

Attack Your SoC Power Challenges with Virtual Prototyping

Stefan Thiel

Gunnar Braun

SYNOPSYS®
Accelerating Innovation



Agenda

Part #1: Power-aware Architecture Definition

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

Power-aware Architecture Definition

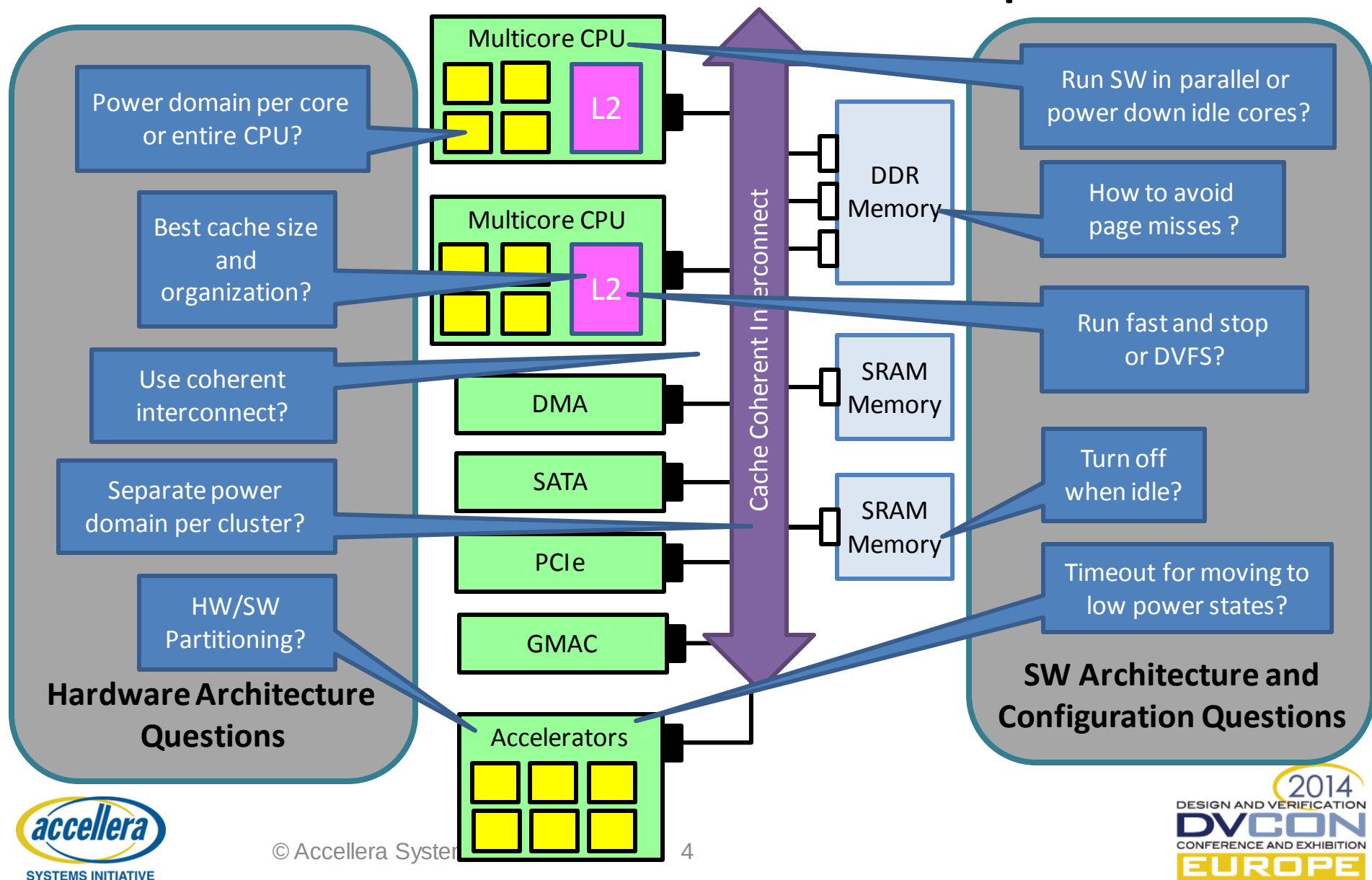
Top three goals for today

Reduce risk of wrong design decision due to late power analysis

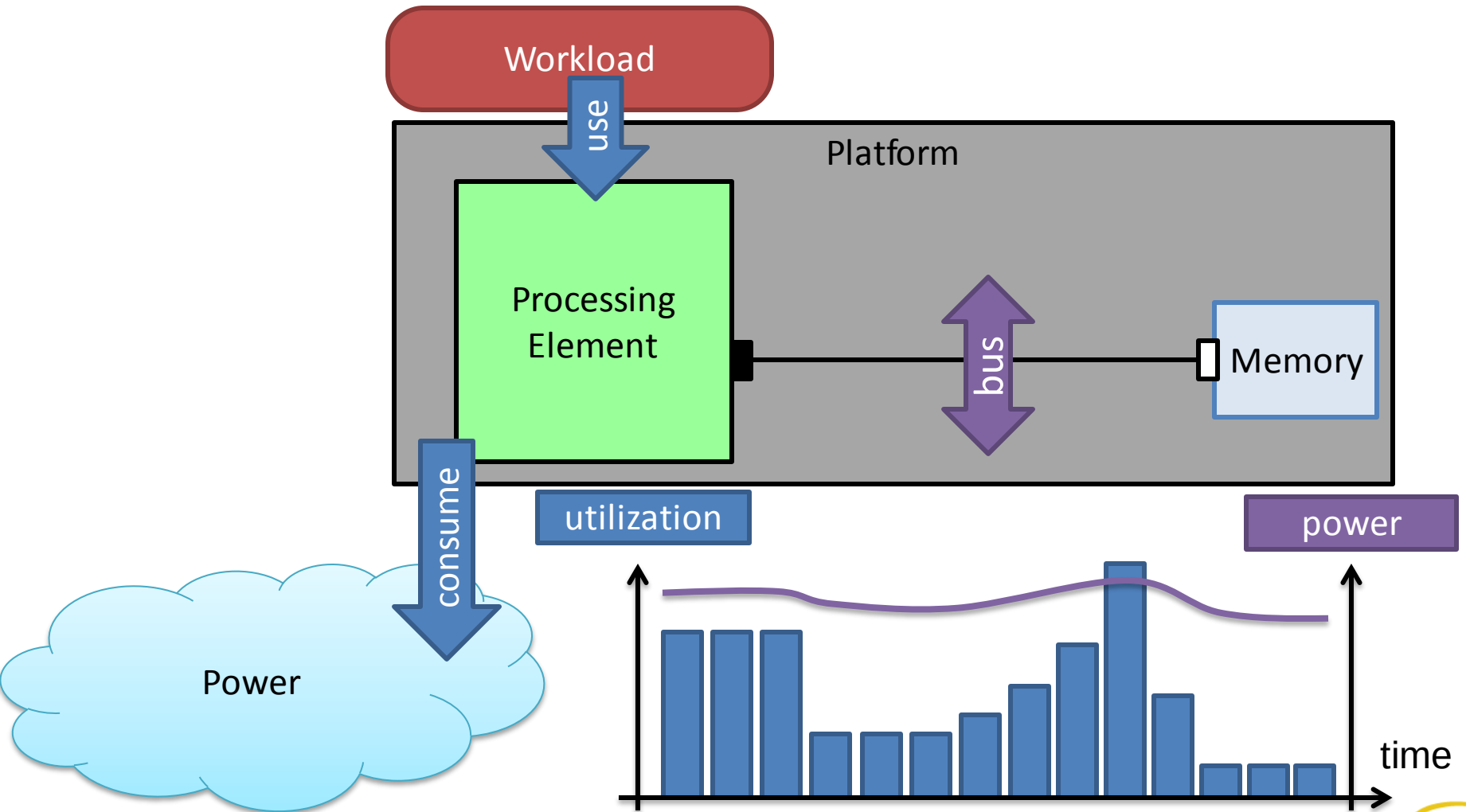
Define the right HW/SW architecture to meet power and performance goals

Define the right Power Management strategy

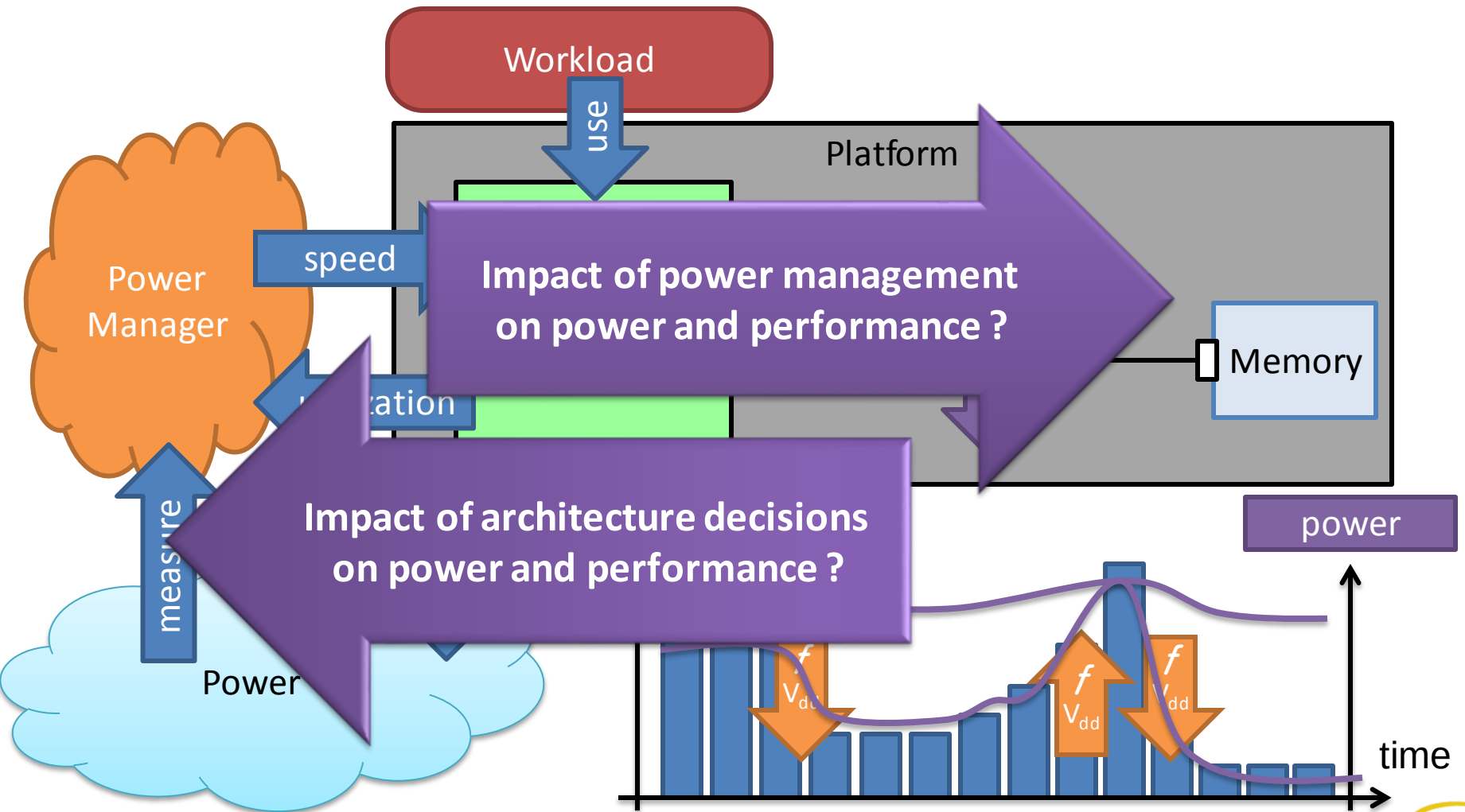
How does architecture affect power?



