    { .cap = 622, .power = 224, }, /* 729MHz */
    { .cap = 819, .power = 367, }, /* 960MHz */
    { .cap = 1024, .power = 670, }, /* 1.2GHz */
};
```
Cluster’s active idle state

At CPU level, all CPUs in the cluster are in one of the idle states (WFI or Off state) and the cluster is not processing any software.

At cluster level, the cluster remains active.

Is Active idle state a C-state or a P-state? The main difference is if we need consider it with different OPP...
## Power of Cluster C-State

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Cluster Power Off State (mW)</th>
<th>Cluster Power On State (mW)</th>
<th>Power of cluster level active idle state (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>208</td>
<td>344</td>
<td>360</td>
<td>16</td>
</tr>
<tr>
<td>432</td>
<td>345</td>
<td>374</td>
<td>29</td>
</tr>
<tr>
<td>729</td>
<td>346</td>
<td>393</td>
<td>47</td>
</tr>
<tr>
<td>960</td>
<td>352</td>
<td>427</td>
<td>75</td>
</tr>
<tr>
<td>1200</td>
<td>367</td>
<td>479</td>
<td>112</td>
</tr>
</tbody>
</table>

Only in cluster power off state will we totally power off whole cluster, in other idle states the cluster actually is powered on even when all CPUs are powered off.

```c
static struct idle_state idle_states_cluster_a53[] = {
    { .power = 47 },    /* active idle state */
    { .power = 47 },    /* WFI state */
    { .power = 47 },    /* CPU Off state */
    { .power = 0 },     /* Cluster Off state */
};
```
## Power of Cluster P-State

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Cluster Power Off State (mW)</th>
<th>Cluster Power On State (mW)</th>
<th>Power of cluster level active idle state (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>208</td>
<td>344</td>
<td>360</td>
<td>16</td>
</tr>
<tr>
<td>432</td>
<td>345</td>
<td>374</td>
<td>29</td>
</tr>
<tr>
<td>729</td>
<td>346</td>
<td>393</td>
<td>47</td>
</tr>
<tr>
<td>960</td>
<td>352</td>
<td>427</td>
<td>75</td>
</tr>
<tr>
<td>1200</td>
<td>367</td>
<td>479</td>
<td>112</td>
</tr>
</tbody>
</table>

