Attack Your SoC Power Challenges with Virtual Prototyping

Stefan Thiel

Gunnar Braun

SYNOPSYS®

Accelerating Innovation

accellera SYSTEMS INITIATIVE

2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Agenda

Part #1: Power-aware Architecture Definition

- Part #2: Power-aware Software Development
- Questions

Power-aware Architecture Definition

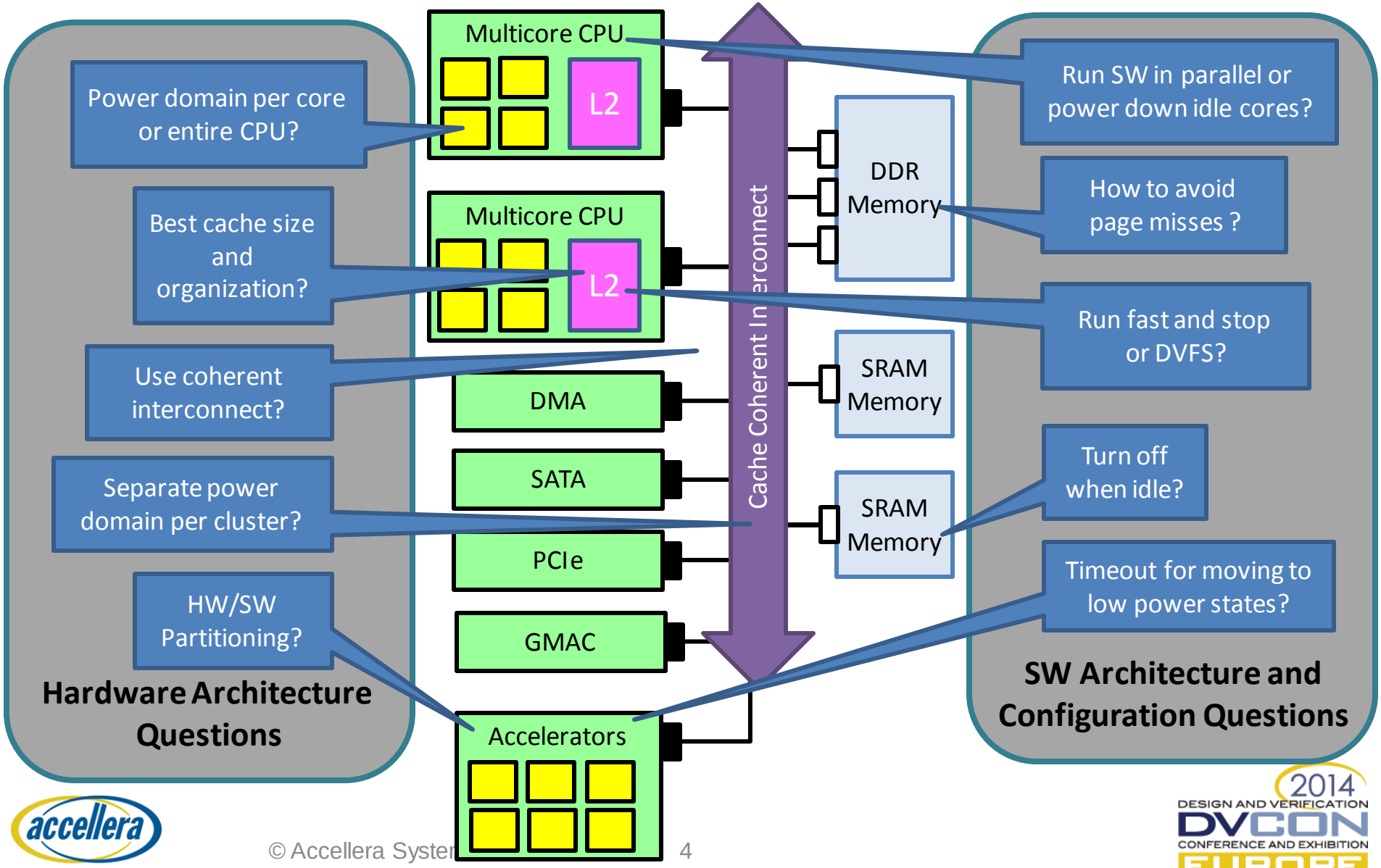
Top three goals for today

Reduce risk of wrong design decision due to late power analysis

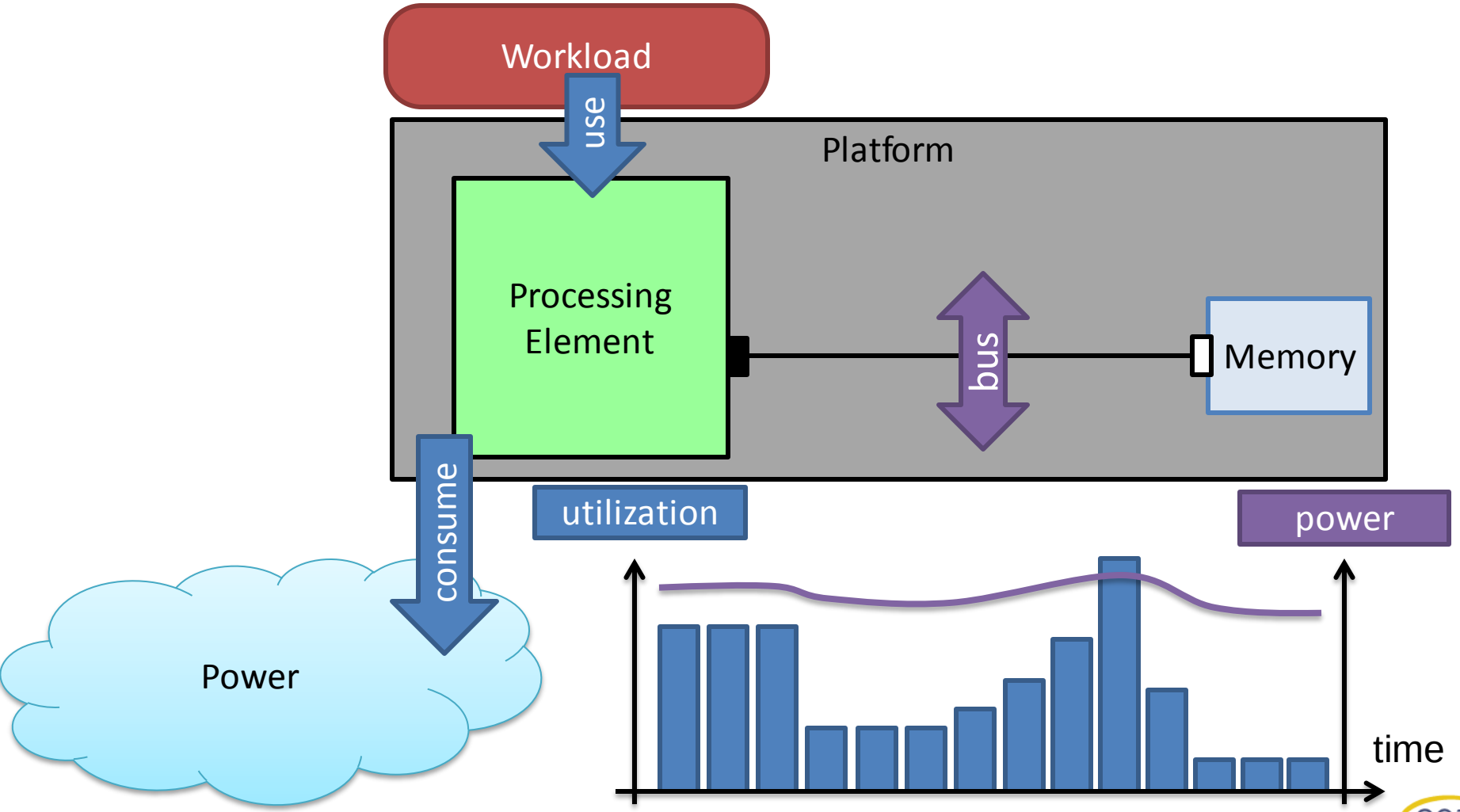
Define the right HW/SW architecture to meet power and performance goals

Define the right Power Management strategy

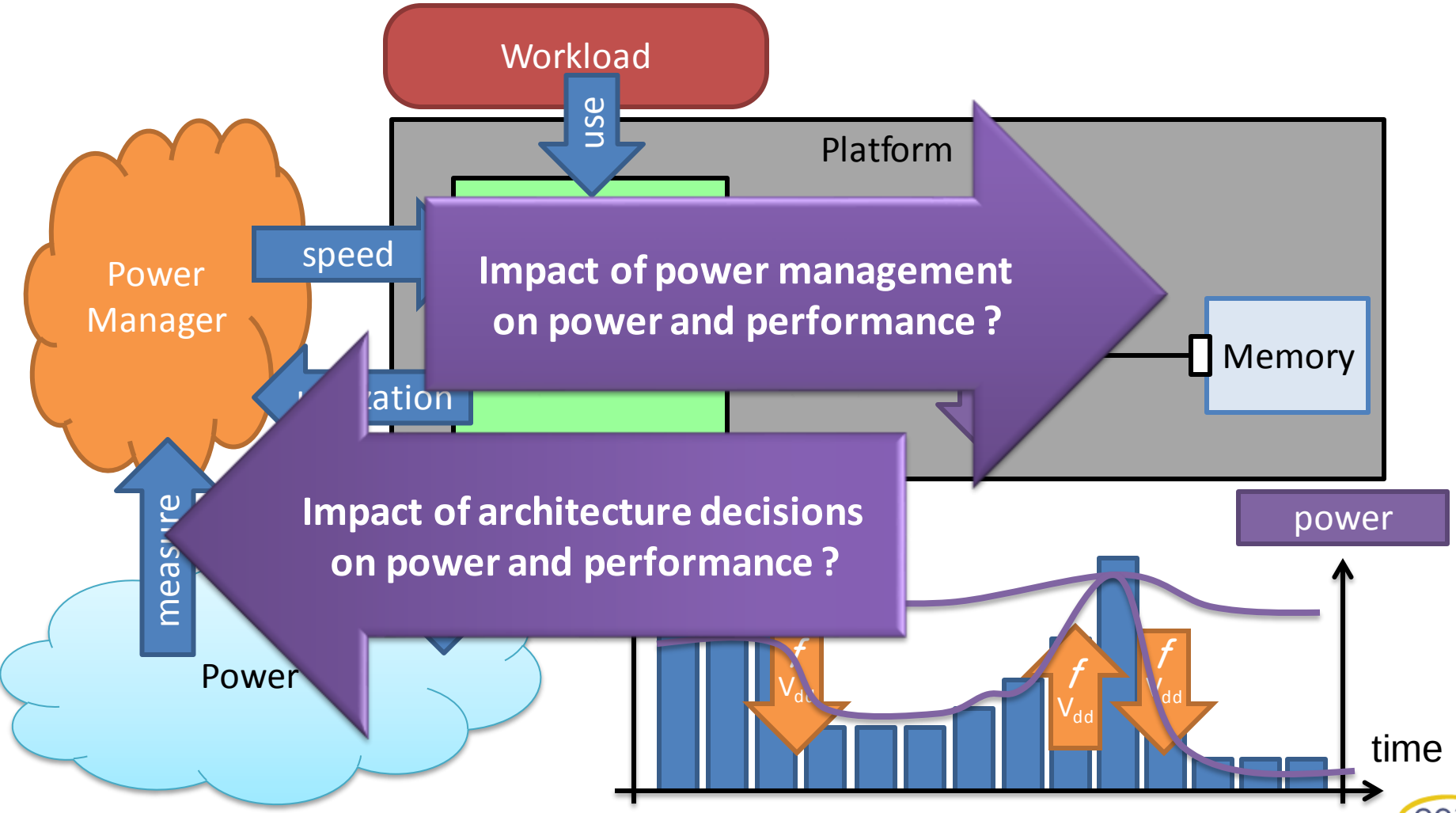
How does architecture affect power?



