SoC Power Management Crash Course

Michael Turquette <mturquette@baylibre.com>

Who am I? And why am I here?

CEO of BayLibre, Inc

- Previously at Texas Instruments, Linaro, San Francisco start-up
- Contributor to various power management-related topics upstream

Author and co-maintainer of the common clk framework

- Merged in 3.4
- Maintenance since May, 2012

Presentation structure

This presentation moves quickly and covers a lot of ground

Interrupt me often! Call out if you disagree, have a question or feel lost

Presentation structure

This presentation moves quickly and covers a lot of ground

Interrupt me often! Call out if you disagree, have a question or feel lost

Most concepts are interchangeable between modern System-on-Chip processors

Terminology tends to lean towards the ARM embedded ecosystem, and less towards the Intel/ACPI view of the world

Presentation structure

This presentation moves quickly and covers a lot of ground

Interrupt me often! Call out if you disagree, have a question or feel lost

Most concepts are interchangeable between modern System-on-Chip processors

Terminology tends to lean towards the ARM embedded ecosystem, and less towards the Intel/ACPI view of the world

We’re flying at about 30,000 feet above sea level (10,000m)

There is a lot of simplification at this altitude

Part 1: PM fundamentals

Power management overview

- The goal of power management (PM) is to consume as little power as needed given the current system state, configuration or use case

Power management overview

- The goal of power management (PM) is to consume as little power as needed given the current system state, configuration or use case

- There are other related goals such as energy management, thermal management and current limiting

Why do we care?

- Battery life
- Data center costs
- Regulatory compliance
- Skin temperature
- Carbon footprint

Physics overview

$$P = IV$$

Physics overview

$$P = IV$$

Instantaneous electrical power is the product of instantaneous voltage and instantaneous current

Physics overview

$$P = IV$$

Instantaneous electrical power is the product of instantaneous voltage and instantaneous current

If either voltage is zero or current is zero then power is zero

Physics overview

$$P = IV$$

Instantaneous electrical power is the product of instantaneous voltage and instantaneous current

If either voltage is zero or current is zero then power is zero

Minimizing power for active use cases and idle use cases is desirable

Physics overview

$$P = IV$$

Instantaneous electrical power is the product of instantaneous voltage and instantaneous current

If either voltage is zero or current is zero then power is zero

Minimizing power for active use cases and idle use cases is desirable

Energy is the integration of power over time

Hardware overview

- Within a system-on-chip (SoC), we generally divide hardware modules into logic and memory

Hardware overview

- Within a system-on-chip (SoC), we generally divide hardware modules into logic and memory

- Modules can be in an active state or an idle state

Hardware overview

- Within a system-on-chip (SoC), we generally divide hardware modules into logic and memory
- Modules can be in an active state or an idle state
- There may be multiple active or running states (performance levels)

Hardware overview

- Within a system-on-chip (SoC), we generally divide hardware modules into logic and memory
- Modules can be in an active state or an idle state
- There may be multiple active or running states (performance levels)
- There may be multiple idle states (deep sleep)

Idle versus Active power savings

Idle

- Saves power when we are not doing work
- Critical sections in Linux device drivers
- Tradeoffs between wakeup latency and power reduction

