Linux Power

Basic, Linux PM, DVFS, OPP, Runtime PM, PM QoS

국민대학교 임베디드 연구실
경 주 현

2017 LG MC 인재육성팀 교육

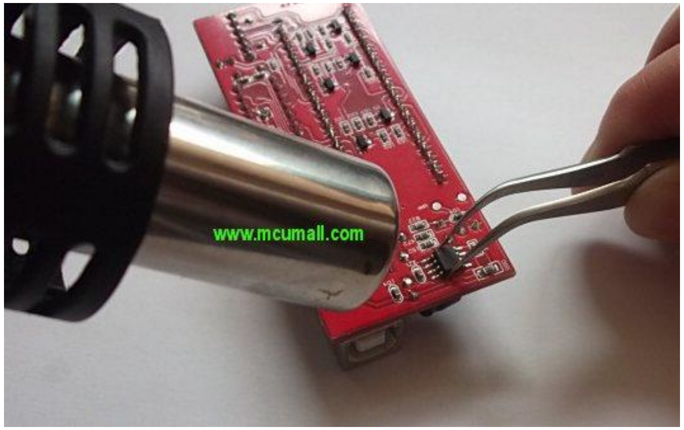
Outline

- Processor Power Management

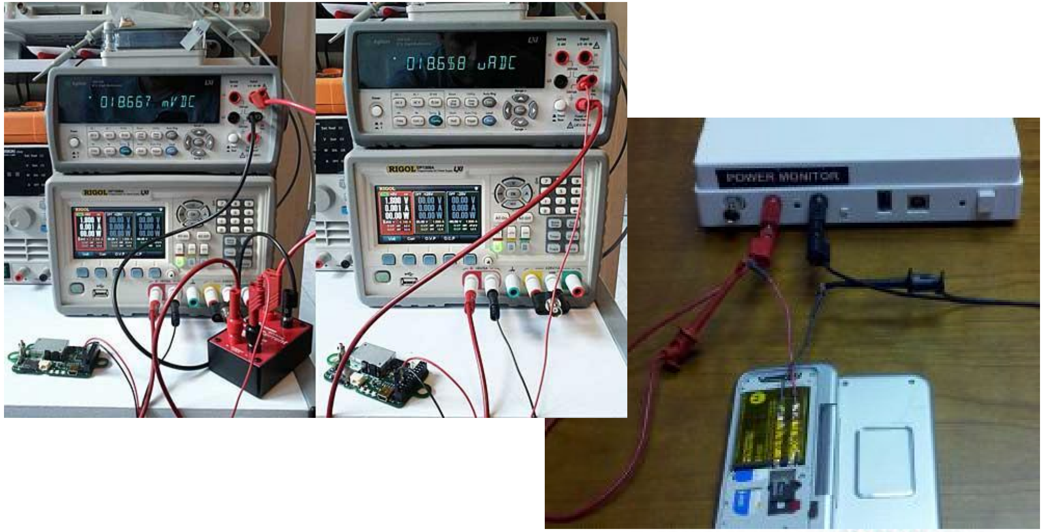
- Linux Power Management

How to reduce power?

How to reduce power?



WIFI, Modem chip, LCD, Touch ....



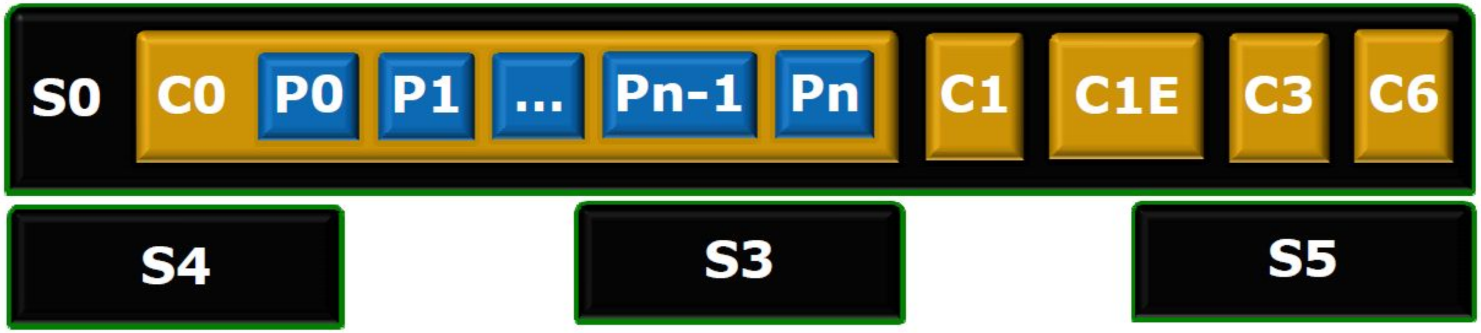
- Power Rail Check
- Total Current Check

Background

- **Energy**
  - joule
- **Power**
  - watt
- **Processor Power Management**
  - P-state, C-state

Processor Power Management

Silicon Power Management Features

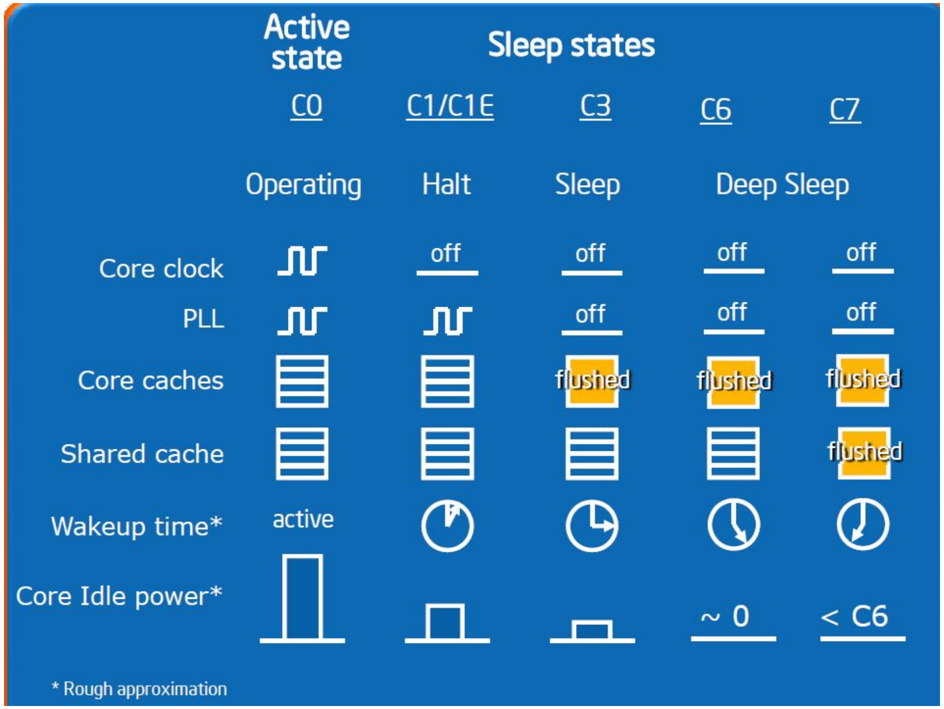


- **P – States are a sub-state of the C0 state**
  - They offer reduced power consumption while the processor is executing code.
- **C-States are a sub-state of the S0 state**
  - They offer reduced power consumption while the system is fully on.
- **S-States**
  - Suspend to idle, Standby, Suspend to ram, suspend to disk

Processor Power Management

- **CPU C-State (Idle Power State)**
  - C0 - Full On State
  - C1 – Auto Halt State
  - C2, C4 (C3, C4, C5), C6

- **CPU P-State (Performance State)**



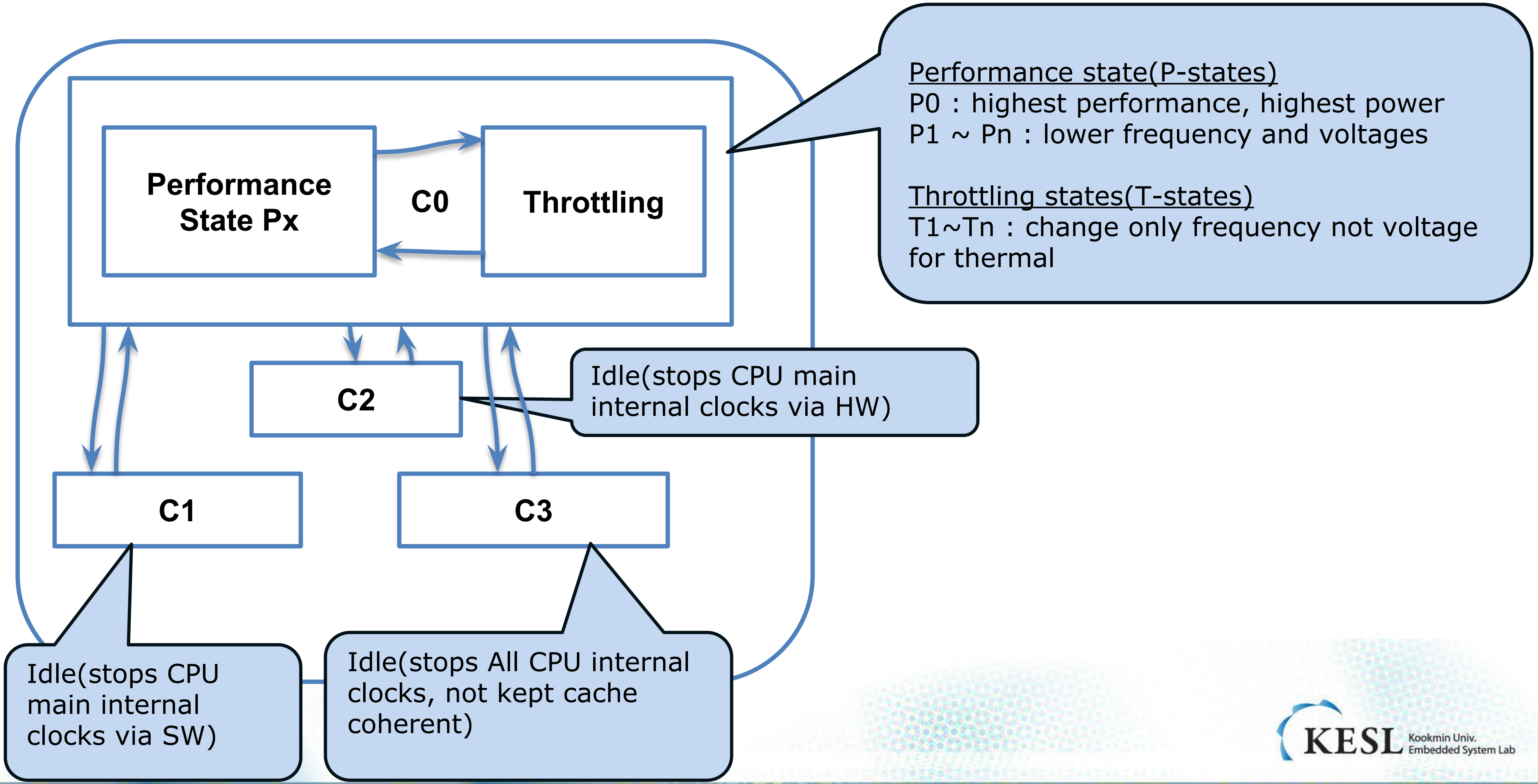
	Active state	Sleep states			
	C0	C1/C1E	C3	C6	C7
	Operating	Halt	Sleep	Deep Sleep	
Core clock	⎍⎍	off	off	off	off
PLL	⎍⎍	⎍⎍	off	off	off
Core caches	≡	≡	flushed	flushed	flushed
Shared cache	≡	≡	≡	≡	flushed
Wakeup time*	active	◷	◷	◷	◷
Core Idle power*	█	▄	▁	~ 0	< C6

* Rough approximation

http://download.intel.com/embedded/technology/PowerManagementforEmbeddedApps.pdf

Silicon Power Management Features

- **Advanced Configuration and Power Interface (ACPI)**



Performance state(P-states)
P0 : highest performance, highest power
P1 ~ Pn : lower frequency and voltages

Throttling states(T-states)
T1~Tn : change only frequency not voltage for thermal

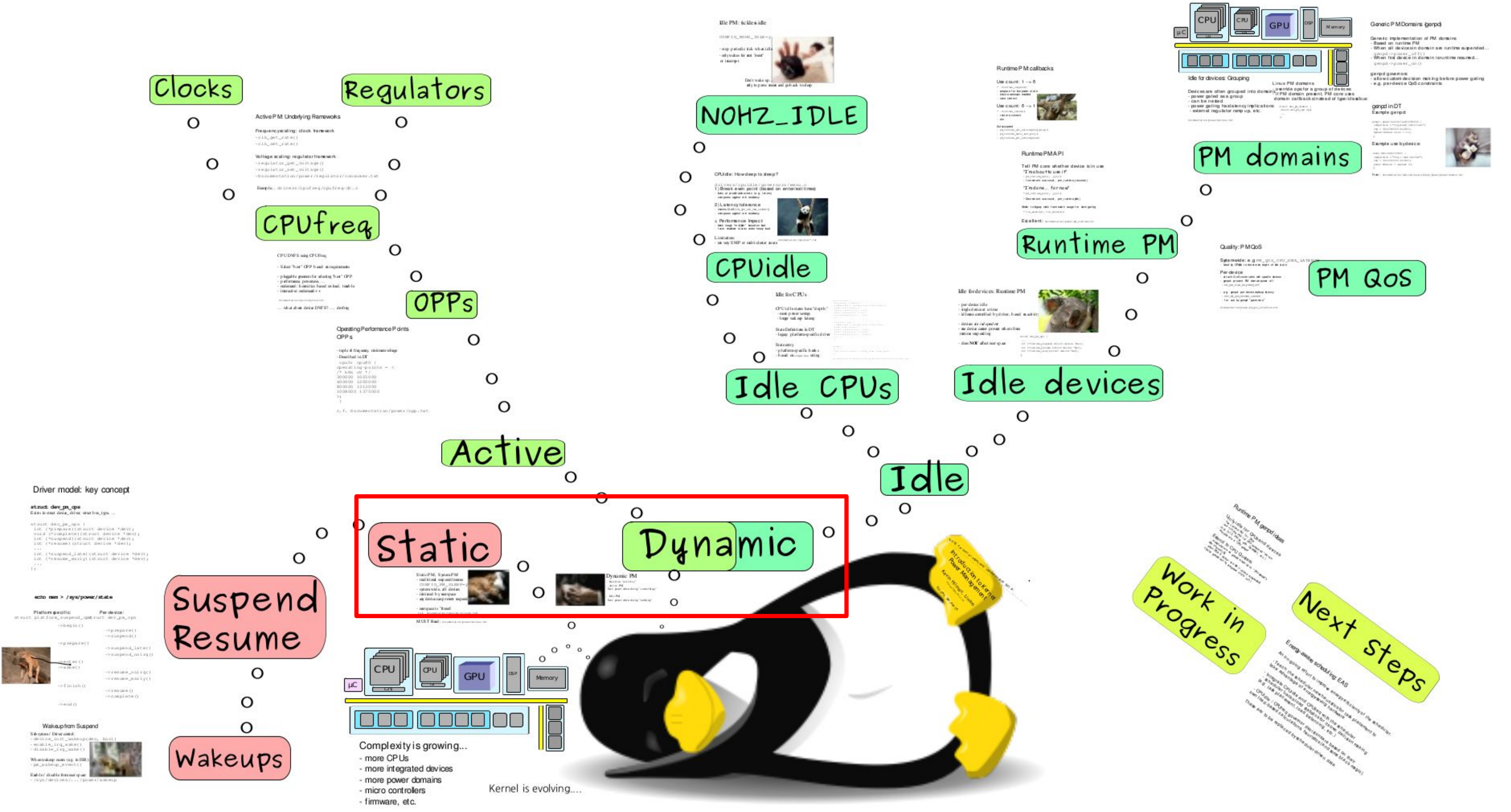
Idle(stops CPU main internal clocks via HW)

Idle(stops CPU main internal clocks via SW)

Idle(stops All CPU internal clocks, not kept cache coherent)

Linux Power Management

Tree view - Dynamic and Static PM



Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

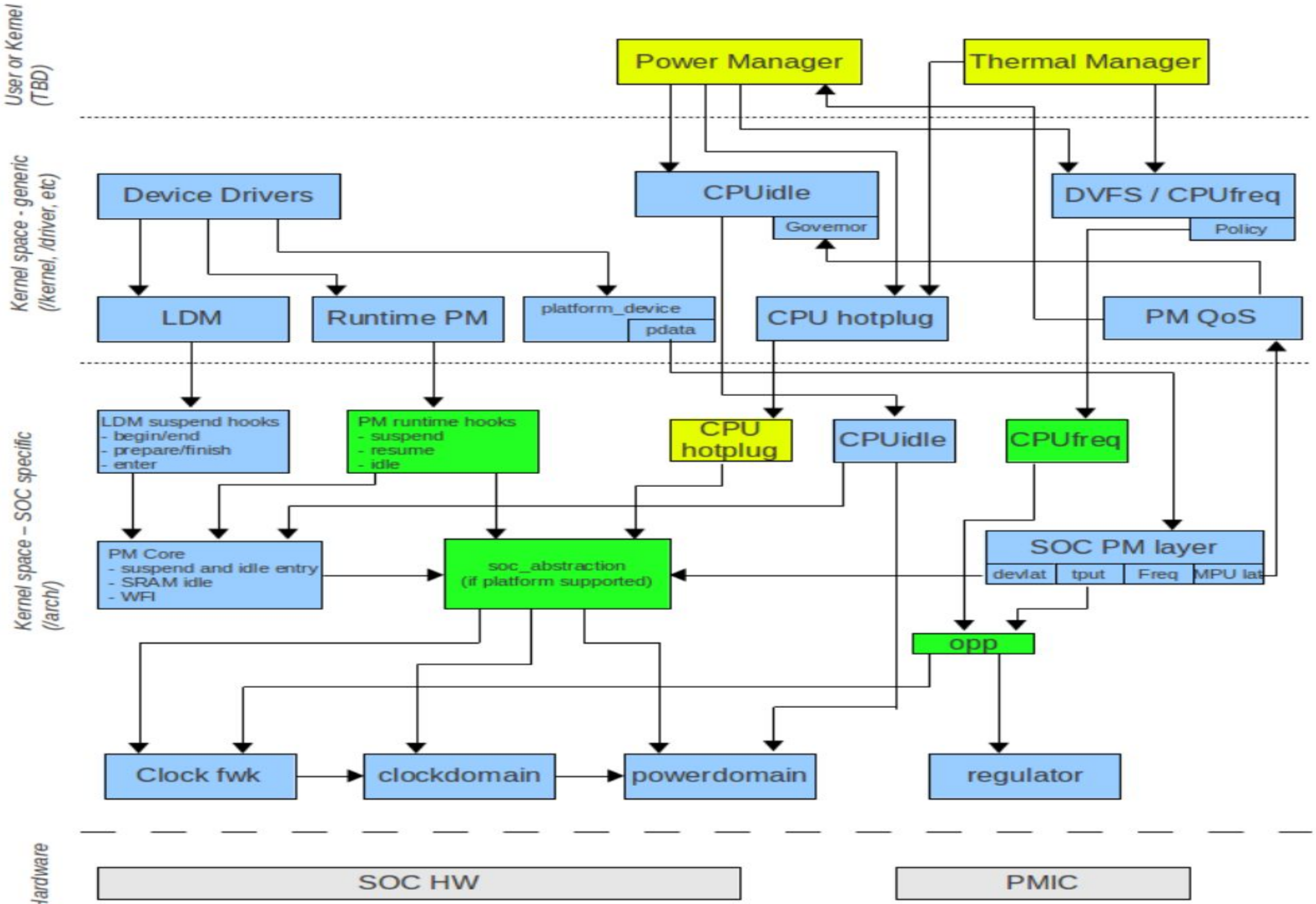
Linux Driver Model

- **Example**

```
static struct device_driver eepro100_driver = {
        .name           = "eepro100",
        .bus            = &pci_bus_type,

        .probe          = eepro100_probe,
        .remove         = eepro100_remove,
        .suspend                = eepro100_suspend,
        .resume         = eepro100_resume,
};
```

Linux Power Framework



User or Kernel (TBD)

Kernel space - generic (/kernel, /driver, etc)

Kernel space – SOC specific (/arch)

Hardware

Power Manager

Thermal Manager

Device Drivers

CPUidle

Governor

DVFS / CPUfreq

Policy

LDM

Runtime PM

platform_device

pdata

CPU hotplug

PM QoS

LDM suspend hooks
- begin/end
- prepare/finish
- enter

PM runtime hooks
- suspend
- resume
- idle

CPU hotplug

CPUidle

CPUfreq

PM Core
- suspend and idle entry
- SRAM idle
- WFI

soc_abstraction (if platform supported)

SOC PM layer

devlat | tput | Freq | MPU lat

opp

Clock fwk

clockdomain

powerdomain

regulator

SOC HW

PMIC

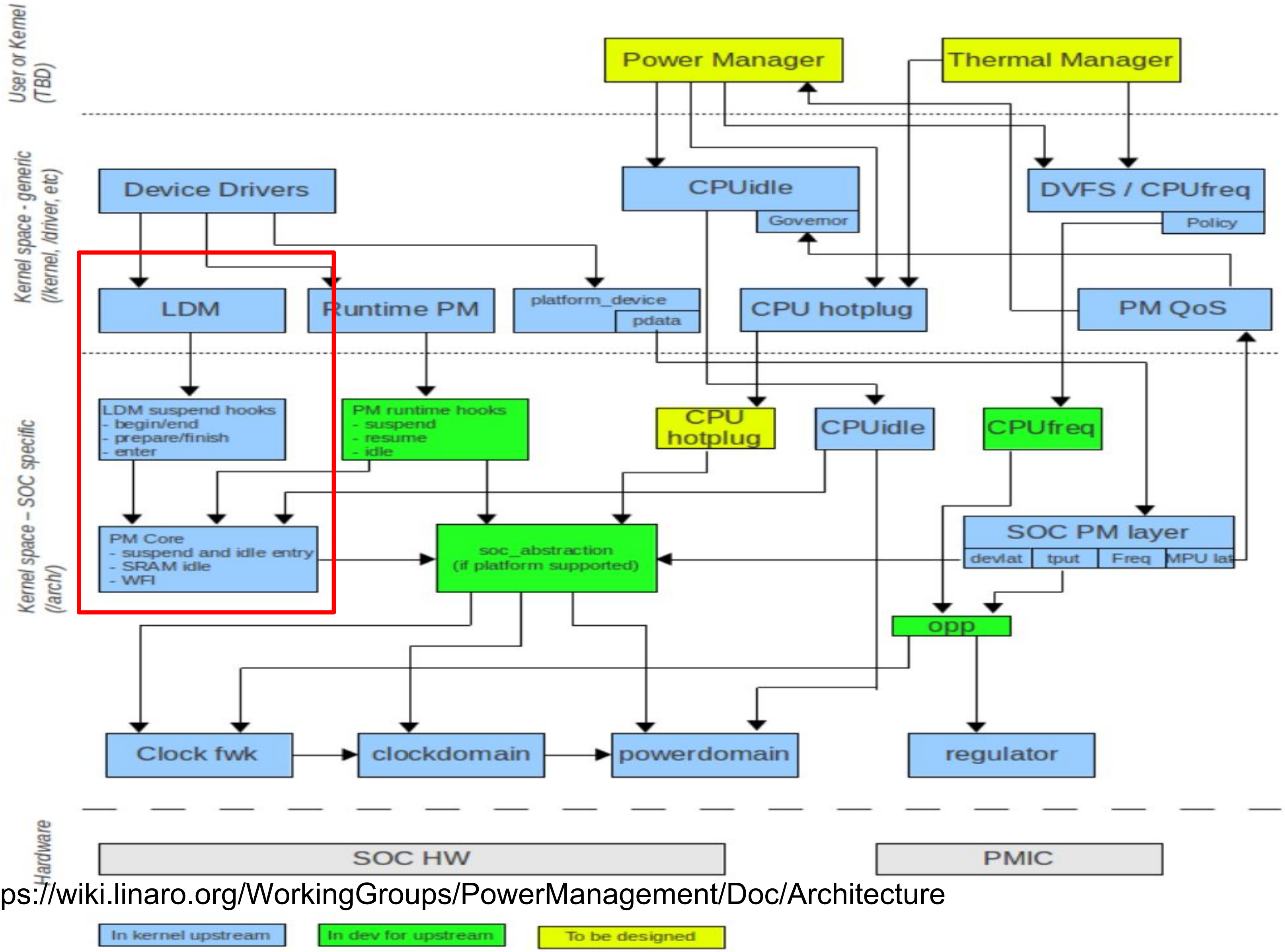
https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