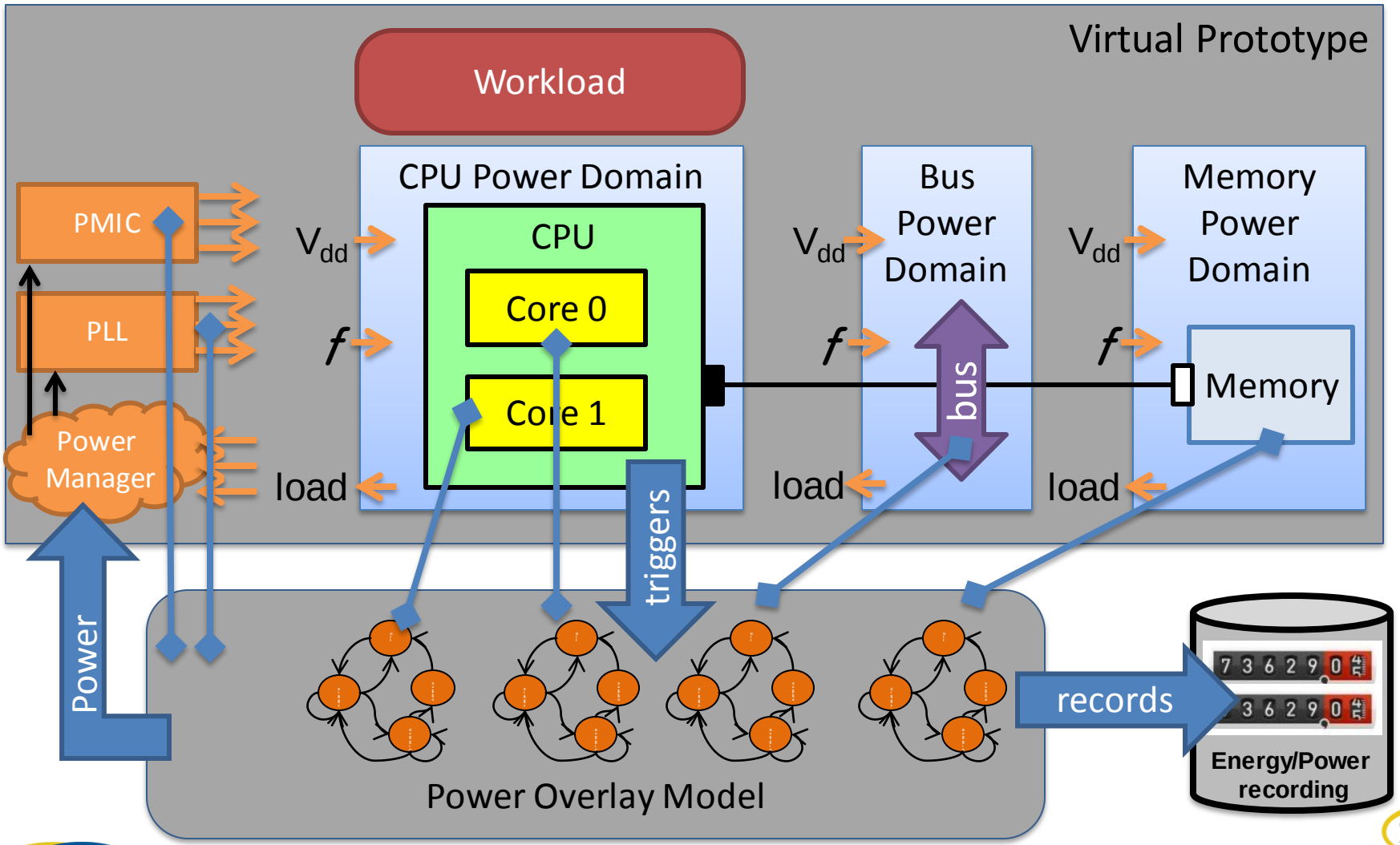
Power and Performance Duality



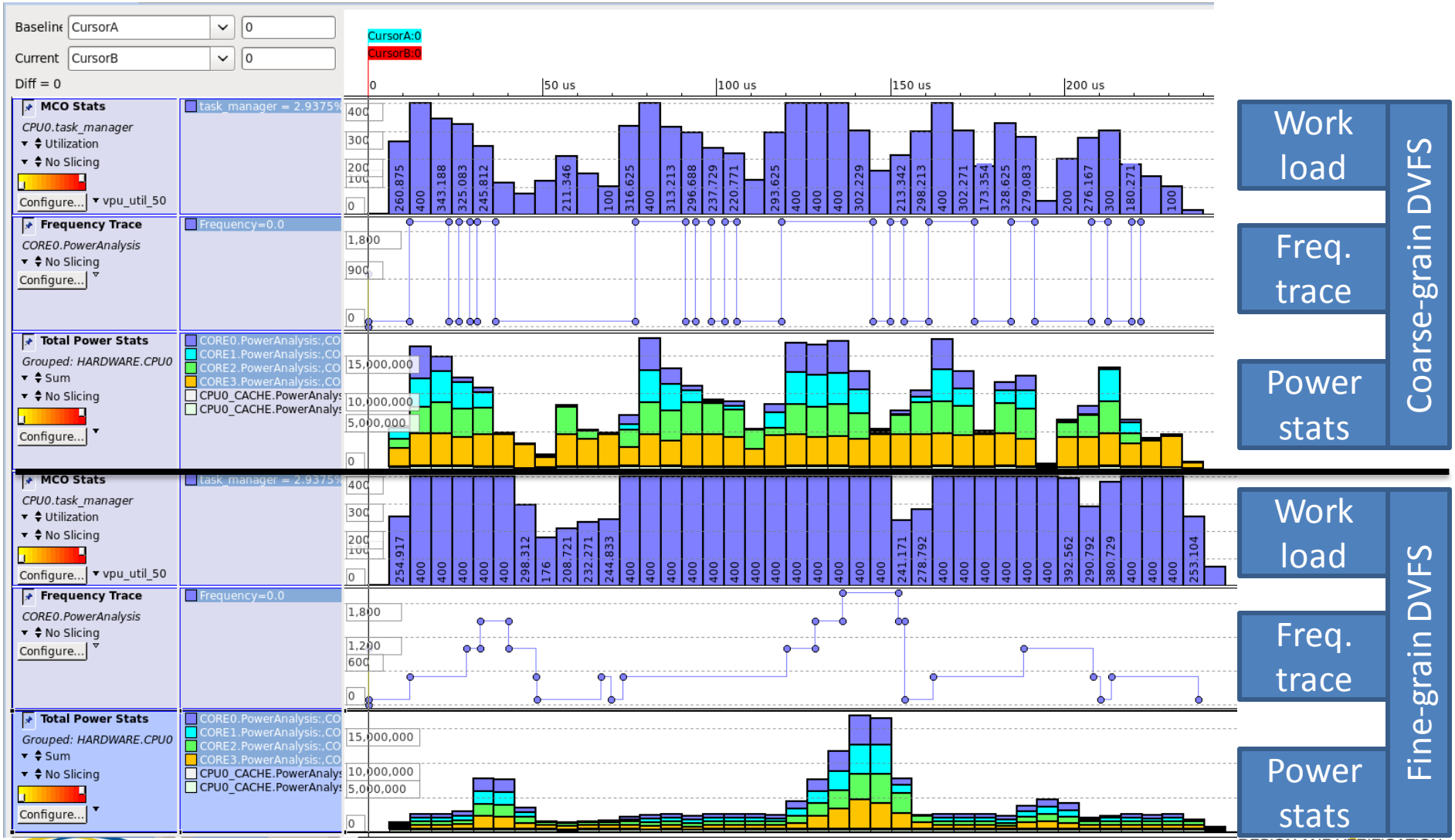
Power and Performance Duality



Virtual Prototyping Approach

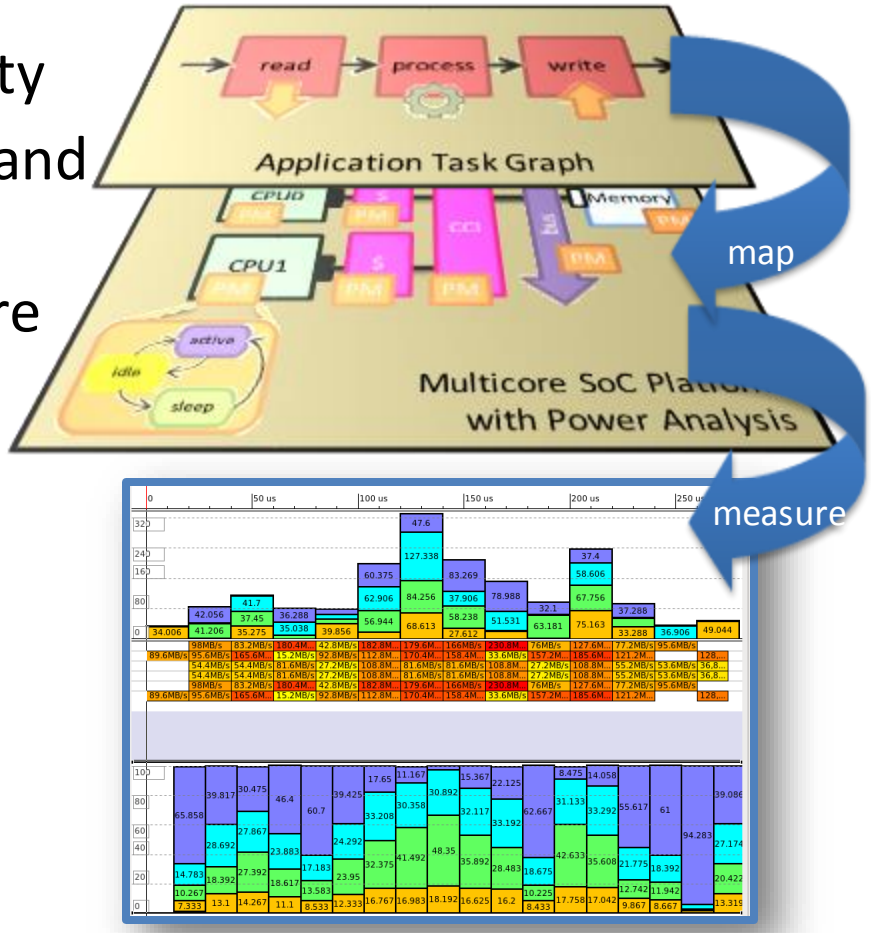


Example: DVFS What-If Analysis



What we just learned

- Reduce risk of late power analysis
- The Power and Performance duality
- Virtual Prototype for early power and performance analysis
  - Create workload model to capture processing and communication requirements
  - Create architecture model to represent processing and communication resources
  - Map workload onto architecture
  - Measure resulting system performance and power analysis



IEEE 1801 UPF System Level Power

- New IEEE 1801 Sub-committee on System Level Power
- Participation from EDA, IP providers and users
  - Active participation from AGGIOS, ARM, Broadcom, Cadence, CSR, Intel, Mentor, Microsoft, ST, Synopsys
- Goal:
  - Standardize format for system level power analysis
  - Enable exchange of power models between groups, companies and EDA tools and across abstraction levels
- Visit
  - http://standards.ieee.org/develop/wg/UPF.html
  - http://www.p1801.org/
  - http://semiengineering.com/system-level-power-modeling-activities-get-rolling/

Power-aware Architecture Definition

Top three goals for today

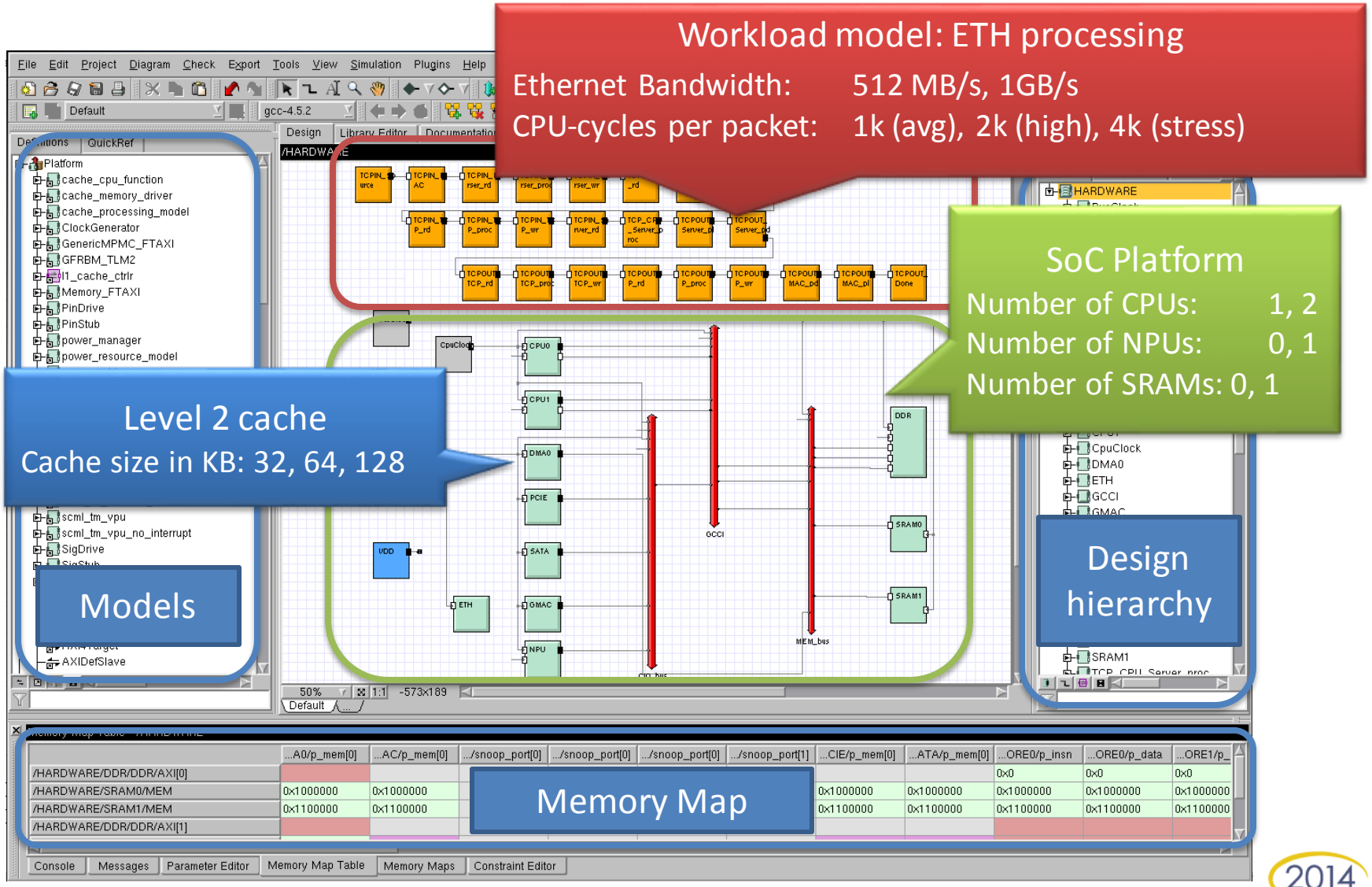
Reduce risk of wrong design decision due to late power analysis