Idle versus Active power savings

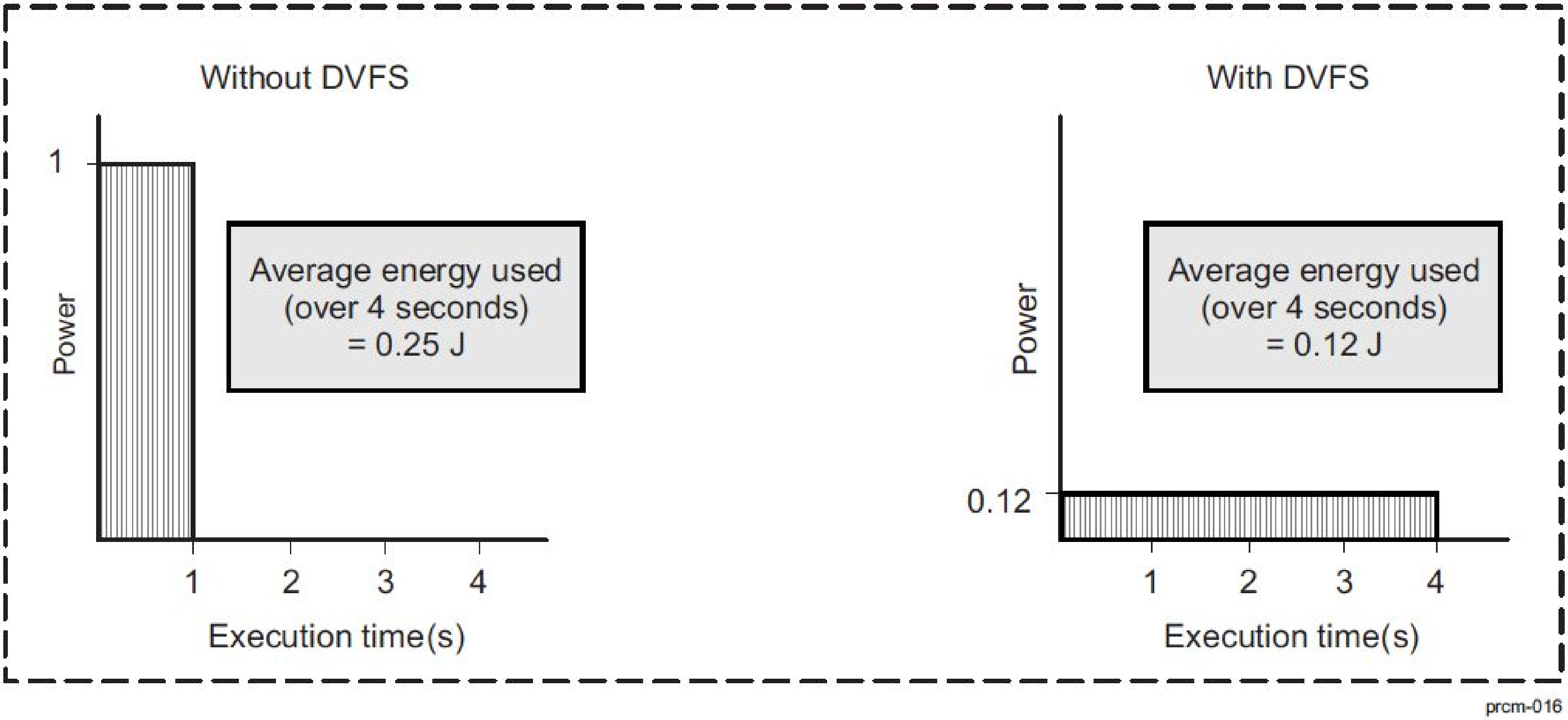
Idle

- Saves power when we are not doing work
- Critical sections in Linux device drivers
- Tradeoffs between wakeup latency and power reduction

Active

- Saves power while we are doing work
- Critical sections are less important
- Tradeoffs between performance and power reduction

Race-to-idle vs Taking-it-slow



Knobs that we control

Voltage

Current

Knobs that we control

Voltage

- While running, a minimum voltage level is required to keep the hardware operating correctly

- Running at different performance levels allows us to scale voltage

- While idle, voltage may be lowered to a very low value while retaining state

Current

Knobs that we control

Voltage

- While running, a minimum voltage level is required to keep the hardware operating correctly
- Running at different performance levels allows us to scale voltage
- While idle, voltage may be lowered to a very low value while retaining state

Current

- Current is drawn by enabled resources such as clock lines, regulators, idle domains and PHYs
- Shutting off these resources when the corresponding devices are inactive (gating) decreases system-wide current draw

How do we control these knobs?

- Memory mapped register interfaces
  - PRCM, PRCMU, CAR, CRM and other IPs within SoC
- Firmware interfaces
  - ACPI, PSCI, SCPI, SCMI, TI-SCI, or stuff using rpmsg
- Communication with Power Management IC (PMIC)
  - I2c or SPI are common methods
  - PMBus or other wrappers also exist
  - GPIO or other line asserts
    - Often combined with WFI/WFE or idle instruction for CPU power management