In kernel upstream

In dev for upstream

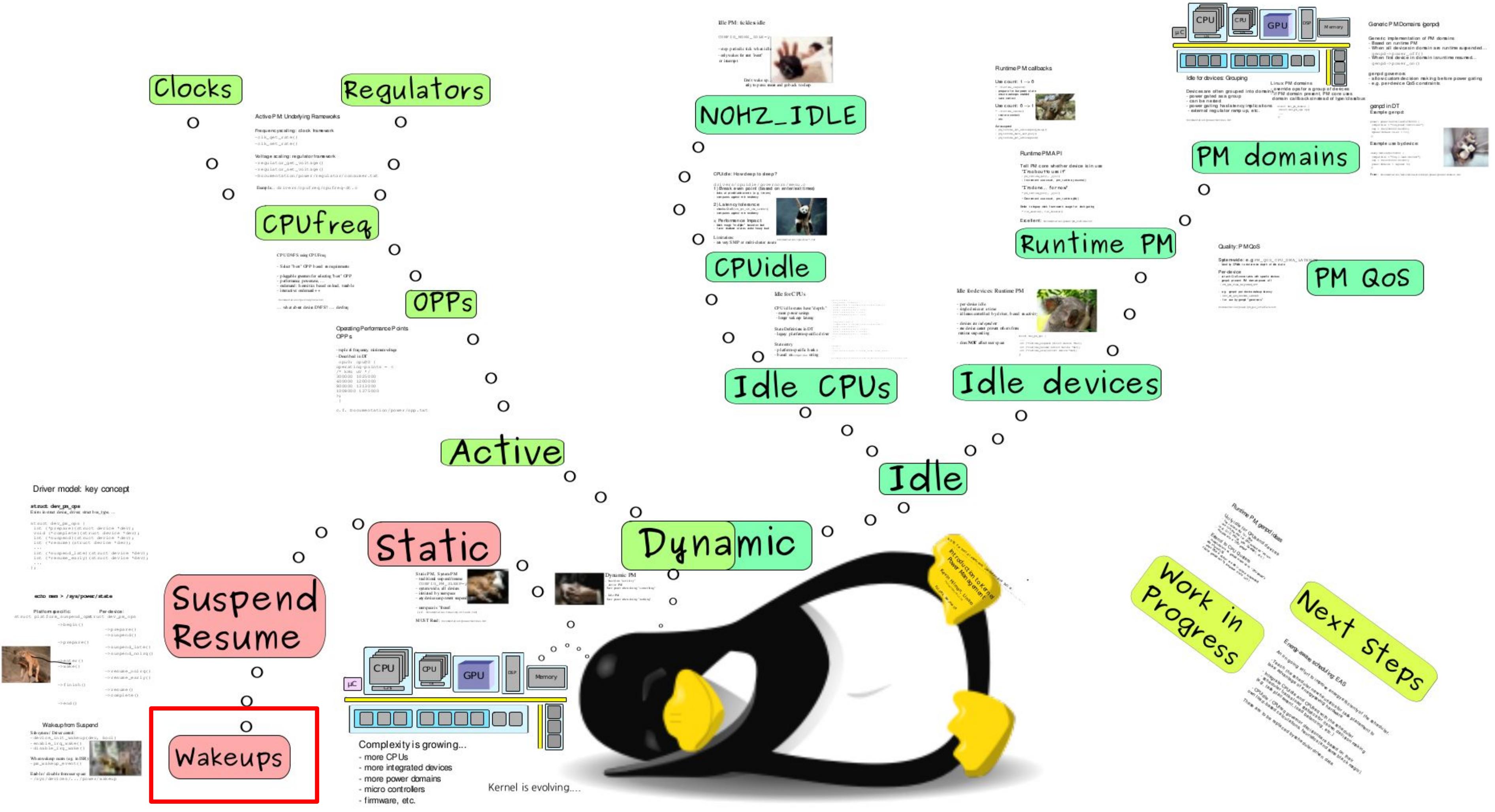
To be designed

Linux Driver Model



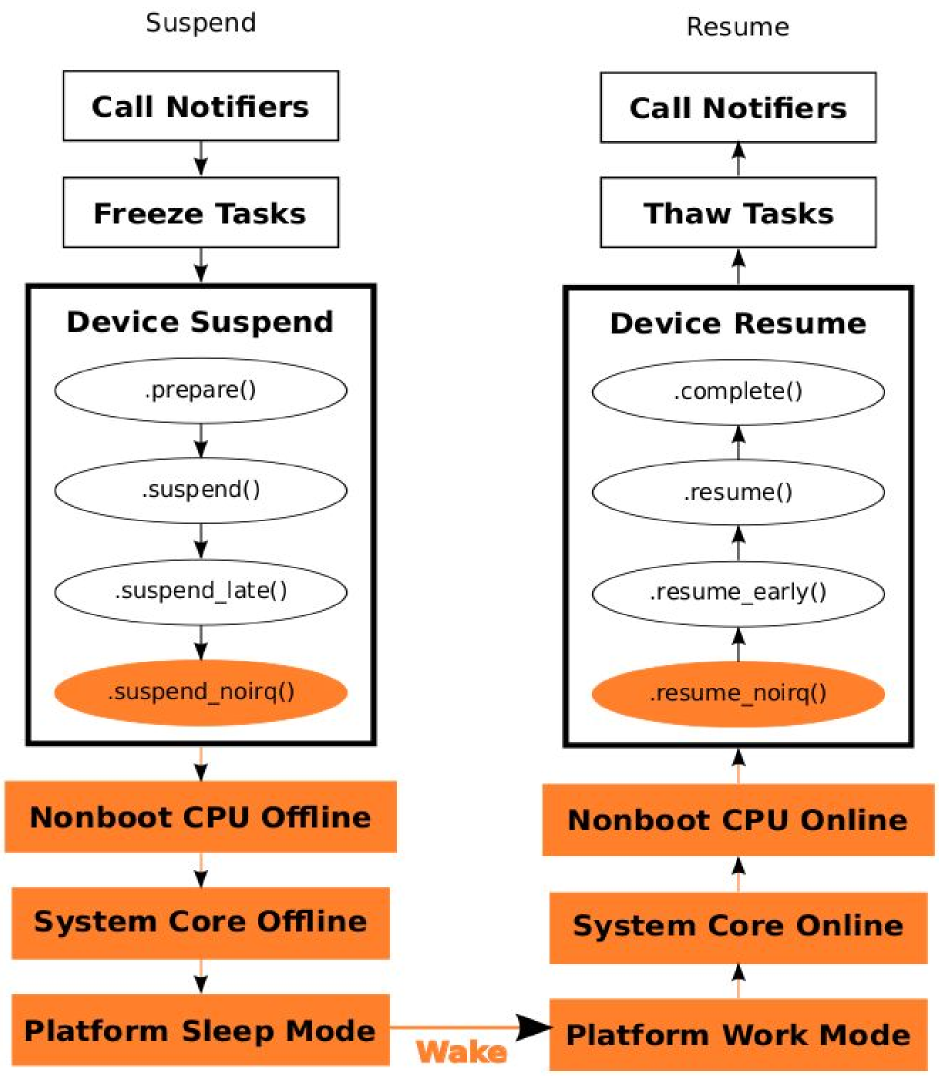
https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

Tree view - Dynamic and Static PM



Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

General System Suspend/Resume Control Flow



Power Management In The Linux* Kernel - Rafael J. Wysocki

Problem

- How to reduce system idle power?

Problem

- How to reduce system idle power?
- What about CPU Hotplugging?

Problem

- How to reduce system idle power?
- What about CPU Hotplugging?
  - Expensive latency
- Solution?

Problem

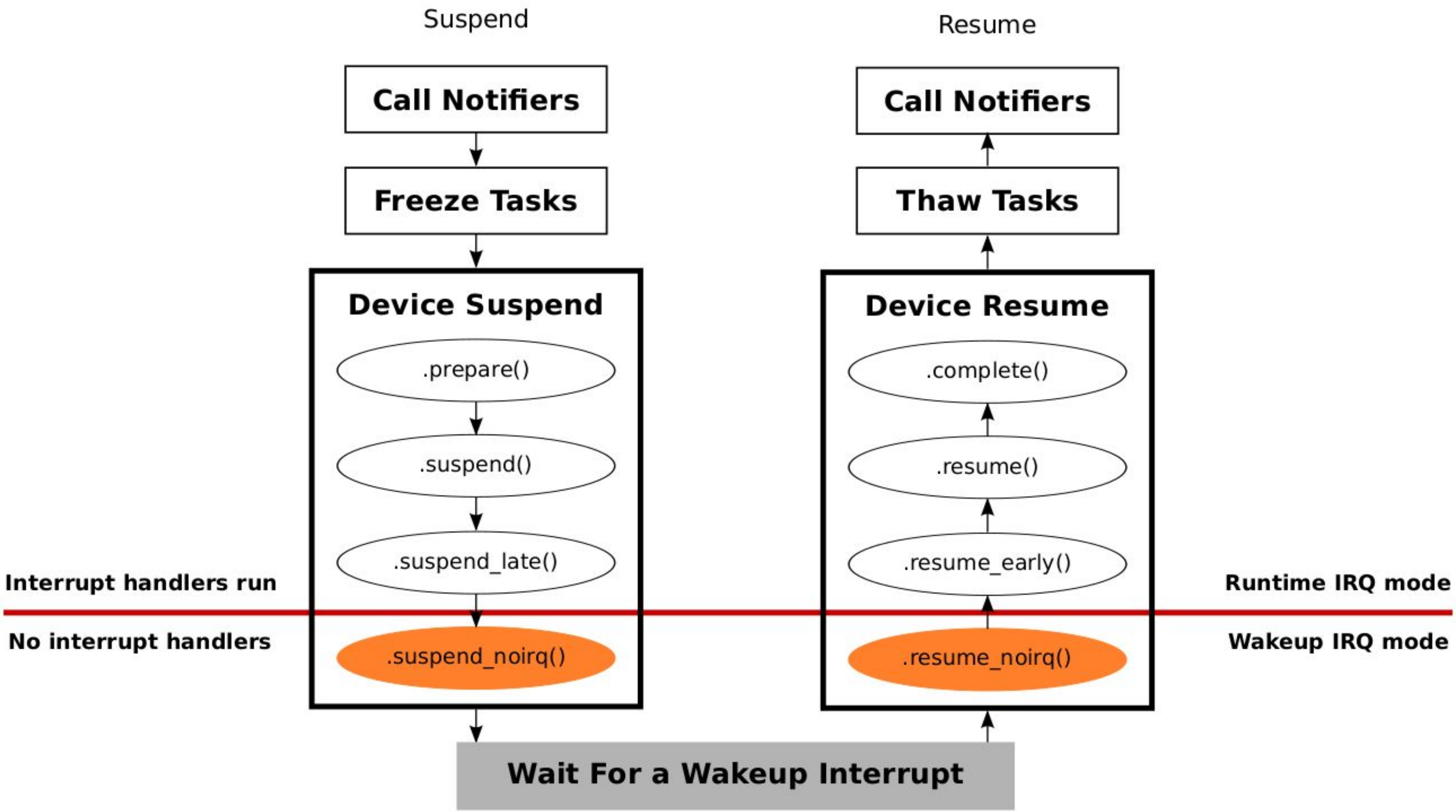
- How to reduce system idle power?

- What about CPU Hotplugging?
  - Expensive latency

- Solution?

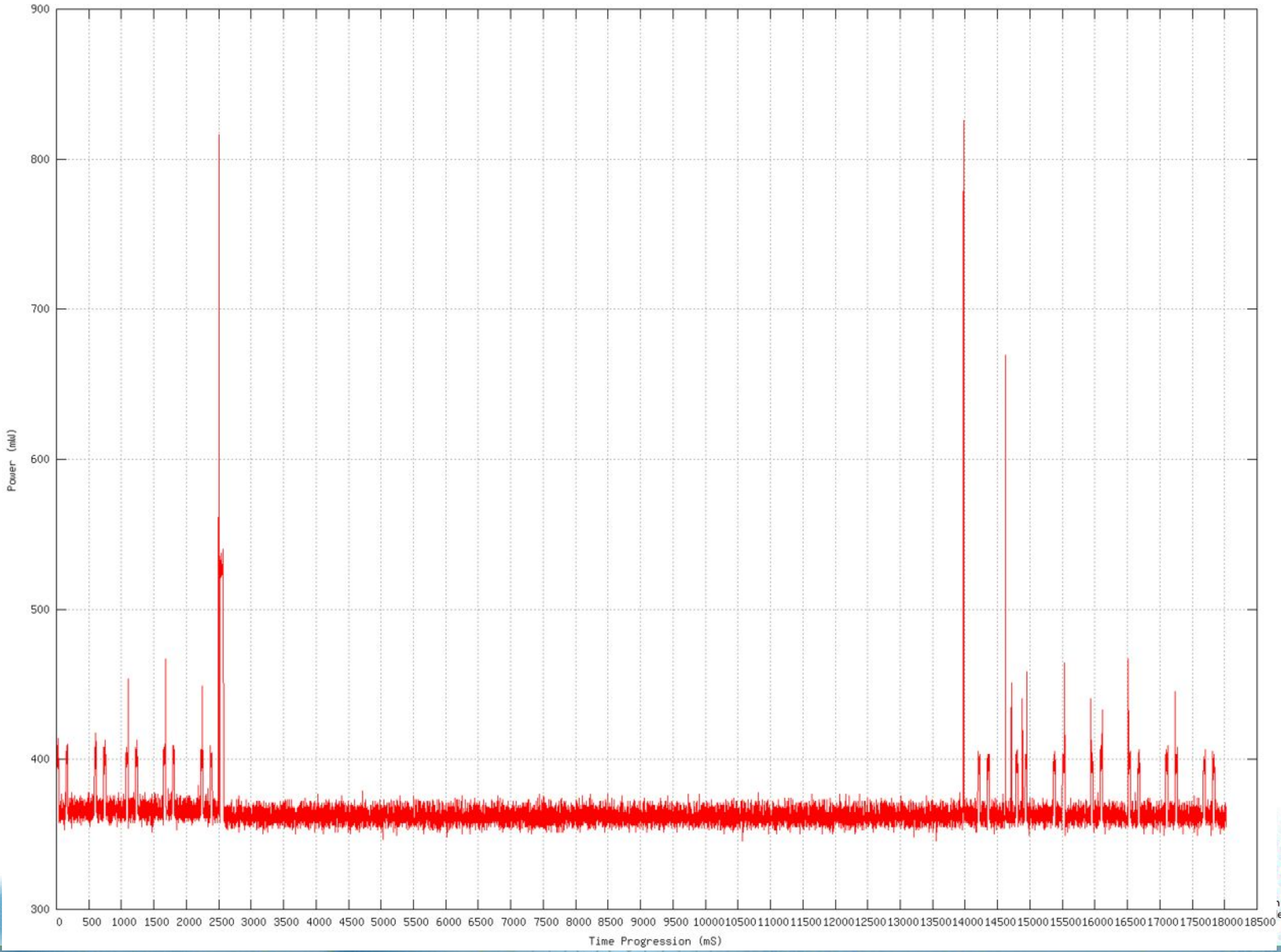
S2I(Suspend to Idle)

Suspend-to-Idle

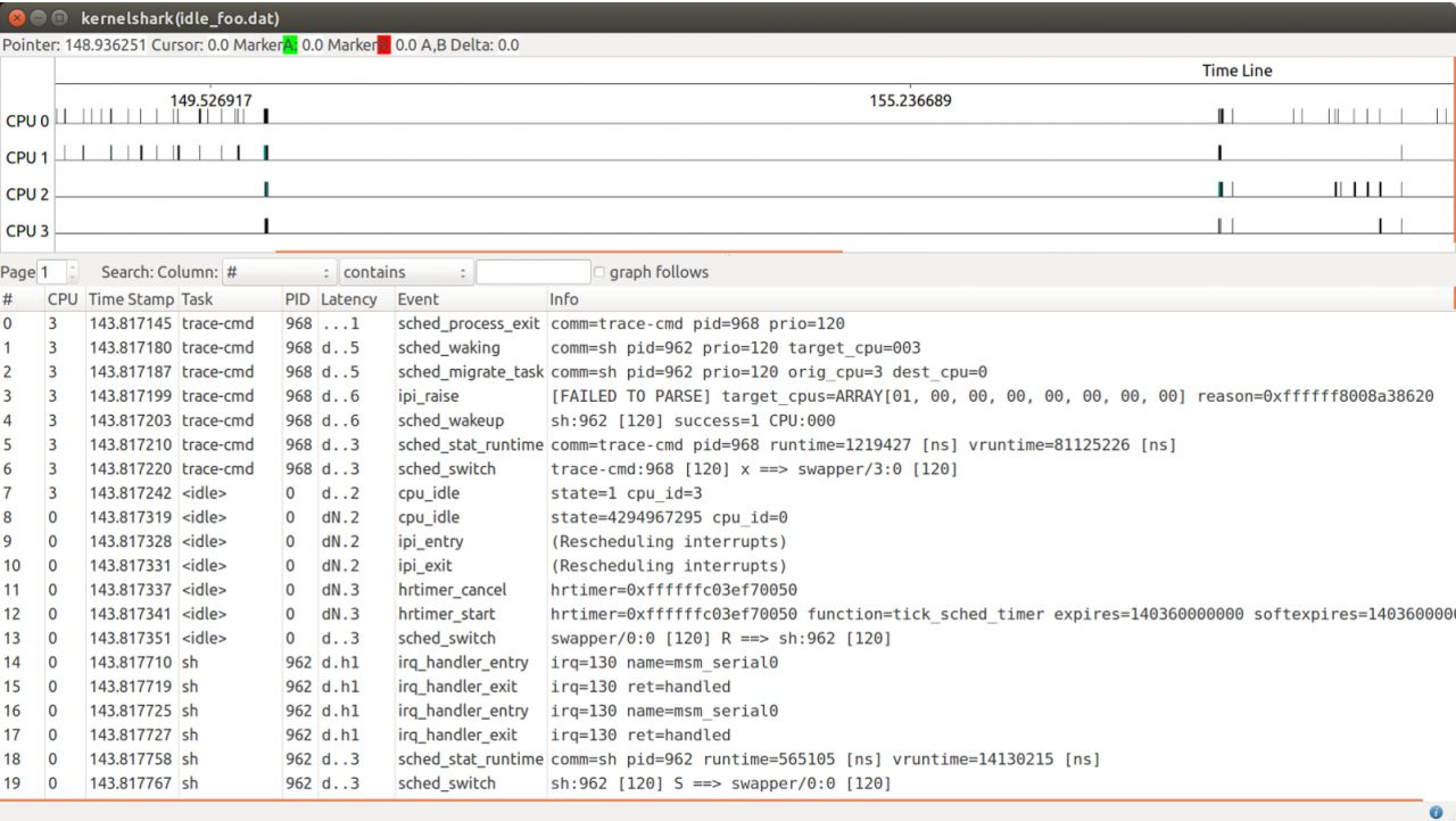


https://www.linaro.org/blog/suspend-to-idle/

Suspend-to-Idle



Suspend-to-Idle

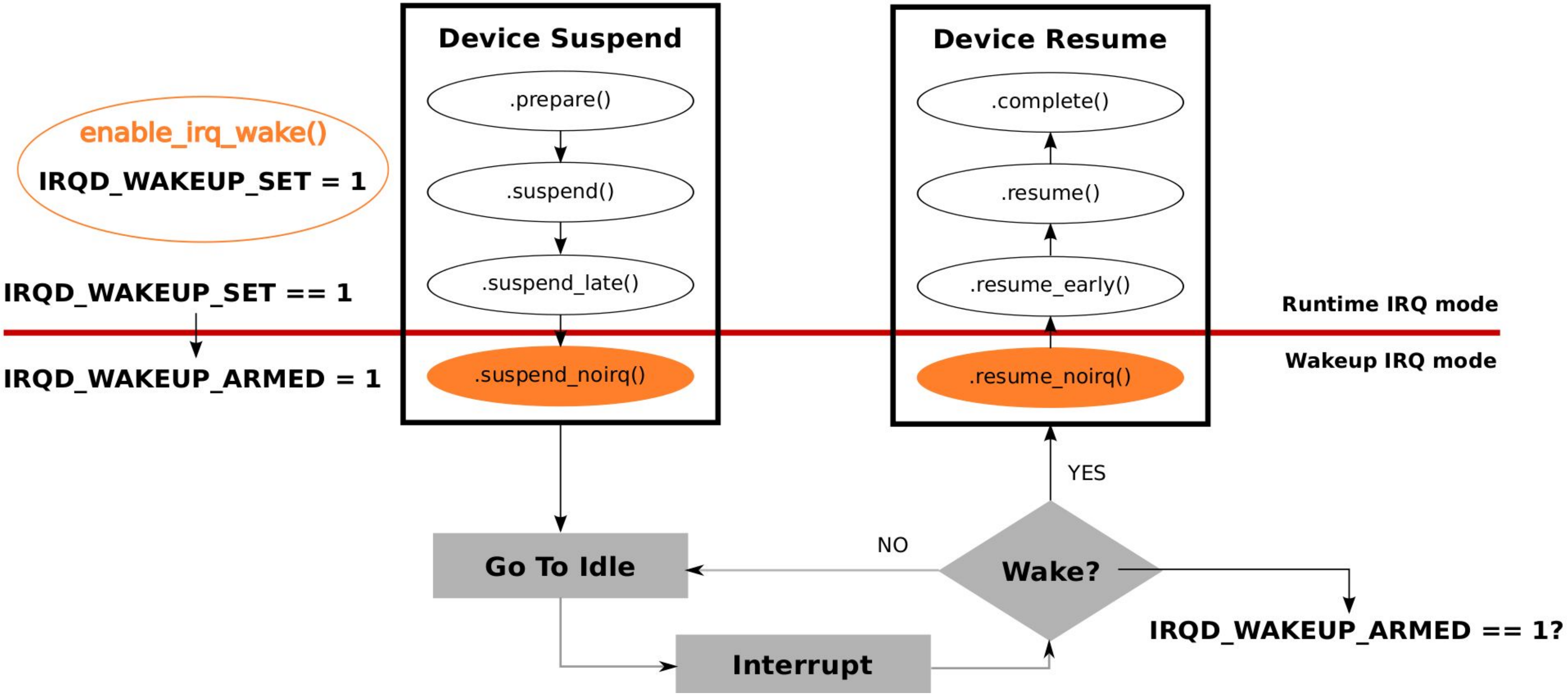


Pointer: 148.936251 Cursor: 0.0 MarkerA: 0.0 Marker🟥 0.0 A,B Delta: 0.0

#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	3	143.817145	trace-cmd	968	...1	sched_process_exit	comm=trace-cmd pid=968 prio=120
1	3	143.817180	trace-cmd	968	d..5	sched_waking	comm=sh pid=962 prio=120 target_cpu=003
2	3	143.817187	trace-cmd	968	d..5	sched_migrate_task	comm=sh pid=962 prio=120 orig_cpu=3 dest_cpu=0
3	3	143.817199	trace-cmd	968	d..6	ipi_raise	[FAILED TO PARSE] target_cpus=ARRAY[01, 00, 00, 00, 00, 00, 00, 00] reason=0xffffff8008a38620
4	3	143.817203	trace-cmd	968	d..6	sched_wakeup	sh:962 [120] success=1 CPU:000
5	3	143.817210	trace-cmd	968	d..3	sched_stat_runtime	comm=trace-cmd pid=968 runtime=1219427 [ns] vruntime=81125226 [ns]
6	3	143.817220	trace-cmd	968	d..3	sched_switch	trace-cmd:968 [120] x ==> swapper/3:0 [120]
7	3	143.817242	<idle>	0	d..2	cpu_idle	state=1 cpu_id=3
8	0	143.817319	<idle>	0	dN.2	cpu_idle	state=4294967295 cpu_id=0
9	0	143.817328	<idle>	0	dN.2	ipi_entry	(Rescheduling interrupts)
10	0	143.817331	<idle>	0	dN.2	ipi_exit	(Rescheduling interrupts)
11	0	143.817337	<idle>	0	dN.3	hrtimer_cancel	hrtimer=0xffffffc03ef70050
12	0	143.817341	<idle>	0	dN.3	hrtimer_start	hrtimer=0xffffffc03ef70050 function=tick_sched_timer expires=140360000000 softexpires=140360000
13	0	143.817351	<idle>	0	d..3	sched_switch	swapper/0:0 [120] R ==> sh:962 [120]
14	0	143.817710	sh	962	d.h1	irq_handler_entry	irq=130 name=msm_serial0
15	0	143.817719	sh	962	d.h1	irq_handler_exit	irq=130 ret=handled
16	0	143.817725	sh	962	d.h1	irq_handler_entry	irq=130 name=msm_serial0
17	0	143.817727	sh	962	d.h1	irq_handler_exit	irq=130 ret=handled
18	0	143.817758	sh	962	d..3	sched_stat_runtime	comm=sh pid=962 runtime=565105 [ns] vruntime=14130215 [ns]
19	0	143.817767	sh	962	d..3	sched_switch	sh:962 [120] S ==> swapper/0:0 [120]