Define the right HW/SW architecture to meet power and performance goals

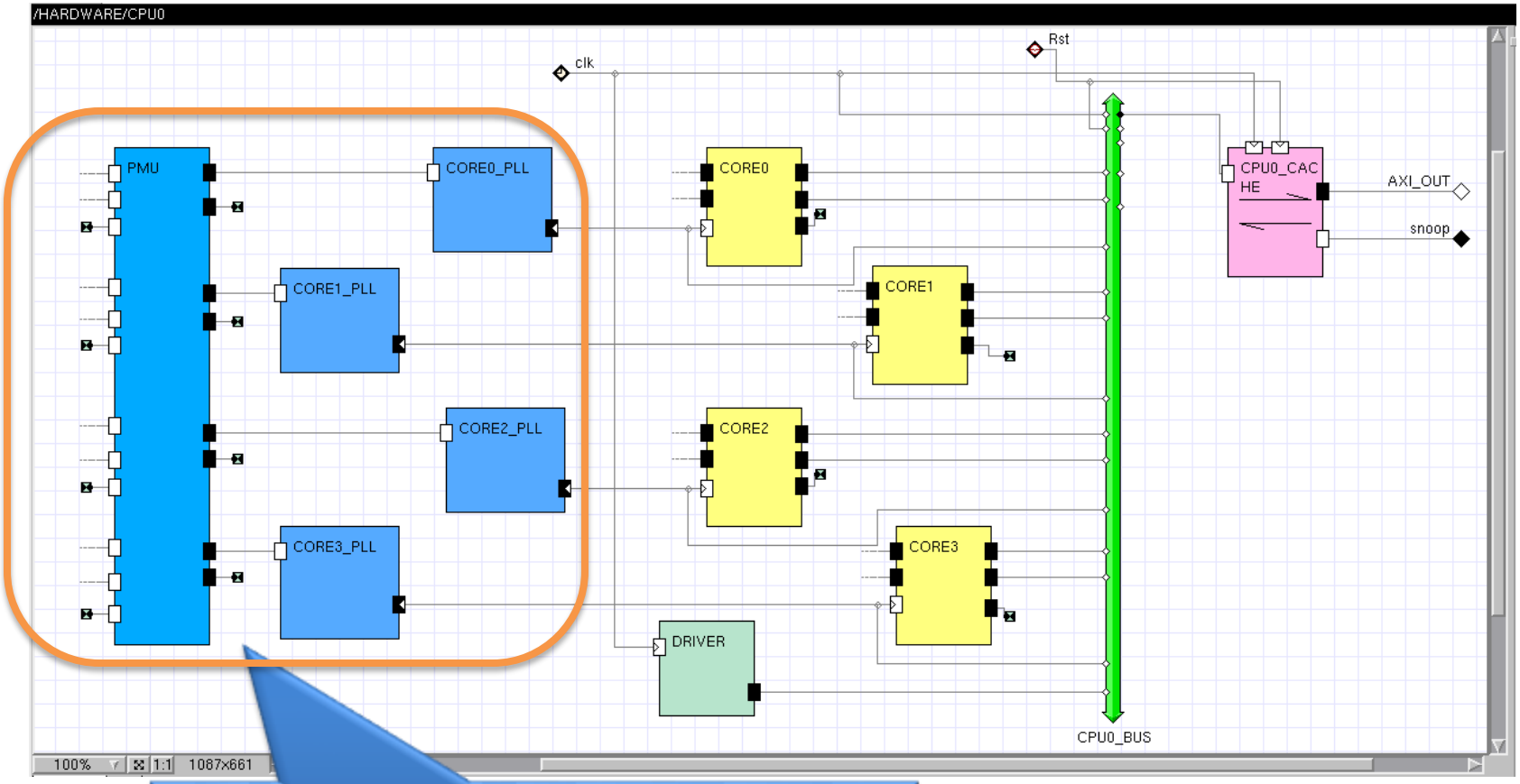
Define the right Power Management strategy

Case Study

Micro Server in Platform Architect MCO



Micro Server in Platform Architect MCO



Power Manager

DVFS-levels: 1, 2, 3, 4,

Thresholds: when to scale up and down

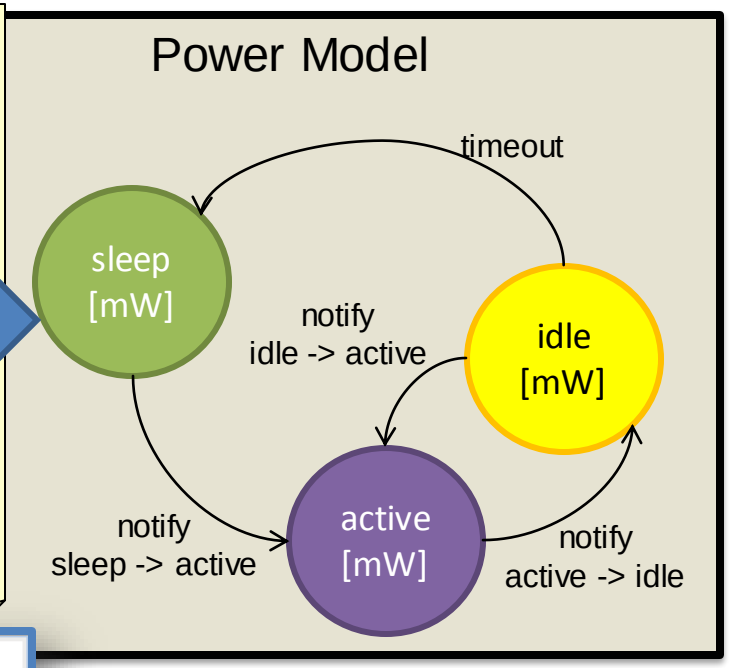
Response delays: slow, med, fast

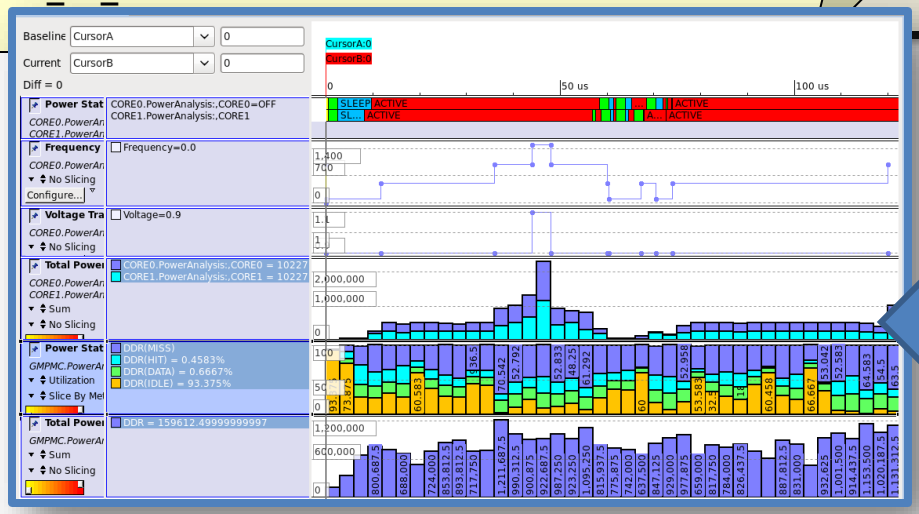
Power Model Instrumentation

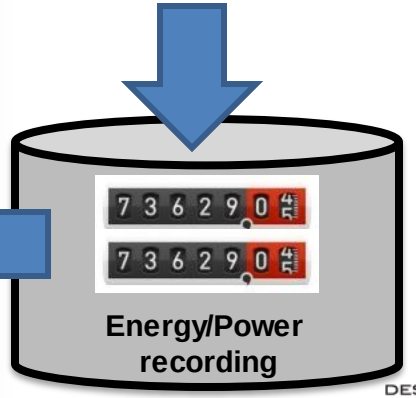
```
set pm [SNPS_PAM create_monitor {Power Analysis} Cpu0Pam CPU0]

$pm set_frequency_input signal CPU0/PLL/clk_in
$pm set_voltage_input   signal CPU0/PMIC/vdd_out

# define states
$pm add_state CORE0 SLEEP     5mW  1.1V   1.5GHz
$pm add_state CORE0 IDLE     70mW  1.1V   1.5GHz
$pm add_state CORE0 ACTIVE  120mW  1.1V   1.5GHz
...
# state transition triggers: signals
$pm link_signal_to_state CORE0 CPU.CORE0.p_notify 0 IDLE
$pm link_signal_to_state CORE0 CPU.CORE0.p_notify 1 ACTIVE
...
# state transition triggers: timeout
$pm link_timeout_to_state 2ms CORE0 SLEEP IDLE
...
```







Power Model Instrumentation - Details

Power State Definition

```
set pm [SNPS_PAM create_monitor {Power Analysis} modem_PAM TOP.modem]

$pm add_state modem_fsm Off       0W
$pm add_state modem_fsm Standby   5mW
$pm add_state modem_fsm Transmit 200mW
$pm add_state modem_fsm Receive  150mW
```

Power Model Instrumentation - Details

Dynamic Voltage and Frequency Scaling

- For each power state we define:
  - Reference dynamic power $\mathbf{P}_{dyn,ref}$
    - for a reference frequency $\mathbf{F}_{dyn,ref}$ and reference voltage $\mathbf{V}_{dyn,ref}$
  - Reference leakage power $\mathbf{P}_{leak,ref}$
    - for a reference voltage $\mathbf{V}_{leak,ref}$
  - Trigger frequency and voltage from Virtual Prototype

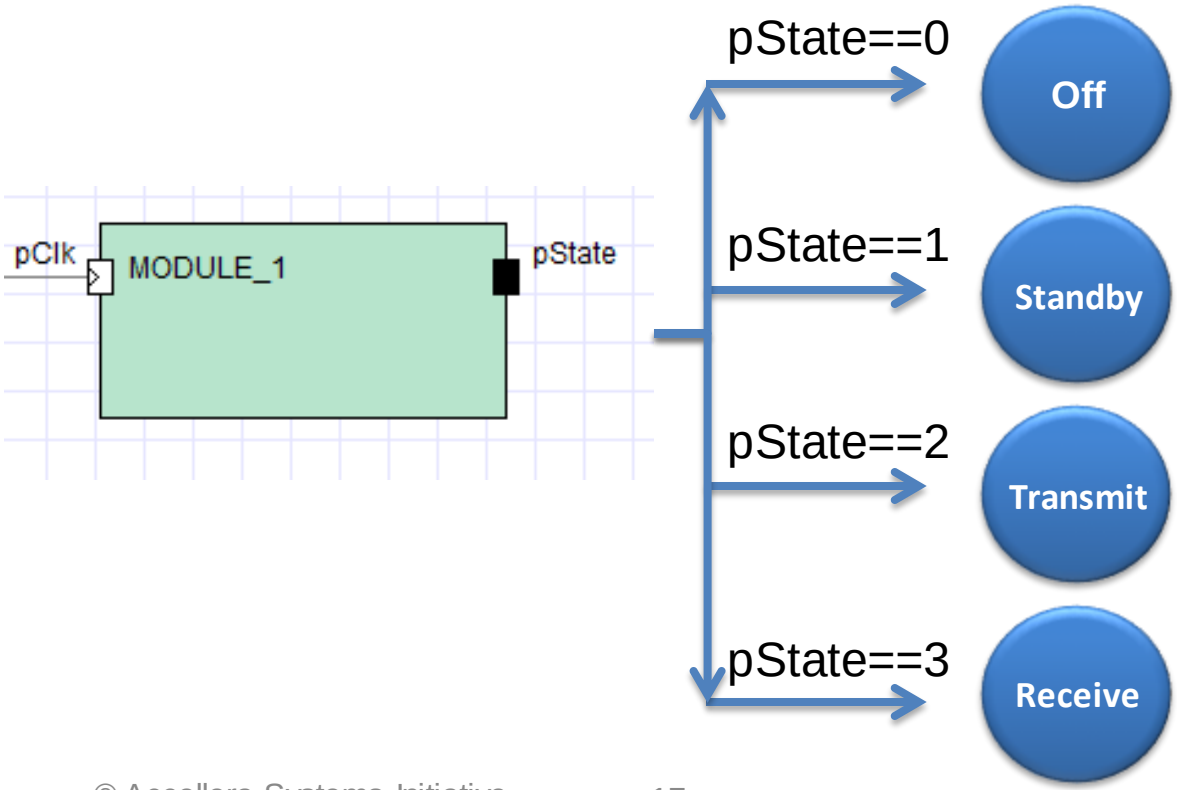
```
$pm set_frequency_input signal TOP/MODULE_3/clk_in
$pm set_voltage_input   signal TOP/PMIC/vdd_out
$pm add_state modem_fsm Off      0W    0W  100MHz 1.1V 1.1V
$pm add_state modem_fsm Standby  5mW   5mW 100MHz 1.1V 1.1V
$pm add_state modem_fsm Transmit 200mW 5mW 100MHz 1.1V 1.1V
$pm add_state modem_fsm Receive  150mW 5mW 100MHz 1.1V 1.1V
```