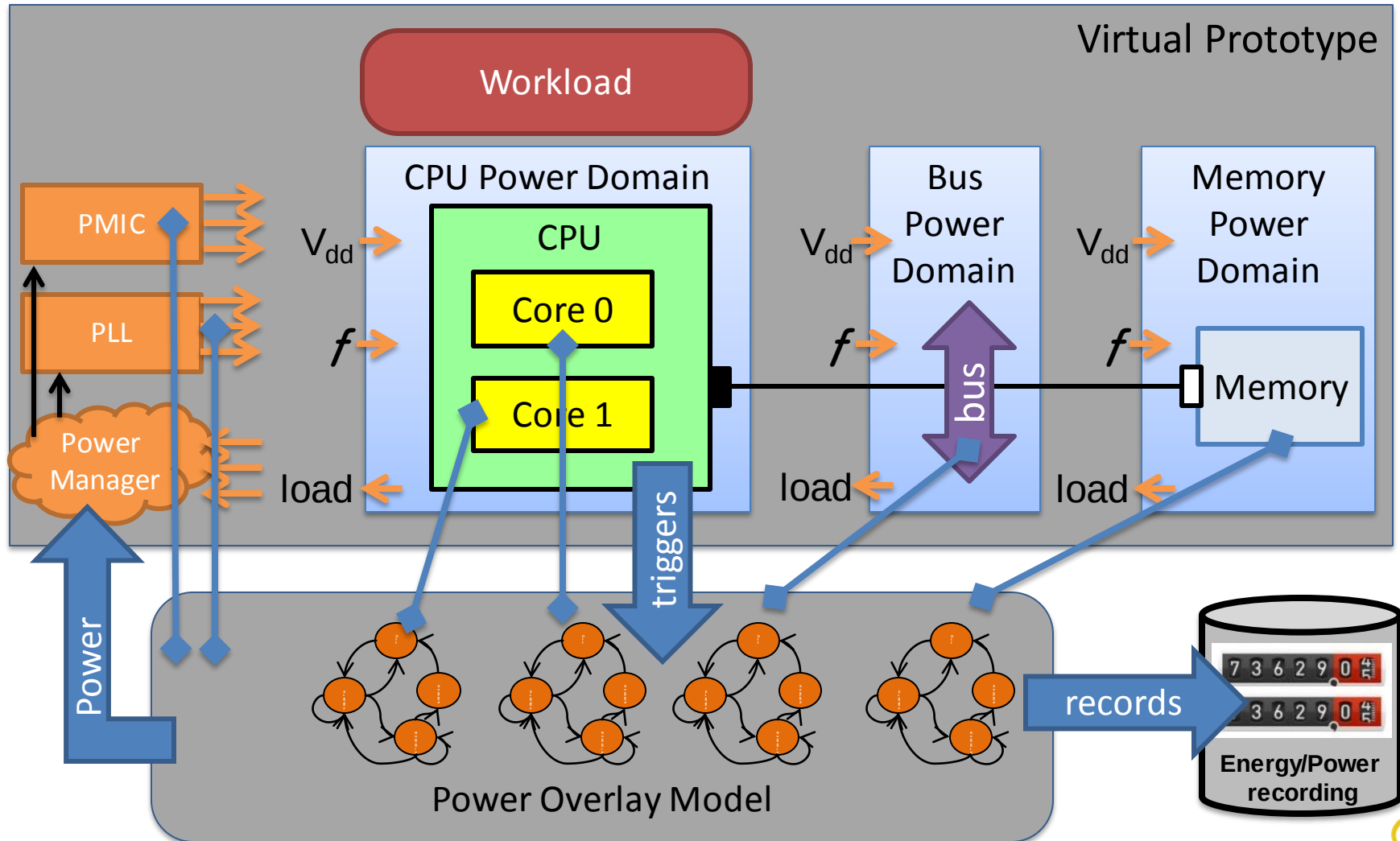
Power and Performance Duality



Power and Performance Duality



Virtual Prototyping Approach



The figure displays system metrics over time, organized into four main sections:

- Top Section:**
 - MCO Stats:** Shows 'task_manager' utilization (2.9375%) and a bar chart of values (e.g., 260.875, 400, 343.188, etc.).
 - Frequency Trace:** Shows 'Frequency=0.0' as a line graph.
- Middle Section:**
 - Total Power Stats:** A stacked bar chart showing power consumption for 'CORE0', 'CORE1', 'CORE2', 'CORE3', 'CPU0_CACHE', and 'CPU0_CACHE'.
- Bottom Section:**
 - MCO Stats:** Similar to the top section, showing 'task_manager' utilization and a bar chart of values (e.g., 254.917, 400, 298.312, etc.).
 - Frequency Trace:** Shows 'Frequency=0.0' as a line graph.

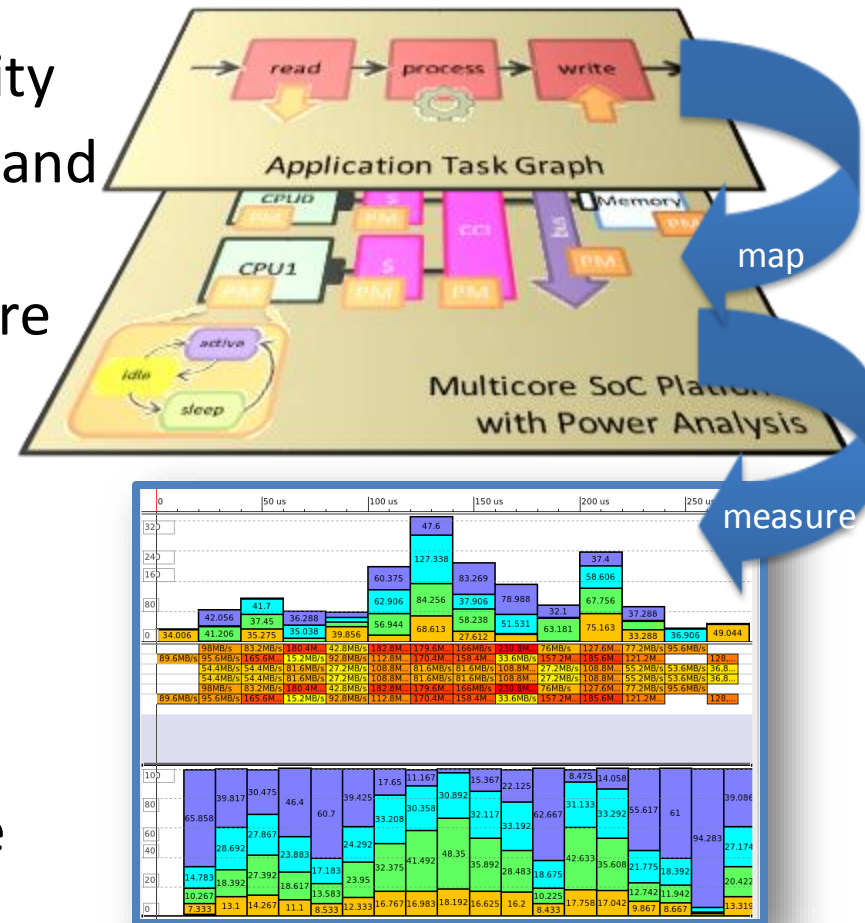
The x-axis represents time in microseconds (us), with markers at 0, 50, 100, 150, and 200. The y-axis for the top and bottom panels ranges from 0 to 400. The y-axis for the middle panel ranges from 0 to 15,000,000. The y-axis for the bottom panel's frequency trace ranges from 0 to 1,800. The bottom panel's power stats y-axis ranges from 0 to 15,000,000. The bottom panel's frequency trace y-axis ranges from 0 to 1,800. The bottom panel's power stats y-axis ranges from 0 to 15,000,000.

Power stats

Fine-grain DVFS

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

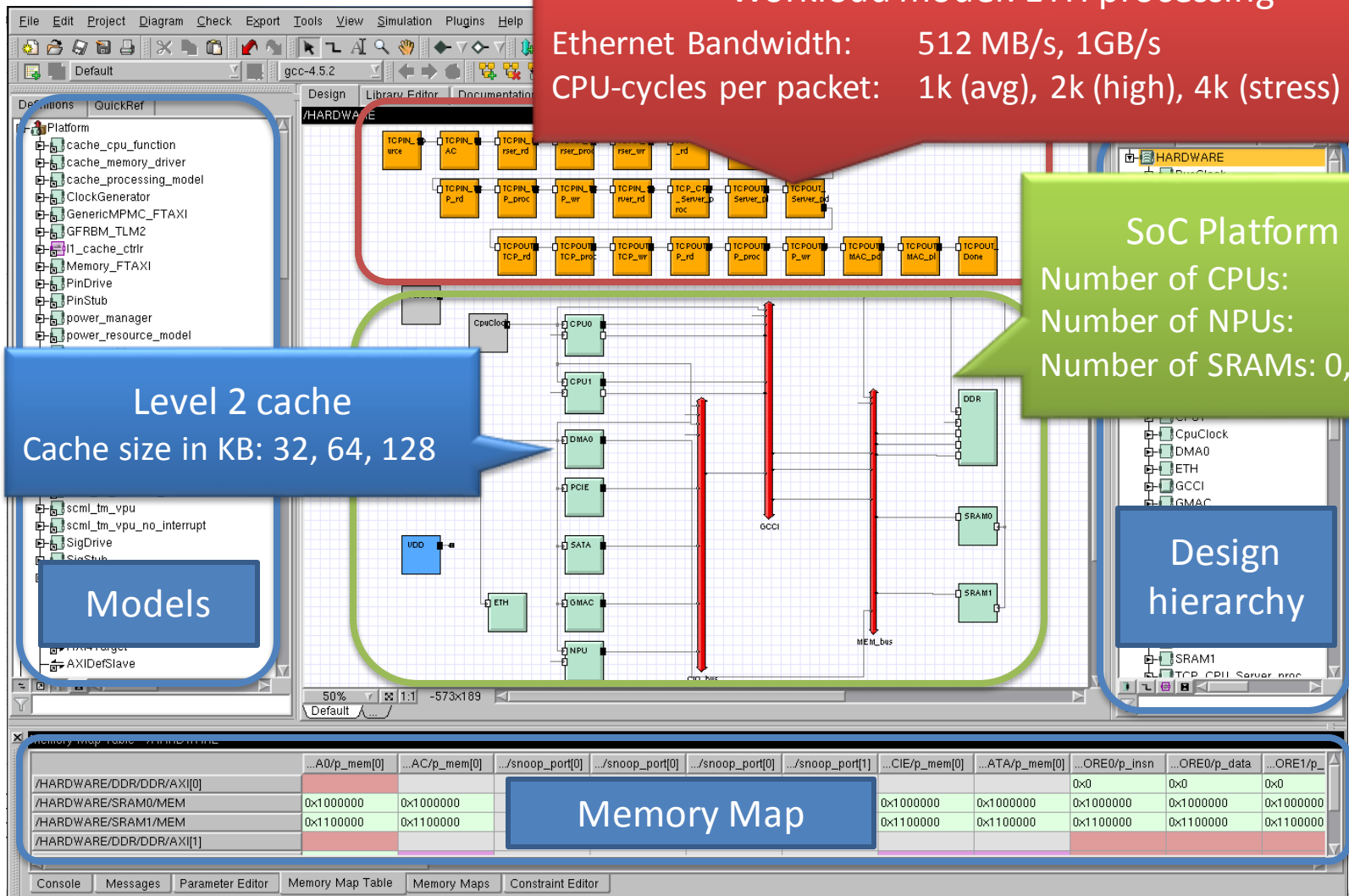
Reduce risk of wrong design decision due to late power analysis