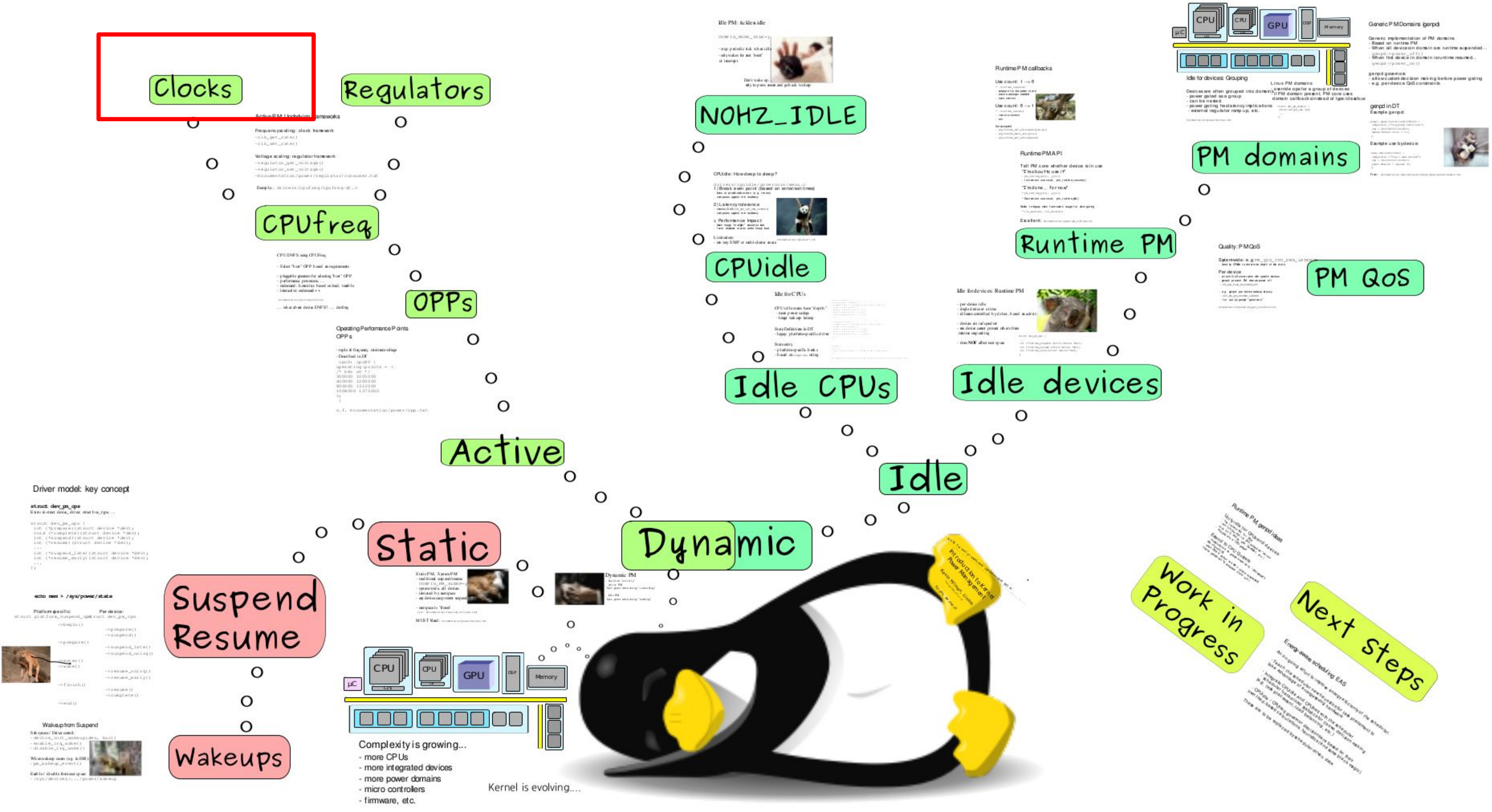
https://www.linaro.org/blog/suspend-to-idle/

Suspend-to-Idle Wakeup



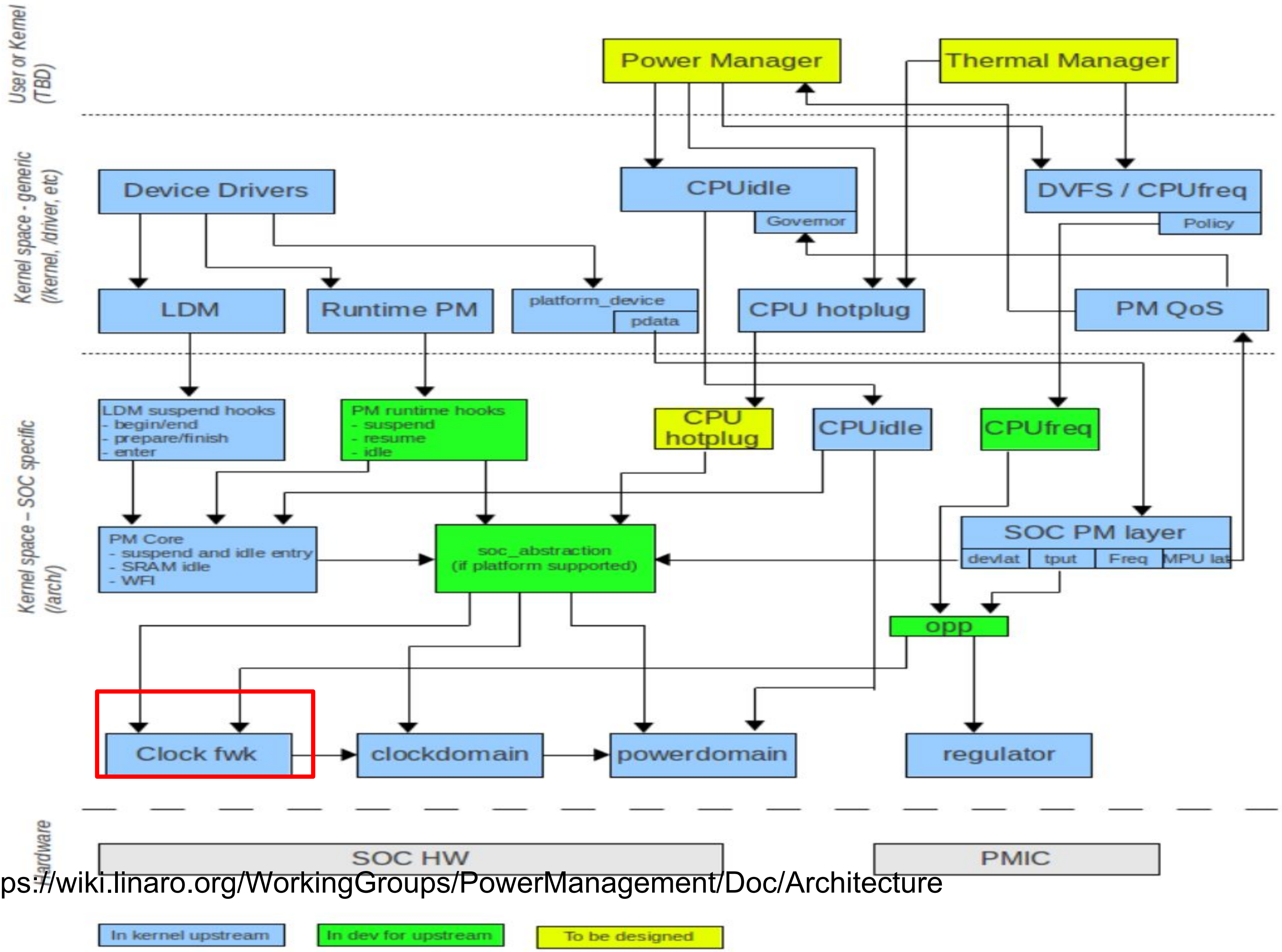
"What Is Suspend-to-Idle and How To Make It Work", Rafael J. Wysocki

Tree view - Dynamic and Static PM



Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

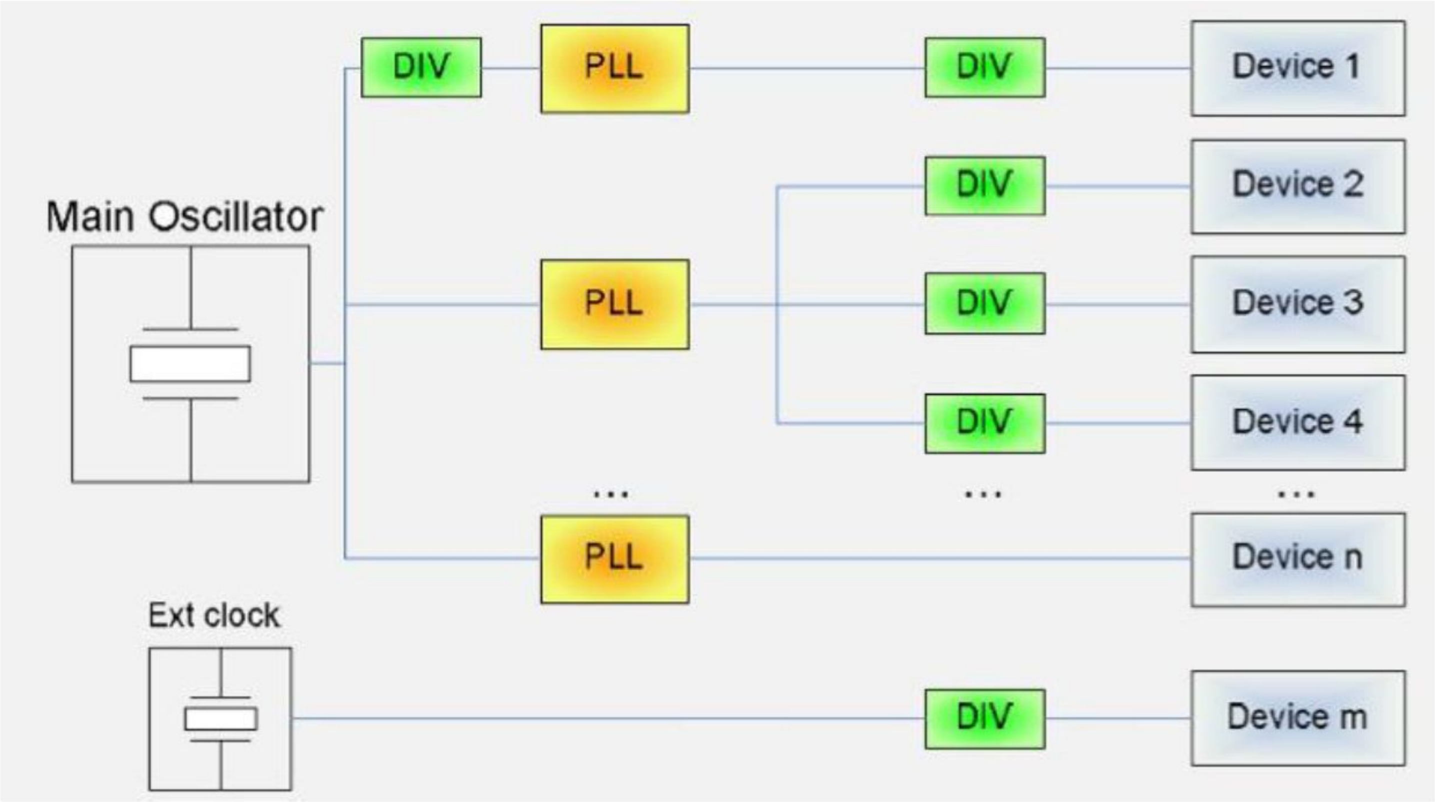
Linux Power Framework



https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

Clock Framework

- **Clock Domains**



Clock Framework – function list

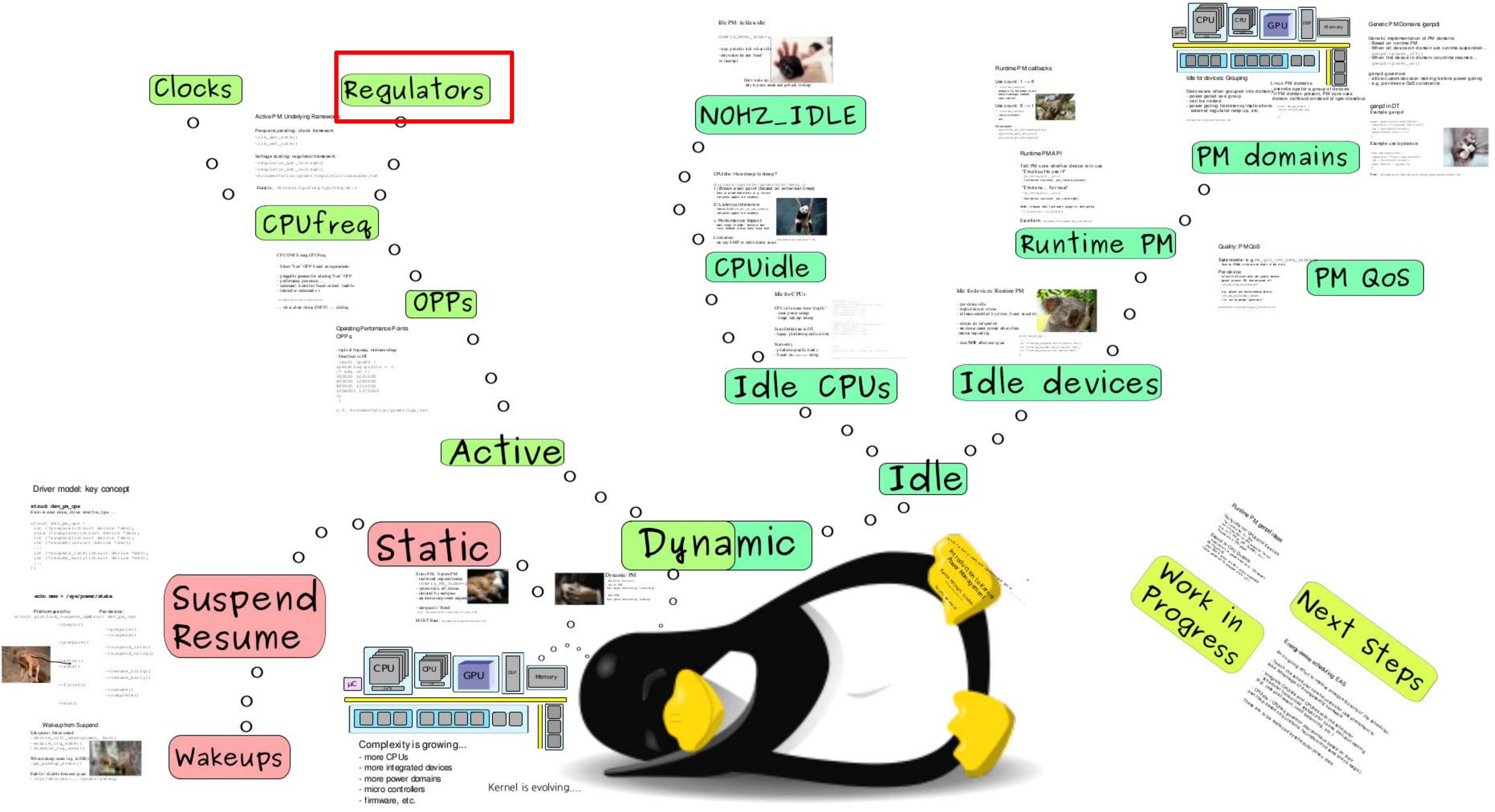
- **clk_get**
  - Lookup and obtain a reference to a clock producer

- **clk_enable**
  - Inform the system when the clock source should be running.

- **clk_disable**
  - Inform the system when the clock source is no longer required.

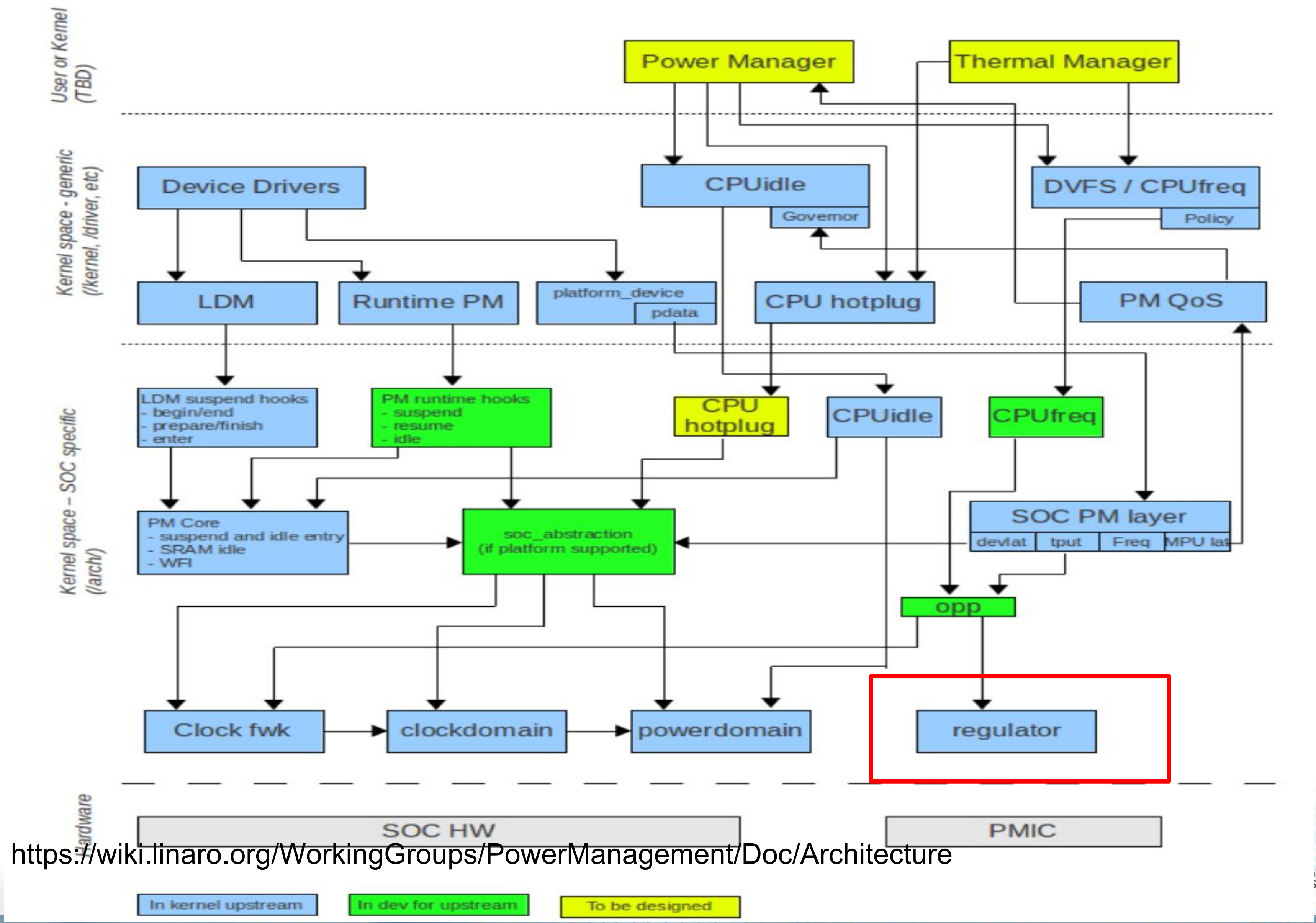
- **clk_put**
  - “free” the clock source

Tree view - Dynamic and Static PM



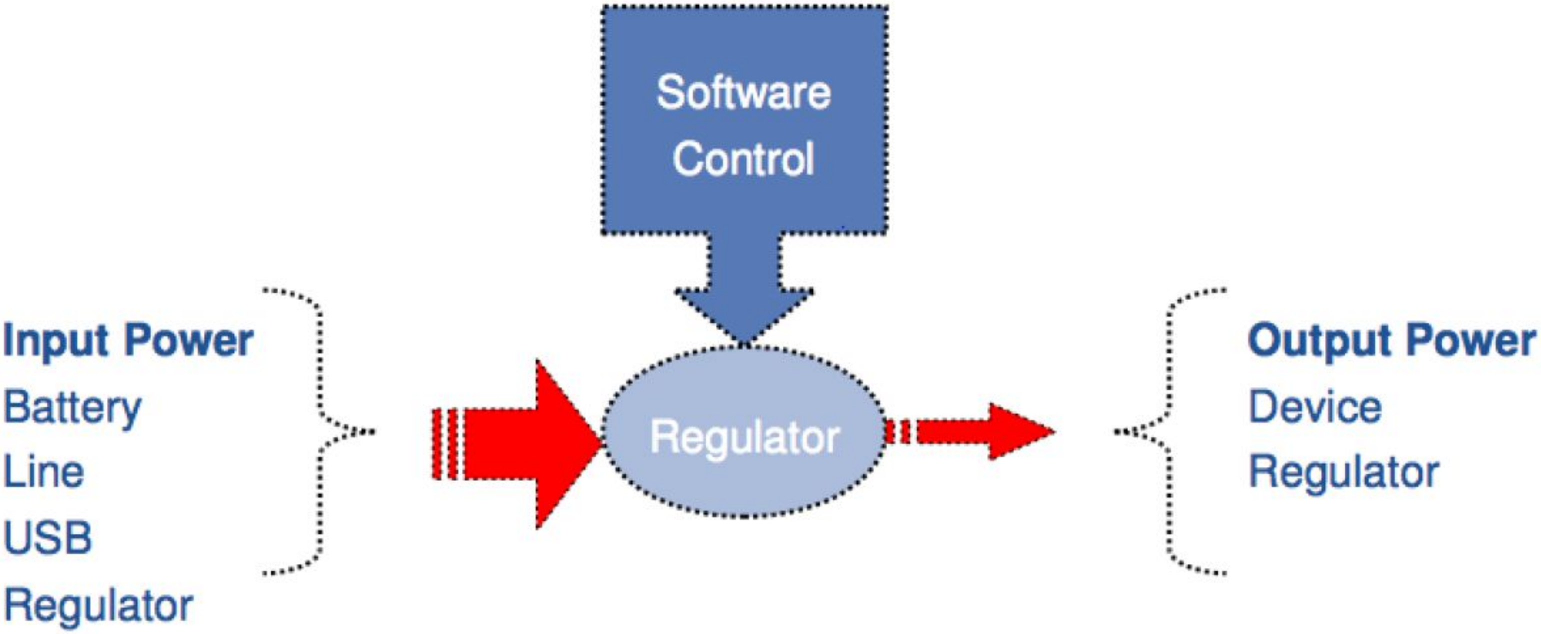
Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