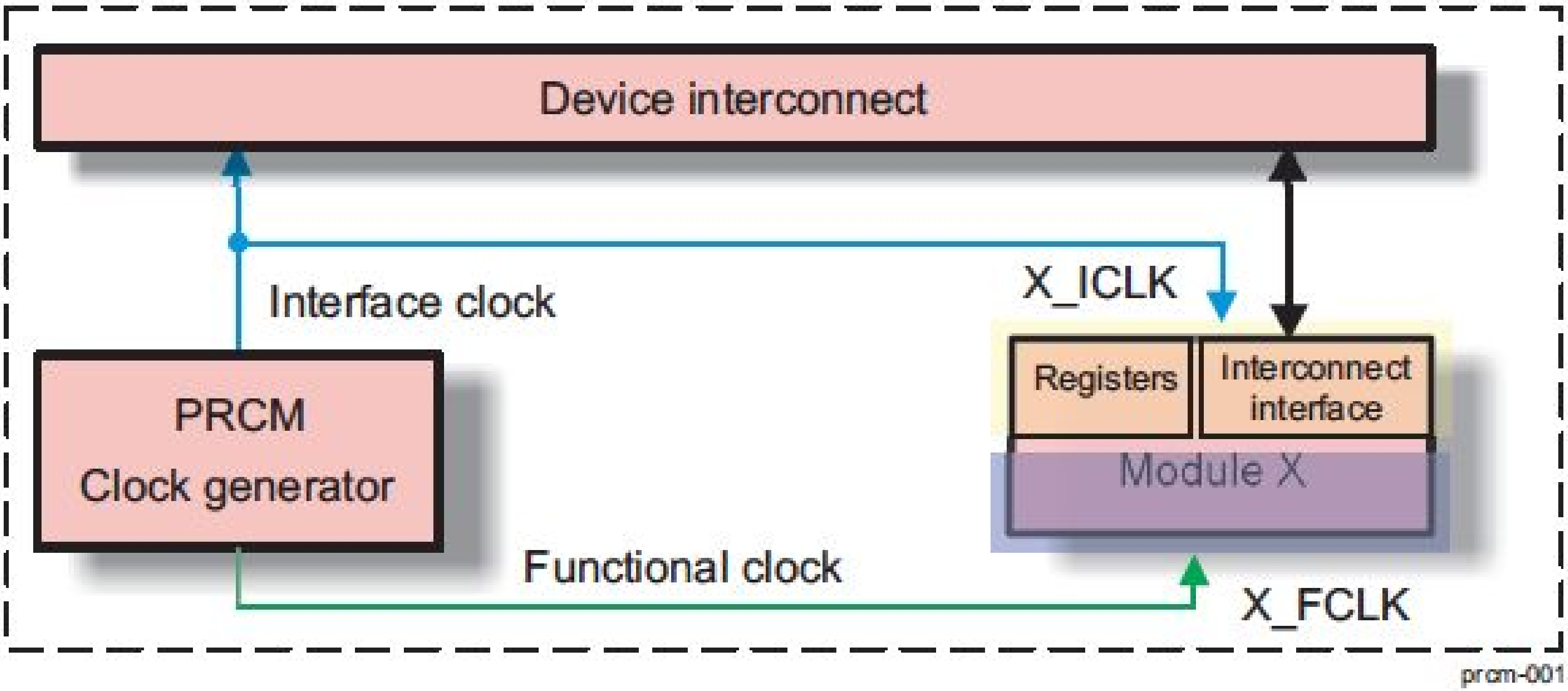
Anyone still awake?