```c
static struct capacity_state cap_states_cluster_a53[] = {
    /* Power per cluster */
    { .cap = 178, .power = 16, },
    { .cap = 369, .power = 29, },
    { .cap = 622, .power = 47, },
    { .cap = 819, .power = 75, },
    { .cap = 1024, .power = 112, },
};
```
### EAS power modeling parameters

#### Cluster level

<table>
<thead>
<tr>
<th>Power per cluster</th>
<th>Cap</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>178</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>369</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>622</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>819</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>1024</td>
<td>112</td>
<td>112</td>
</tr>
</tbody>
</table>

#### CPU level

<table>
<thead>
<tr>
<th>Power per cpu</th>
<th>Cap</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>208MHz</td>
<td>178</td>
<td>69</td>
</tr>
<tr>
<td>432MHz</td>
<td>369</td>
<td>124</td>
</tr>
<tr>
<td>729MHz</td>
<td>622</td>
<td>224</td>
</tr>
<tr>
<td>960MHz</td>
<td>819</td>
<td>367</td>
</tr>
<tr>
<td>1.2GHz</td>
<td>1024</td>
<td>670</td>
</tr>
</tbody>
</table>
Overview for IPA power model

1. Report temperature, reach trip points
2. Gather actor's request budget
3. Get dynamic power
4. Get static power
5. Updated allocated budget
6. Update CPUFreq policy

IPA

Thermal sensor driver

CPU Cooling Device

Power Model for CPU actor

CPUFreq Framework
Power model based on dynamic power and static leakage

- Dynamic power
- SoC temperature impacts static leakage
- Cluster static leakage
- CPU static leakage
Power model as a linear equation

\[ \text{power} = "\text{capacitance}" \times (\text{freq} \times \text{volt}^2) + \text{static\_power} \]

Dynamic power

\[ Y = m \times X + b \]

Can easily calculate best-fit values for \( m \) and \( b \) using the \text{LINEST} function found in most spreadsheet programs.
## Review cluster power

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Voltage (v)</th>
<th>Cluster Power Off State (mW)</th>
<th>Cluster P-state (mW)</th>
<th>Cluster Power (mW)</th>
<th>F * V^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>208</td>
<td>1.04</td>
<td>344</td>
<td>360</td>
<td>16</td>
<td>225</td>
</tr>
<tr>
<td>432</td>
<td>1.04</td>
<td>345</td>
<td>374</td>
<td>29</td>
<td>467</td>
</tr>
<tr>
<td>729</td>
<td>1.09</td>
<td>346</td>
<td>393</td>
<td>47</td>
<td>866</td>
</tr>
<tr>
<td>960</td>
<td>1.18</td>
<td>352</td>
<td>427</td>
<td>75</td>
<td>1337</td>
</tr>
<tr>
<td>1200</td>
<td>1.33</td>
<td>367</td>
<td>479</td>
<td>112</td>
<td>2123</td>
</tr>
</tbody>
</table>
Linear regression on the cluster power

Linear regression for cluster power

- Power (mW)
- L.R Power Model

Gradient gives “capacitance” = 0.051

Intercept gives static power (~5mW)
Review CPU power

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Voltage (v)</th>
<th>Cluster P-state (mW)</th>
<th>CPU P-state (mW)</th>
<th>CPU Dynamic Power (mW)</th>
<th>F * V^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>208</td>
<td>1.04</td>
<td>360</td>
<td>429</td>
<td>69</td>
<td>225</td>
</tr>
<tr>
<td>432</td>
<td>1.04</td>
<td>374</td>
<td>498</td>
<td>124</td>
<td>467</td>
</tr>
<tr>
<td>729</td>
<td>1.09</td>
<td>393</td>
<td>617</td>
<td>224</td>
<td>866</td>
</tr>
<tr>
<td>960</td>
<td>1.18</td>
<td>427</td>
<td>794</td>
<td>367</td>
<td>1337</td>
</tr>
<tr>
<td>1200</td>
<td>1.33</td>
<td>479</td>
<td>1149</td>
<td>670</td>
<td>2123</td>
</tr>
</tbody>
</table>
Linear regression on the CPU power

A negative intercept (static power) makes no sense physically and strongly suggests the line is nonlinear. In the graph we see the highest two OPPs’ power increase sharply.

Gradient (capacitance) = 0.317

Intercept (static power) is negative value -27
Potential solutions to generate a model

● Three possible options
  ○ Can't have negative static power because `get_static_power()` cannot return negative number, but we could force linear regression through zero if we accept that there will be some flaws in the model.
  ○ Could correct for temperature. It is likely the highest two OPPs start to curve upwards as die temperature rises and static leakage increases.
  ○ Could modify kernel to remove IPA model and use EAS figures directly. These physical measurements already include a contribution from die temperature...

● We didn’t collect temperature data for Hikey and DT bindings for EAS are not yet agreed so, for now, linear regression through zero is the simplest choice.
Linear regression through zero on the CPU power

Using linear regression through zero gives worst case error of ~10%

Intercept (static power) is 0

Gradient (capacitance) = 0.298
IPA power coefficients

```c
struct cluster_power_coefficients cluster_data = {
    .dyn_coeff = 311,
    (0.298 + (0.054 / 4 CPUs)) * 1000 = 311
};
```

A simplistic power modeling without taking account of static leakage:

1. Given the non-linearity for CPU power data then modeling static power is pointless
2. Updated value for cluster “capacitance” which is now forced through zero as well
3. Error in model can be tolerated in some use cases; dynamic power remains main contributor to power consumption
4. PID controller’s integral term will partly compensate for temperature drift
Decide sustainable power based on OPP

Temperature increases very quickly when 8CPUs@1.2GHz, meaning the power consumption is above max sustainable level.

Temperature increases slowly when 8CPUs@960MHz, system is only very slightly above sustainable power.

When CPU runs at 729MHz the temperature doesn’t increase. Its power consumption is less than the sustainable power.

<table>
<thead>
<tr>
<th>OPP (MHz)</th>
<th>Sustainable power</th>
</tr>
</thead>
<tbody>
<tr>
<td>729</td>
<td>2155</td>
</tr>
<tr>
<td>960</td>
<td>3326</td>
</tr>
<tr>
<td>1200</td>
<td>5285</td>
</tr>
</tbody>
</table>
Agenda

- Background
- Low power states
- Measurement on Hikey
- Generate power model
- Conclusions
Conclusions

● IPA was very effective on HiKey even with a poor model
  ○ Increase of ~40% performance compared to step-wise governor

● Allow EAS model to be used for IPA
  ○ Allows direct link between observed values to power mode
  ○ Using EAS values directly is convenient for simple systems although it makes it impossible to explicitly model die temperature effects

● Recommend to record temperature alongside power data
  ○ If possible enable temperature sensor for SoC before gathering power info

● Other suggestions
  ○ Use CFS’s CPU utilization for accurate duty cycle and calculate dynamic power
Thanks!

Linaro Premium Services includes the delivery of tailored training courses for Linaro Club and Core members. This presentation is an abridgement of material prepared as part our training programme.