Linux Power Framework



https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

Regulator Framework - 1



Input Power

Battery

Line

USB

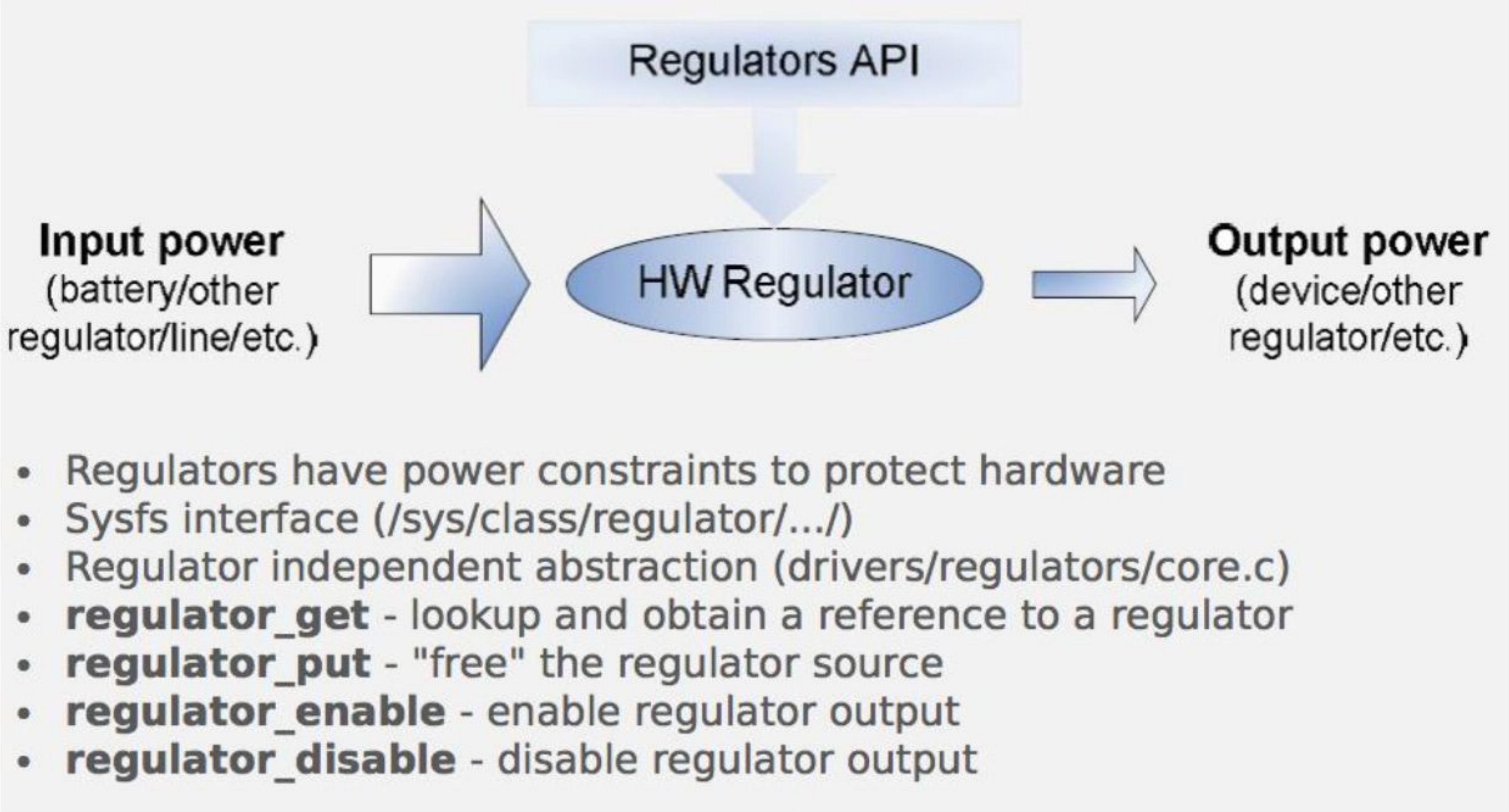
Regulator

Output Power

Device

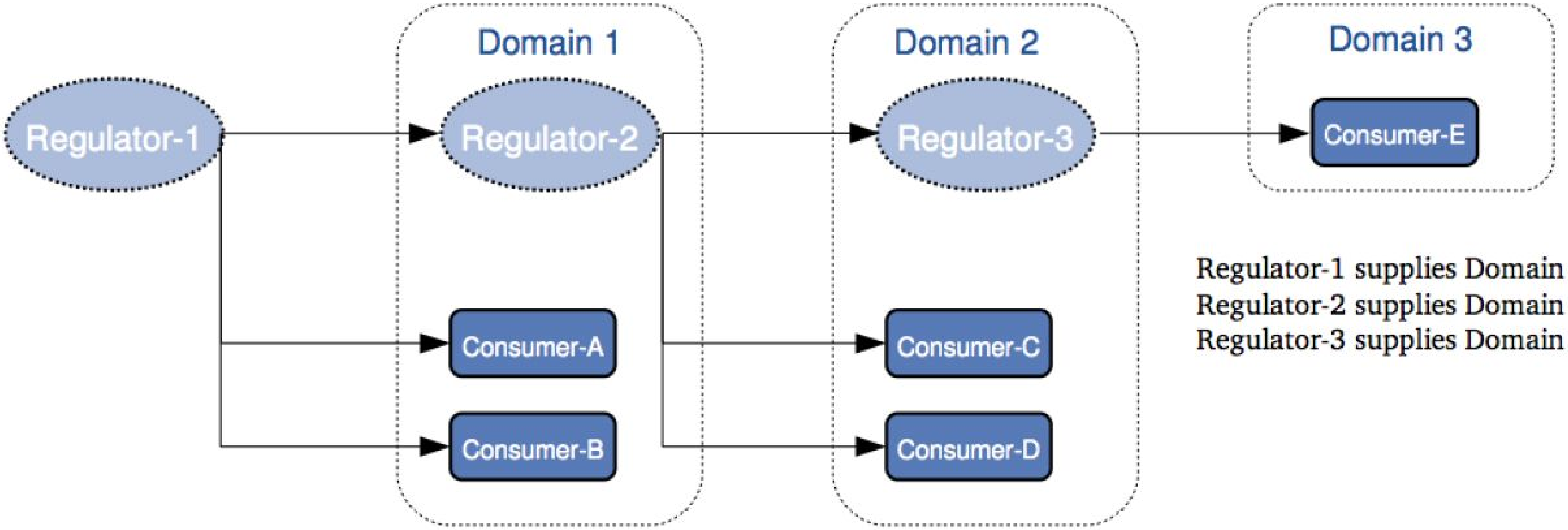
Regulator

Regulator Framework - 4


Regulators API
Input power
(battery/other regulator/line/etc.)
HW Regulator
Output power
(device/other regulator/etc.)

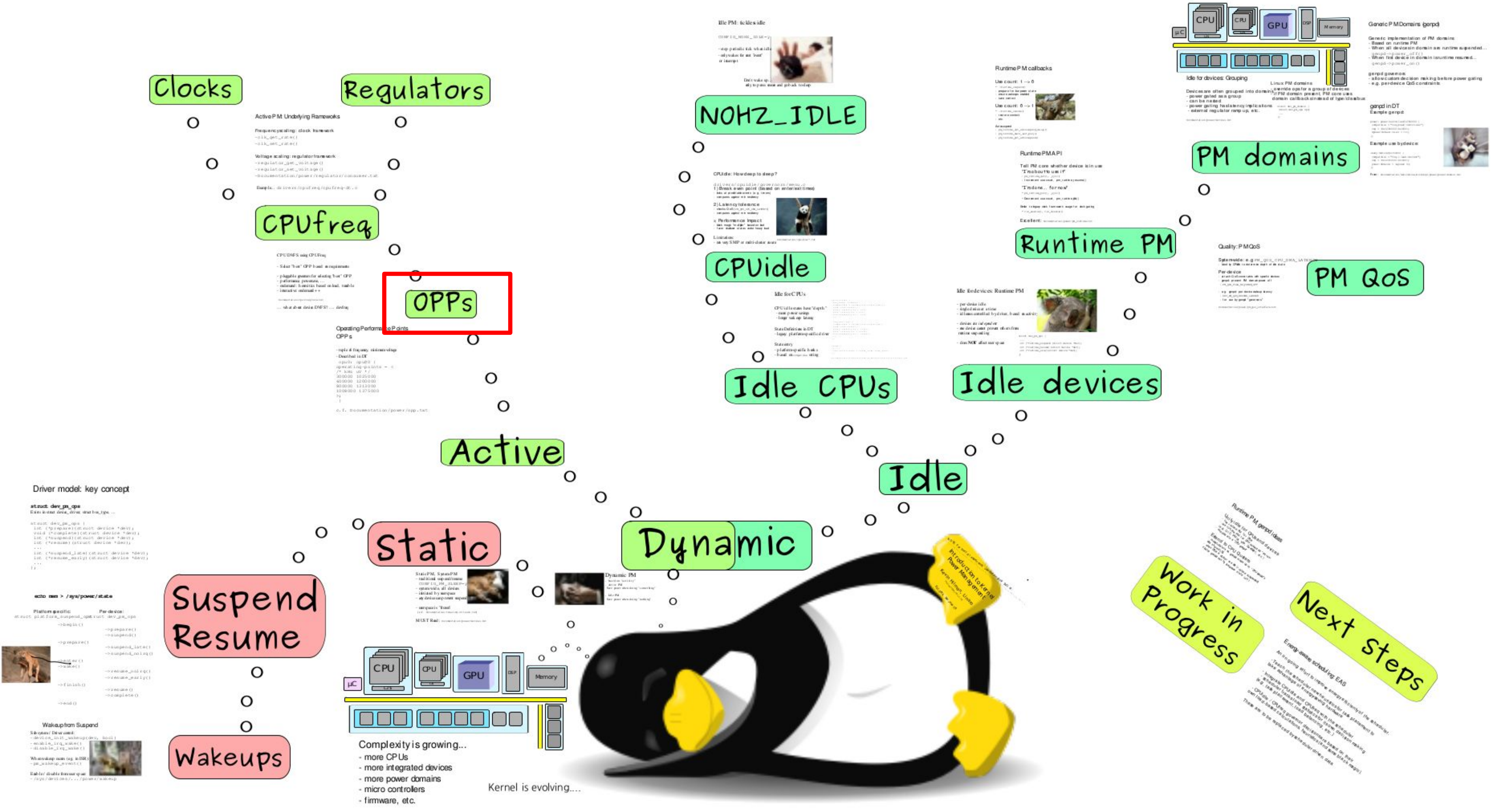
- Regulators have power constraints to protect hardware
- Sysfs interface (/sys/class/regulator/.../)
- Regulator independent abstraction (drivers/regulators/core.c)
- **regulator_get** - lookup and obtain a reference to a regulator
- **regulator_put** - "free" the regulator source
- **regulator_enable** - enable regulator output
- **regulator_disable** - disable regulator output

Regulator Framework - 2



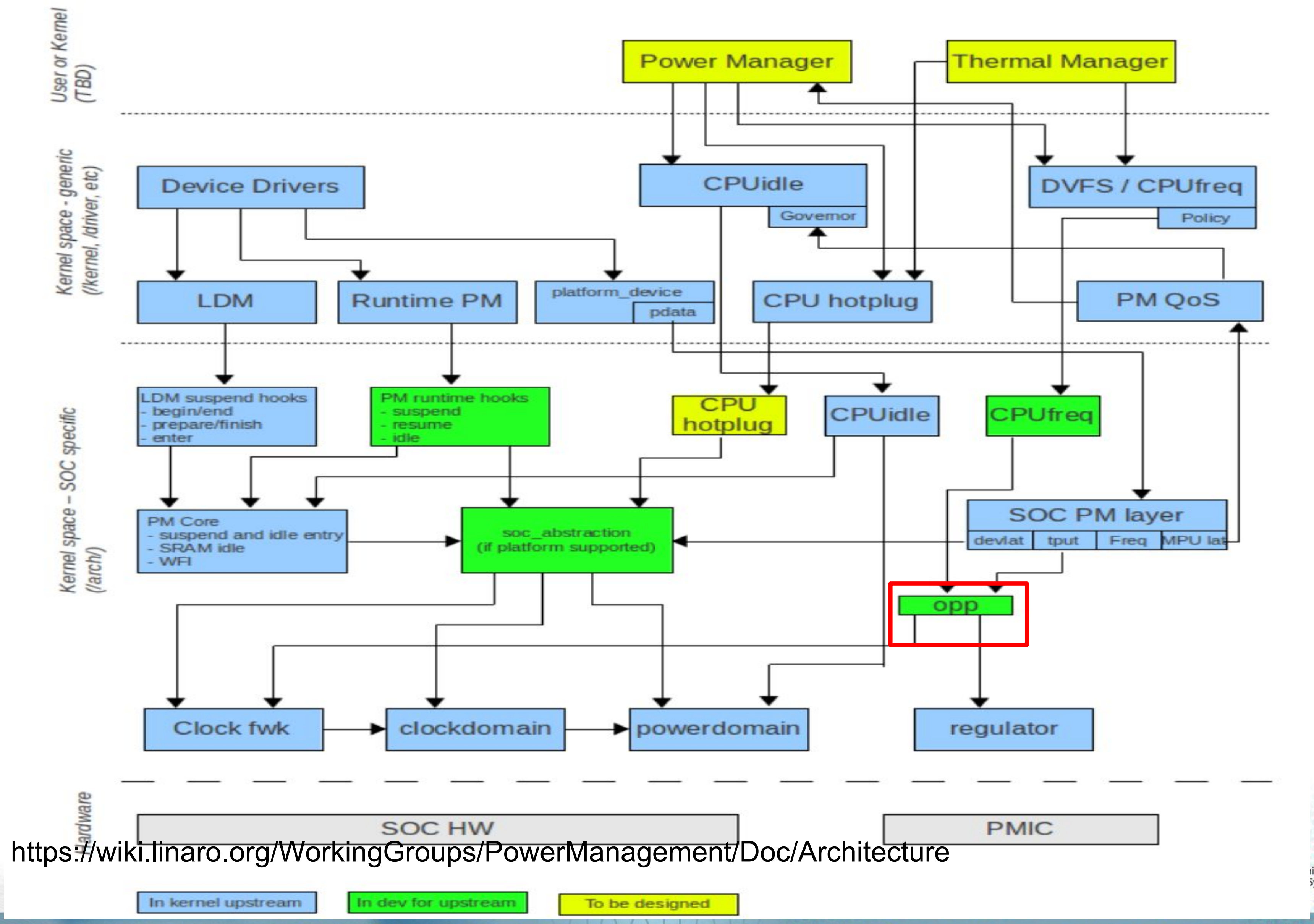
Regulator-1 supplies Domain 1
Regulator-2 supplies Domain 2
Regulator-3 supplies Domain 3

Tree view - Dynamic and Static PM



Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

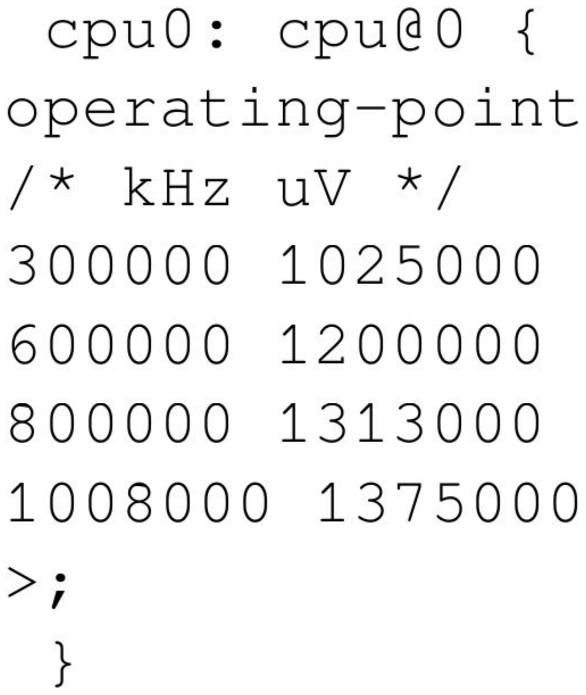
Linux Power Framework



https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

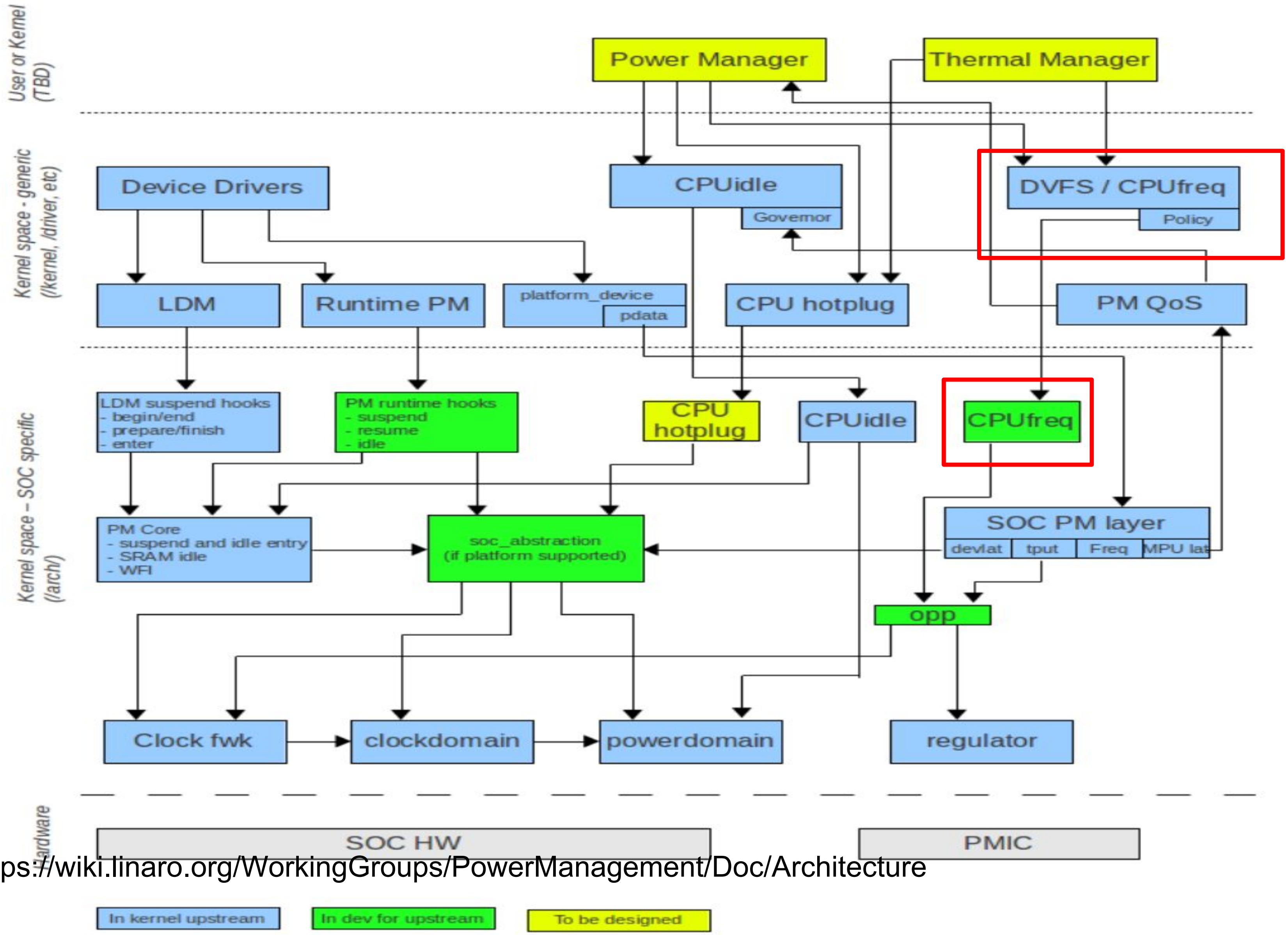
OPP - Operating Performance Point

- Tuple of frequency, minimum voltage.
- Described in DT



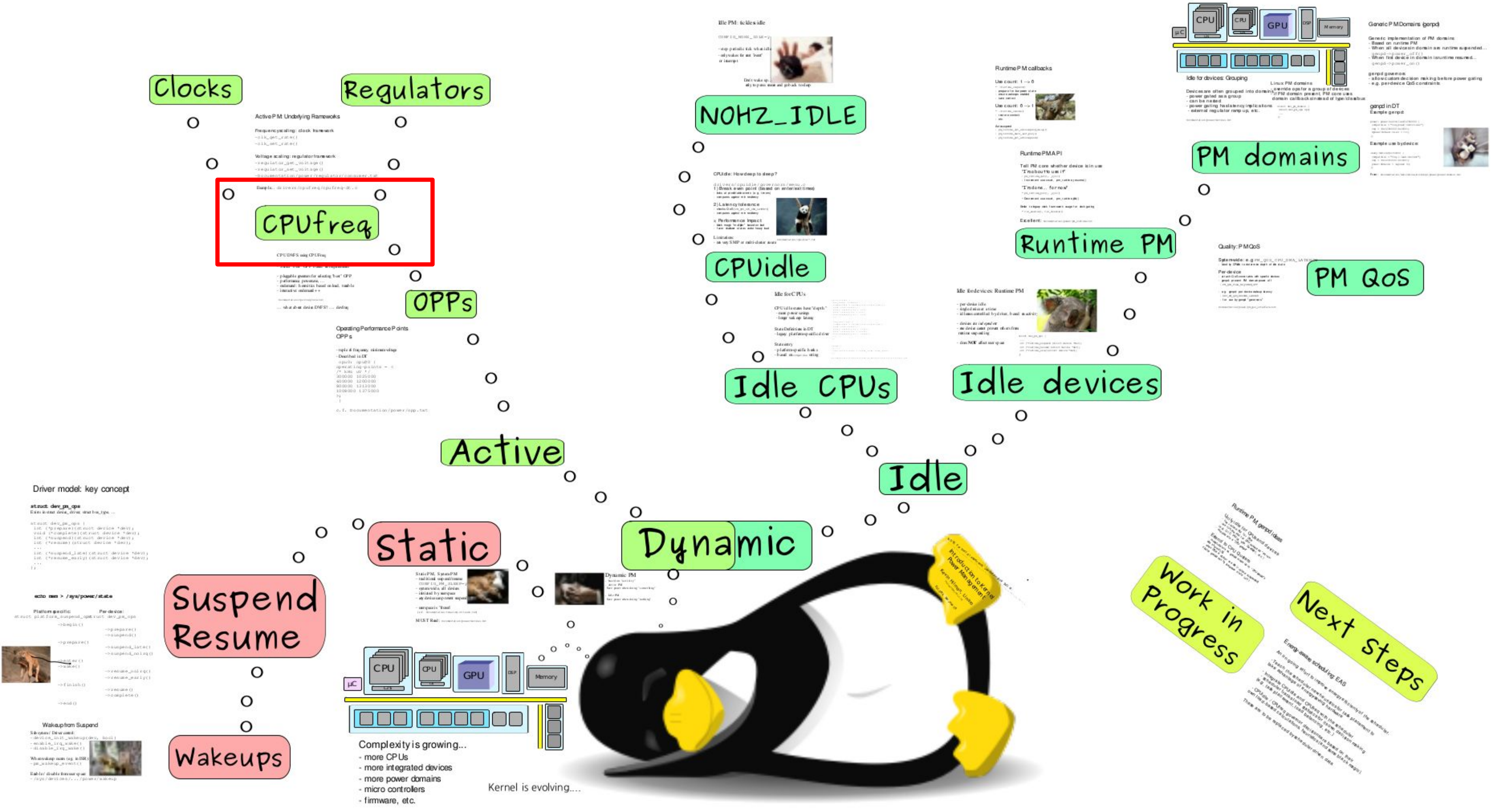
```
 cpu0: cpu@0 {
operating-points = <
/* kHz uV */
300000 1025000
600000 1200000
800000 1313000
1008000 1375000
>;
 }
```

Linux Power Framework



https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

Tree view - Dynamic and Static PM



Active PM: Underlying Frameworks

Frequency scaling: clock framework

Voltage scaling: regulator framework

Operating Performance Points
OPPs

Driver model: key concept

Wakeup from Suspend

Complexity is growing...
- more CPUs
- more integrated devices
- more power domains
- micro controllers
- firmware, etc.