Putting it together

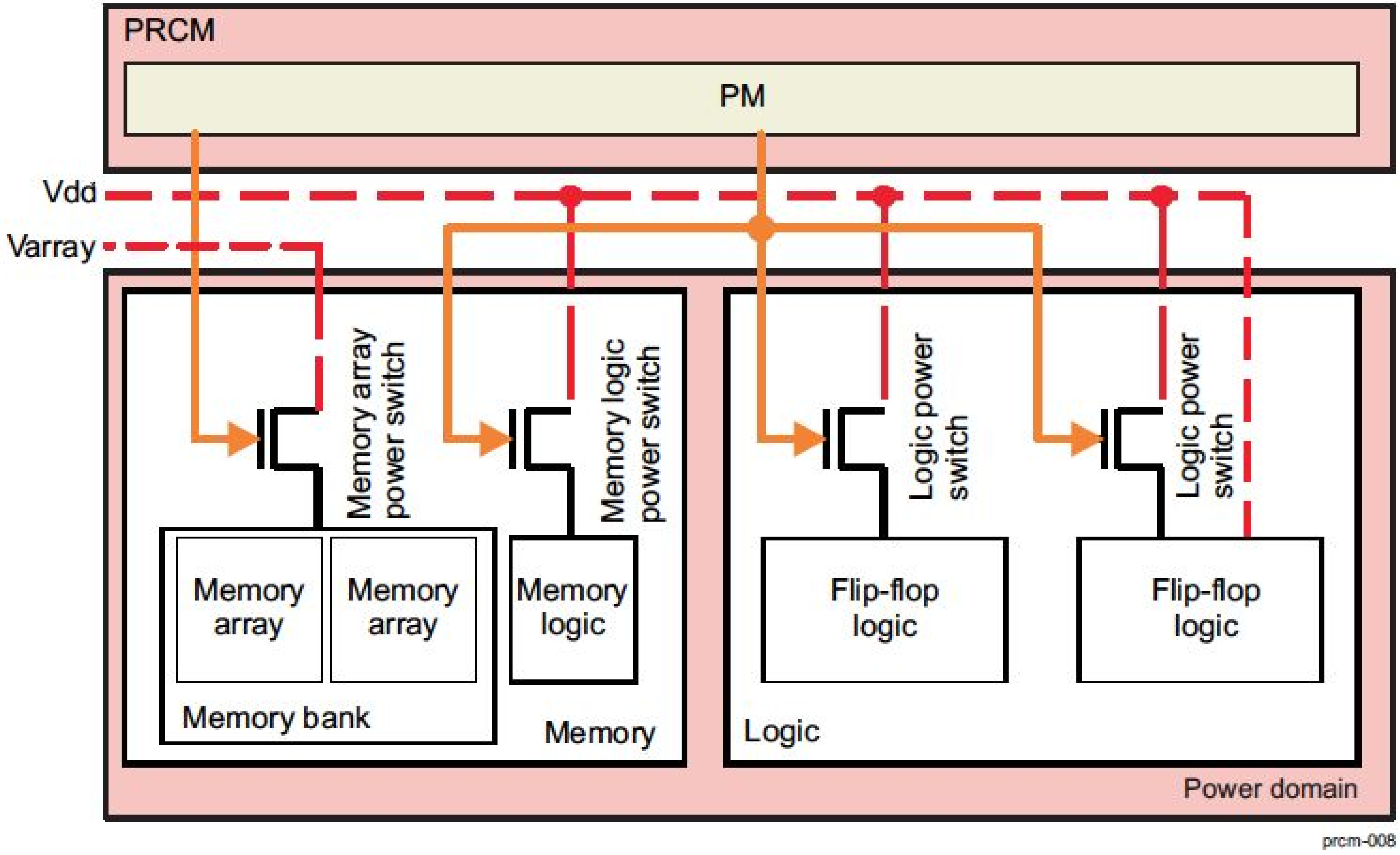
But first, a quick review!

- Try to optimize voltage and current, for both active and idle use cases
- Modern SoCs allow for fine-grained power management
- Controlling power resources is complicated
- Various policies and schemes for saving power

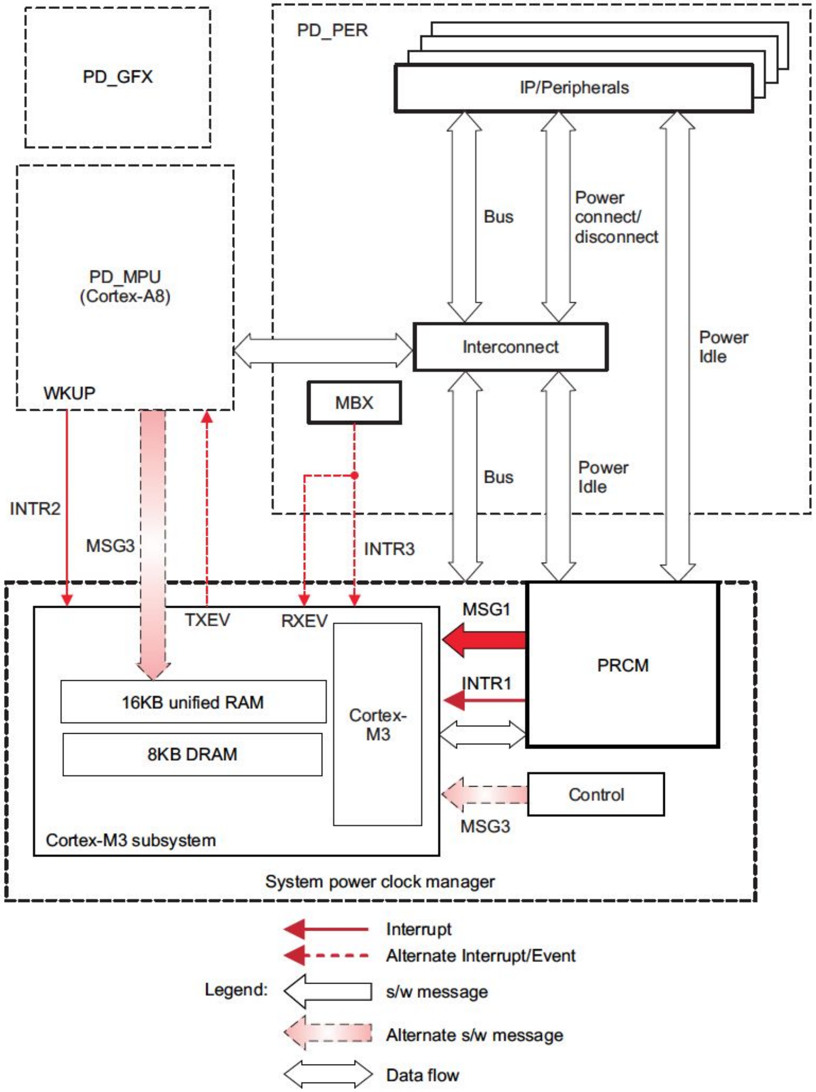
Putting it together: modules & IP blocks



