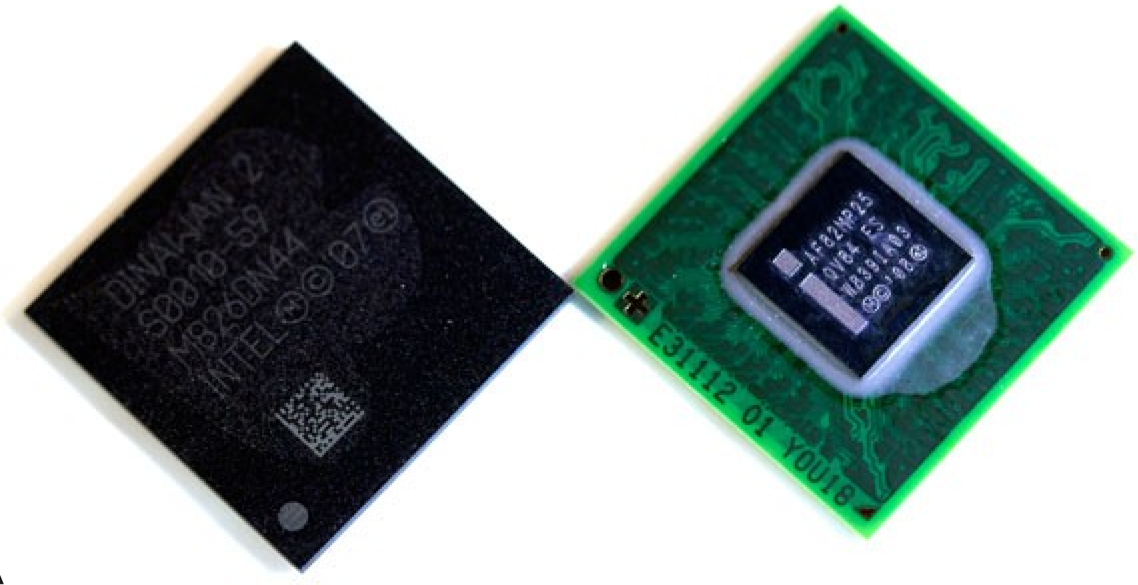
Linux Power Management Experiences on Moorestown