Kernel is evolving....

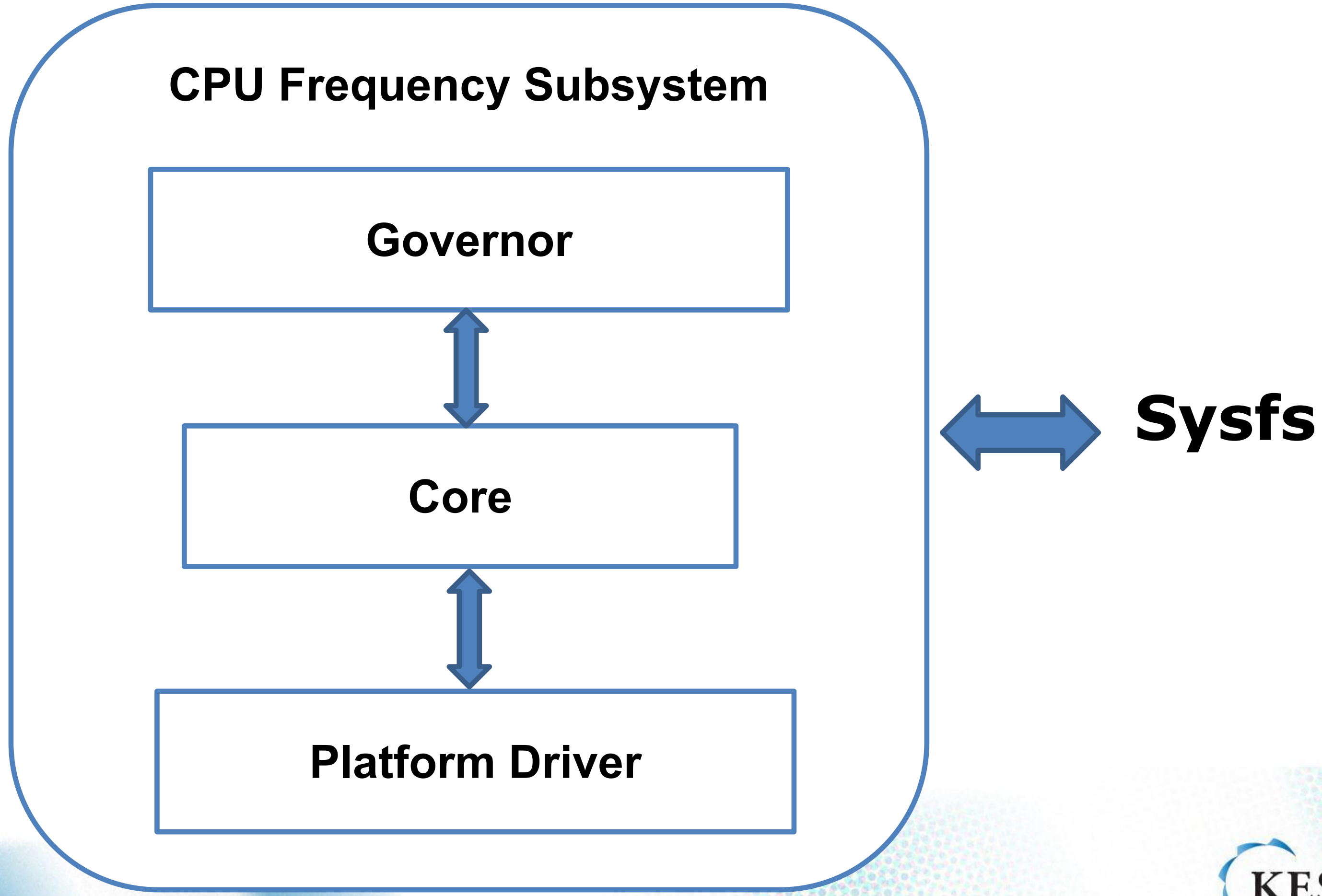
Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

CPU Frequency Governor

- 4.5 and Earlier version of Linux -

CPU Frequency Subsystem

- Linux CPU frequency subsystem manages CPU frequency scaling.



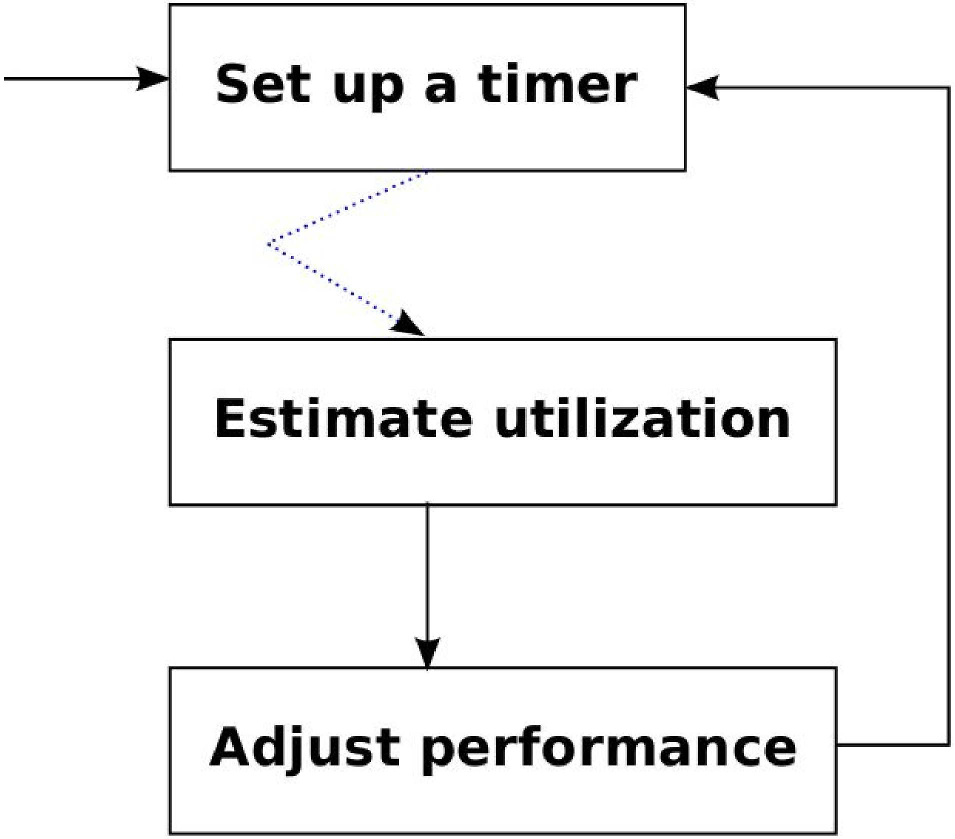
CPU Frequency Subsystem

- **Comprise three parts**
  - Governor
    - Implemented CPU frequency scaling policy
    - Governors in Linux Kernel
      - Performance, Powersave, Userspace, Ondemand, Conservative, Interactive
  - Core
    - Provide generic interface for governor and platform driver to communicate with each other
  - Platform driver
    - SoC-chip specific cpu frequency scaling driver
      - Thermal throttle, DVFS, and so on

Governors In Linux Kernel

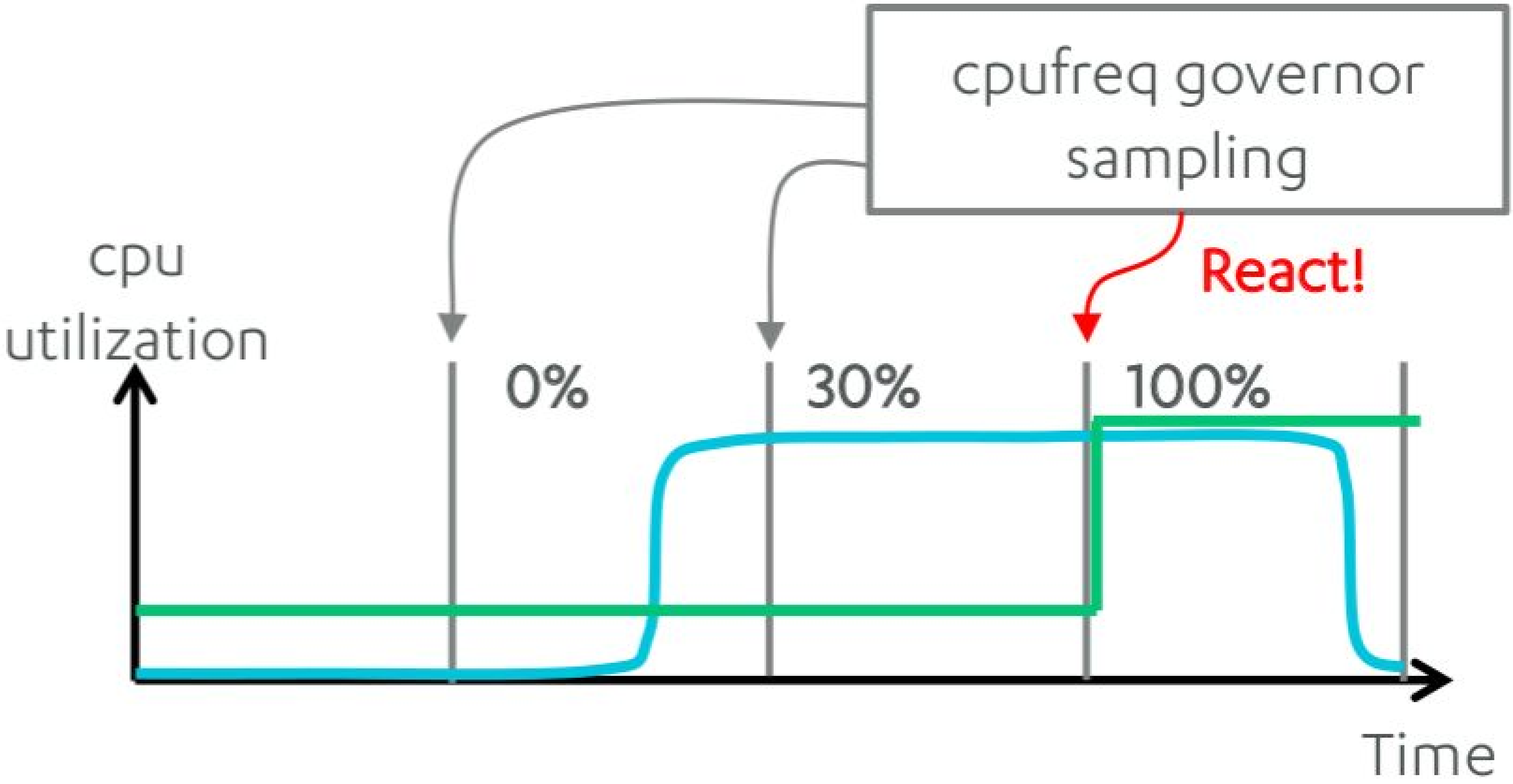
- **Powersave governor**
  - Set frequency statically to lowest available frequency.
- **Performance governor**
  - Set frequency statically to highest available frequency.
- **Userspace governor**
  - Set frequency manually.
- **Ondemand governor**
  - Scale frequency based on CPU utilization.
- **Interactive governor**
  - More responsive to interactive workloads.

Governors In Linux Kernel



"CPUFreq and The Scheduler Revolution in CPU Power Management", Rafael J. Wysocki

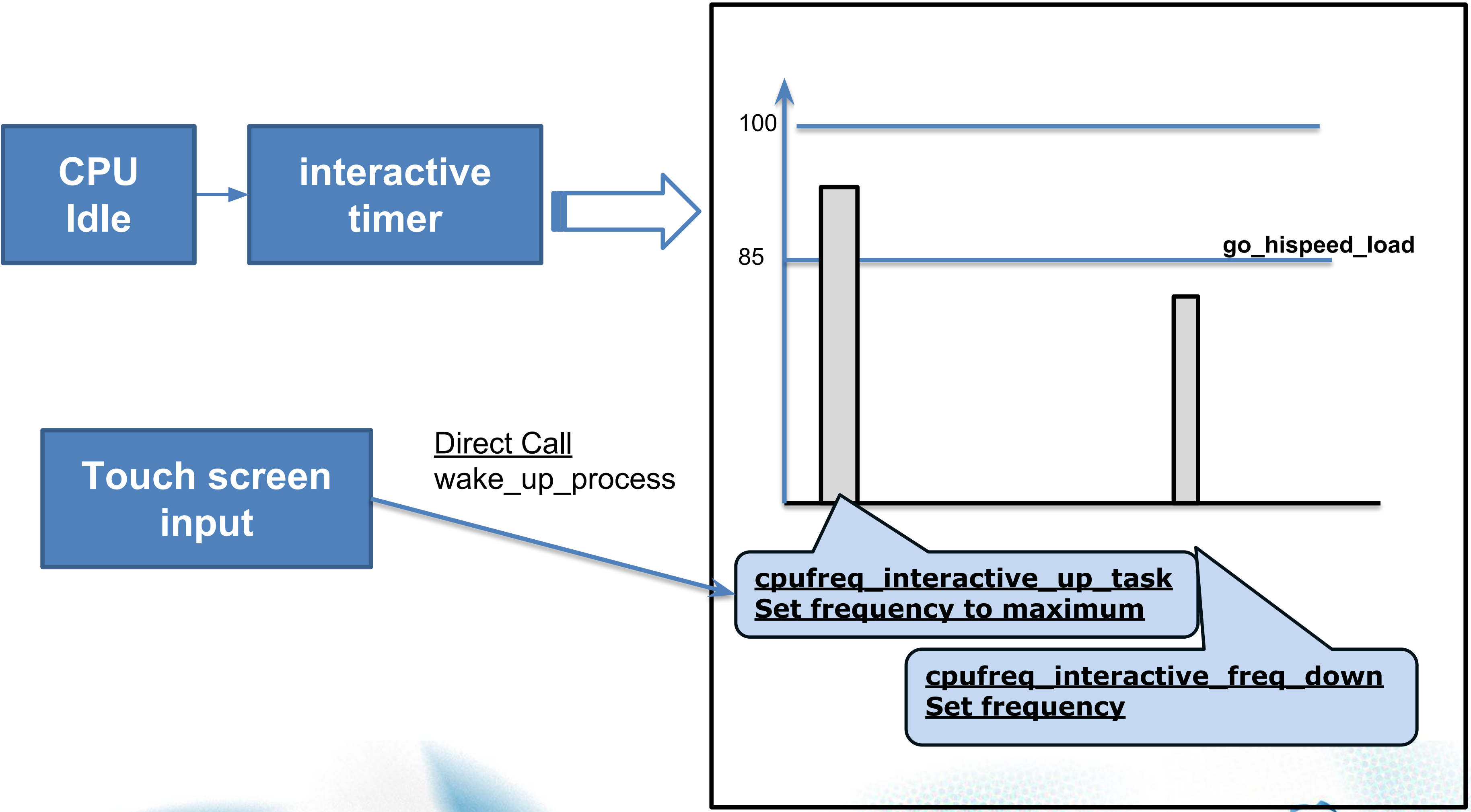
DVFS in Linux (cpufreq)



"EAS Update", Amit Kucheria et. ,al. ARM

Ex) Interactive Governor

- **Interactive**



CPU Idle

interactive timer

Touch screen input

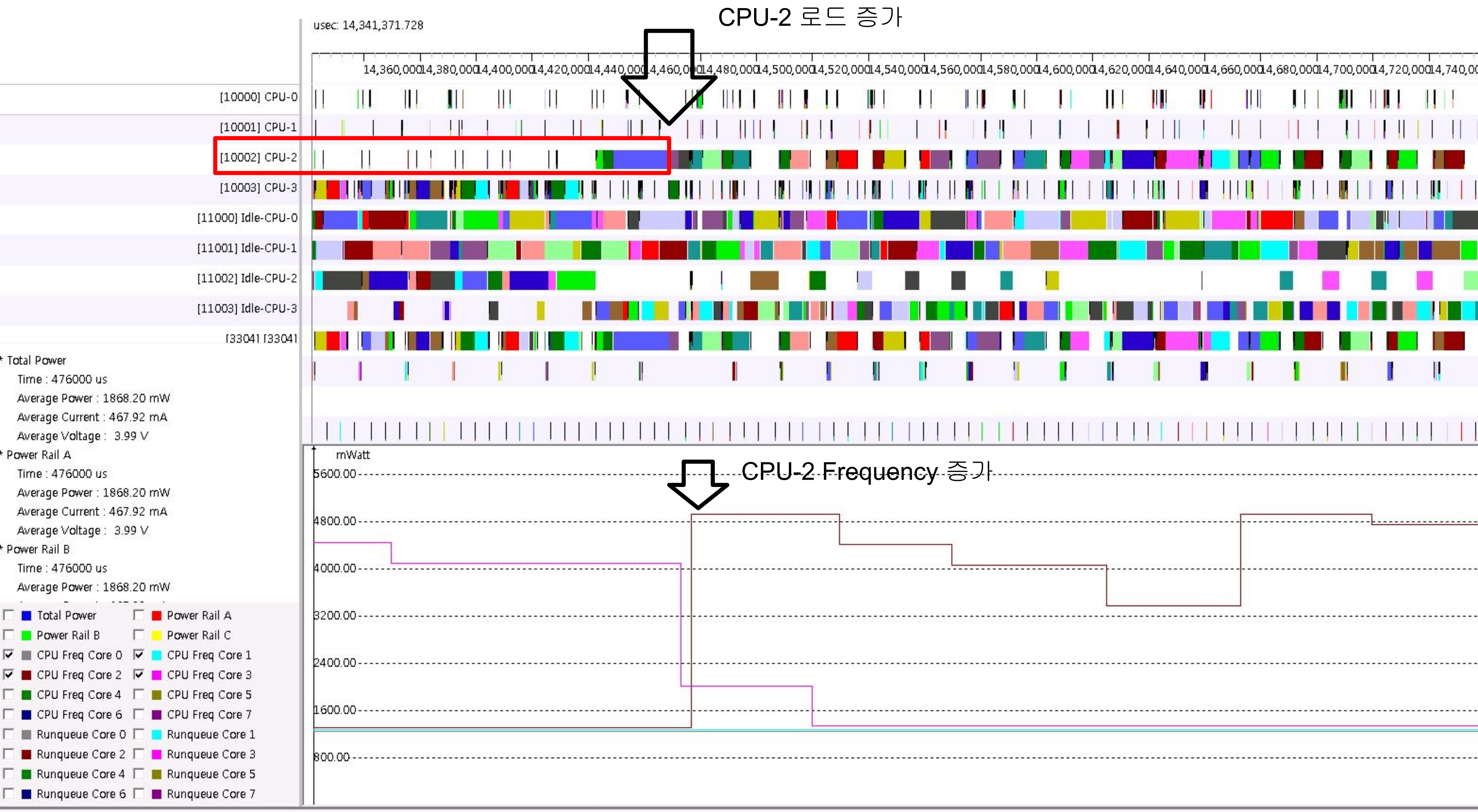
Direct Call
wake_up_process

cpufreq_interactive_up_task
Set frequency to maximum

cpufreq_interactive_freq_down
Set frequency

Interactive

Interface



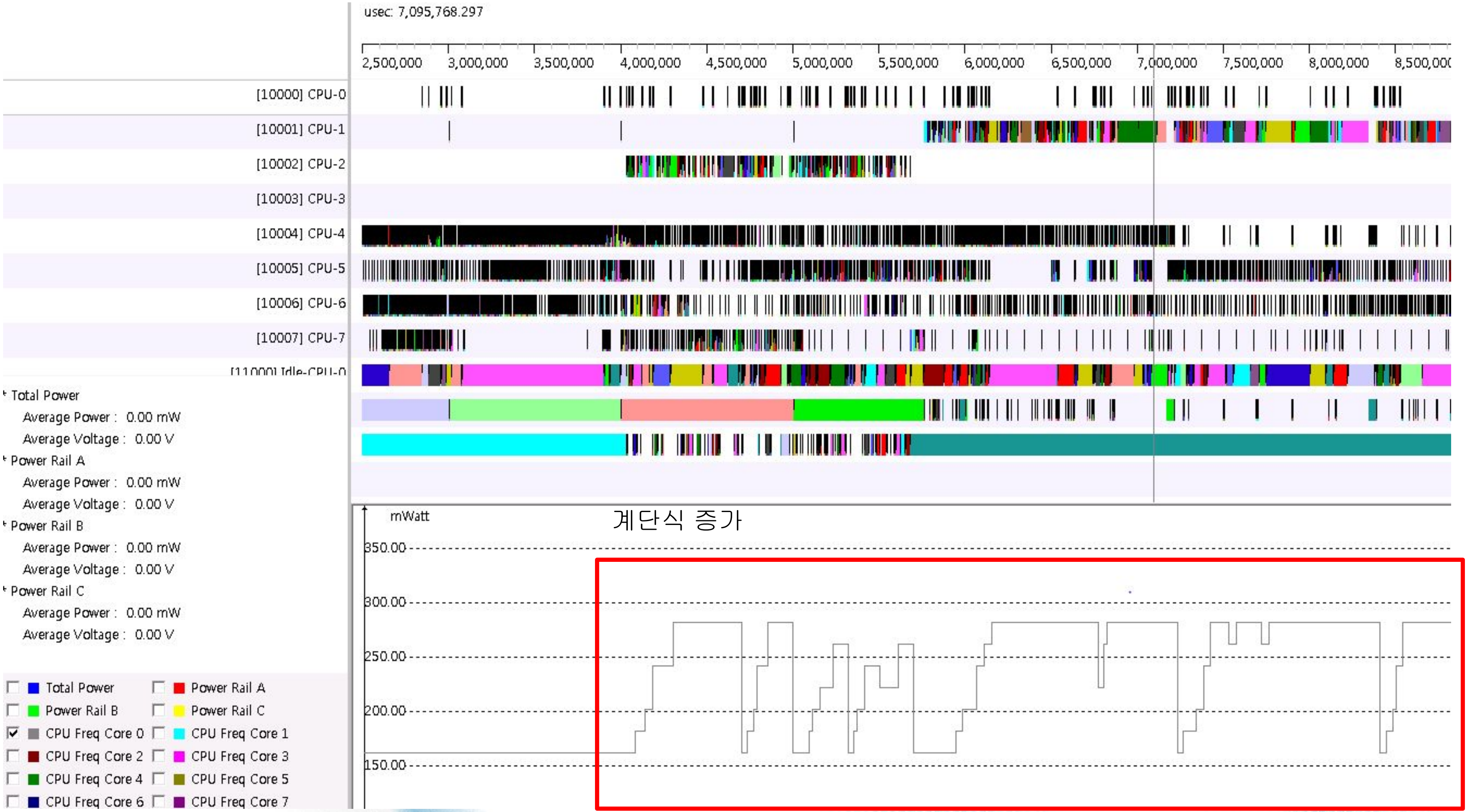
CPU-2 로드 증가

CPU-2 Frequency 증가

```
* Total Power
    Time : 476000 us
    Average Power : 1868.20 mW
    Average Current : 467.92 mA
    Average Voltage :  3.99 V
* Power Rail A
    Time : 476000 us
    Average Power : 1868.20 mW
    Average Current : 467.92 mA
    Average Voltage :  3.99 V
* Power Rail B
    Time : 476000 us
    Average Power : 1868.20 mW
```