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

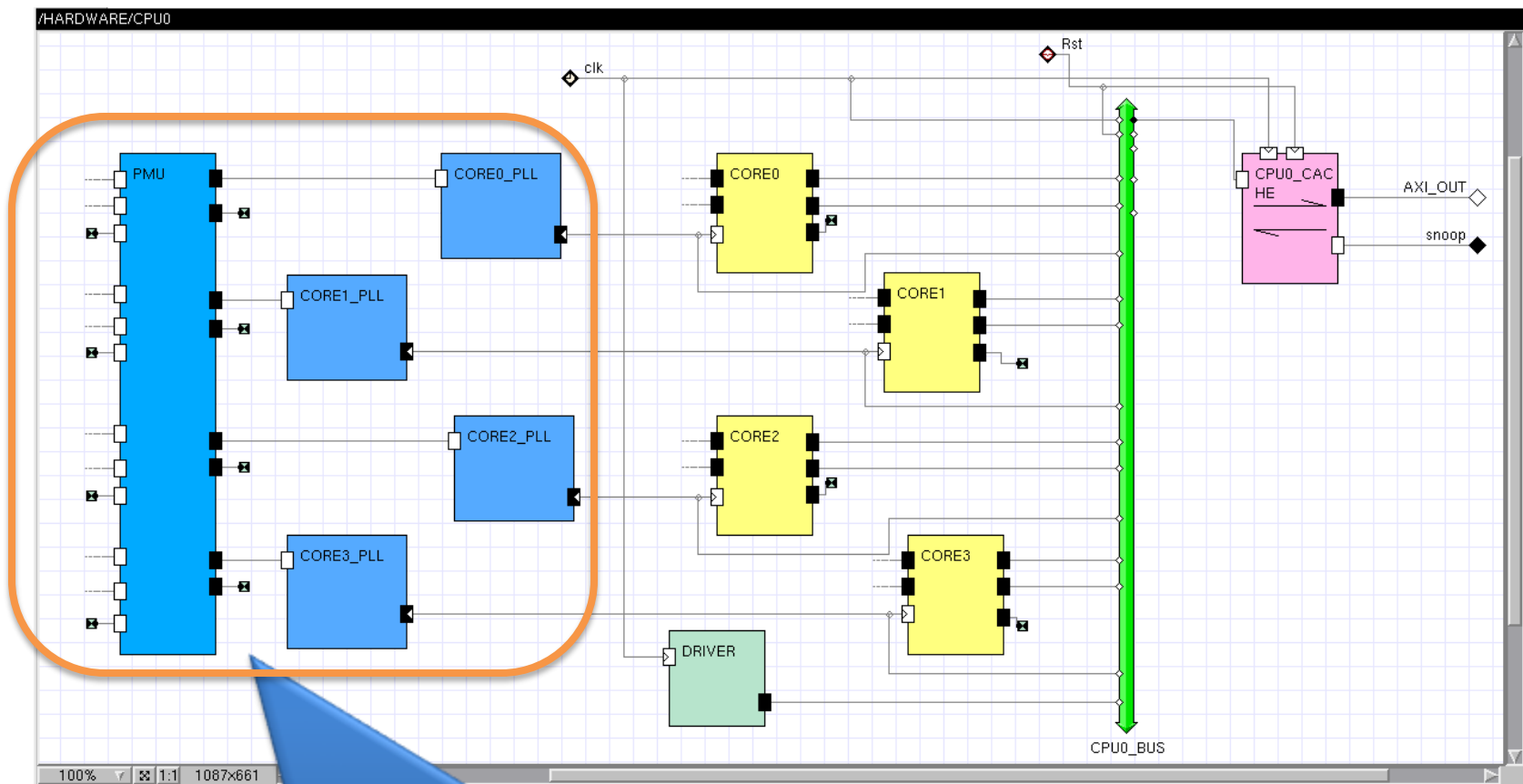
Define the right Power Management strategy

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
```

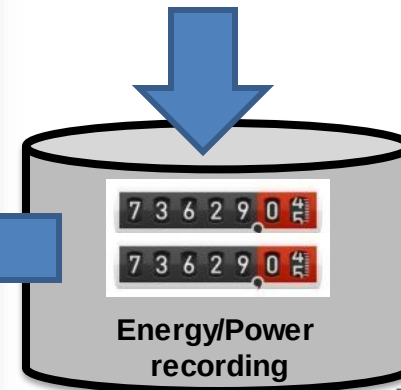
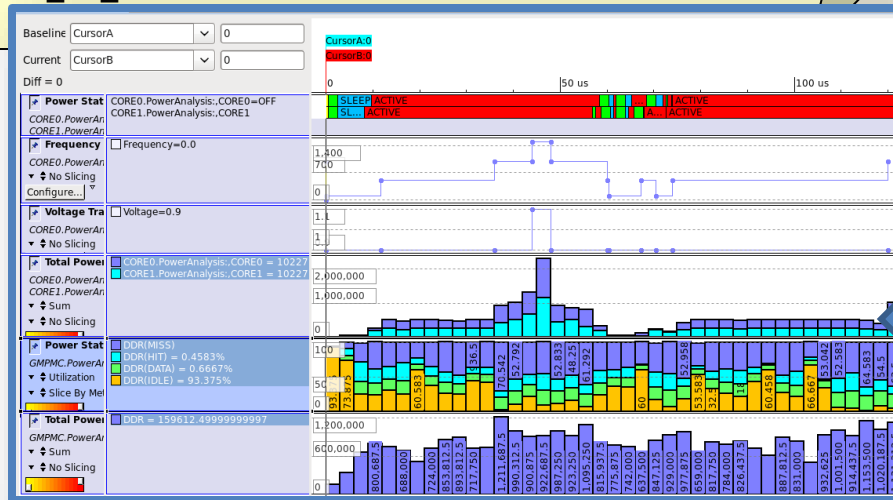
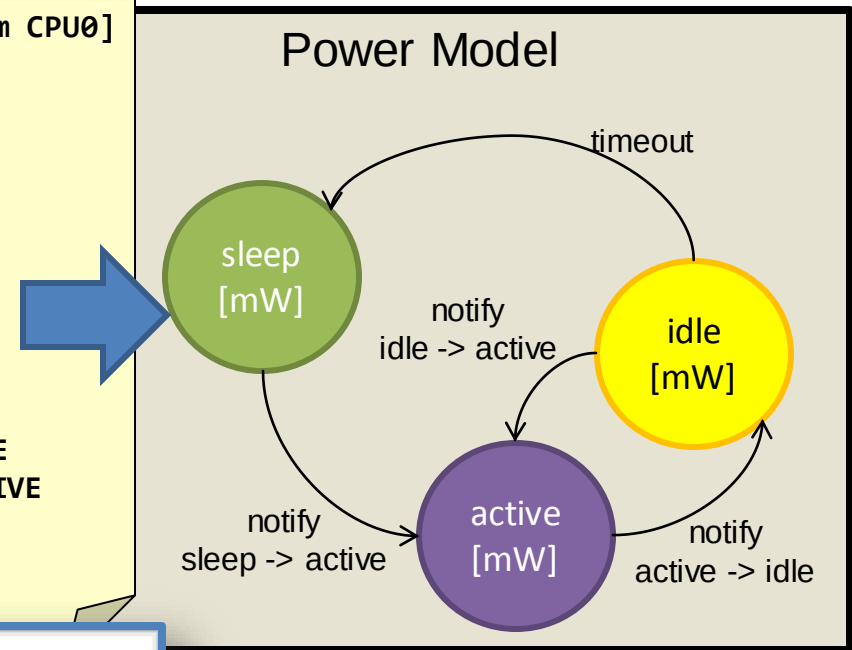
```
$pm link_signal_to_state CORE0 CPU.CORE0.p_notify 1 ACTIVE
```