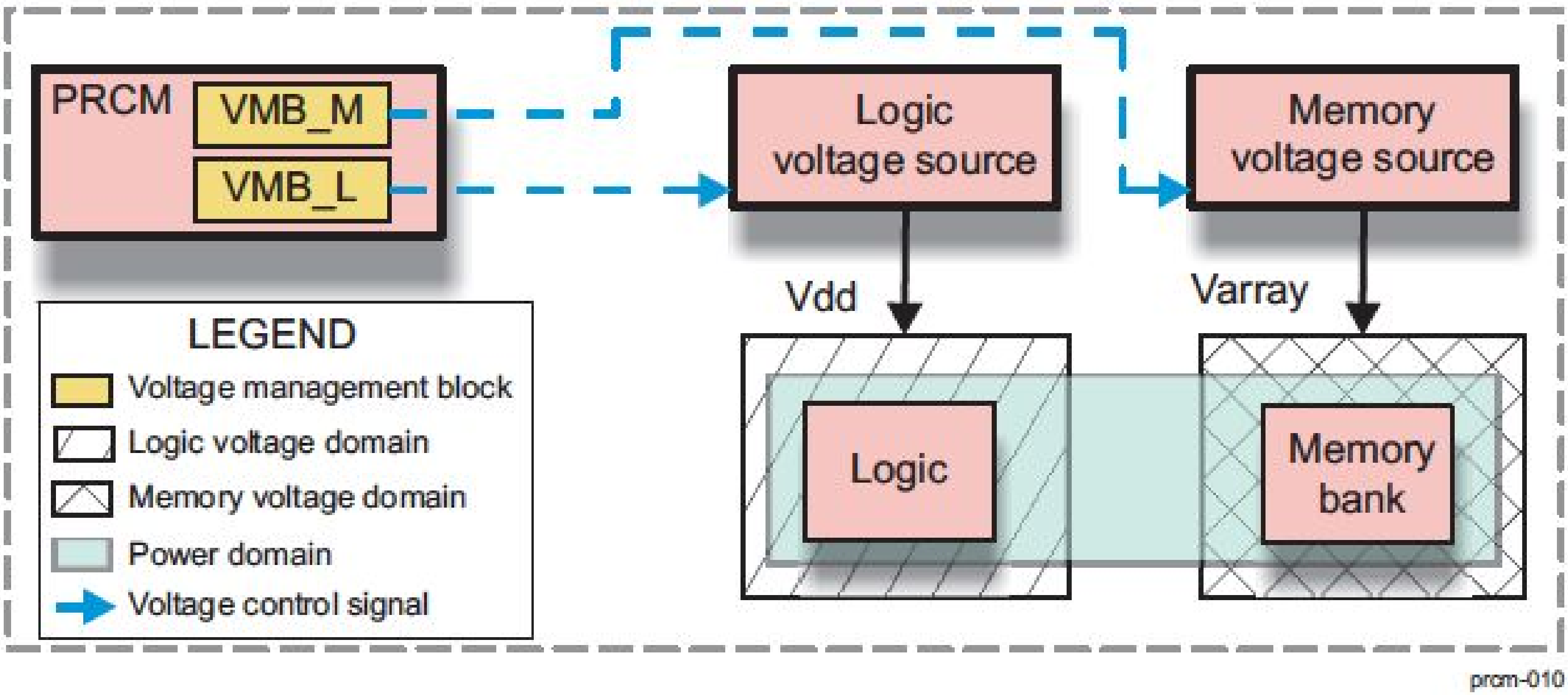
Putting it together: idle domains



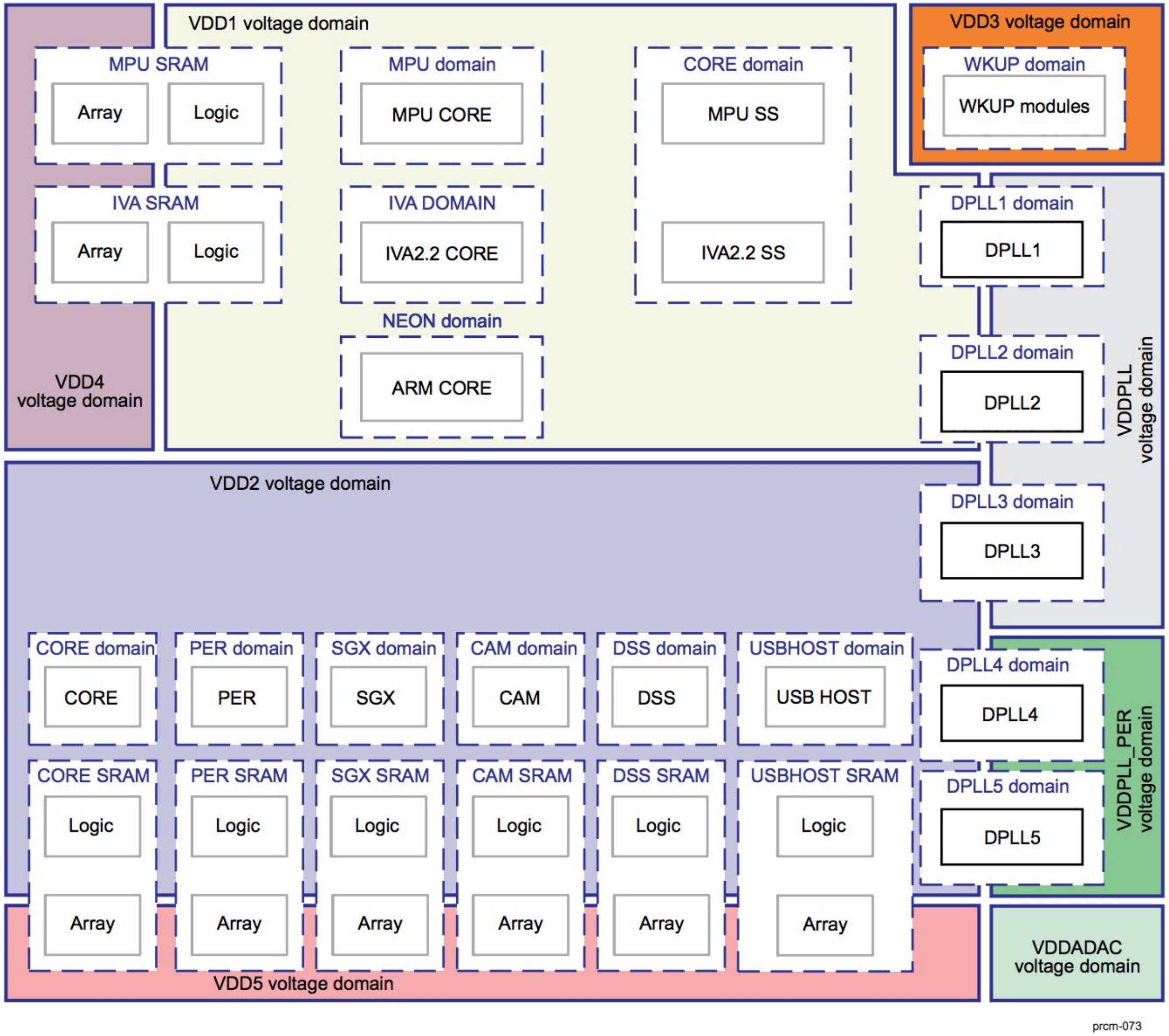
Example silicon: AM335x



Putting it together: performance domains



Example silicon: OMAP3



Part 2: Finally, the Linux stuff!