☐ 🟦 Total Power	☐ 🟥 Power Rail A
☐ 🟩 Power Rail B	☐ 🟨 Power Rail C
☑ ■ CPU Freq Core 0	☑ ■ CPU Freq Core 1
☑ ■ CPU Freq Core 2	☑ ■ CPU Freq Core 3
☐ ■ CPU Freq Core 4	☐ ■ CPU Freq Core 5
☐ ■ CPU Freq Core 6	☐ ■ CPU Freq Core 7
☐ ■ Runqueue Core 0	☐ ■ Runqueue Core 1
☐ ■ Runqueue Core 2	☐ ■ Runqueue Core 3
☐ ■ Runqueue Core 4	☐ ■ Runqueue Core 5
☐ ■ Runqueue Core 6	☐ ■ Runqueue Core 7

Boost



Total Power
 Average Power : 0.00 mW
 Average Voltage : 0.00 V

Power Rail A
 Average Power : 0.00 mW
 Average Voltage : 0.00 V

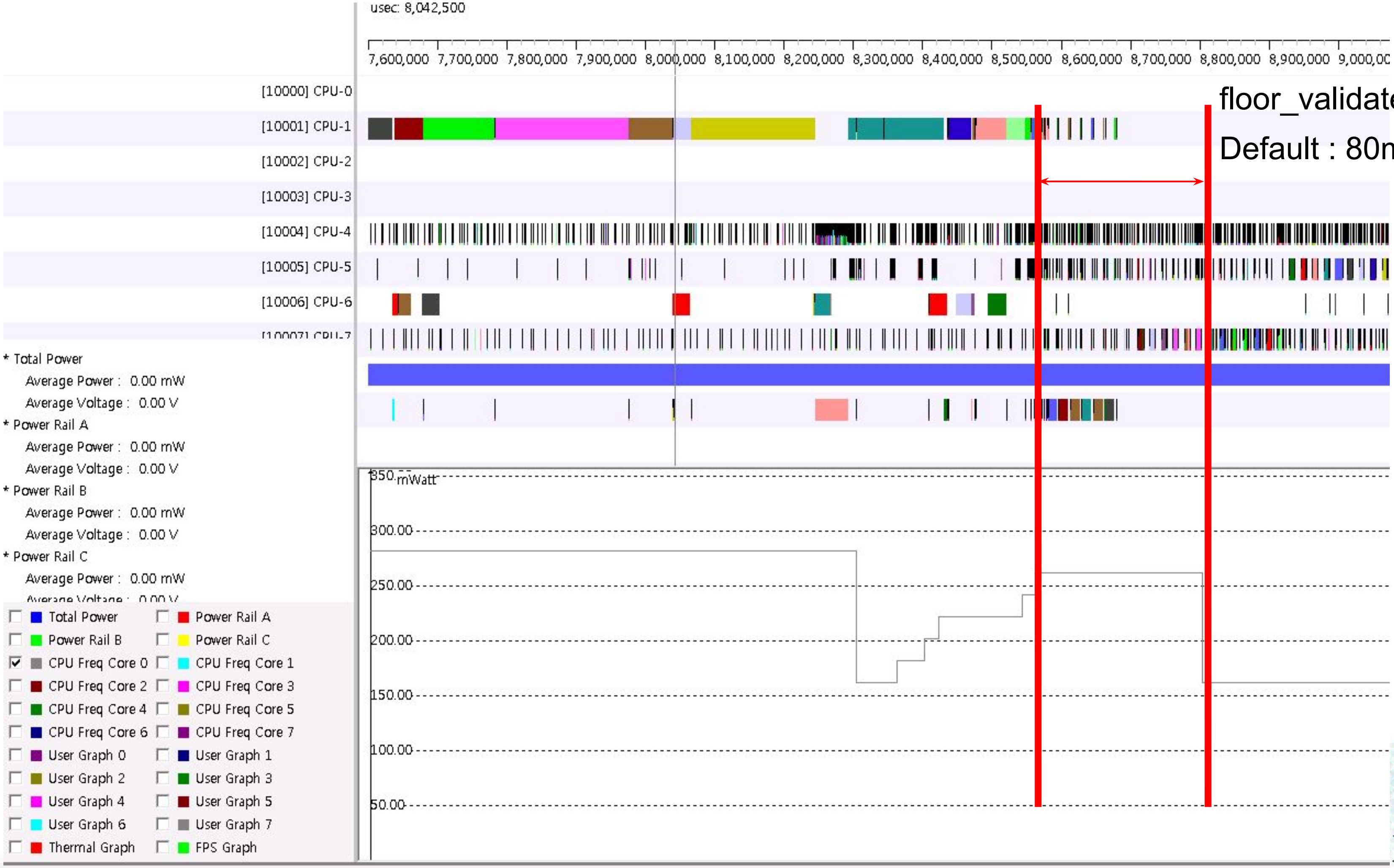
Power Rail B
 Average Power : 0.00 mW
 Average Voltage : 0.00 V

Power Rail C
 Average Power : 0.00 mW
 Average Voltage : 0.00 V

☐ 🟦 Total Power ☐ 🟥 Power Rail A
☐ 🟩 Power Rail B ☐ 🟨 Power Rail C
☑ ■ CPU Freq Core 0 ☐ ■ CPU Freq Core 1
☐ ■ CPU Freq Core 2 ☐ ■ CPU Freq Core 3
☐ ■ CPU Freq Core 4 ☐ ■ CPU Freq Core 5
☐ ■ CPU Freq Core 6 ☐ ■ CPU Freq Core 7

계단식 증가

Floor validate_time



* Total Power

Average Power : 0.00 mW

Average Voltage : 0.00 V

* Power Rail A

Average Power : 0.00 mW

Average Voltage : 0.00 V

* Power Rail B

Average Power : 0.00 mW

Average Voltage : 0.00 V

* Power Rail C

Average Power : 0.00 mW

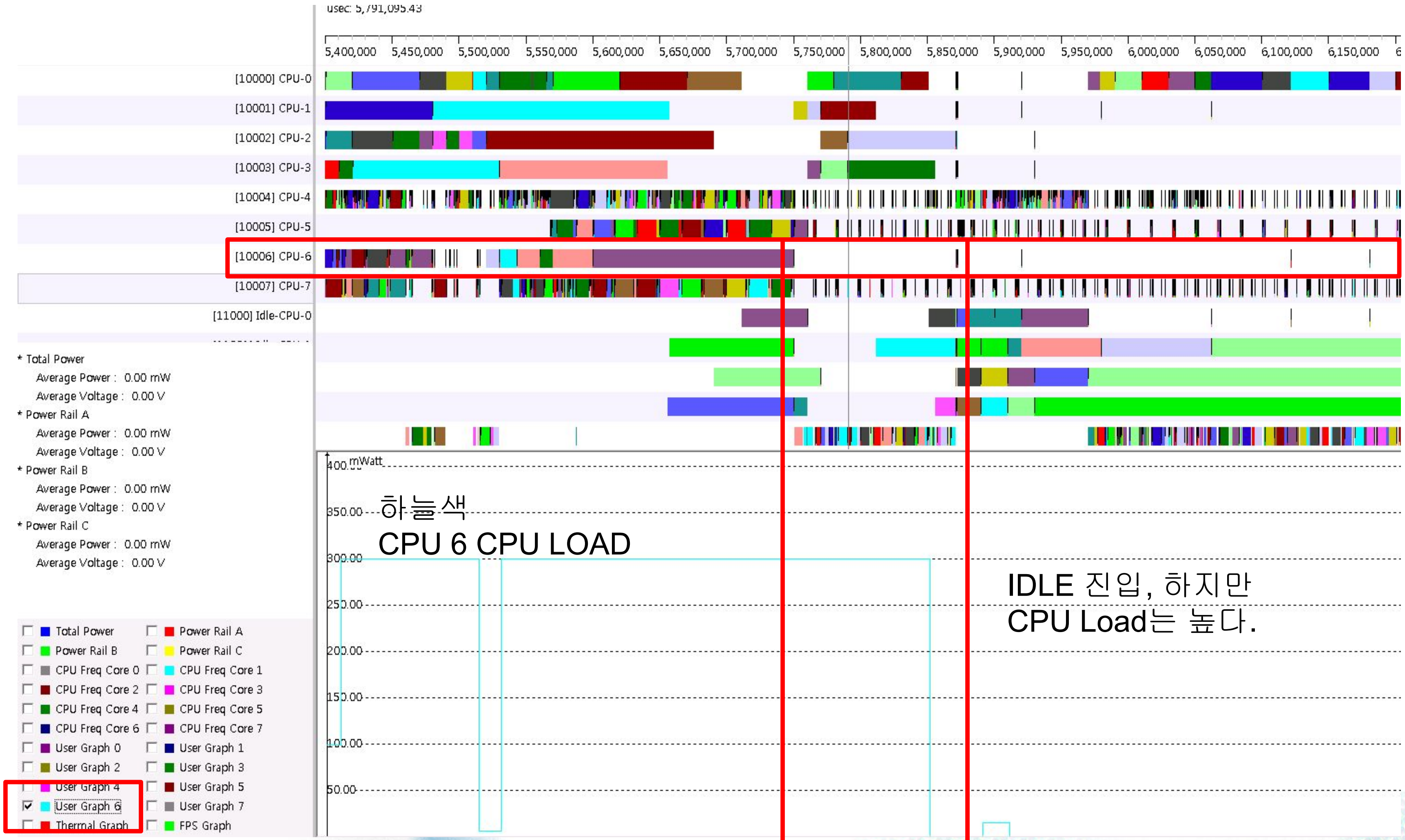
Average Voltage : 0.00 V

☐ 🟦 Total Power	☐ 🟥 Power Rail A
☐ 🟩 Power Rail B	☐ 🟨 Power Rail C
☑ ■ CPU Freq Core 0	☐ ■ CPU Freq Core 1
☐ ■ CPU Freq Core 2	☐ ■ CPU Freq Core 3
☐ ■ CPU Freq Core 4	☐ ■ CPU Freq Core 5
☐ ■ CPU Freq Core 6	☐ ■ CPU Freq Core 7
☐ ■ User Graph 0	☐ ■ User Graph 1
☐ ■ User Graph 2	☐ ■ User Graph 3
☐ ■ User Graph 4	☐ ■ User Graph 5
☐ ■ User Graph 6	☐ ■ User Graph 7
☐ 🟥 Thermal Graph	☐ 🟩 FPS Graph

floor_validate_time

Default : 80ms

Deferrable timer



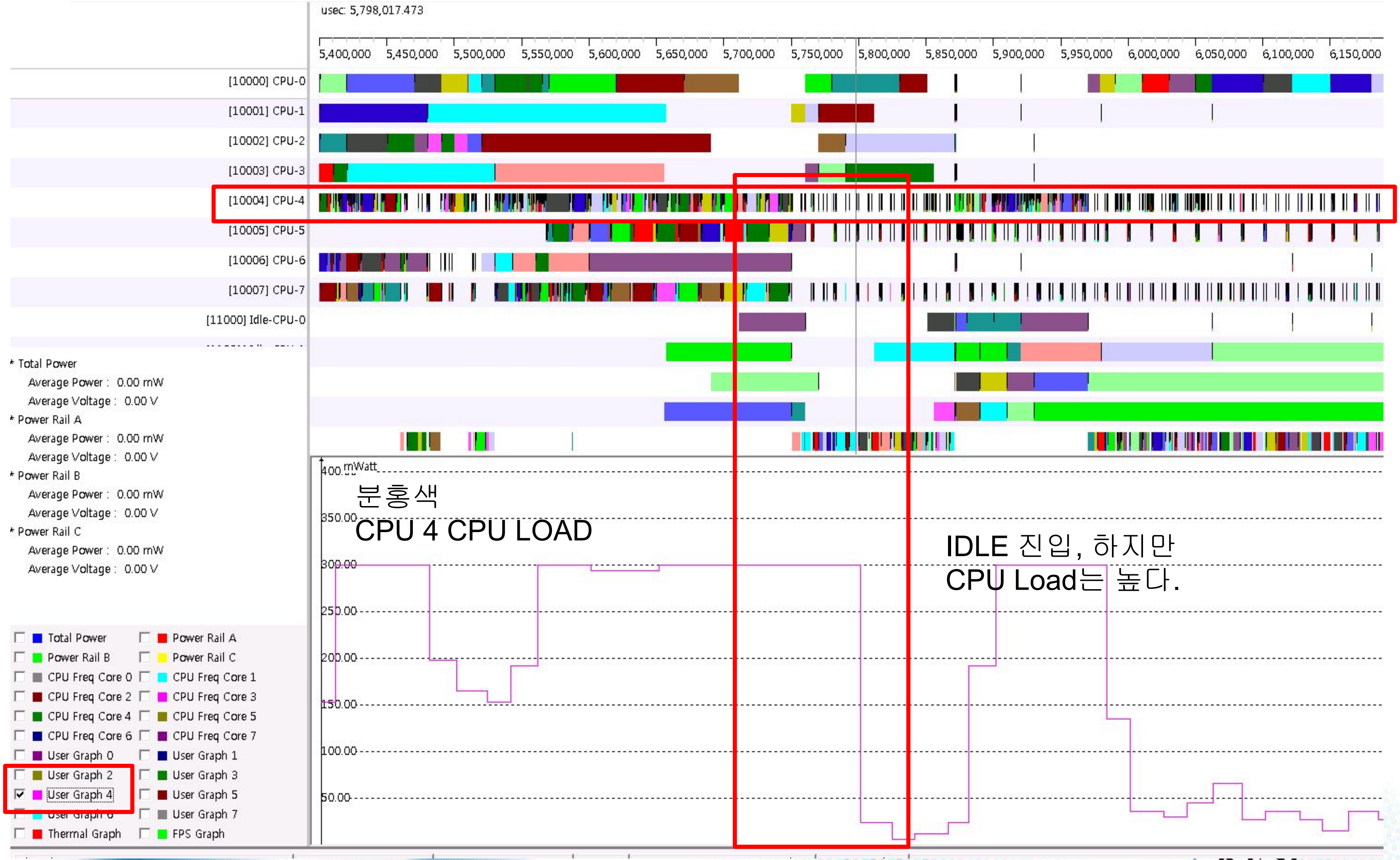
* Total Power
  Average Power : 0.00 mW
  Average Voltage : 0.00 V
* Power Rail A
  Average Power : 0.00 mW
  Average Voltage : 0.00 V
* Power Rail B
  Average Power : 0.00 mW
  Average Voltage : 0.00 V
* Power Rail C
  Average Power : 0.00 mW
  Average Voltage : 0.00 V

☐ 🟦 Total Power	☐ 🟥 Power Rail A
☐ 🟩 Power Rail B	☐ 🟨 Power Rail C
☐ ■ CPU Freq Core 0	☐ ■ CPU Freq Core 1
☐ ■ CPU Freq Core 2	☐ ■ CPU Freq Core 3
☐ ■ CPU Freq Core 4	☐ ■ CPU Freq Core 5
☐ ■ CPU Freq Core 6	☐ ■ CPU Freq Core 7
☐ ■ User Graph 0	☐ ■ User Graph 1
☐ ■ User Graph 2	☐ ■ User Graph 3
☐ ■ User Graph 4	☐ ■ User Graph 5
☑ ■ User Graph 6	☐ ■ User Graph 7
☐ 🟥 Thermal Graph	☐ 🟩 FPS Graph

하늘색
CPU 6 CPU LOAD

IDLE 진입, 하지만
CPU Load는 높다.

Deferrable timer



* Total Power

Average Power : 0.00 mW

Average Voltage : 0.00 V

* Power Rail A

Average Power : 0.00 mW

Average Voltage : 0.00 V

* Power Rail B

Average Power : 0.00 mW

Average Voltage : 0.00 V

* Power Rail C

Average Power : 0.00 mW

Average Voltage : 0.00 V

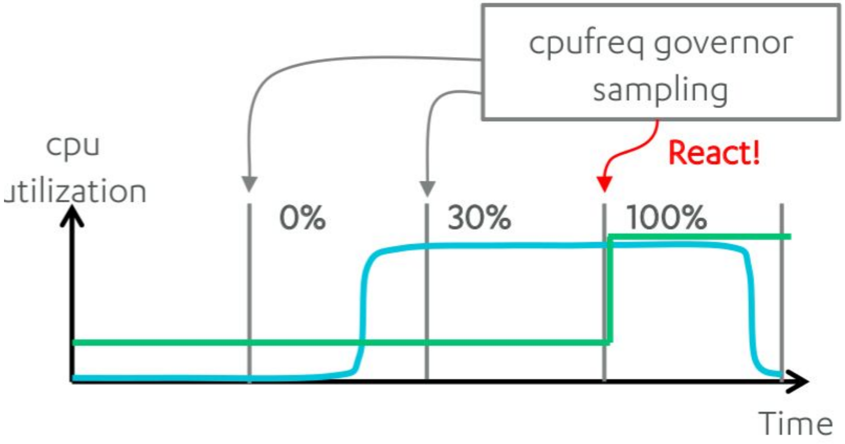
□ ■ Total Power □ ■ Power Rail A
□ ■ Power Rail B □ ■ Power Rail C
□ ■ CPU Freq Core 0 □ ■ CPU Freq Core 1
□ ■ CPU Freq Core 2 □ ■ CPU Freq Core 3
□ ■ CPU Freq Core 4 □ ■ CPU Freq Core 5
□ ■ CPU Freq Core 6 □ ■ CPU Freq Core 7
□ ■ User Graph 0 □ ■ User Graph 1
□ ■ User Graph 2 □ ■ User Graph 3
☑ ■ User Graph 4 □ ■ User Graph 5
□ ■ User Graph 6 □ ■ User Graph 7
□ ■ Thermal Graph □ ■ FPS Graph

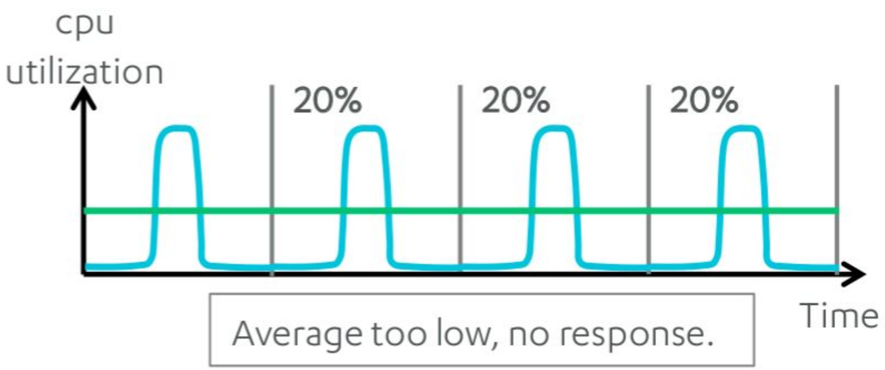
분홍색
CPU 4 CPU LOAD

IDLE 진입, 하지만
CPU Load는 높다.