$\mathbf{F}_{dyn,ref}$ $\mathbf{V}_{dyn,ref}$ $\mathbf{V}_{leak,ref}$

Power Model Instrumentation - Details

Driving Power States from a (HW) signal

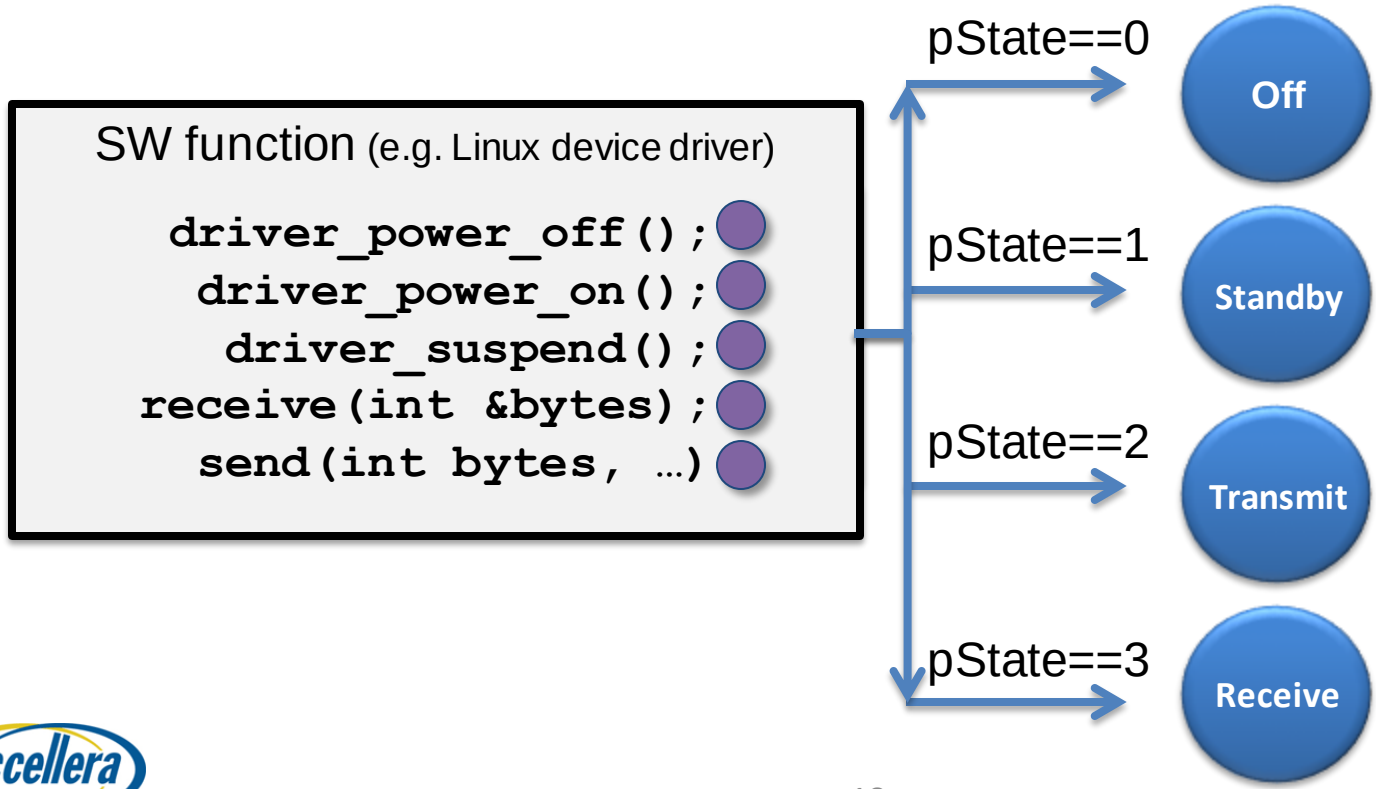
```
$pm link_signal_to_state TOP/MODULE_1/pState 0 0xf modem_fsm Off
$pm link_signal_to_state TOP/MODULE_1/pState 1 0xf modem_fsm Standby
$pm link_signal_to_state TOP/MODULE_1/pState 2 0xf modem_fsm Transmit
$pm link_signal_to_state TOP/MODULE_1/pState 3 0xf modem_fsm Receive
```



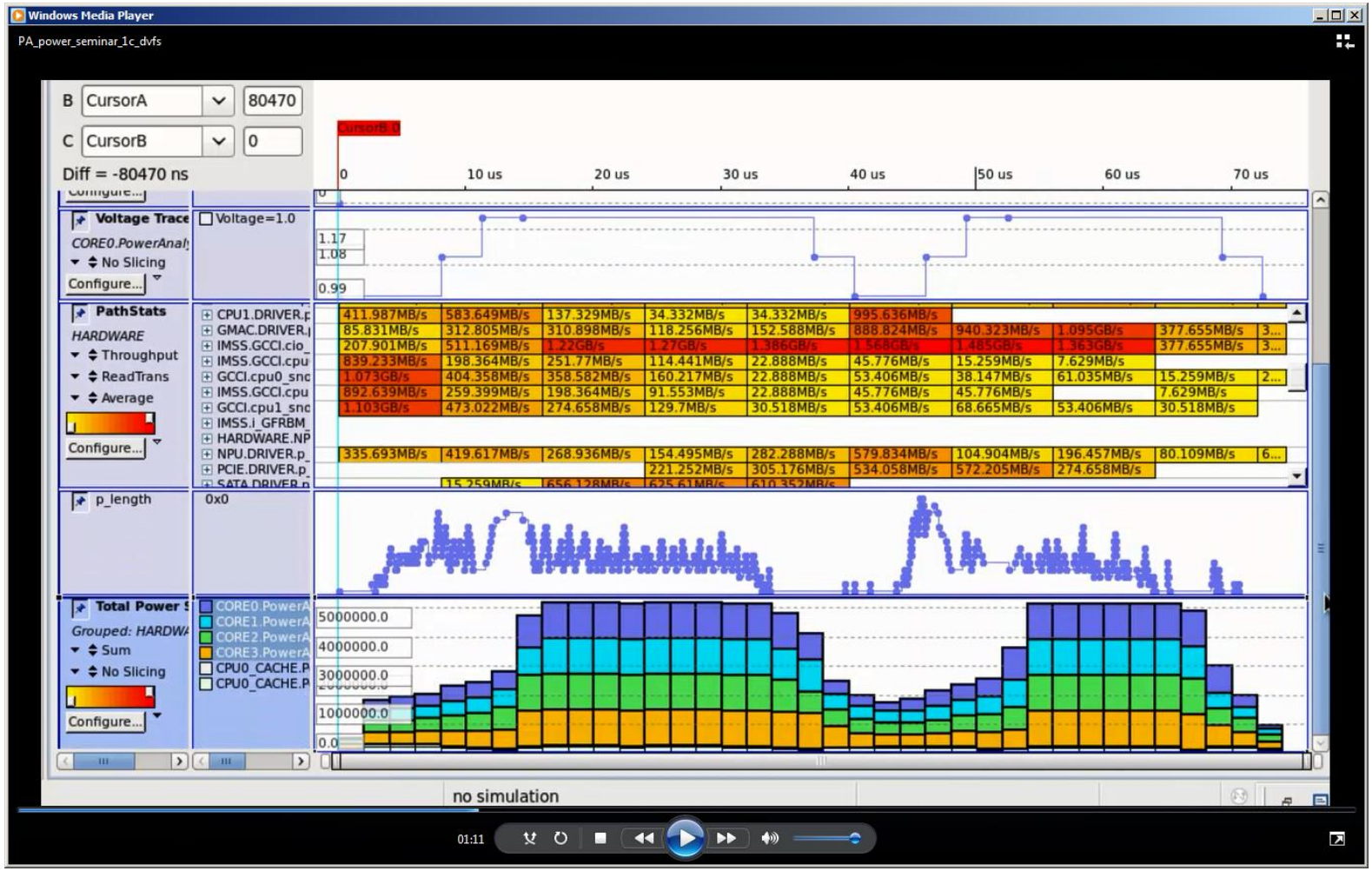
Power Model Instrumentation - Details

Driving Power States from Software

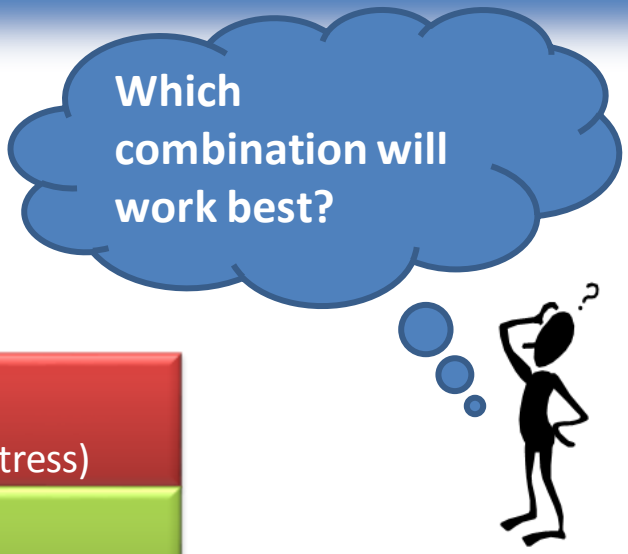
```
$pm link_instruction_address_to_state modem_fsm driver_power_off Off
$pm link_instruction_address_to_state modem_fsm driver_power_on  Standby
$pm link_instruction_address_to_state modem_fsm send             Transmit
$pm link_instruction_address_to_state modem_fsm receive          Receive
```



Video: Analyze Power Management



Parameters to Explore



- Architecture and workload parameters

Workload	Ethernet Bandwidth:	512 MB/s, 1GB/s
	CPU-cycles per packet:	1k (avg), 2k (high), 4k (stress)
Platform	Number of CPUs:	1, 2
	Number of NPUs:	0, 1
	Number of SRAMs:	0, 1
L2 cache	Size in KB:	32, 64, 128

6 parameters
144 combinations

- DVFS power management related parameters

DVFS-levels:	1, 2, 3, 4, 5
Thresholds (up/down):	2/1, 4/1, 8/1, 4/2, 8/2, 8/4
Response delays in us:	1, 2, 3, 4, 5

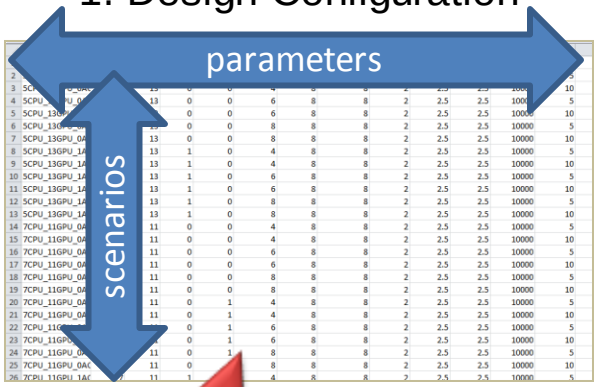
3 parameters
150 combinations

➔ Total of 21600 combinations ???

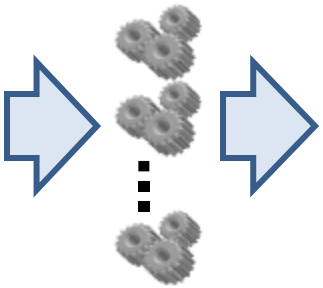
➔ Sensitivity Analysis + Divide and Conquer!