```
...
```

```
# 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 $P_{\text{dyn,ref}}$
 - for a reference frequency $F_{\text{dyn,ref}}$ and reference voltage $V_{\text{dyn,ref}}$
 - Reference leakage power $P_{\text{leak,ref}}$
 - for a reference voltage $V_{\text{leak,ref}}$
 - 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
```

$F_{\text{dyn,ref}}$

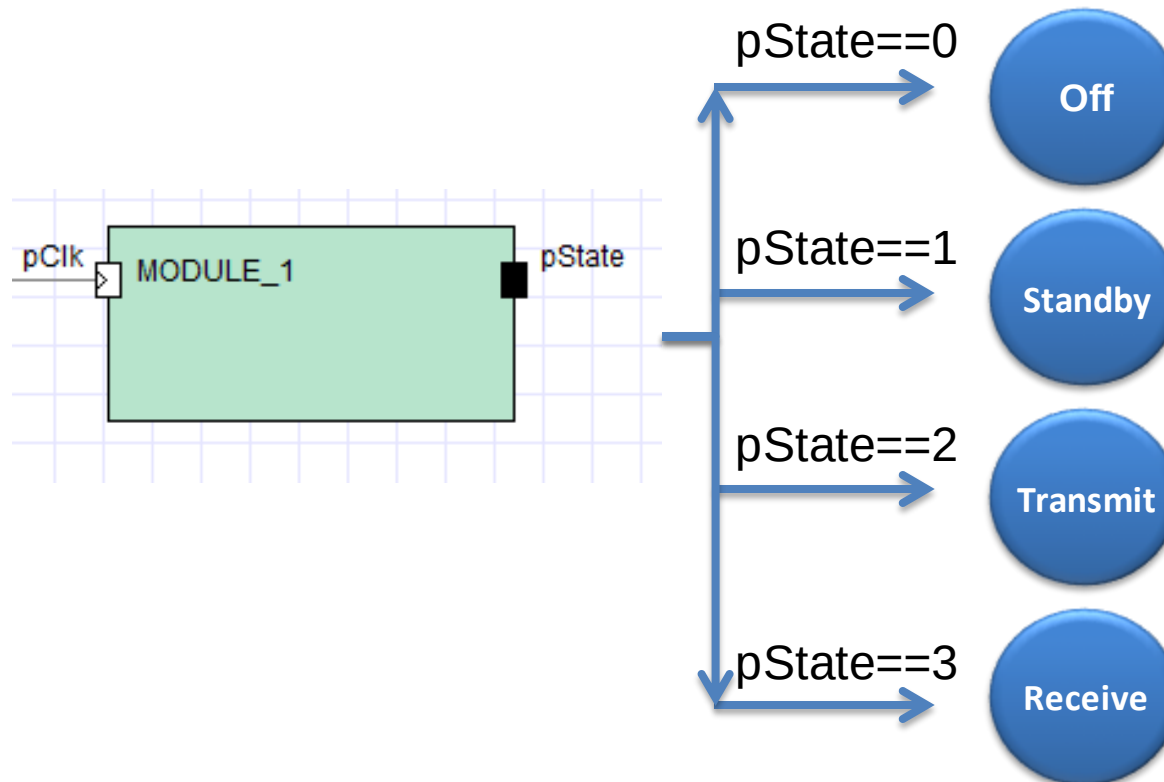
$V_{\text{dyn,ref}}$

$V_{\text{leak,ref}}$

Power Model Instrumentation - Details

Driving Power States from a (HW) signal

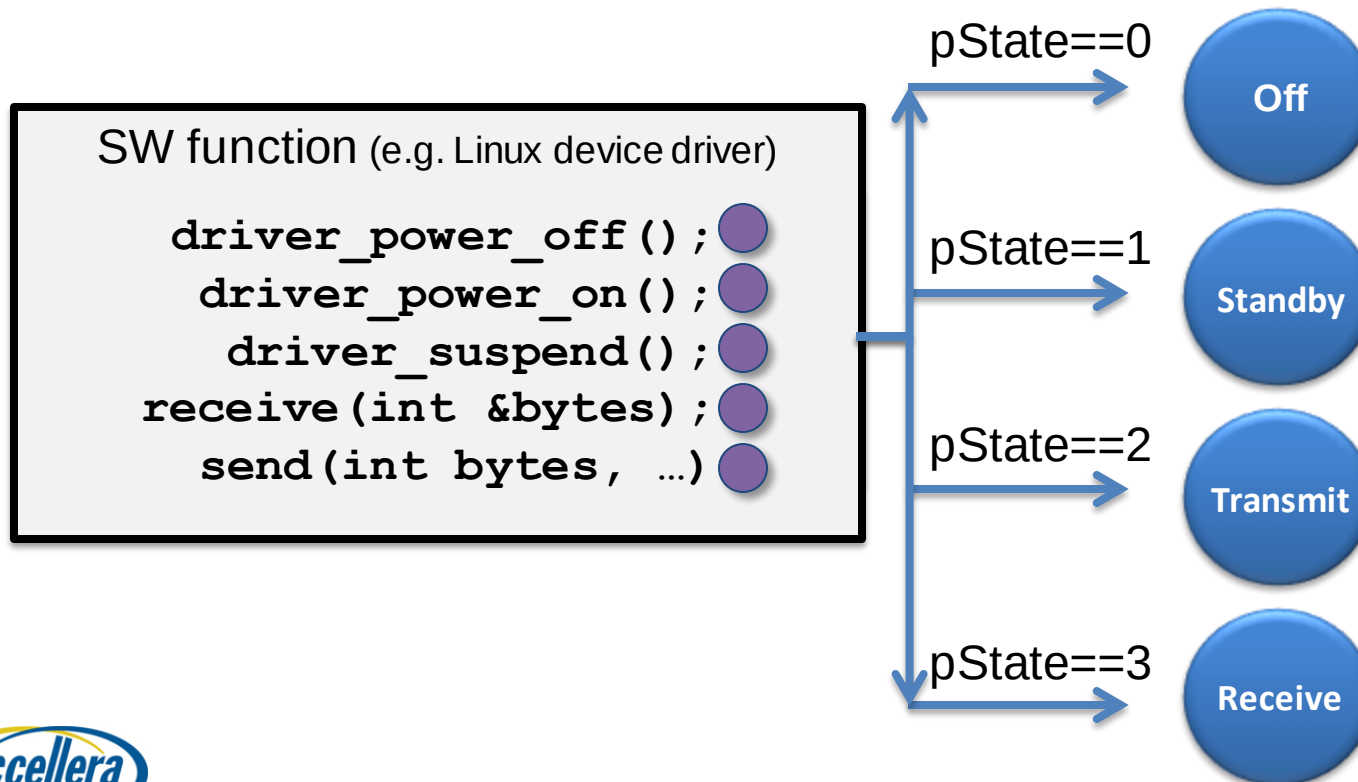
```
$pm link_signal_to_state TOP/MODULE_1/pState 0 0xf modem_fsm Off
$pm link_signal_to_state TOP/MODULE_1/pState 1 0xf modem_fsm Standby
$pm link_signal_to_state TOP/MODULE_1/pState 2 0xf modem_fsm Transmit
$pm link_signal_to_state TOP/MODULE_1/pState 3 0xf modem_fsm Receive
```



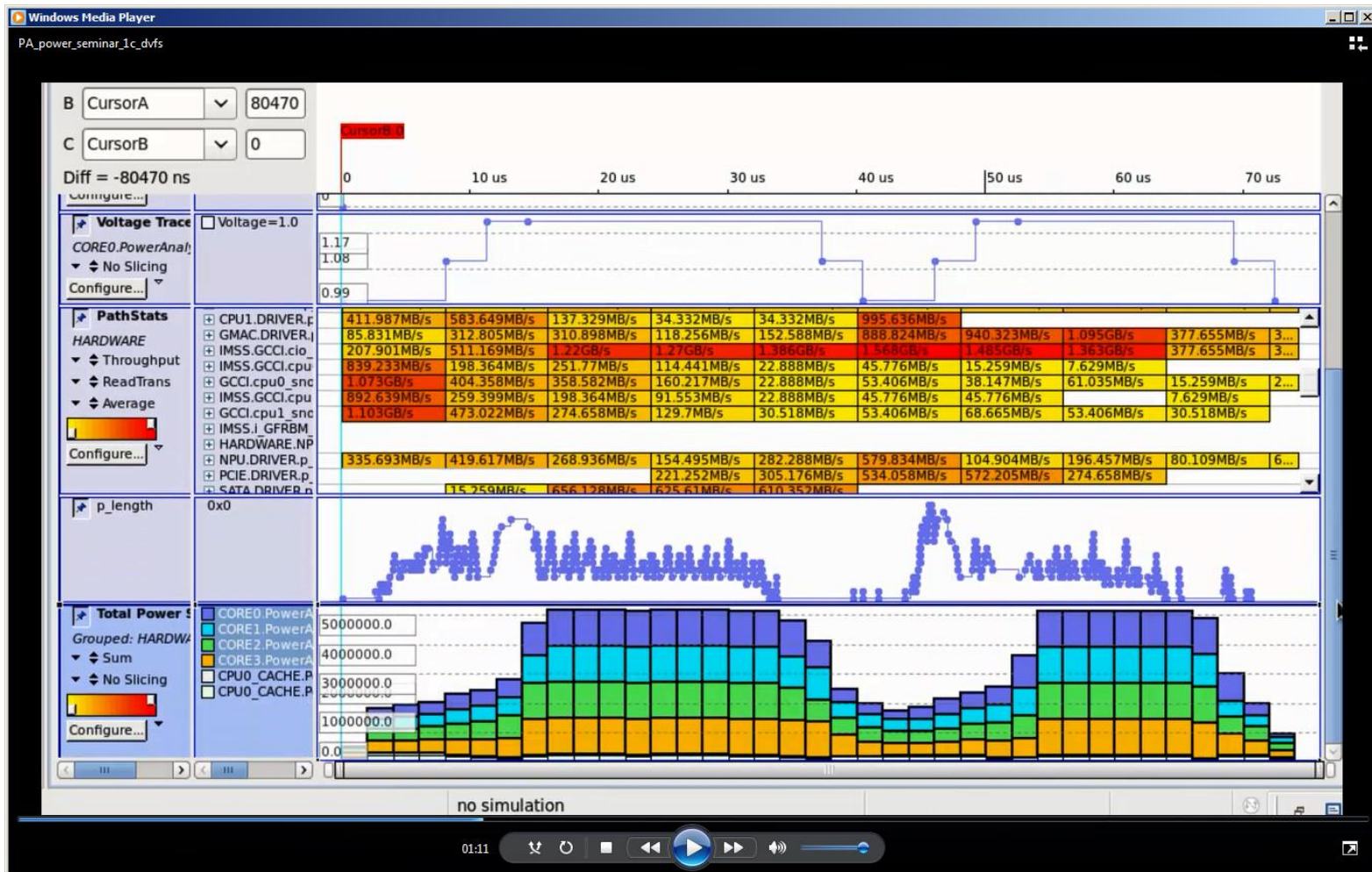
Power Model Instrumentation - Details

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

parameters

scenarios

2. Simulation sweep

3. Metric Extraction

parameters

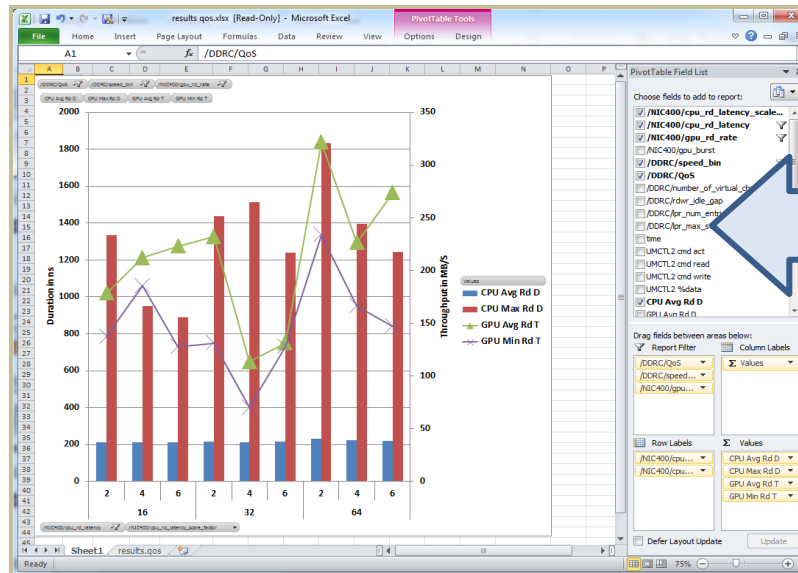
metrics

results

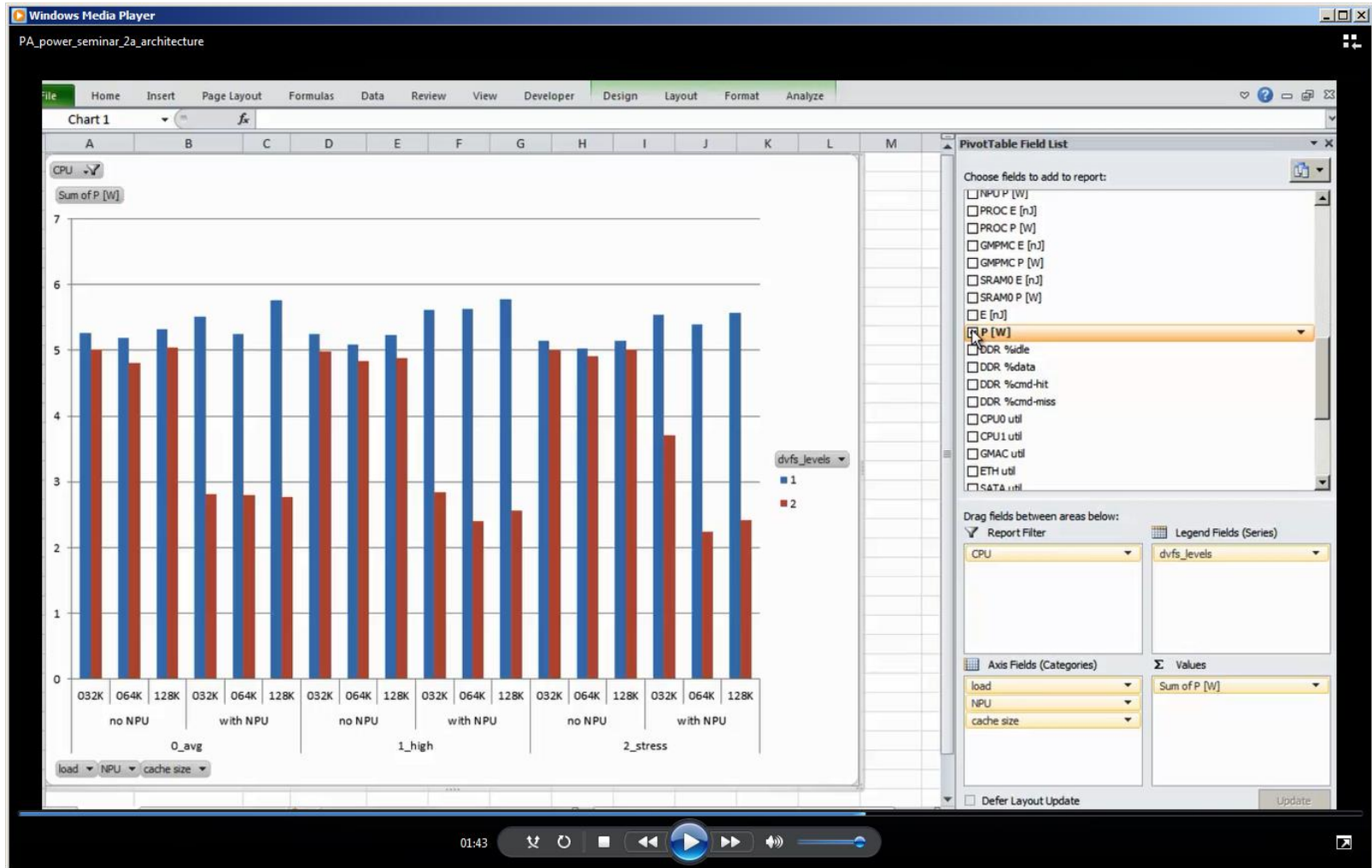
Spreadsheet In

4. Sensitivity Analysis

Spreadsheet Out



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

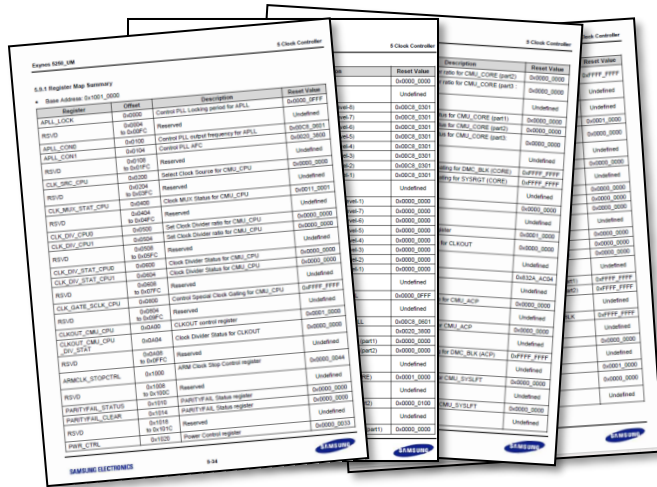
- Control clocks and voltages
- Example: set voltage regulator to 1.1V, set clock to 1GHz
- Initiated by operating system power management

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 ACLKEN 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_Manual_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_Manual_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.
```