Idling devices in Linux, 1/4

Runtime PM + Generic PM Domains

The hardware will be in an active state after a device driver calls pm_runtime_get()

The hardware might acquiesce into an idle state after a device driver calls pm_runtime_put()

These form critical sections in the code where work is done

Idling devices in Linux, 2/4

Runtime PM + Generic PM Domains

genpd is the driver framework for controlling the power management hardware and resources

Idle domains and power domains in hardware can be modeled in this framework, and then client devices **attach** to these domains

genpd is the hardware-specific backend for the hardware-independent Runtime PM interface

Idling devices in Linux, 3/4

Runtime PM + Generic PM Domains

include/linux/pm_domain.h

Documentation/devicetree/bindings/power/power_domain.txt

include/linux/pm_runtime.h

Documentation/power/runtime_pm.txt

Idling devices in Linux, 4/4

Runtime PM + Generic PM Domains

http://elinux.org/images/0/08/ELC-2010-Hilman-Runtime-PM.pdf

http://elinux.org/images/1/18/Elc2011_damm.pdf

http://elinux.org/images/1/14/Last_One_Out,_Turn_Off_The_Lights.pdf

CPUidle

Scheduler has a dedicated idle thread

Idle thread calls into the CPUidle driver subsystem

CPUidle driver programs CPUs, clusters & packages into sleep states based on next estimate work

drivers/cpuidle/cpuidle.c

Documentation/cpuidle/*.txt

CPUidle vs Runtime PM & genpd, 1/2

We already have Runtime PM and genpd for managing hardware idle states

Why do something different for CPUs?

- Predates Runtime PM & genpd
- Written by CPU vendors, versus platform/SoC vendors

CPUidle vs Runtime PM & genpd, 2/2

Efforts are ongoing to unify these subsystems

https://linuxplumbersconf.org/2015/ocw/system/presentations/3075/original/One%20idle%20to%20rule%20them%20all.pdf

PM QoS, 1/2

How do we select the idle state?

Per-device PM Quality of Service!

Wake-up latency constraints limit idle state depth

Fast wake-up constraint means shallow idle state

Slow wake-up constraint (or not constraint at all) means deeper idle state

PM QoS, 2/2

include/linux/pm_qos.h

pm_qos_update_request(request, latency);

Affects the hardware idle state when pm_runtime_put() is called

System Suspend & Resume

How is it different from Runtime PM?

The “close your laptop lid” use case

Tells the scheduler to stop … scheduling

struct dev_pm_ops might be replaced with Runtime PM callbacks?

CPUfreq

Similar to CPUidle; controls CPU frequency/performance

Variety of governors or policies

Device Tree bindings have greatly simplified writing drivers for ARM platforms

drivers/cpufreq/*.c

include/linux/cpufreq.h

Devfreq

CPUfreq-like subsystem for managing device performance policy

Extremely similar codebase compared to CPUfreq

Uses governors as policies to select performance target

Best for DDR, memory busses and non-CPU processors such as GPUs, DSPs or other offload engines/accelerators

Operating Performance Points

OPPs are frequency & voltage pairs

In fact, they are tuples of performance state information:

- Clock frequency
- Regulator voltage
- Performance “level”
- State-change sequencing

Used by CPUfreq and Devfreq

Runtime PM for performance?

CPUfreq and Devfreq provide some performance management in Linux

Currently Linux does not have a generic performance power management solution that is similar to what Runtime PM & genpd do for idle power management

I'm interested in fixing this problem. Let me know if you are too!

Things we didn’t have time to talk about

- Process nodes
  - Static and dynamic leakage
  - Cold/nominal/hot bins, also called strong/nominal/weak bins
- Adaptive voltage scaling
  - Silicon aging & tin foil hats
- Instrumenting boards for power measurement
  - Shameless plug: buy ACME! http://baylibre.com/acme/
- Energy Aware Scheduling

Attribution

AM335x Technical Reference Manual (Rev. O)

- http://www.ti.com/lit/pdf/spruh73

OMAP4430 ES2.x Technical Reference Manual (Rev. AP)

- http://www.ti.com/lit/pdf/swpu231

linux-pm@vger.kernel.org mailing list

Questions?