Sensitivity Analysis

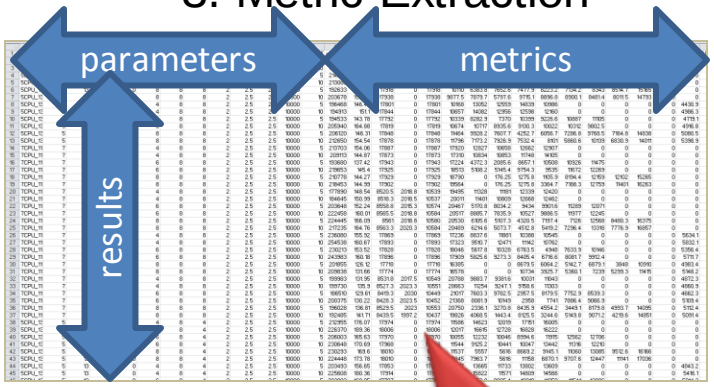
1. Design Configuration



2. Simulation sweep



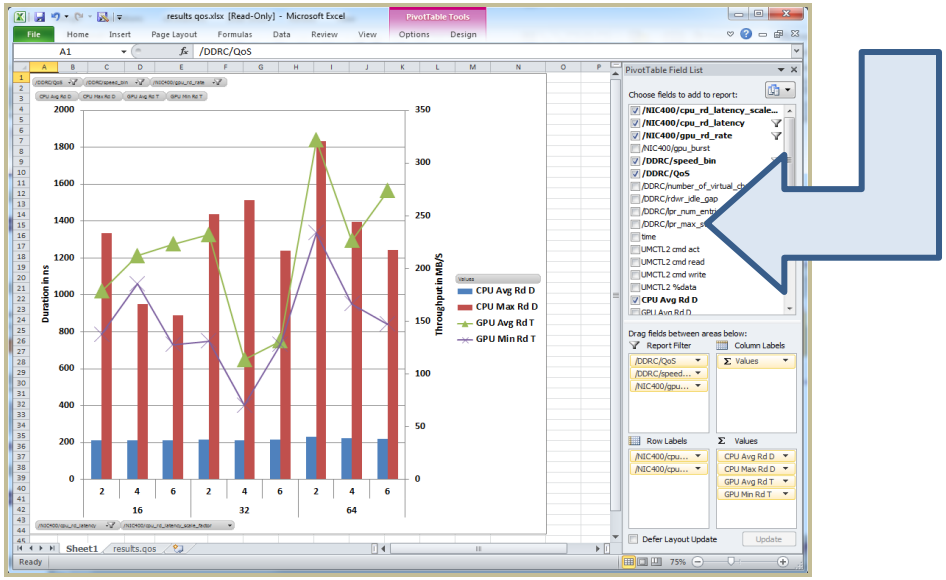
3. Metric Extraction



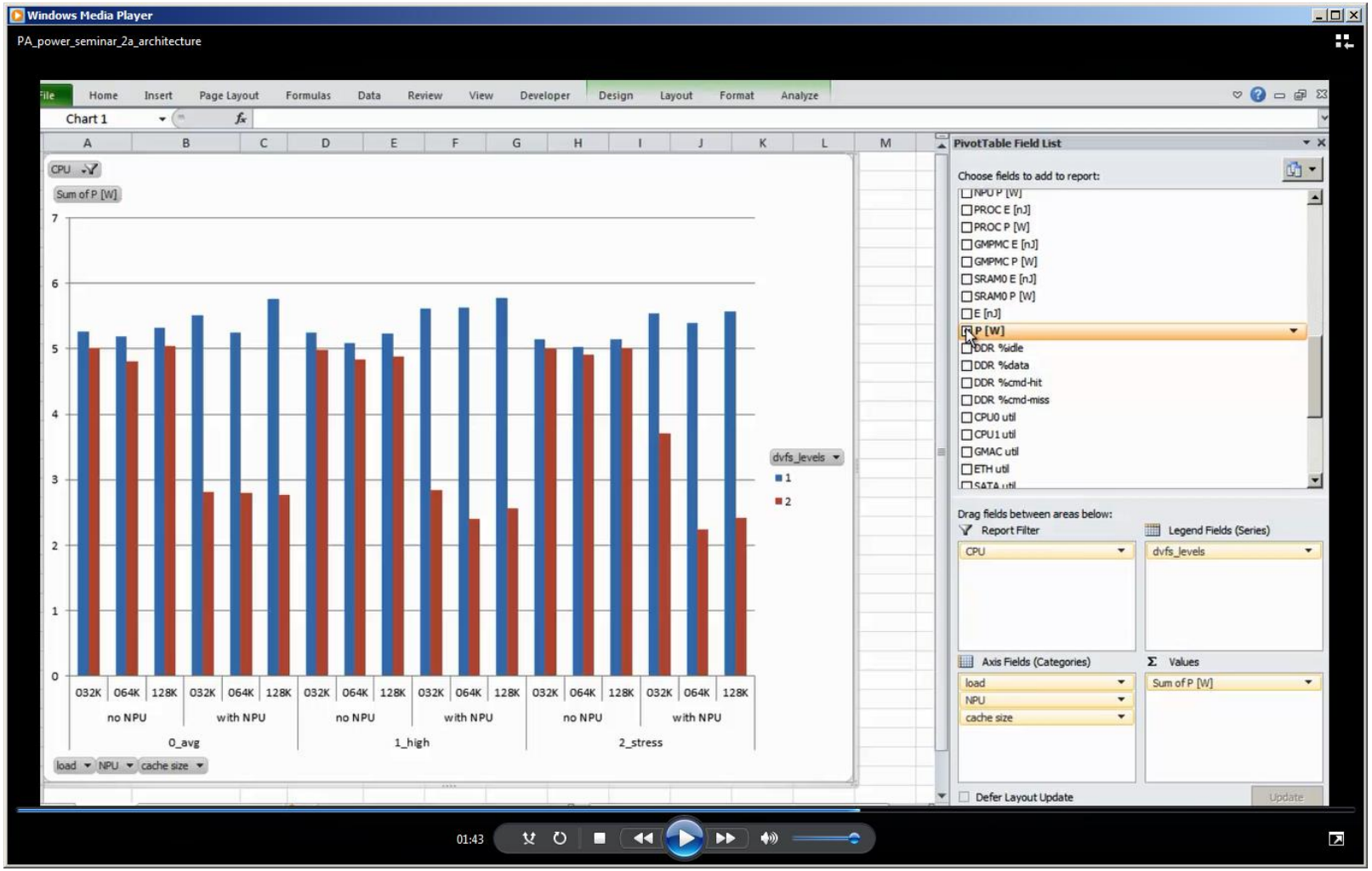
Spreadsheet In

Spreadsheet Out

4. Sensitivity Analysis



Video: Explore Architecture



Parameters to Explore

- Best architecture configuration:

Workload	Ethernet Bandwidth:	512 MB/s, 1GB/s
	CPU-cycles per packet:	1k (avg), 2k (high), 4k (stress)
Platform	Number of CPUs:	**2**
	Number of NPUs:	**1**
	Number of SRAMs:	**1**
L2 cache	Size in KB:	**64**

Results from first sweep

- DVFS power management related parameters

DVFS-levels:	1, 2, **3**, 4, 5
Thresholds (up/down):	2/1, 4/1, 8/1, 4/2, **8/2, 8/4**
Response delays in us:	1, **2, 3**, 4, 5

3 parameters
150 combinations

Summary – Power aware Architecture Definition

Reduce risk of wrong design decision due to late power analysis

- Power and Performance duality: Performance impacts power, power (management) impacts performance
- Early power analysis important for taking the right design decisions

Define the right HW/SW architecture to meet power and performance goals

- Power aware HW/SW partitioning, cache and cache coherency analysis, interconnect/memory optimization

Define the right power management strategy

- Grouping of components into power domains
- Run-fast-then-stop vs. DVFS
- Power management thresholds, time-outs and delays

Agenda

- Part #1: Power-aware Architecture Definition
- Part #2: Power-aware Software Development
- Questions

Power-aware Software Development

Top three goals for today

Reduce risk of late power management software development

Improve robustness of power management software

Reveal software bugs that can drain the battery

How can software affect power?

Software controls the activity of the power consumers

Power management – application & use-case level

- Selection of best effort service
- Example: Turn off WiFi and use 3G when user idle
- Based on user's performance/power needs

Power management – operating system level (OSPM)

- Runtime control of (sub-) system power modes
- Example: Drive WiFi subsystem into standby
- Steered by application level performance/power needs

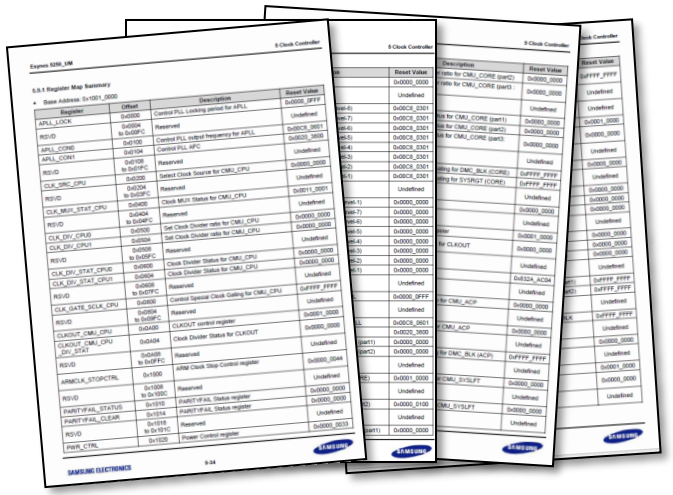
Power control – firmware level

- Control clocks and voltages
- Example: set voltage regulator to 1.1V, set clock to 1GHz
- Initiated by operating system power management

Power Management complexity

Example: Mobile Application Processor

200 pages of clock programming



Specification for each register:

Exynos 5250_UM — 5 Clock Controller

5.9.1.6 CLK_DIV_CPU0

- Base Address: 0x1001_0000
- Address = Base Address + 0x0500, Reset Value = 0x0000_0000

Name	Bit	Type	Description	Reset Value
RSVD	[31]	–	Reserved	0x0
ARM2_RATIO	[30:28]	RW	DIV_ARM2 clock divider Ratio ARMCLK = DOUT_ARM/(ARM2_RATIO + 1)	0x0
RSVD	[27]	–	Reserved	0x0
APLL_RATIO	[26:24]	RW	DIV_APLL clock divider Ratio SCLK_APLL = MOUT_APLL/(APLL_RATIO + 1)	0x0
RSVD	[23]	–	Reserved	0x0
PCLK_DBG _RATIO	[22:20]	RW	DIV_PCLK_DBG clock divider Ratio PCLK_DBG = ATCLK/(PCLK_DBG_RATIO + 1)	0x0
RSVD	[19]	–	Reserved	0x0
ATB_RATIO	[18:16]	RW	DIV_ATB clock divider Ratio ATCLKEN ratio => ARMCLK/(ATB_RATIO + 1)	0x0
RSVD	[15]	–	Reserved	0x0
PERIPH_RATIO	[14:12]	RW	DIV_PERIPH clock divider Ratio PERIPHCLKEN ratio => ARMCLK/(PERIPH_RATIO + 1)	0x0
RSVD	[11]	–	Reserved	0x0
ACP_RATIO	[10:8]	RW	DIV_ACP clock divider Ratio ACLKENS ratio => ARMCLK/(ACP_RATIO + 1)	0x0
RSVD	[7]	–	Reserved	0x0
CPUD_RATIO	[6:4]	RW	DIV_CPUD clock divider Ratio ACLK_CPUD = ARMCLK/(CPUD_RATIO + 1)	0x0
RSVD	[3]	–	Reserved	0x0
ARM_RATIO	[2:0]	RWX	DIV_ARM clock divider Ratio DOUT_ARM = MOUT_ARM/(ARM_RATIO + 1)	0x0

```
410 static struct clksrc_clk exynos5_clk_dout_armclk = {
411         .clk    = {
412                 .name           = "dout_armclk",
413                 .parent         = &exynos5_clk_mout_cpu.clk,
414         },
415         .reg_div = { .reg = EXYNOS5_CLKDIV_CPU0, .shift = 0, .size = 3
},
416 };
```

Clock definition in Linux

Source: http://www.samsung.com/global/business/semiconductor/file/product/Exynos_5_Dual_User_Manaul_Public_REV100-0.pdf

Power Management complexity

Example: Mobile Application Processor

Programmer's Manual – The devil is in the detail:

Caution:	It should be guaranteed that S/W does not access IPs whose clock is gated. It may cause system failure.

Caution:	It should be guaranteed that the ratio between freq (MCLK_CDREX) and freq (ACLK_CDREX) is kept as "2 to 1" all the time.

Typical software problems that can cause days, or often weeks of debugging!

Source: http://www.samsung.com/global/business/semiconductor/file/product/Exynos_5_Dual_User_Manaul_Public_REV100-0.pdf

Fighting bugs with power impact

Typical scenario – Here Linux Kernel Mailing List

```
Hey Kukjin, Andrzej,
     I recently started playing around with functionfs, and have noticed
some strange behavior with my origen board.

If I enable the FunctionFS gadget driver, I see the board hang at boot here:

[    2.360000] USB Mass Storage support registered.
```

```
LDO3 and LDO8 are used for powering both device and host phy controllers.
These regulators are not handled in USB host driver. Hence we get
unexpected behaviour when the regulators are disabled elsewhere.

It would be best to keep these regulators always on.

Signed-off-by: Tushar Behera <tushar.behera@linaro.org>
```

```
So your patch worked great for me! Thanks for the analysis and the patch!
```

Source: Feb. 2013, https://lkml.org/lkml/2013/2/26/608

Fighting bugs with power impact

Typical scenario – Here Linux Kernel Mailing List

> It would be best to keep these regulators always on.
No, it's just workaround patch.

It should be handled at USB drivers.
we usually used this scheme enable USB power always. but it consumes lots of power.
There's no need to enable usb power when there's no usb connection.

So I suggest to enable power when usb is connected only.

In our case, micro IC detects the usb connection and enable usb power at that time.

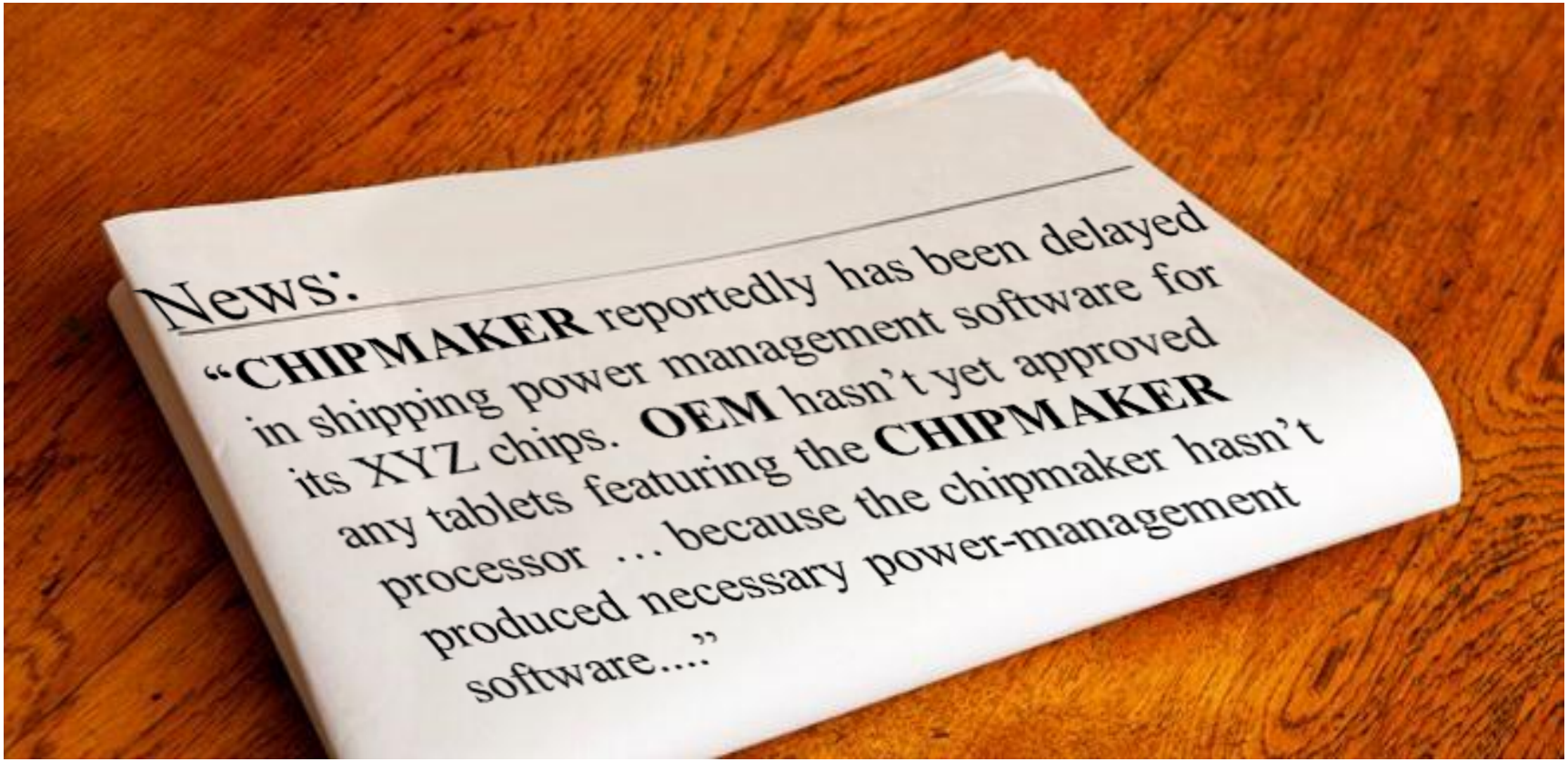
Thank you,
Kyungmin Park

Works fine, but

Source: Feb. 2013, https://lkml.org/lkml/2013/2/26/608

Shift-Left with Virtual Prototyping

Start earlier – Late Power Management software can have catastrophic consequences on project schedules