LD03 and LD08 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
```



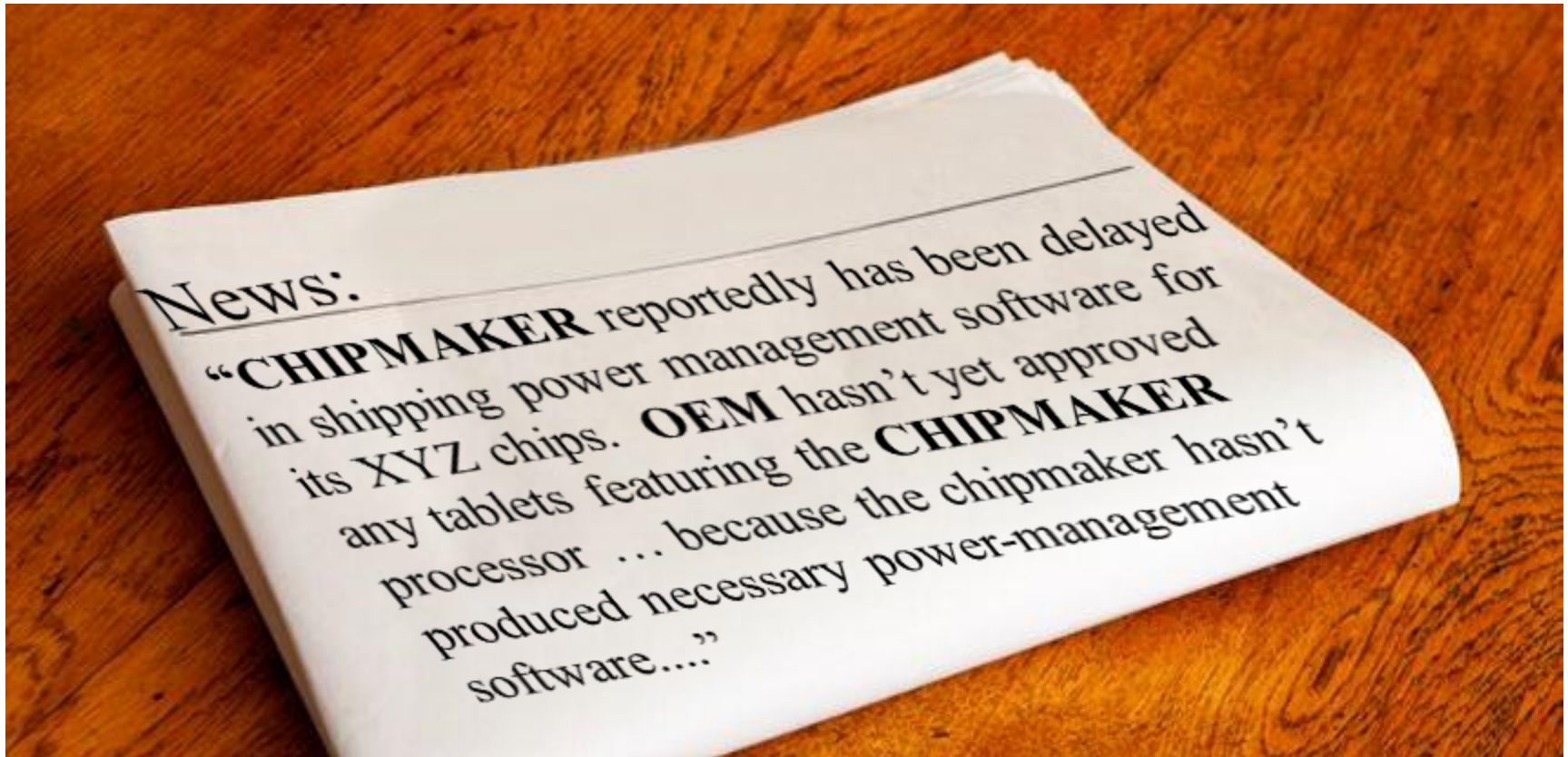
Works fine, but



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

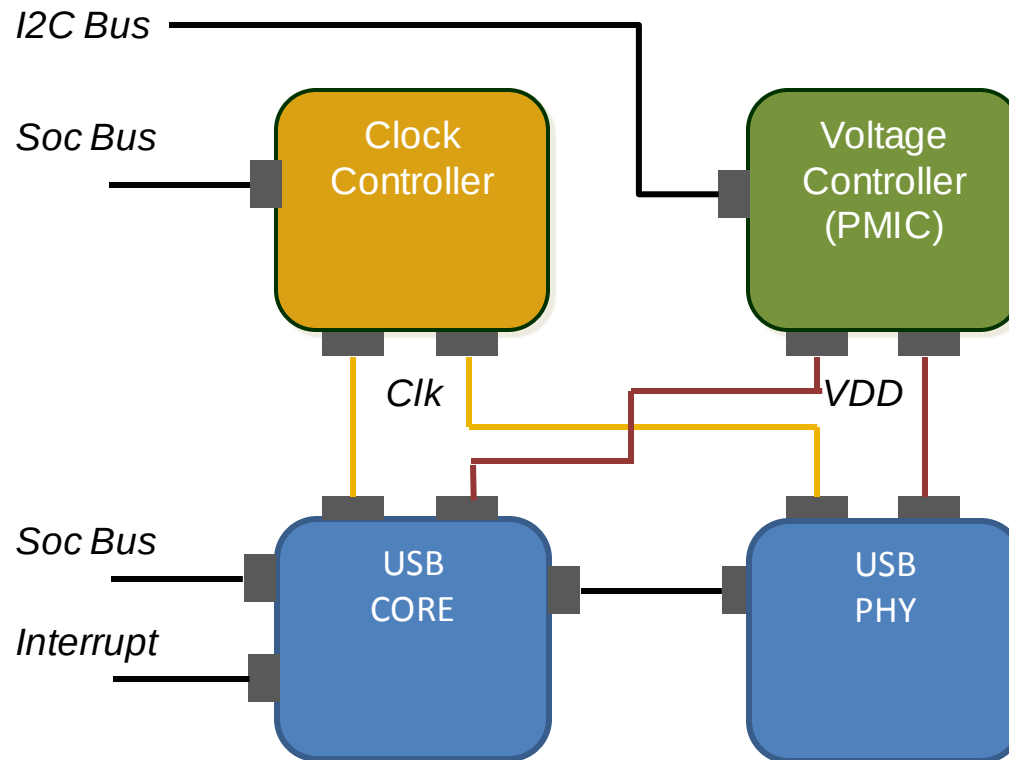
How

- 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)

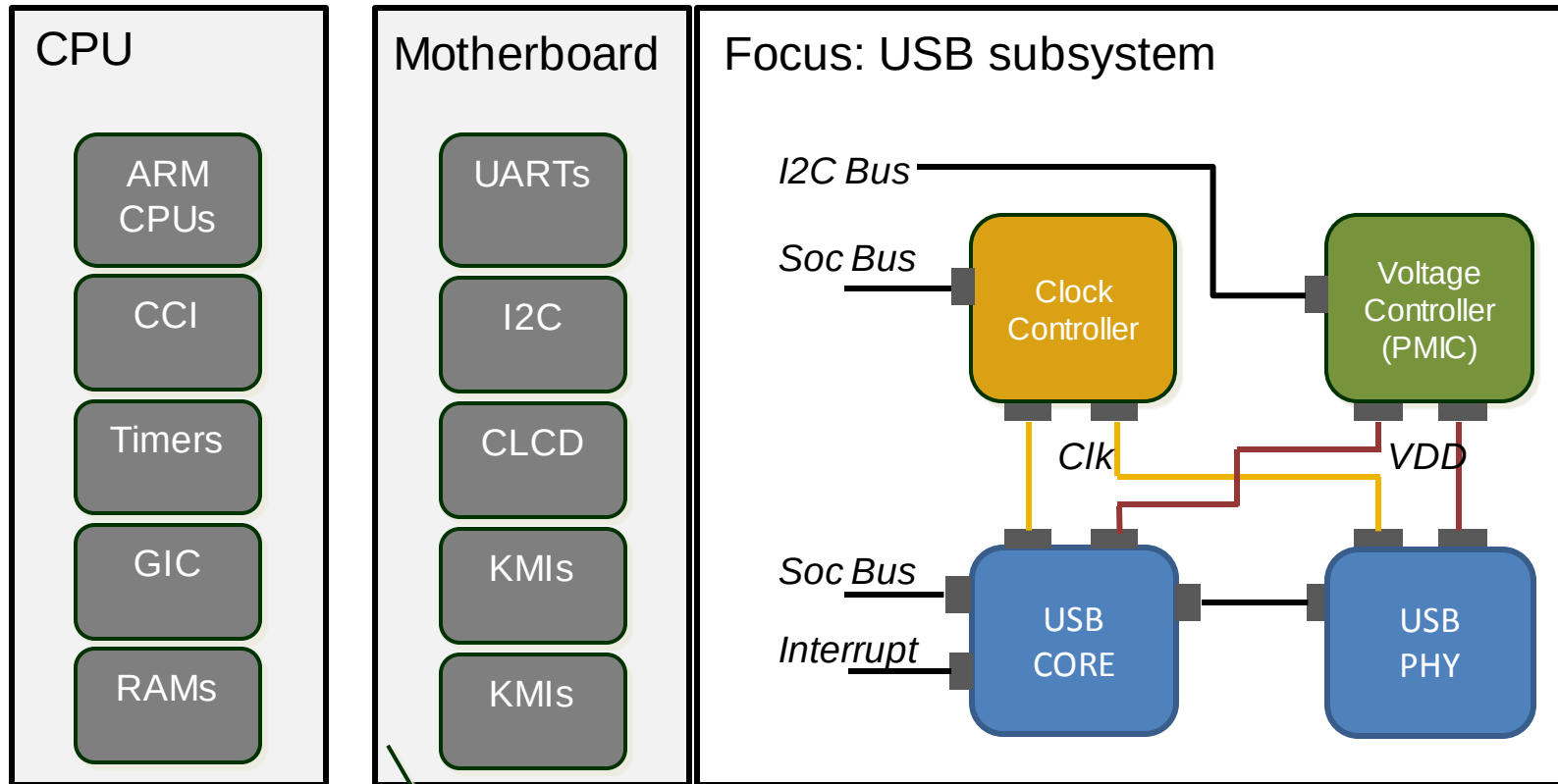
Case Study: SoC Power Management

USB subsystem with our specific PMIC and Clock Controller



Case Study: SoC Power Management

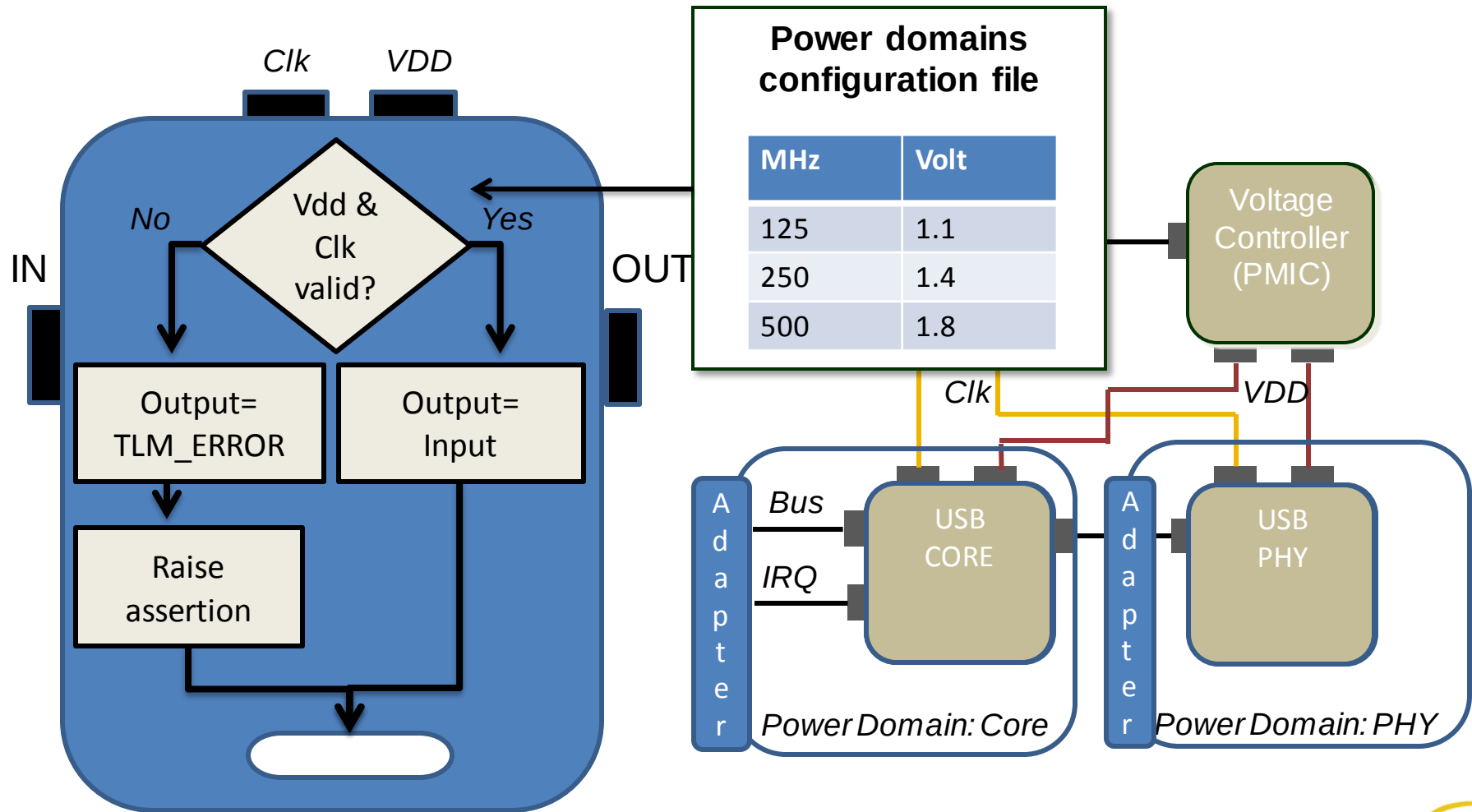
Combine with available VDK for ARM Versatile Express and software



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 Managment 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

VDK_uATX_CortexA57x1_usb_pm - Linux Console (HARDWARE:UART0)