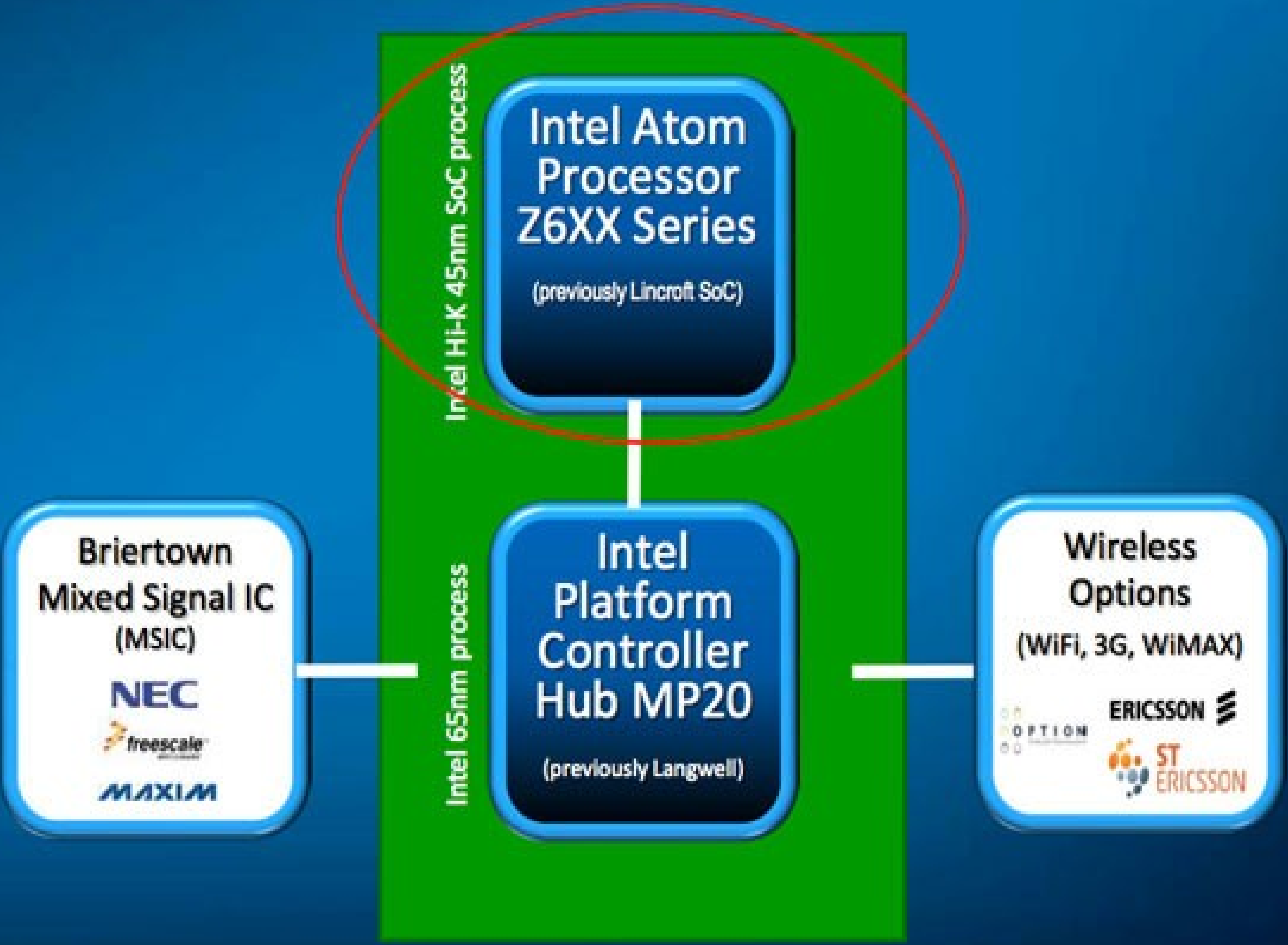
Kristen Accardi
Len Brown

Intel Open Source Technology Center



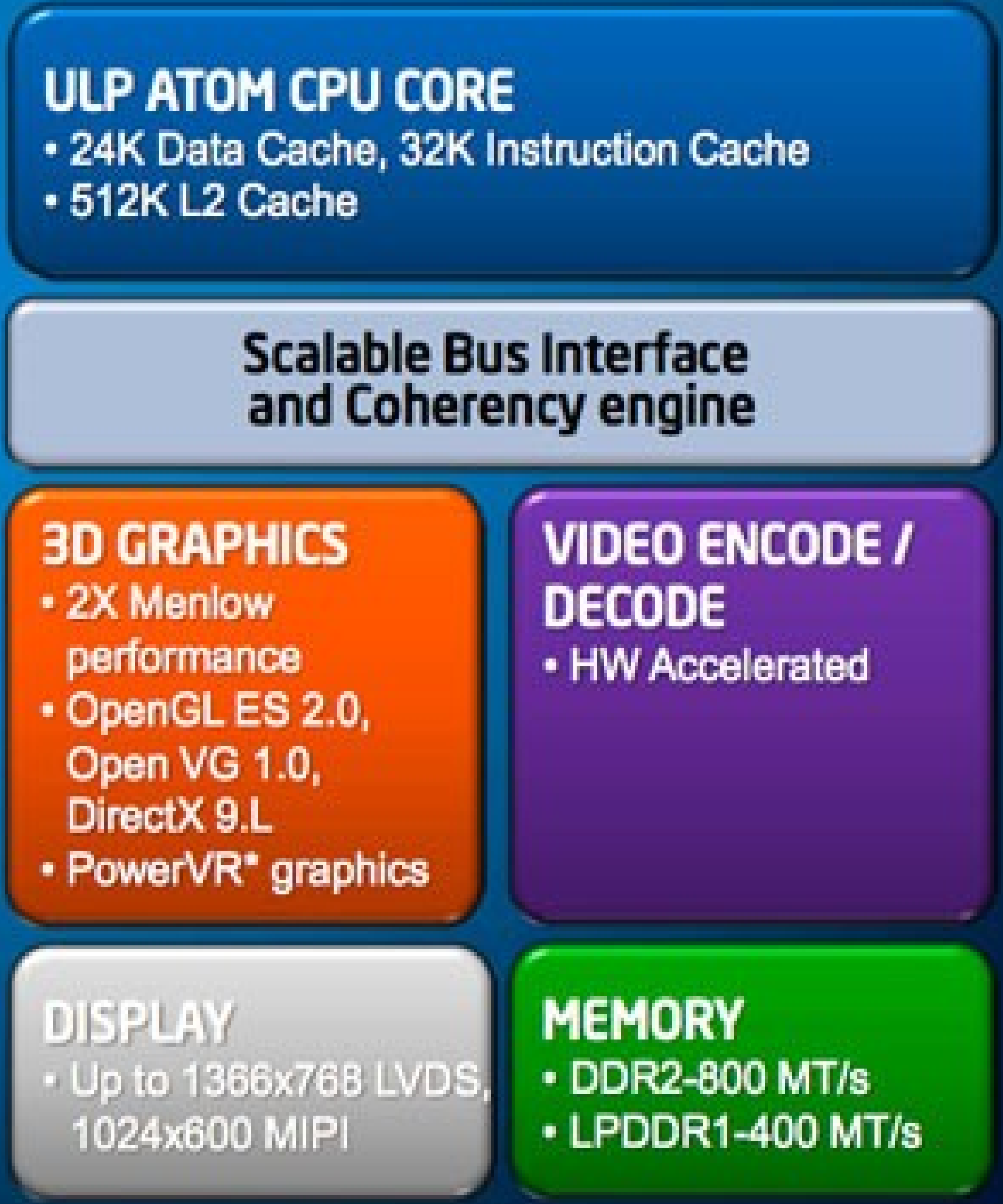
Linux Plumbers Conference
Sept. 8, 2011 Santa Rosa, CA