Virtual Prototyping approach

Goals

- Enable most critical software development tasks
- As early as possible
- Aligned with software project schedule

Needs

- Develop and deploy virtual prototypes (VP) incrementally
- Enable different software teams to develop in parallel
- Multiple VPs are created with different focus

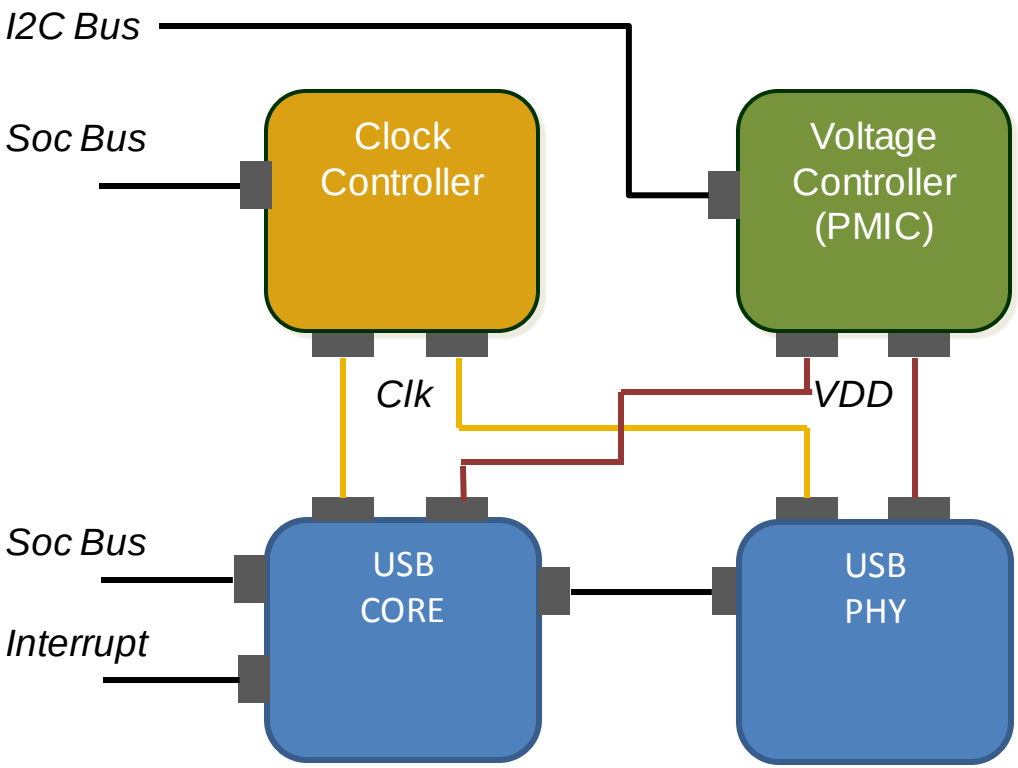
How

- Model subsystems to support most critical software tasks
- Leverage existing or generic models for simulation of subsystems outside the software development focus

Its here not a goal of the virtual prototype to represent all the hardware to develop all the software (availability would be too late to make any impact)

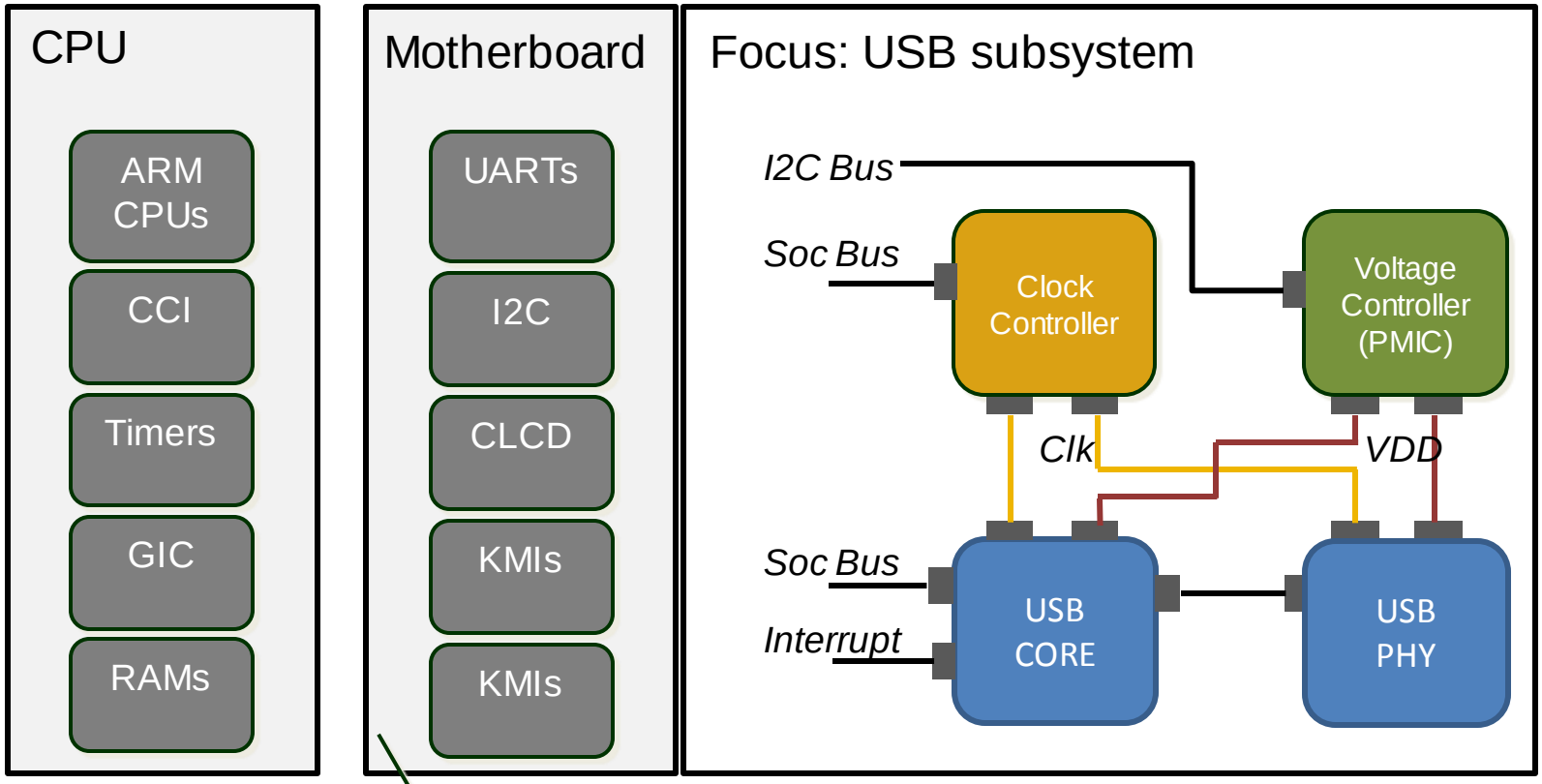
Case Study: SoC Power Management

USB subsystem with our specific PMIC and Clock Controller



Case Study: SoC Power Management

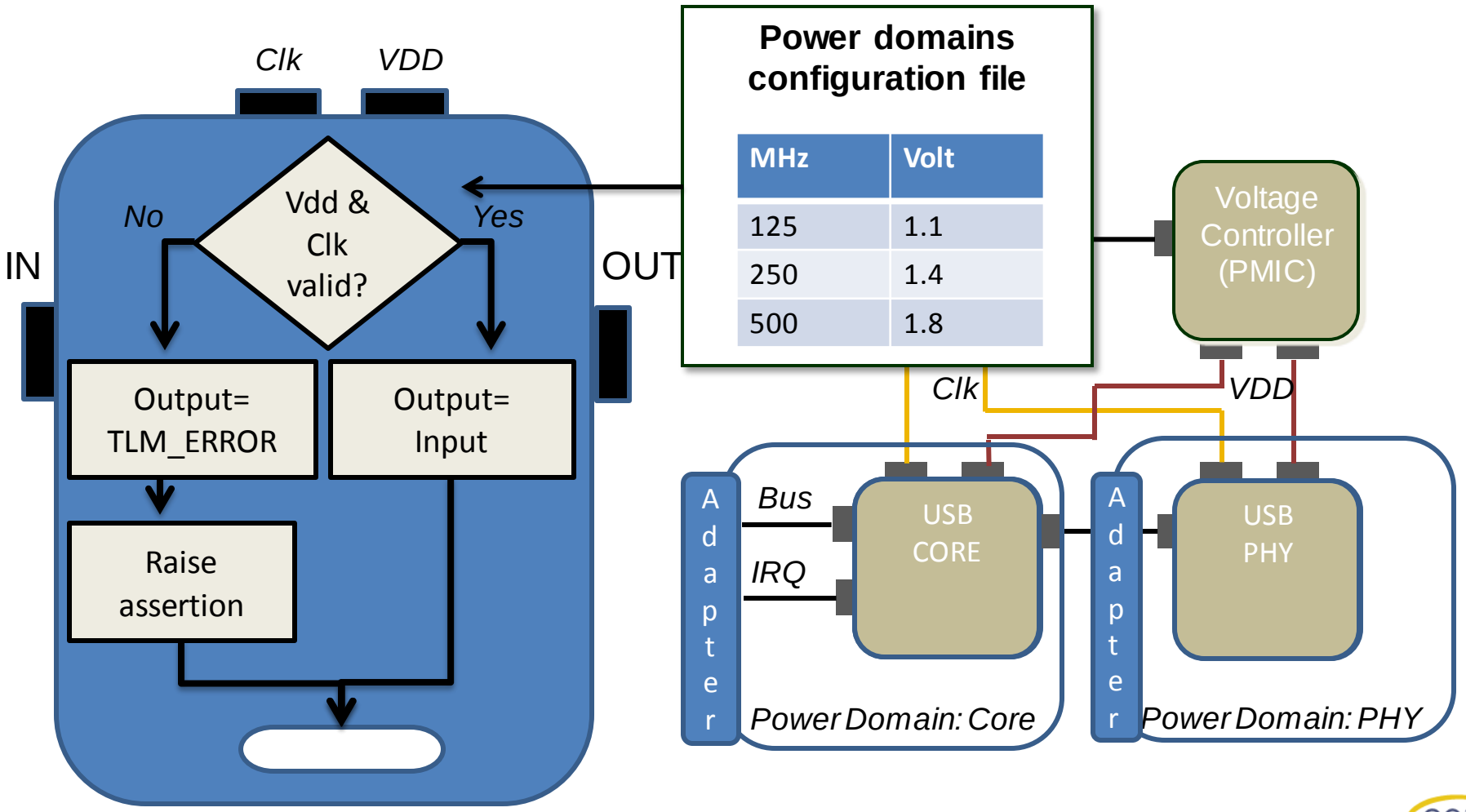
Combine with available VDK for ARM Versatile Express and software



Readily available pre-assembled VP building blocks.
Part of VDK for the ARM Versatile Express prototyping system
Allows running stock Linaro Linux kernel and filesystem images

Case Study: SoC Power Management

Add power domain information for the USB subsystem



What we just learned

- The scope of a Virtual Prototype need to match the requirements of the software team
  - What is needed to enable the key critical software tasks?
  - A mix of generic and specific HW components
  - Assembly & modeling tools assist the specific HW model creation
  - Enables earliest availability
- A VP can accurately model power management hardware
  - Clock & voltage trees
  - Power Managament IC (PMIC)
  - Clock controller
  - Power domain isolation