```
SLUB: Genslabs=11, Hwalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Hierarchical RCU implementation.
RCU restricting CPUs from NR_CPUS=4 to nr_cpu_ids=1.
NR_IRQS:64 nr_irqs:64 0
GIC CPU mask not found - kernel will fail to boot.
GIC CPU mask not found - kernel will fail to boot.
Architected local timer running at 100.00MHz (phys).
Console: colour dummy device 80x25
Calibrating delay loop (skipped), value calculated using timer frequency.. 200.00 BogoMIPS (lpj=1000000)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 256
hw perfevents: enabled with arm/armv8-pmu PMU driver, 7 counters available
Brought up 1 CPUs
SMP: Total of 1 processors activated (200.00 BogoMIPS).
devtmpfs: initialized
atomic64 test passed
regulator-dummy: no parameters
NET: Registered protocol family 16
vds: 2 pages (1 code, 1 data) at base fffffffc0005d4000
hw-breakpoint: found 6 breakpoint and 4 watchpoint registers.
Serial: AMBA PL011 UART driver
1c090000.uart: ttyAMA0 at MMIO 0x1c090000 (irq = 37) is a PL011 rev1
console [ttyAMA0] enabled
1c0a0000.uart: ttyAMA1 at MMIO 0x1c0a0000 (irq = 38) is a PL011 rev1
1c0b0000.uart: ttyAMA2 at MMIO 0x1c0b0000 (irq = 39) is a PL011 rev1
1c0c0000.uart: ttyAMA3 at MMIO 0x1c0c0000 (irq = 40) is a PL011 rev1
```

DB2_1: DW USB3

Simulated USB Model

Cable

- Bus Status ---
- ☒ VBus
- ☒ Suspended
- ☒ Roles are reversed
- Orientation ---
- A-B ☒ B-A
- Speed ---
- ☒ Low/Full Speed
- ☐ High Speed

Remote USB: Physical

Controls For When Remote USB Is Physical

AutoPlay

Removable Disk (F:)

General options

Open folder to view files using Windows Explorer

[View more AutoPlay options in Control Panel](#)

Welcome to Synopsys VDK For ARM Cortex v8 Processors!