Moorestown Platform Overview


Intel Hi-K 45nm SoC process
Intel Atom Processor Z6XX Series
(previously Lincroft SoC)
Intel 65nm process
Intel Platform Controller Hub MP20
(previously Langwell)
Briertown Mixed Signal IC (MSIC)
NEC
freescale
MAXIM
Wireless Options
(WiFi, 3G, WiMAX)
OPTION
ERICSSON
ST ERICSSON

Intel® Atom™ Processor Z6XX Series
(Previously Codenamed Lincroft SOC)

- Highly Integrated SoC
- Combines Atom core, 3D Graphics, and Video Encode / Decode
- Compact Package: 13.8 x 13.8 x 1.1 mm³ (0.5mm pitch)
- High Performance at Low Power
  - 19 power islands w/ on-die clock and power gating
  - New ultra low power states (S0i1, S0i3)
  - Hardware Accelerated Video Encode/Decode
  - Optimized 3D graphics
  - Based on Intel Hi-k 45nm LP SoC process



ULP ATOM CPU CORE
- 24K Data Cache, 32K Instruction Cache
- 512K L2 Cache

Scalable Bus Interface and Coherency engine

3D GRAPHICS
- 2X Menlow performance
- OpenGL ES 2.0, Open VG 1.0, DirectX 9.L
- PowerVR* graphics

VIDEO ENCODE / DECODE
- HW Accelerated

DISPLAY
- Up to 1366x768 LVDS, 1024x600 MIPI

MEMORY
- DDR2-800 MT/s
- LPDDR1-400 MT/s

Intel Platform Controller Hub MP20

(Previously Codenamed Langwell)

- Highly Integrated PCH reduces power, footprint, and cost
- Incorporates Handheld I/Os and Media Accelerators
- Compact Package: 14 x 14 x 1.3 mm^3 (0.5mm pitch)
- Ultra Low Power
  - Low leakage 65nm LP process
  - Fine-grained power management
  - Enables Low active/idle platform power

SYSTEM CONTROLLER
- Low Power 32-bit RISC core

IMAGE PROCESSING (Camera support)
- 5MP & VGA Camera
- Dedicated Image Signal Processor