Power-aware Software Development

Top three goals for today

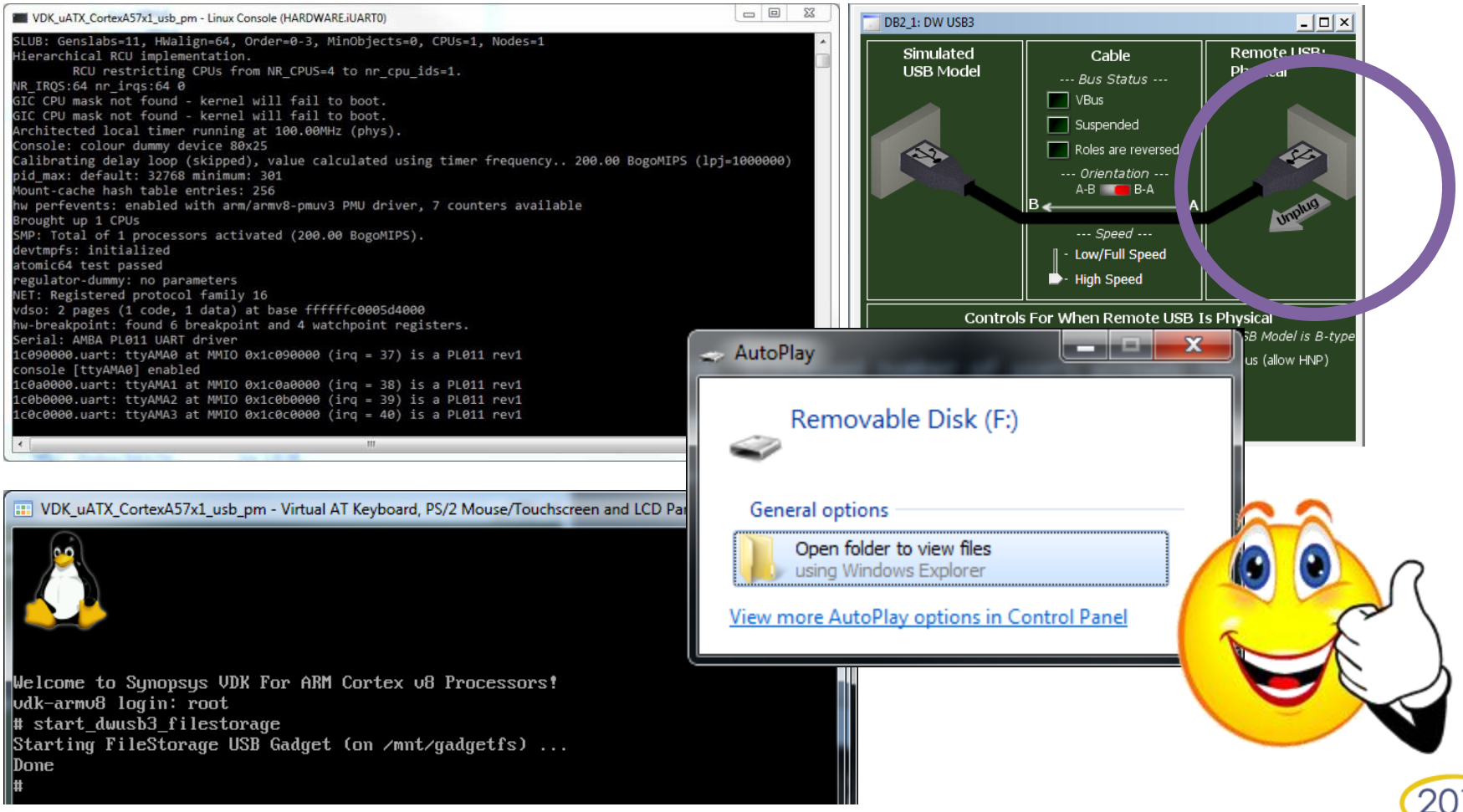
Reduce risk of late power management software development

Improve robustness of power management software

Reveal software bugs that can drain the battery

Case Study: Normal OS & driver operation

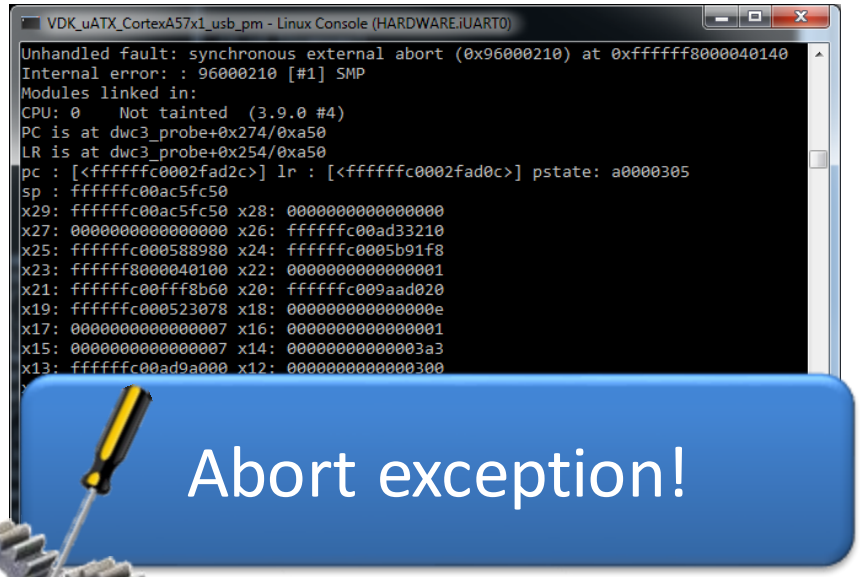
Booting and using USB for a file storage gadget



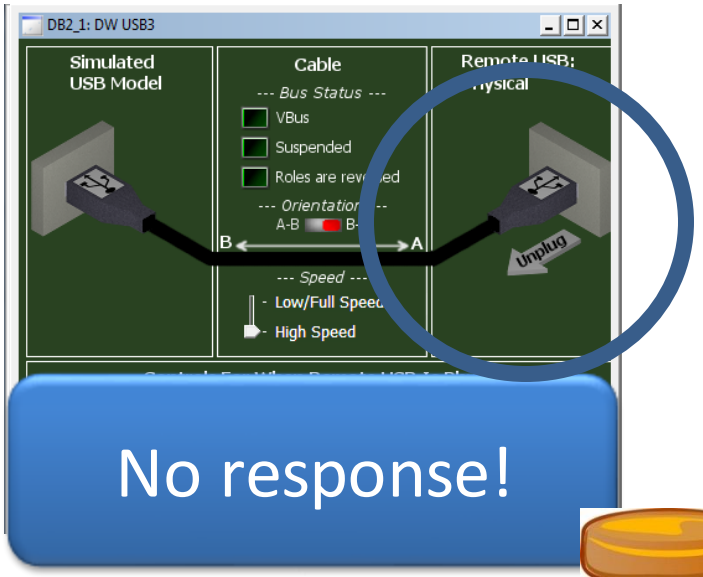
Case Study: Driver faults from power bug

Booting and using USB for a file storage gadget

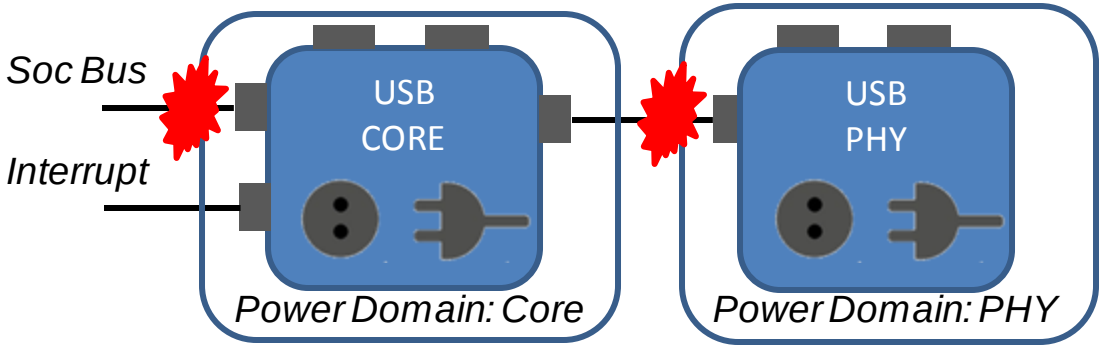
- Unpowered core
- Unpowered PHY



Abort exception!



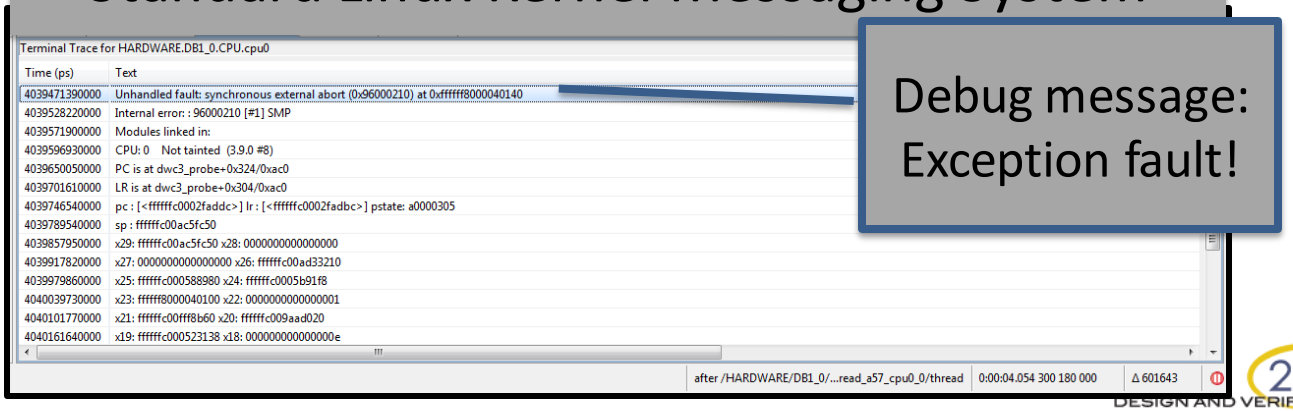
No response!



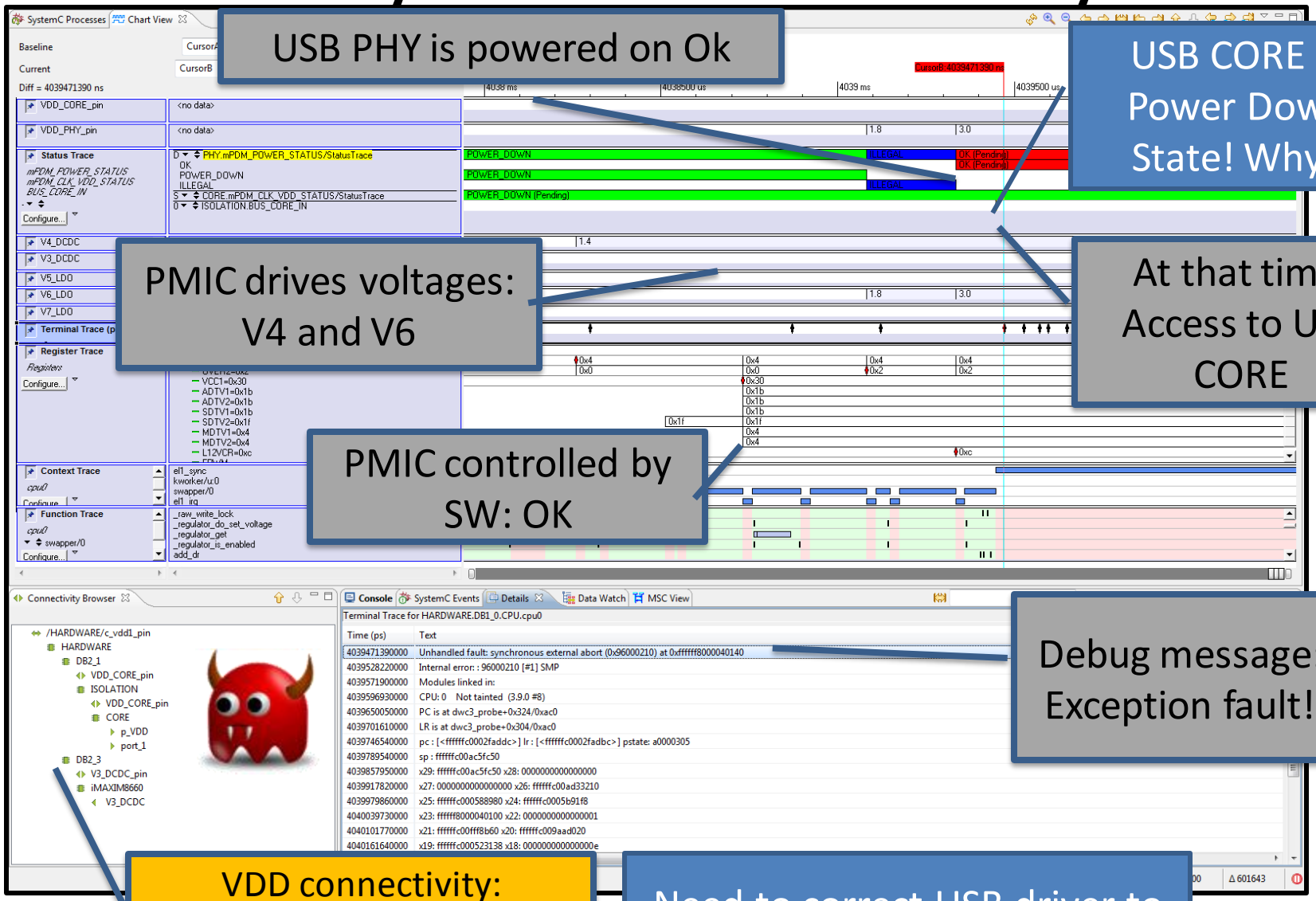
Case Study: Root cause analysis

Using a VDK,
there is more to see!

Standard Linux Kernel Messaging System



Case Study: Root cause analysis



What we just learned

- Virtual prototypes do accurately simulate power management fault scenarios
  - Software developer can reproduce and observe same defects like on hardware
  - Deterministic repetition for debug and testing
- Virtual prototypes help accelerating root cause analysis
  - Visibility and traceability of any HW or SW property
  - Cross correlation of HW and SW power management

Power-aware Software Development

Top three goals for today

Reduce risk of late power management software development

Improve robustness of power management software

Reveal software bugs that can drain the battery

Software bugs impacting power consumption – the reality today

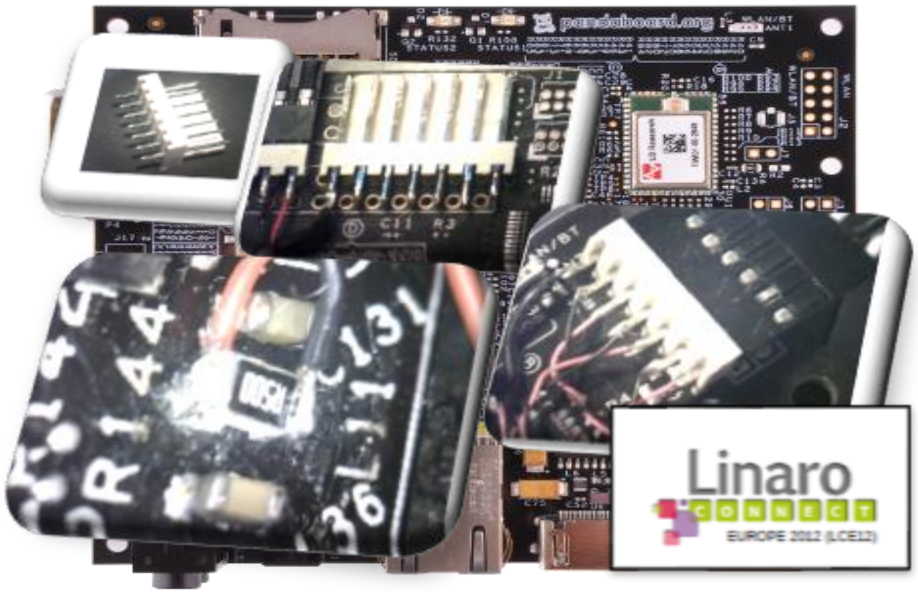
Small software bugs can have big impact on power	• Invisible to the developer • Only revealed in long term scenario measurements
Impact is use-case dependent	• Talk time might not be impacted at all • Standby time might be reduced by multiple hours
Software developers are often unaware	• No means to test and use-case analyze during development • Power bugs continuously slip into production firmware

How to analyze power bugs?