```
vdv-armv8 login: root
# start_dwusb3_filestorage
Starting FileStorage USB Gadget (on /mnt/gadgetfs) ...
Done
#
```

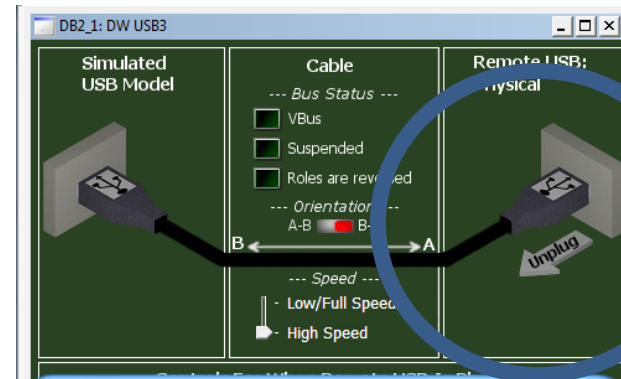
Case Study: Driver faults from power bug

Booting and using USB for a file storage gadget

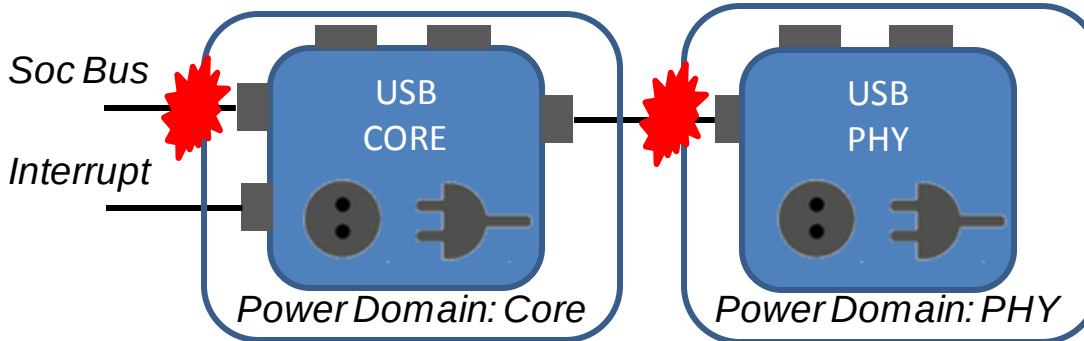
- Unpowered core
- Unpowered PHY