DVFS in Linux (cpufreq)

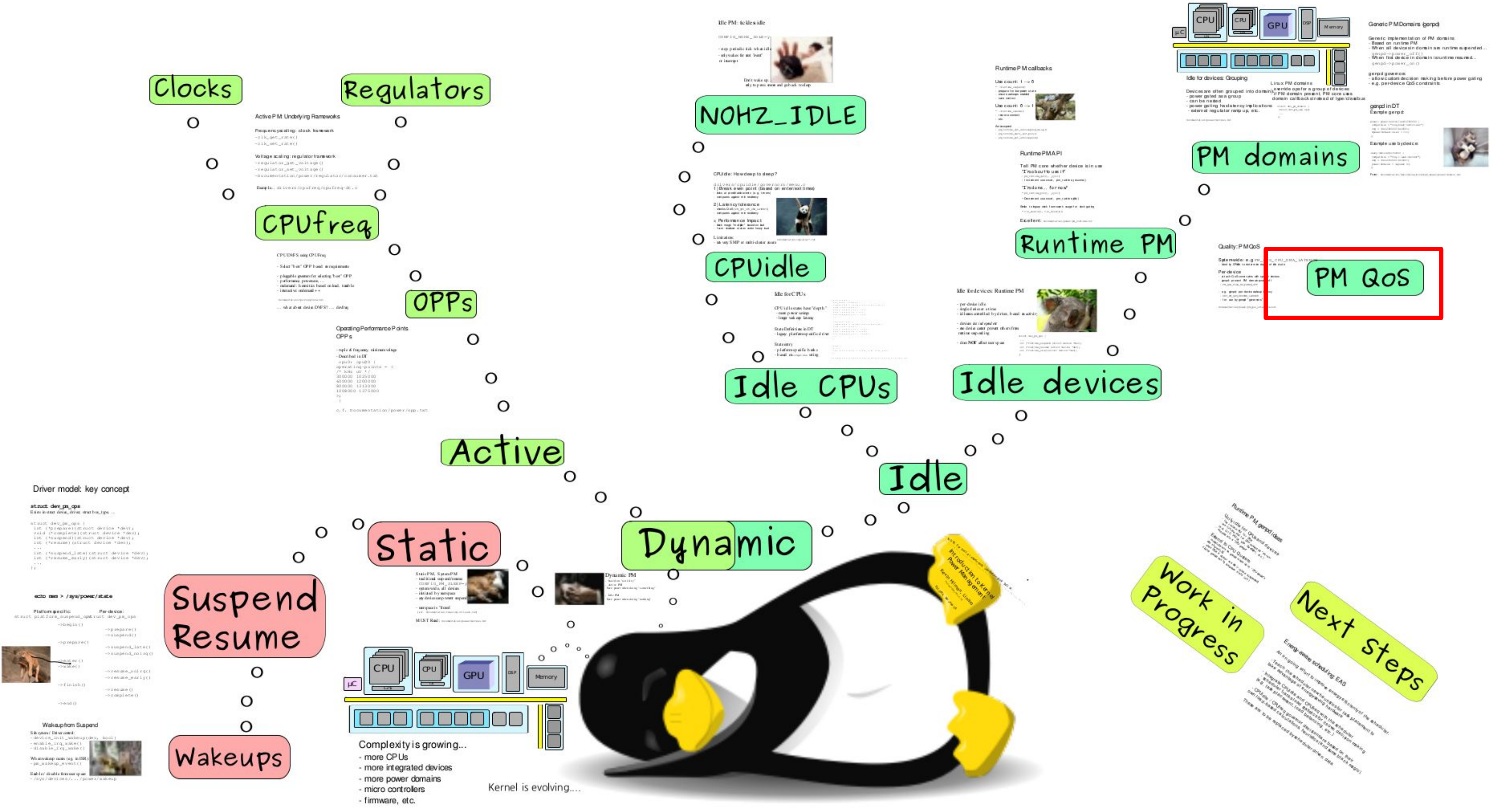
- **Problem**
  - Sampling based governors are slow to respond and hard to tune.
- **Sampling too fast**
  - Freq changes for small utilization spikes.
- **Sampling too slow**
  - Average too low





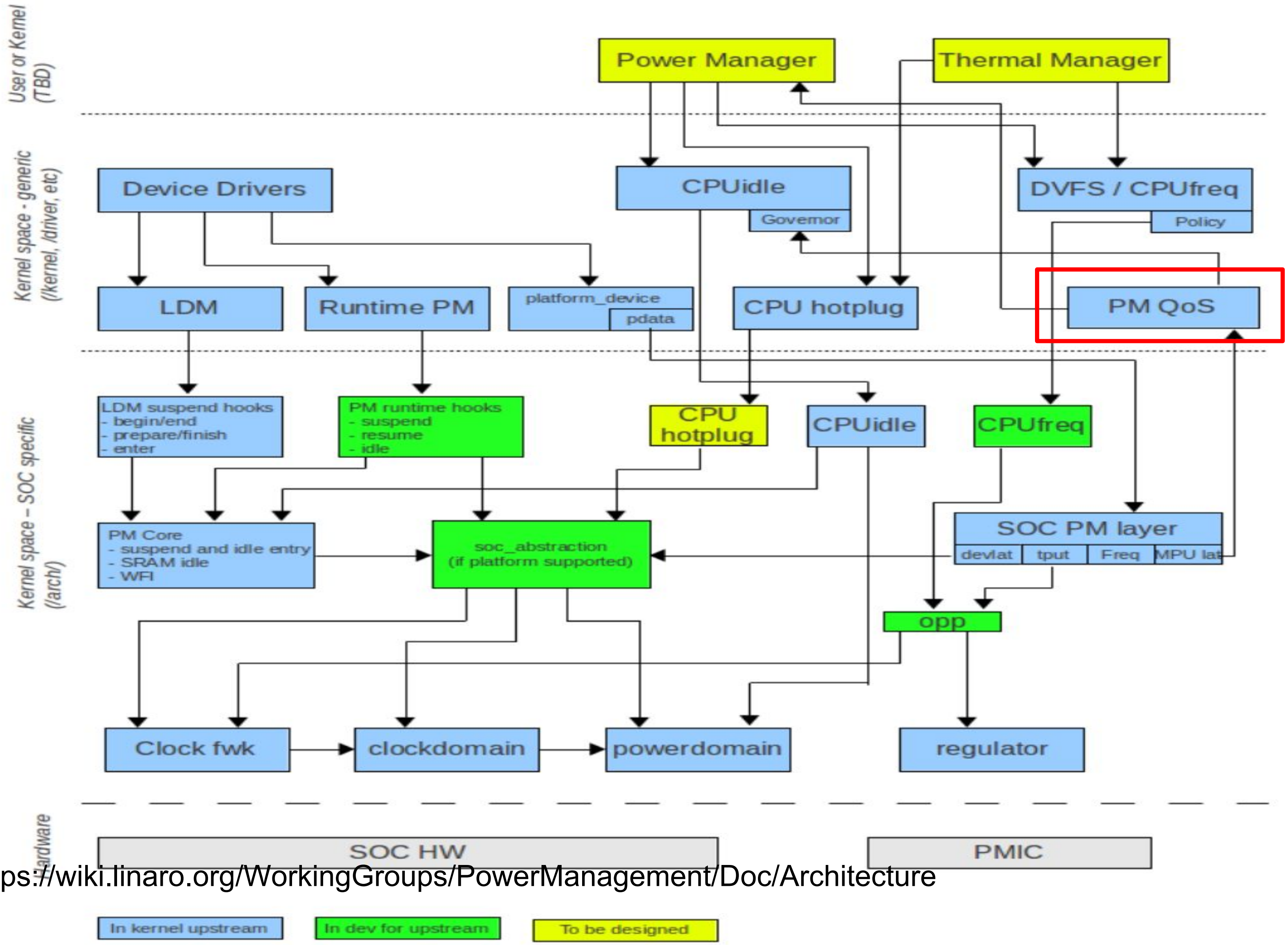
"EAS Update", Amit Kucheria et. ,al. ARM

Tree view - Dynamic and Static PM



Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

Linux Power Framework



https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

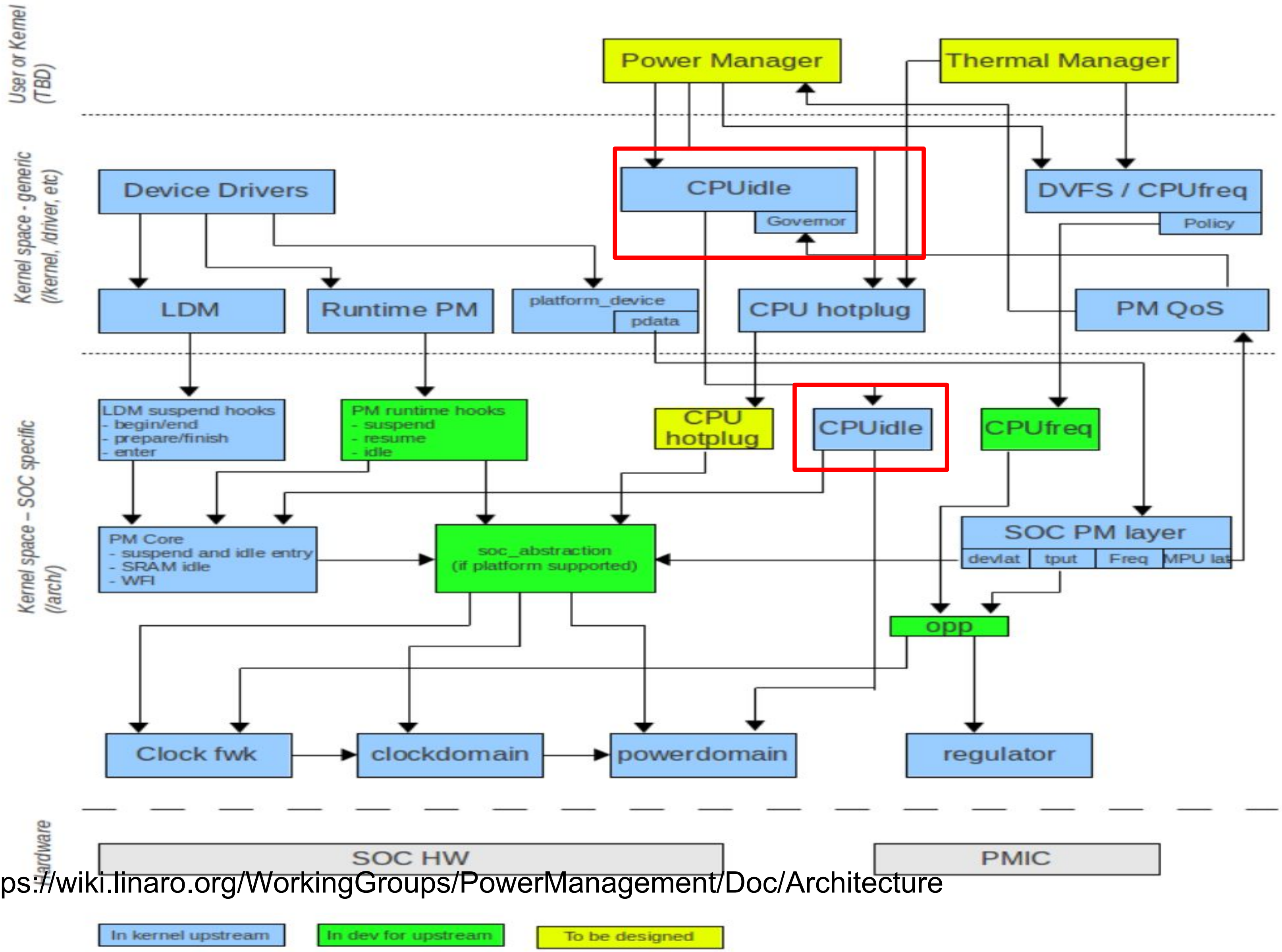
PM_QOS

- **Goal : to reduce energy**

- **A coordination mechanism**
  - User's performance needs

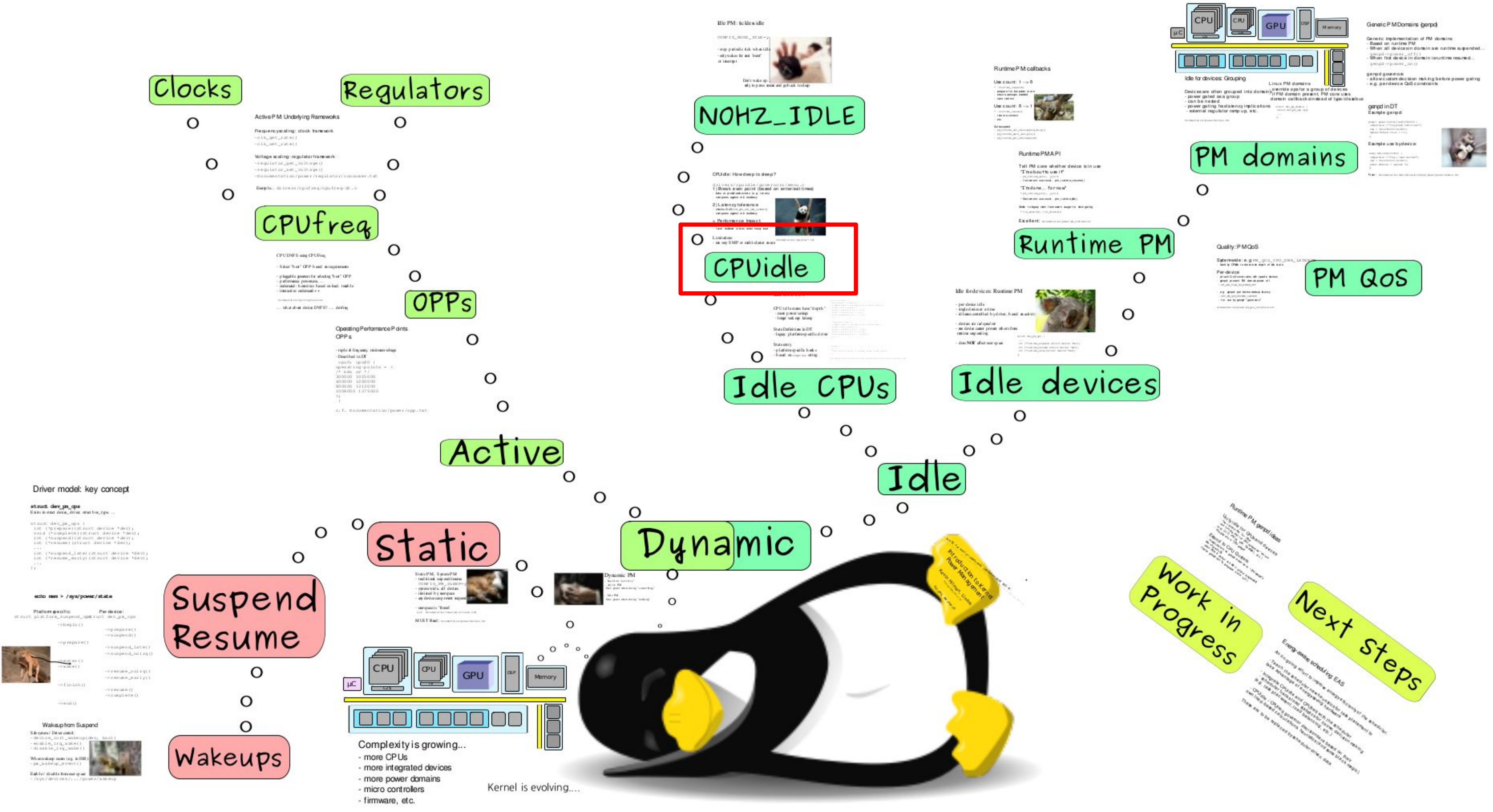
- **Examples**
  - PM QoS with cpuidle
  - Per-device PM QoS with runtime pm.

Linux Power Framework



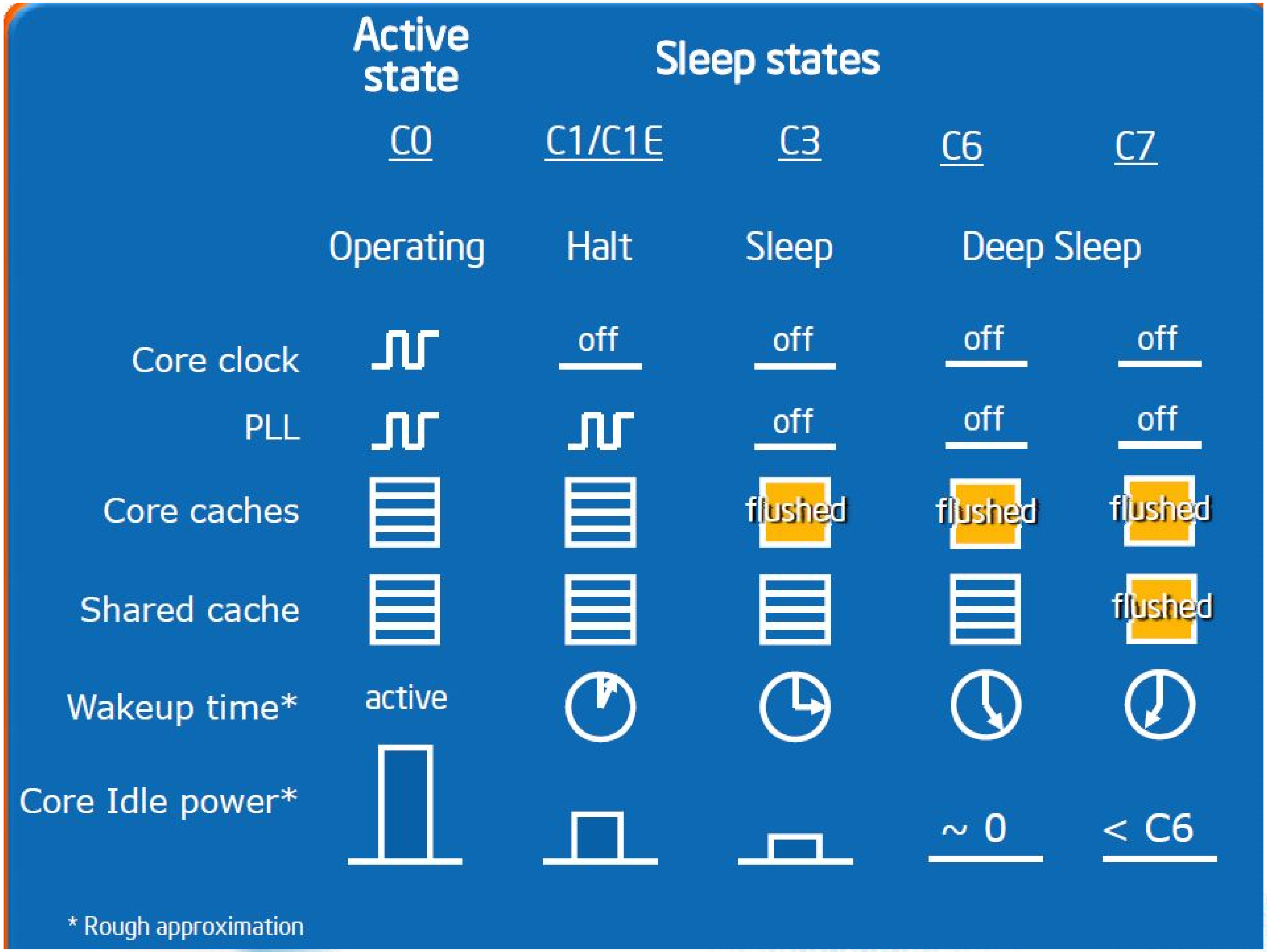
https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

Tree view - Dynamic and Static PM



Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

Processor Power Management

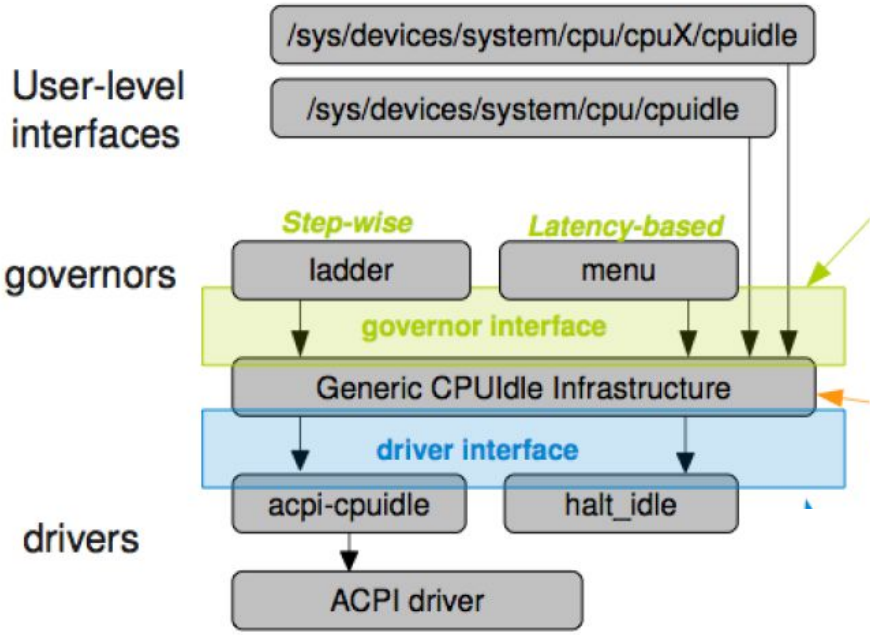


	Active state	Sleep states			
	C0	C1/C1E	C3	C6	C7
	Operating	Halt	Sleep	Deep Sleep	
Core clock	⎍	off	off	off	off
PLL	⎍	⎍	off	off	off
Core caches	☰	☰	flushed	flushed	flushed
Shared cache	☰	☰	☰	☰	flushed
Wakeup time*	active	🕐	🕒	🕔	🕖
Core Idle power*	▮	▬	▁	~ 0	< C6

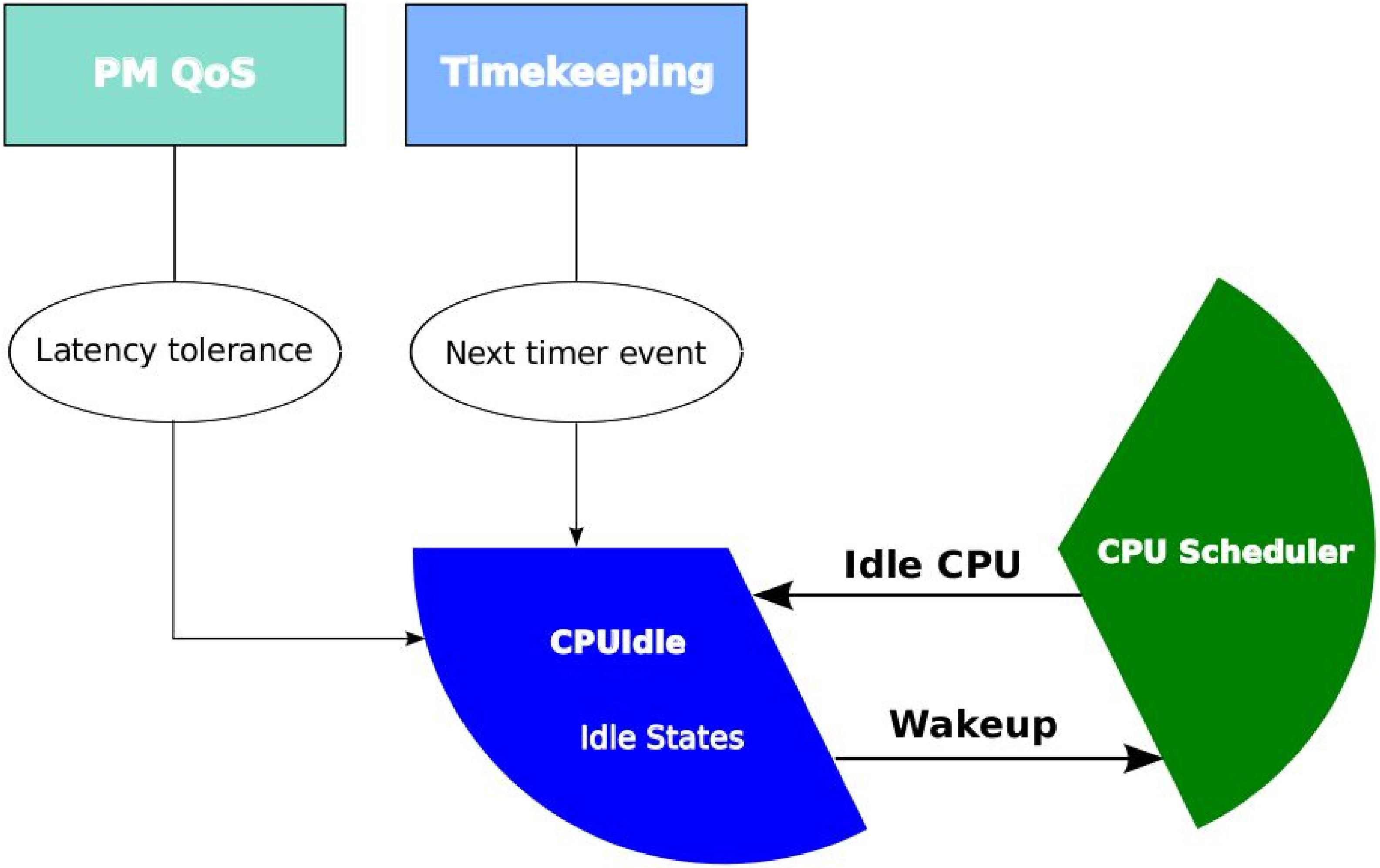
* Rough approximation

http://download.intel.com/embedded/technology/PowerManagementforEmbeddedApps.pdf

CPU idle - C-state management



CPU idle


PM QoS
Timekeeping
Latency tolerance
Next timer event
CPUIdle
Idle States
Idle CPU
Wakeup
CPU Scheduler

Menu governor: How deep to sleep?