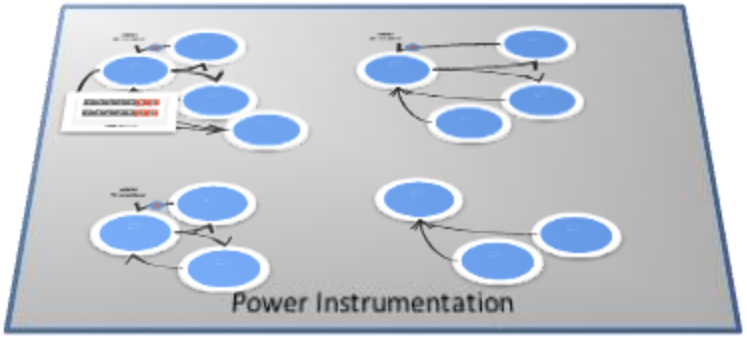
Soldering skills required…

The hardware way…

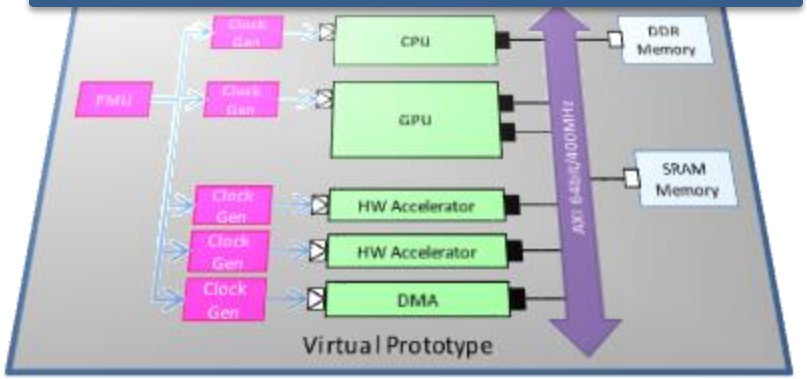
- Use a 7-way 0.1” header for each probe (3 channels)
- Superglue the back to a spare region of the board
- Add 0V connection
- Add twisted pairs back to the shunt
- Remove old inductor, solder the shunt in place
- Add the other end of the wires to the shunt



The Virtualizer way…



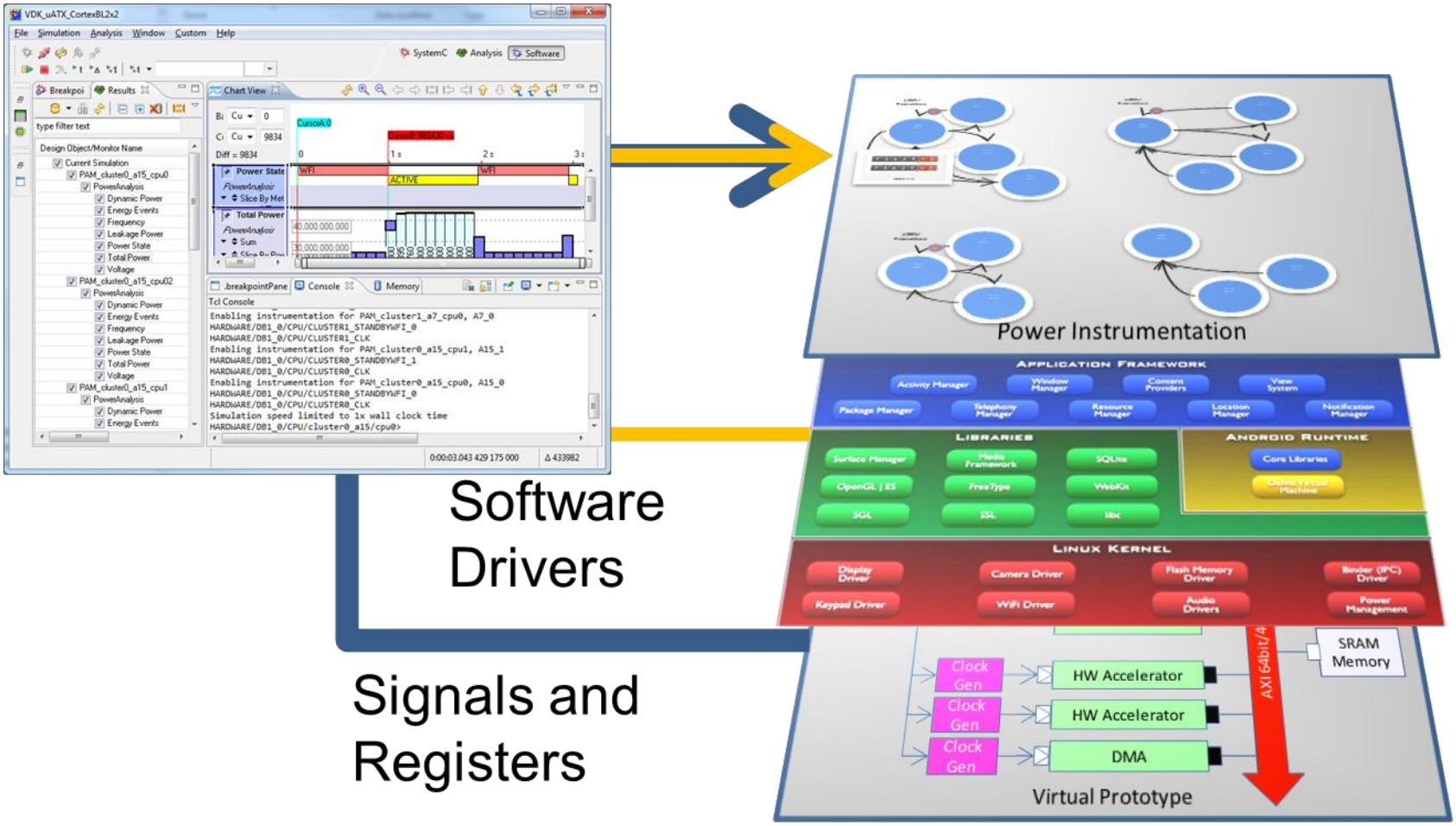
Dynamic power analysis
Instrumentation overlay



Source: How to measure SoC power, Andy Green, TI Landing Team lead, Linaro

Dynamic power analysis

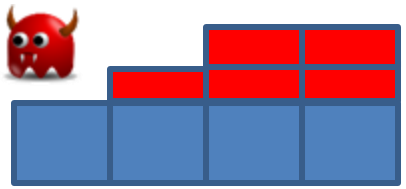
Using instrumentation scripts in Virtualizer VDK



USB Power Model

Abstract power model

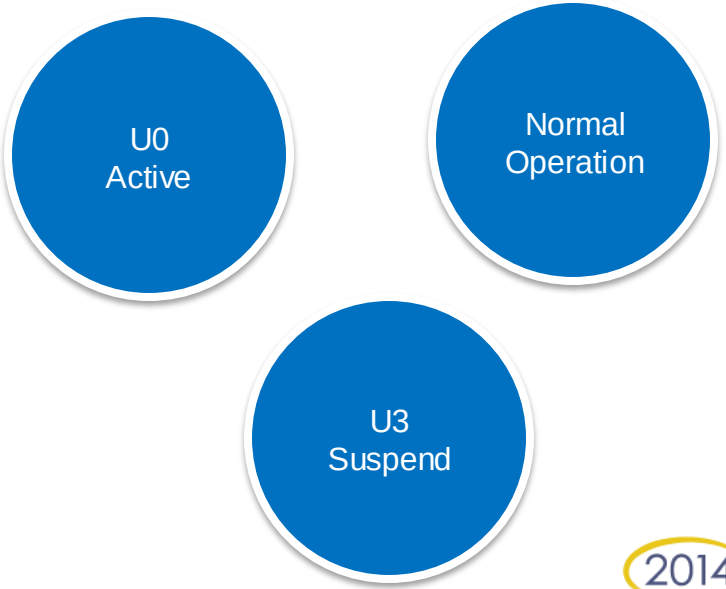
- Approximate power per component at a reference frequency & voltage
- Good enough to find bug mentioned in introduction!



From K Park:
> **It would be best to keep these regulators always on.**
No, it's just workaround patch. It should be handled at USB drivers. We usually used this scheme enable USB power always. but **it consumes lots of power.** There's no need to enable usb power when there's no usb connection.

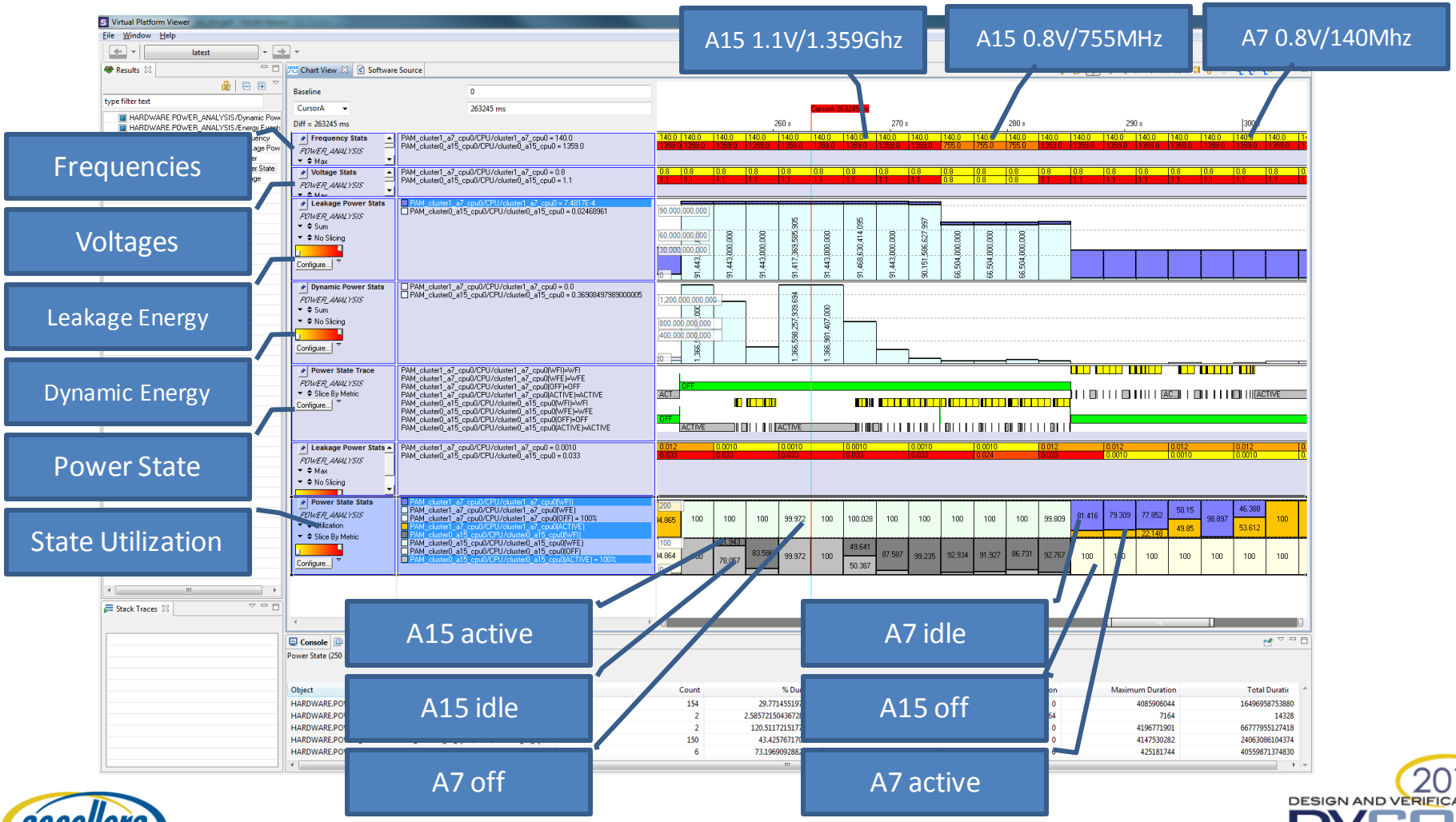
Detailed power model

- Approximate power for each power mode in a component
- Needed for run-time power management SW



Dynamic power analysis

big.LITTLE processing – Task migration with DVFS



What we just learned

- Power awareness has higher priority than power accuracy for software developers
  - Even a simple on/off power model can reveal severe defects
  - Power models can be realized at multiple levels of abstraction
- Power analysis can be added as an overly to a virtual prototype
  - Less intrusive than cutting rails and soldering shunts
  - Accuracy in the same ballpark as HW based measurement

Power-aware Software Development

Top three goals for today

Reduce risk of later power management software

- Complete software development earlier with Virtual prototypes
- Simulate power domain control (PMIC and clock controller)

Improve robustness of power management software

- Simulate fault scenarios such as unpowered hardware
- Efficient root cause analysis from HW though SW stack

Reveal software bugs that can drain the battery

- Simulate approximate power consumption
- Expose power consumption defects to the SW developer

Questions?

Thank You