```
VDK_uATX_CortexA57x1_usb_pm - Linux Console (HARDWARE.iUART0)
Unhandled fault: synchronous external abort (0x9600210) at 0xffffffff8000040140
Internal error: : 9600210 [#1] SMP
Modules linked in:
CPU: 0 Not tainted (3.9.0 #4)
PC is at dwc3_probe+0x274/0xa50
LR is at dwc3_probe+0x254/0xa50
pc : [<fffffc0002fad2c>] lr : [<fffffc0002fad0c>] pstate: a000305
sp : ffffffc00ac5fc50
x29: ffffffc00ac5fc50 x28: 0000000000000000
x27: 0000000000000000 x26: ffffffc00ad33210
x25: ffffffc000588980 x24: ffffffc0005b91f8
x23: ffffffc000040100 x22: 0000000000000001
x21: ffffffc00fffb600 x20: ffffffc009aad020
x19: ffffffc000523078 x18: 000000000000000e
x17: 0000000000000007 x16: 0000000000000001
x15: 0000000000000007 x14: 00000000000003a3
x13: ffffffc00ad9a000 x12: 0000000000000300
```

Abort exception!



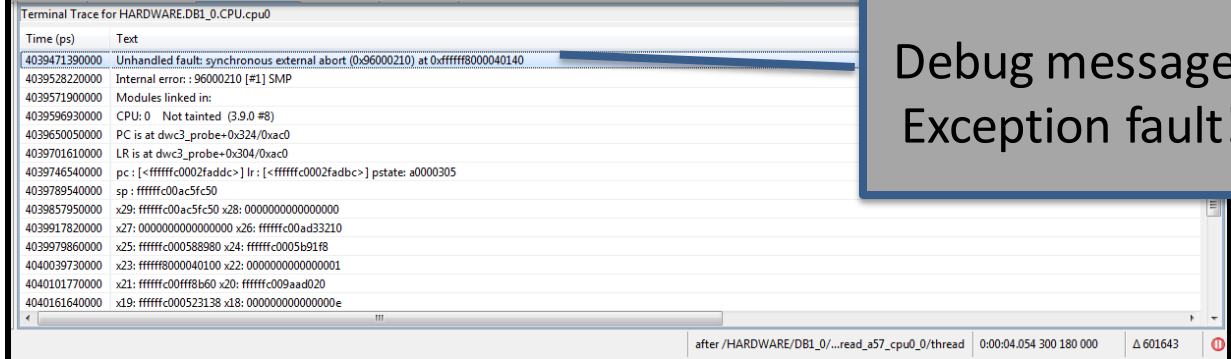
No response!



Case Study: Root cause analysis

Using a VDK,
there is more to see!

Standard Linux Kernel Messaging System



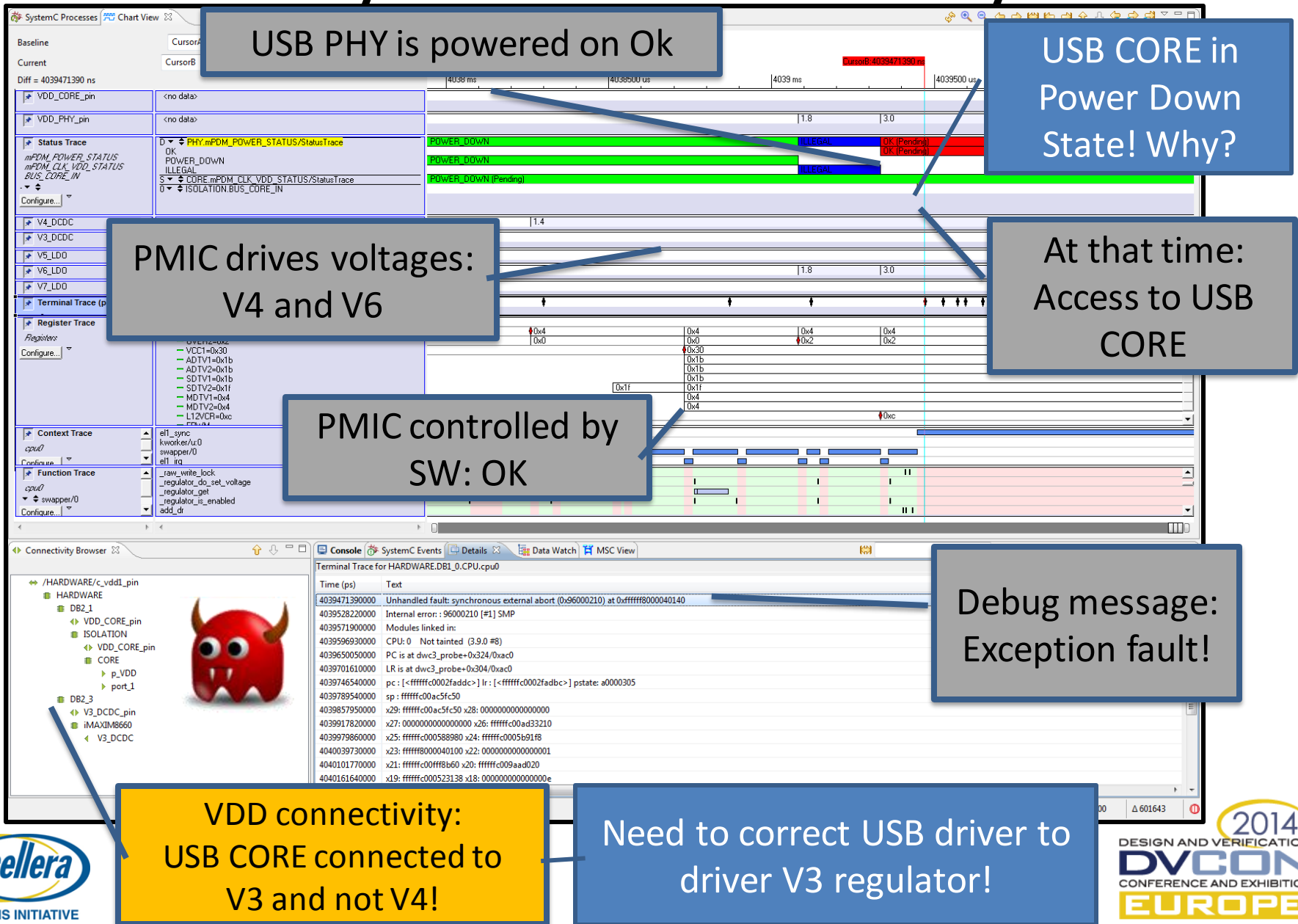
Terminal Trace for HARDWARE.DB1_0.CPU.cpu0

Time (ps)	Text
4039471390000	Unhandled fault: synchronous external abort (0x96000210) at 0xfffff8000040140
4039528220000	Internal error: 96000210 [#1] SMP
4039571900000	Modules linked in:
4039596930000	CPU: 0 Not tainted (3.9.0 #8)
4039650050000	PC is at dwc3_probe+0x324/0xac0
4039701610000	LR is at dwc3_probe+0x304/0xac0
4039746540000	pc : [<fffffc0002faddc>] lr : [<fffffc0002fadbc>] pstate: a0000305
4039789540000	sp : fffffc00ac5fc50
4039857950000	x29: fffffc00ac5fc50 x28: 0000000000000000
4039917820000	x27: 0000000000000000 x26: fffffc00ad33210
4039979860000	x25: fffffc000588980 x24: fffffc0005b91f8
4040039730000	x23: fffffc0000040100 x22: 0000000000000001
4040101770000	x21: fffffc00fffb60 x20: fffffc009aad020
4040161640000	x19: fffffc000523138 x18: 000000000000000e

after /HARDWARE/DB1_0/...read_a57_cpu0_0/thread 0:00:04.054 300 180 000 Δ 601643

Debug message:
Exception fault!

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

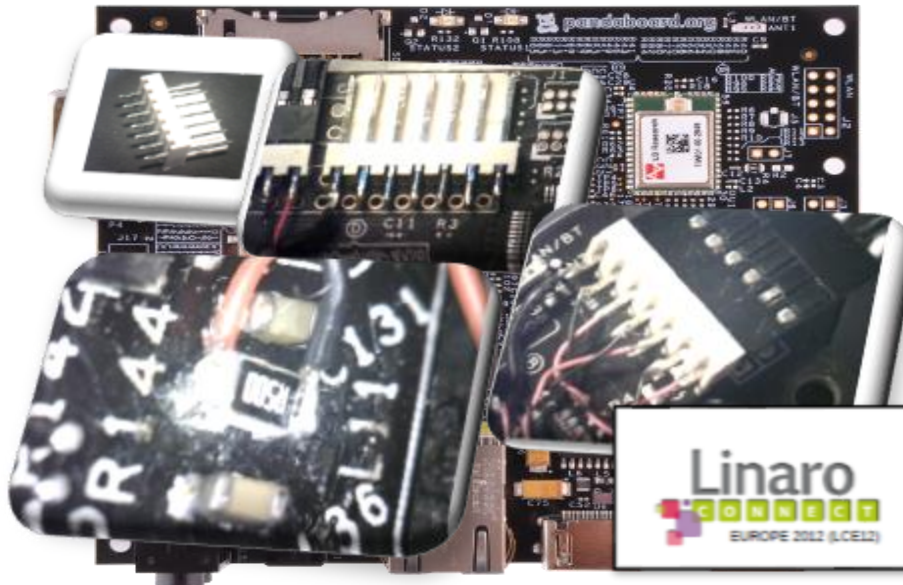
- No means to test and use-case analyze during development
- Power bugs continuously slip into production firmware

How to analyze power bugs?

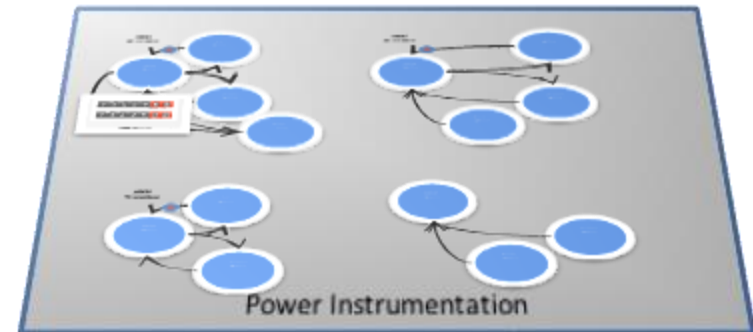
Soldering skills required...

The hardware way...

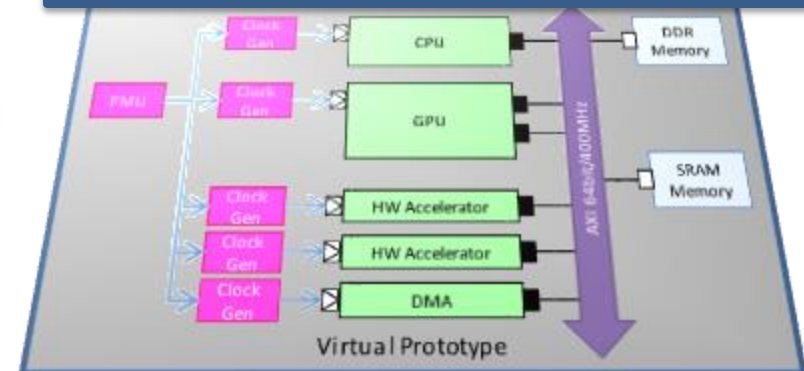
- Use a 7-way 0.1" header for each probe (3 channels)
- Superglue the back to a spare region of the board
- Add 0V connection
- Add twisted pairs back to the shunt
- Remove old inductor, solder the shunt in place
- Add the other end of the wires to the shunt



The Virtualizer way...



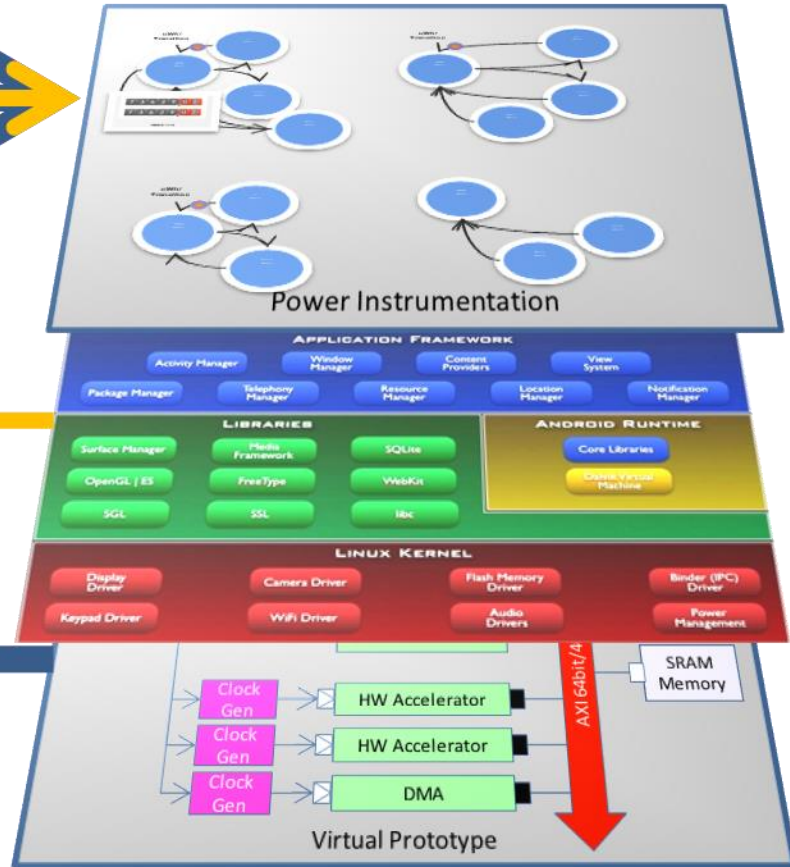
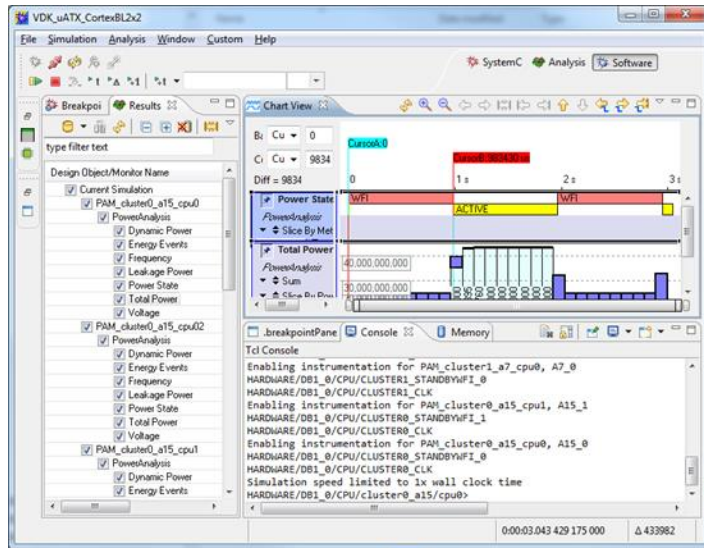
Dynamic power analysis
Instrumentation overlay



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

Dynamic power analysis

Using instrumentation scripts in Virtualizer VDK



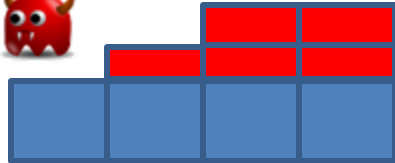
Software
Drivers

Signals and
Registers

USB Power Model

Abstract power model

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



USB PHY Power
USB Core Power
Other

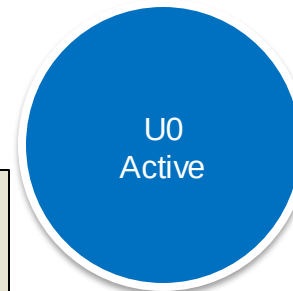
From K Park:

> **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.

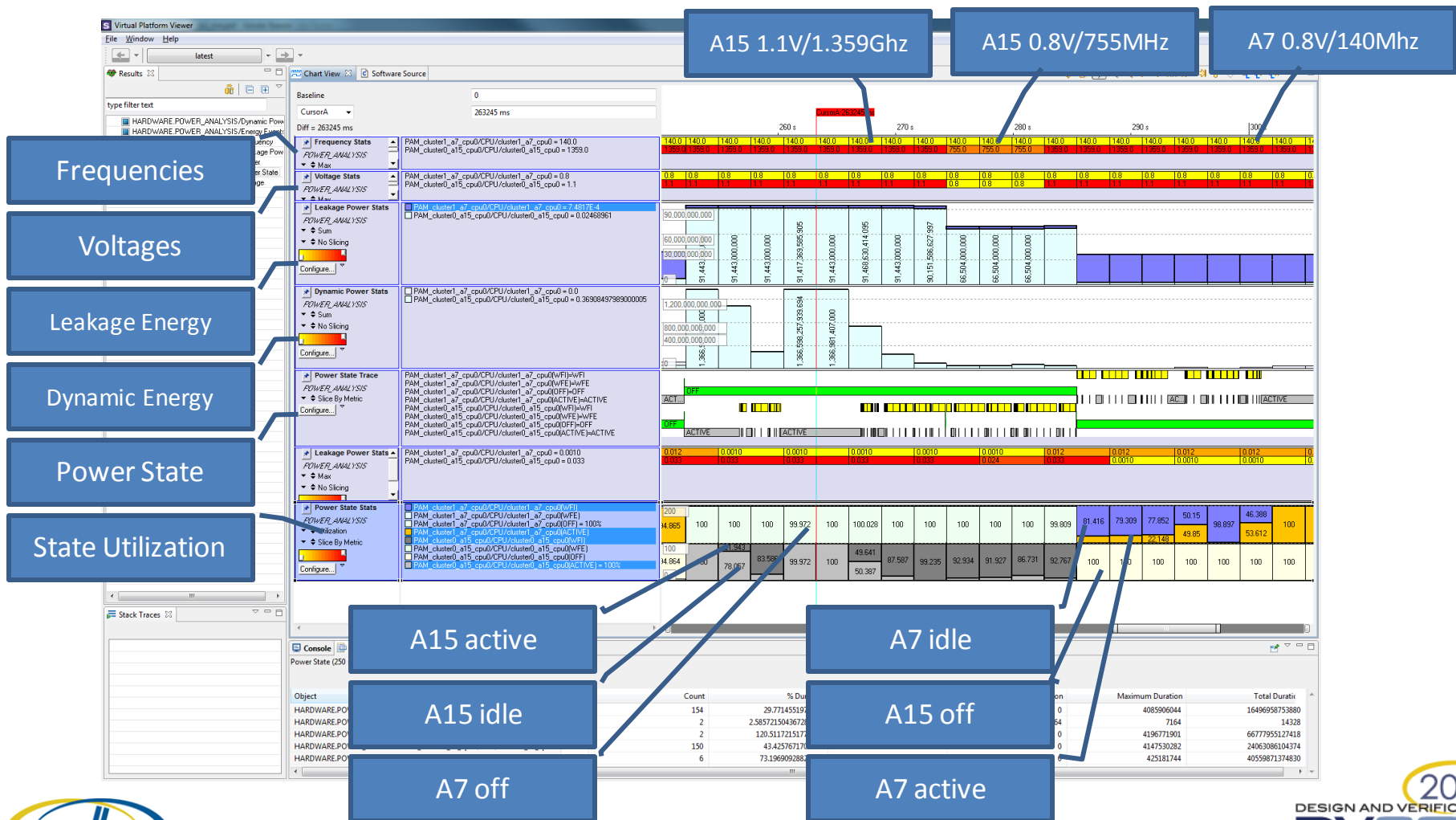
Detailed power model

- 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 through 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