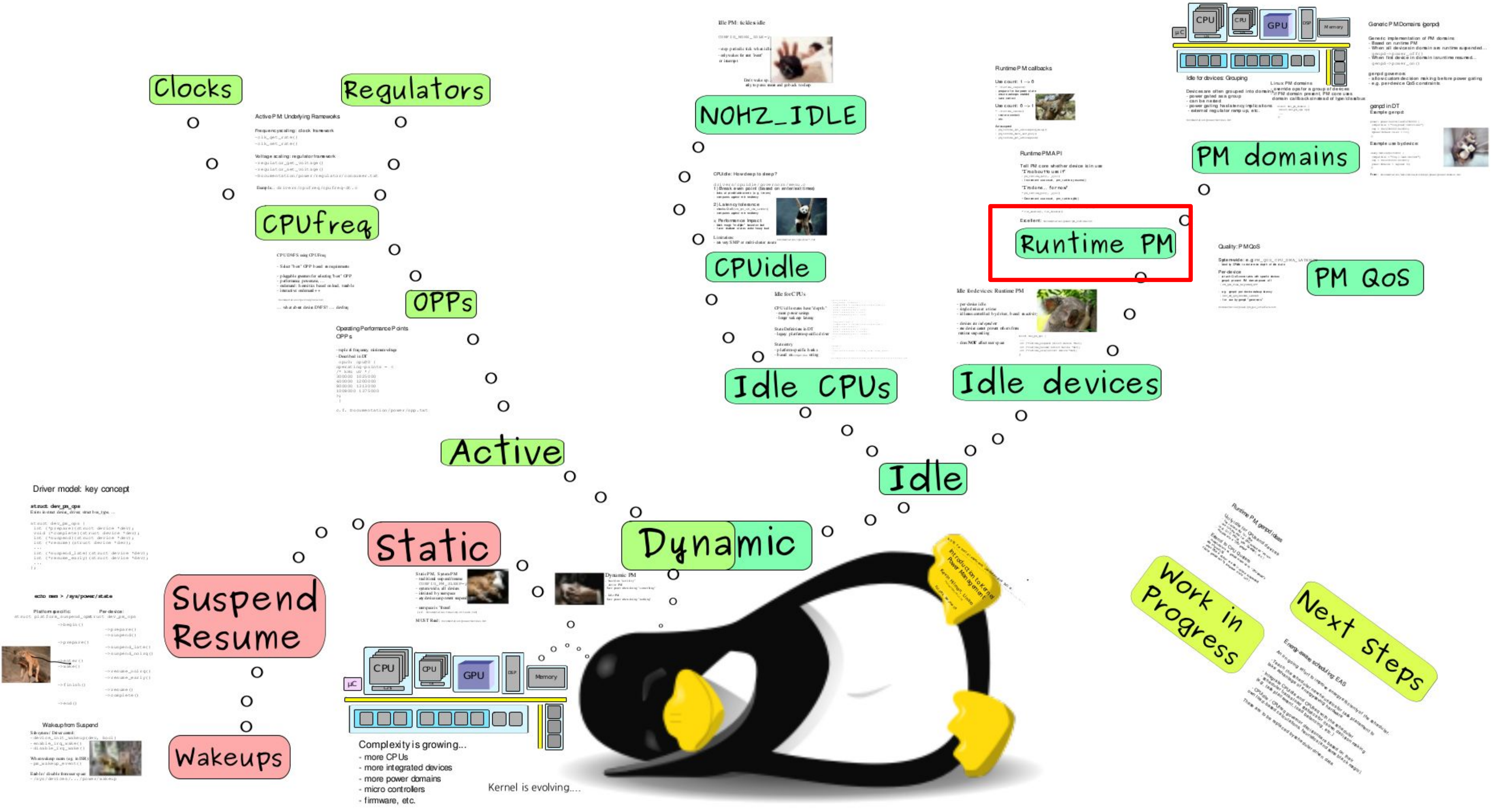
- **Energy break**
  - next timer event
  - historic behavior

- **Performance impact**
  - nr_iowaiters, cpu_load

- **Latency tolerance**
  - checks QoS (PM_QOS_CPU_DMA_LATENCY)

Per-device Power Management

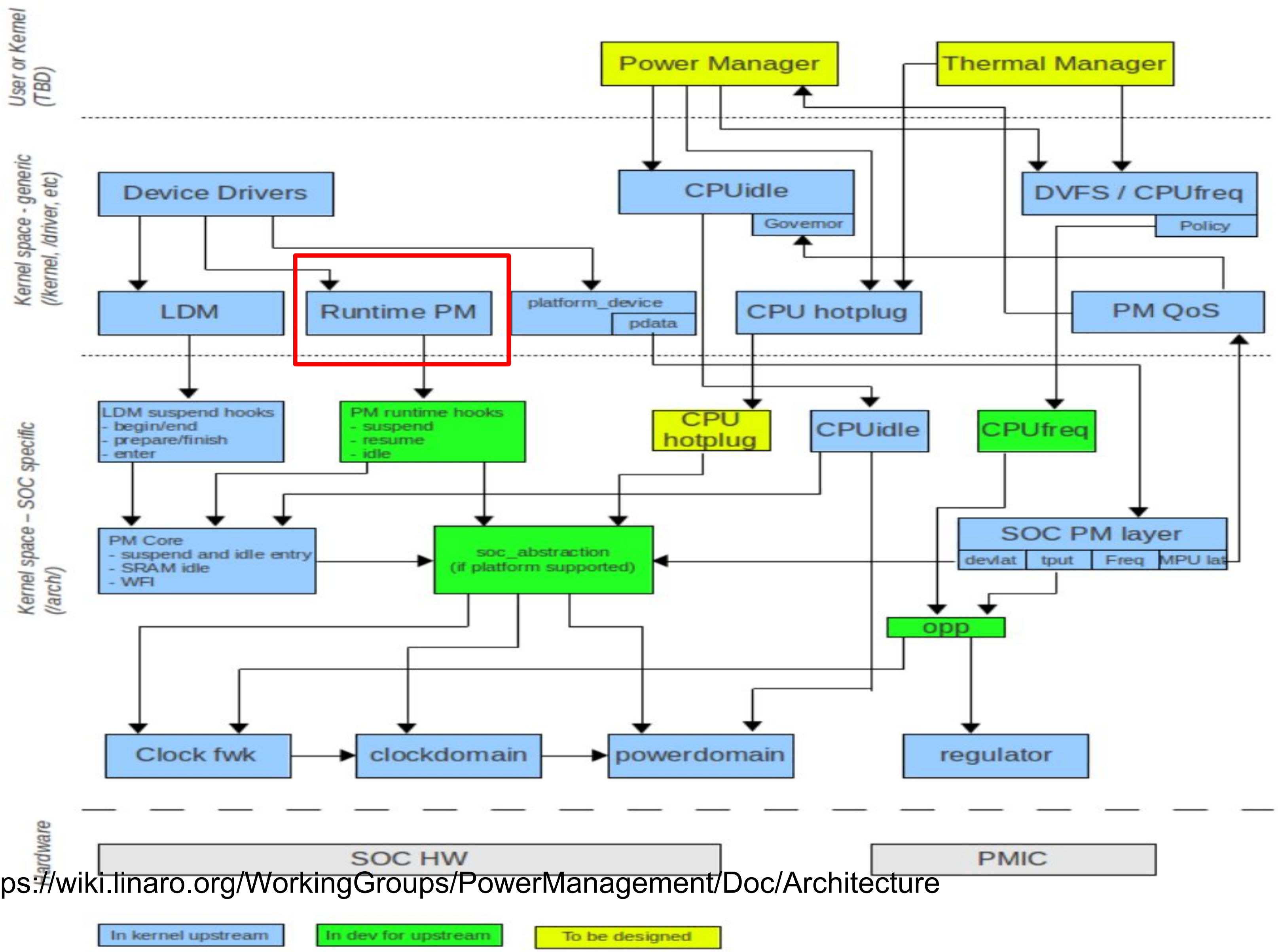
Tree view - Dynamic and Static PM



Complexity is growing...
- more CPUs
- more integrated devices
- more power domains
- micro controllers
- firmware, etc.

Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

Linux Power Framework



https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

Runtime Power Management

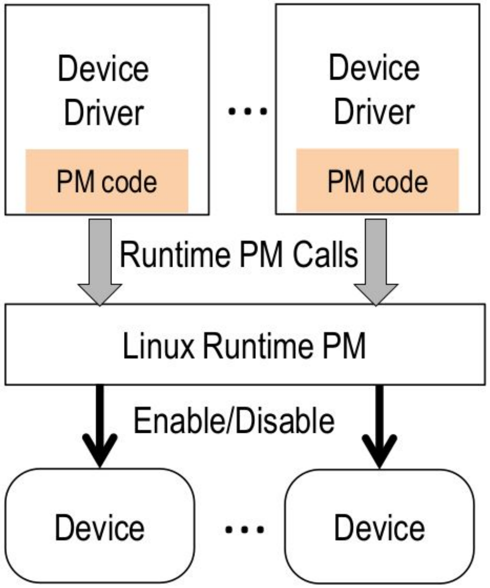
- Why’s Runtime PM
- What’s Runtime PM
- How’s Runtime PM

Problem

- System sleep is not enough to decrease runtime energy consumption
- Devices may depend on another device

Why's Runtime PM

- **System sleep is not enough to decrease runtime energy consumption**
- **Devices may depend on another device**
- **Solution**
  - Runtime PM



Runtime PM callback and Helper Functions

- **ksrc/include/linux/pm.h :**

```
struct dev_pm_ops {
 …
 int (*runtime_suspend)(struct device *dev);
 int (*runtime_resume)(struct device *dev);
 int (*runtime_idle)(struct device *dev);
};
```

- **Ksrc/include/linux/pm_runtime.h**
  - pm_runtime_xxx();

Linux Support for Device Runtime PM

- Track the number of concurrent users of device

- use a per-device reference counter
  - _get(), _put()

Runtime PM API: _get(), _put()

- **Tell PM core whether device is in use**

- **I need the device**
  - pm_runtime_get(), _sync(), _noresume()
  - Increment use count, pm_runtime_resume()

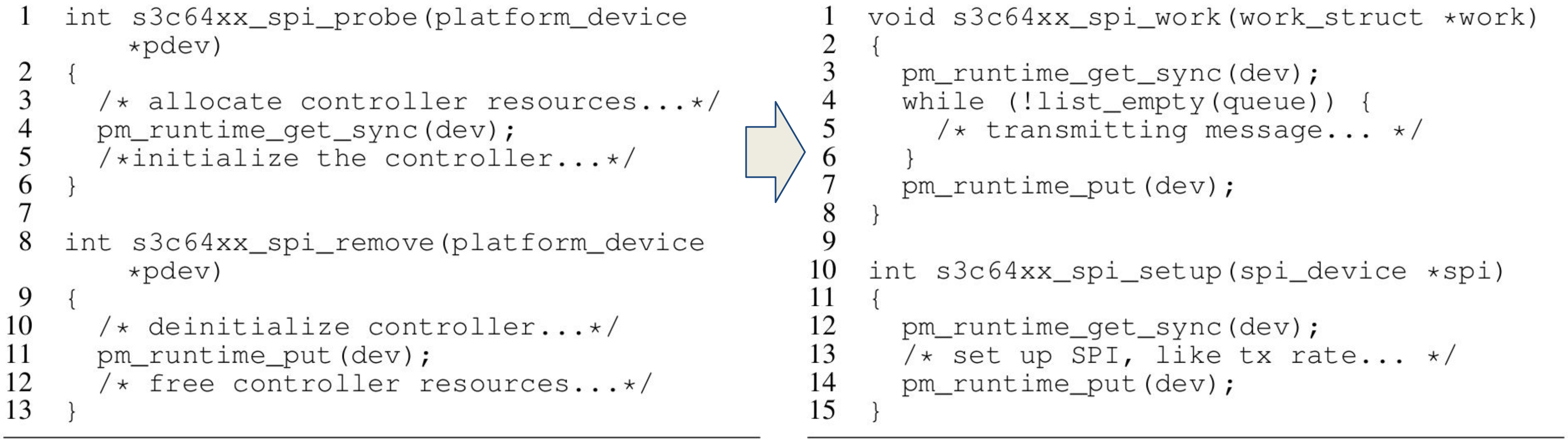
- **I'm done**
  - pm_runtime_put(), _sync, _noidle()
  - Decrement use count, pm_runtime_idle()

Autosuspend

- **Some device should not allow devices to be suspend**
  - until they have been inactive for some minimum period.
- **A common heuristic**
  - similarly a way of PM qos
- **Init**
  - pm_runtime_set_suspended(dev);
  - pm_runtime_use_autosuspend(dev);
  - pm_runtime_set_autosuspend_delay(dev, DELAY);
- **I need the device**
  - pm_runtime_get_sync(dev);
- **I'm done**
  - pm_runtime_put_sync_autosuspend(dev);

However, Real world - Runtime PM

- **Exynos SPI driver**
- **Runtime PM often supported after long delay.**



```
int s3c64xx_spi_probe(platform_device *pdev)
{
  /* allocate controller resources...*/
  pm_runtime_get_sync(dev);
  /*initialize the controller...*/
}

int s3c64xx_spi_remove(platform_device *pdev)
{
  /* deinitialize controller...*/
  pm_runtime_put(dev);
  /* free controller resources...*/
}
```

```
void s3c64xx_spi_work(work_struct *work)
{
  pm_runtime_get_sync(dev);
  while (!list_empty(queue)) {
    /* transmitting message... */
  }
  pm_runtime_put(dev);
}

int s3c64xx_spi_setup(spi_device *spi)
{
  pm_runtime_get_sync(dev);
  /* set up SPI, like tx rate... */
  pm_runtime_put(dev);
}
```

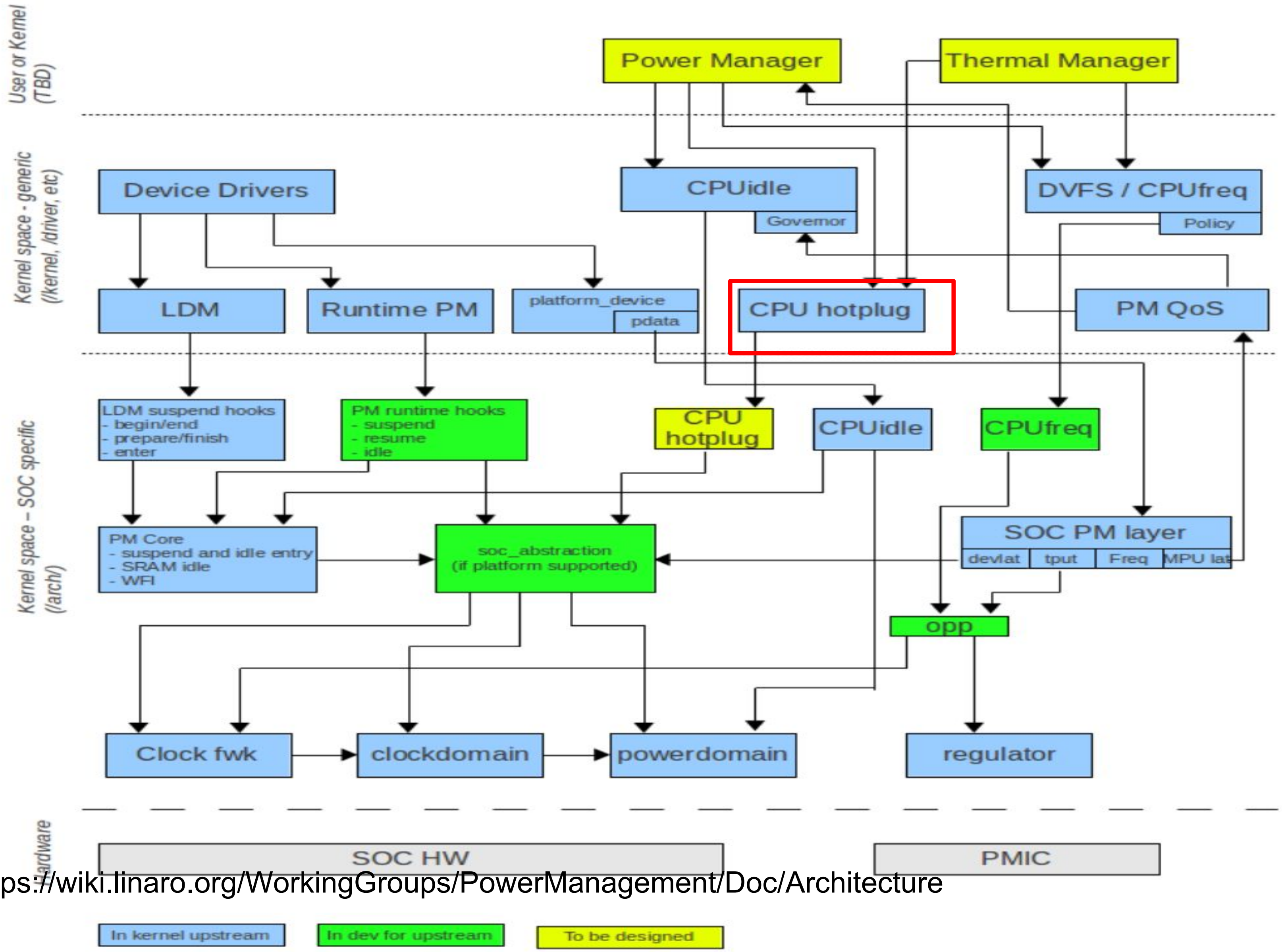
Hand-tuned PM in the Exynos SPI driver

However, Real world - Runtime PM

- **Complex drivers make it hard to do runtime PM.**
  - It is similar with a Lock.

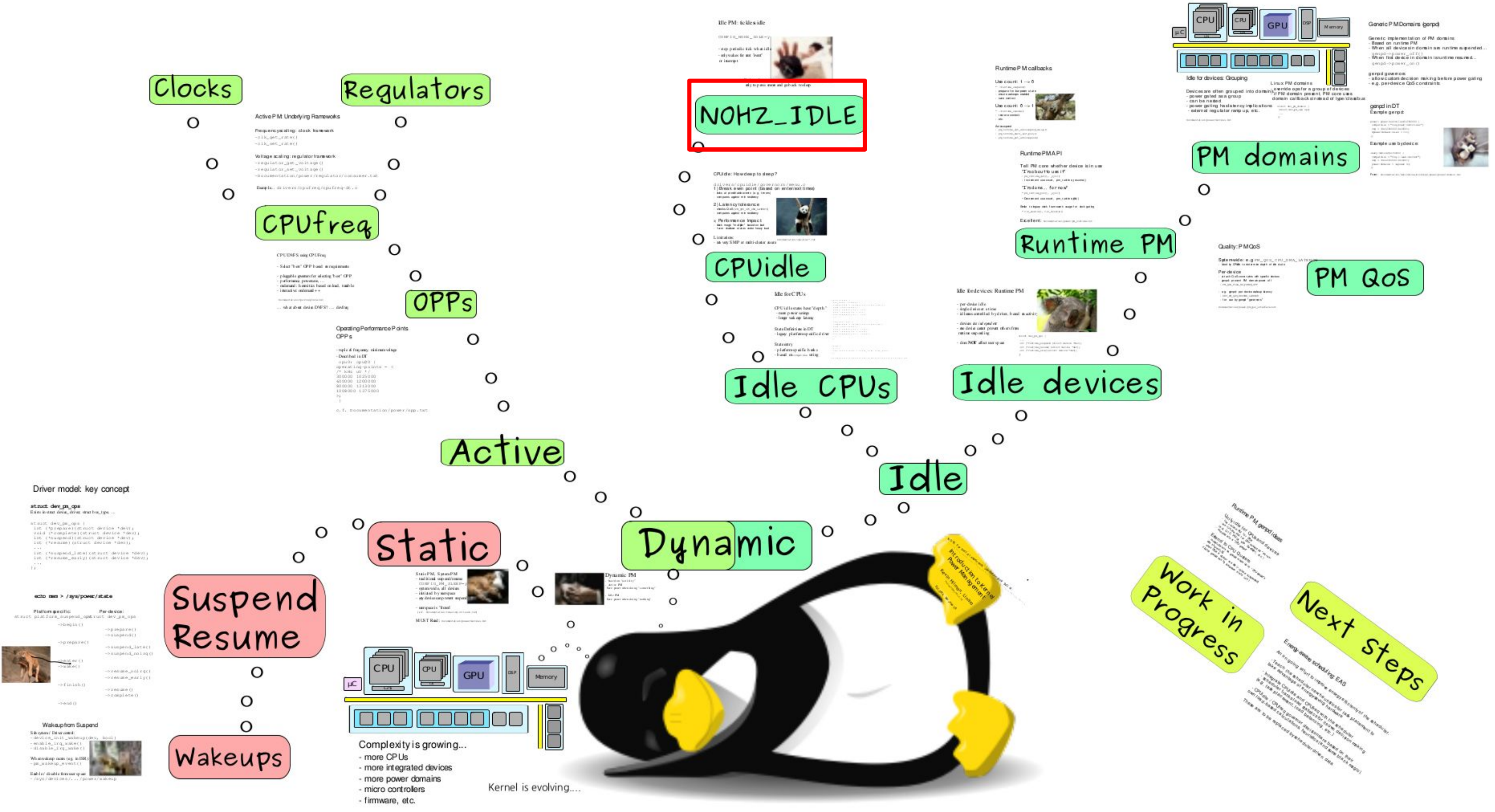
- **The impact of bad drivers.**
  - Because clock, power, voltage depends on the domains.
  - sharing the same clock source.
  - sharing the same power
  - sharing the same voltage

Hotplug



https://wiki.linaro.org/WorkingGroups/PowerManagement/Doc/Architecture

NOHZ_IDLE



Reference from "Introduction to Kernel Power Management" Kevin Hilman, Linaro

Queue work on power efficient wq

- **Problem**
  - Work-queues can be performance or power oriented.
  - keep them running on a single cpu
    - it remains cache hot
  - but consider big.LITTLE platform

Queue work on power efficient wq

- **Problem**
  - Work-queues can be performance or power oriented.
  - keep them running on a single cpu
    - it remains cache hot
  - but consider big.LITTLE platform

- **Solution**
  - Power oriented WQ
  - **Give scheduler the liberty**
    - to choose target cpu for running work handler.

- **Power efficient WQ**
  - workqueue is allocated with WQ_UNBOUND flag
  - Idle time for many cpus has increased considerably on **big.LITTLE platform**

Thermal - iphone, galaxy

Thermal - G6

Thermal

Problem?

Thermal

Spread out

Today

process/thread -> performance -> multi-core performance scalability-> lock-> bottleneck-> cache cohere system problem-> per-core partition approach

Next Step.

Energy-aware scheduling: EAS

1. **CFS scheduler(User level, Tools, Kernel level)**
2. **Load Balancer(Group Scheduling, Bandwidth Control, PELT)**
3. **EAS features**

Reference

- https://pdos.csail.mit.edu/6.828/2016/schedule.html
- http://web.mit.edu/6.033
- http://www.rdrop.com/~paulmck/
- “**Is Parallel Programming Hard, And If So, What Can You Do About It?”**
- Davidlohr Bueso. 2014. Scalability techniques for practical synchronization primitives. *Commun. ACM* 58

http://queue.acm.org/detail.cfm?id=2698990

- **"CPUFreq and The Scheduler Revolution in CPU Power Management", Rafael J. Wysocki**
- **https://sites.google.com/site/embedwiki/oses/linux/pm/pm-qos**
- **https://intl.aliyun.com/forum/read-916**
- **User-level threads : co-routines**

  http://www.gamedevforever.com/291

  https://www.youtube.com/watch?v=YYtzQ355_Co

- **Scheduler Activations**
  - https://cgi.cse.unsw.edu.au/~cs3231/12s1/lectures/SchedulerActivations.pdf
- **https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)**
- **http://jake.dothome.co.kr/**
- **http://www.linuxjournal.com/magazine/completely-fair-scheduler?page=0,0**
- **https://www2.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/6_CPU_Scheduling.html**
- **"Energy Aware Scheduling", Byungchul Park, LG Electronic**
- **“Update on big.LITTLE scheduling experiments”, ARM**
- **“EAS Update” 2015 september ARM**
- **"EAS Overview and Integration Guide”, ARM TR**
- **“Drowsy Power Management”, Matthew Lentz, SOSP 2015**
- **https://www.slideshare.net/nanik/learning-aosp-android-hardware-abstraction-layer-hal**
- https://www.youtube.com/watch?v=oTGQXqD3CNI
- https://www.youtube.com/watch?v=P80NcKUKpuo
- https://lwn.net/Articles/398470/
- “SCHED_DEADLINE: It’s Alive!”, ARM, 2017