AUDIO ACCELERATION
- Support for audio codecs
- I2S interface

CRYPTO ACCELERATION
- Secure Boot

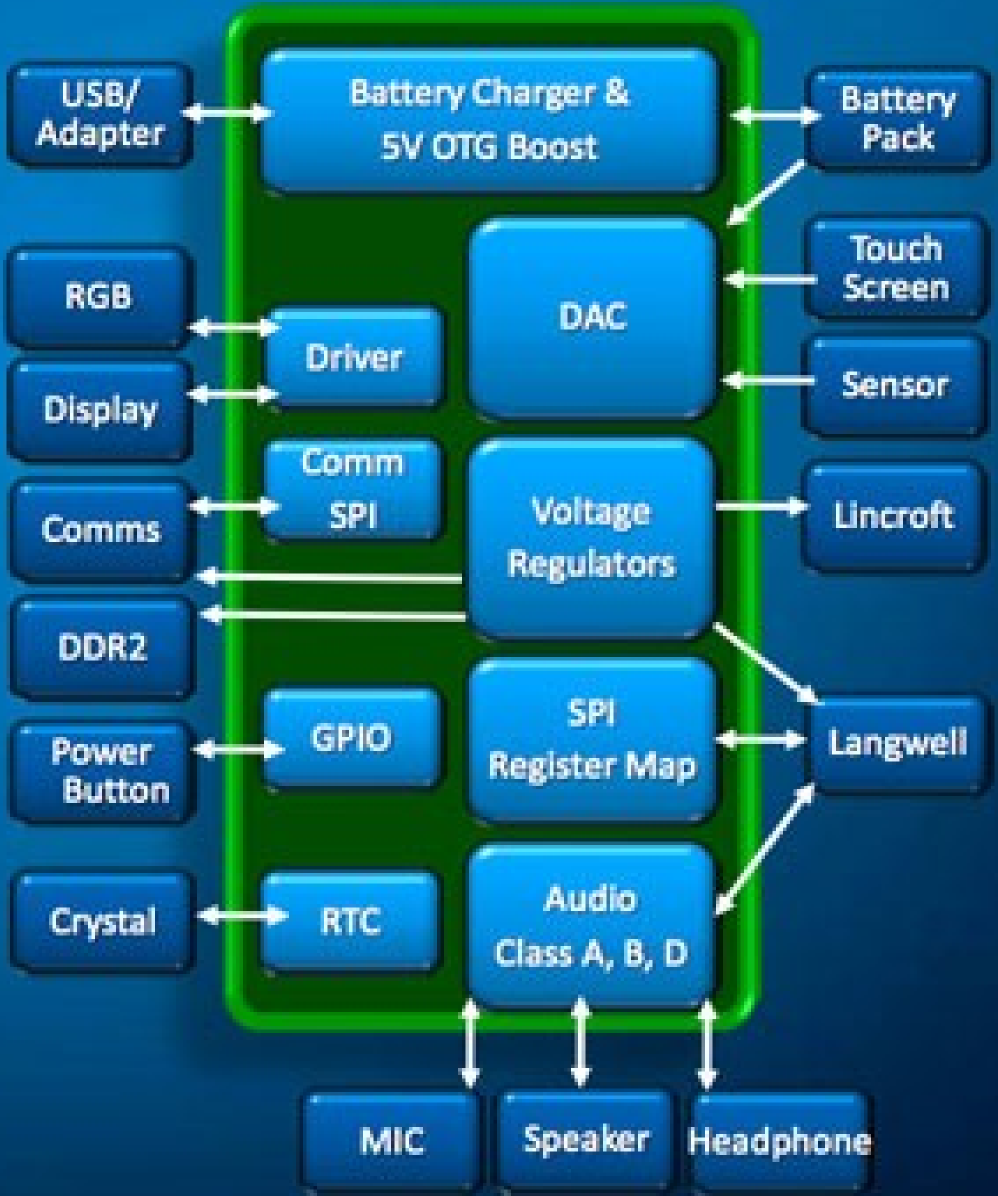
IO
- USB 2.0 EHCI Controller, USB OTG 1 Host/Device port
- HDMI Support

Solid State Drive

intel

Mixed Signal IC (MSIC)
(Previously Codenamed Briertown)

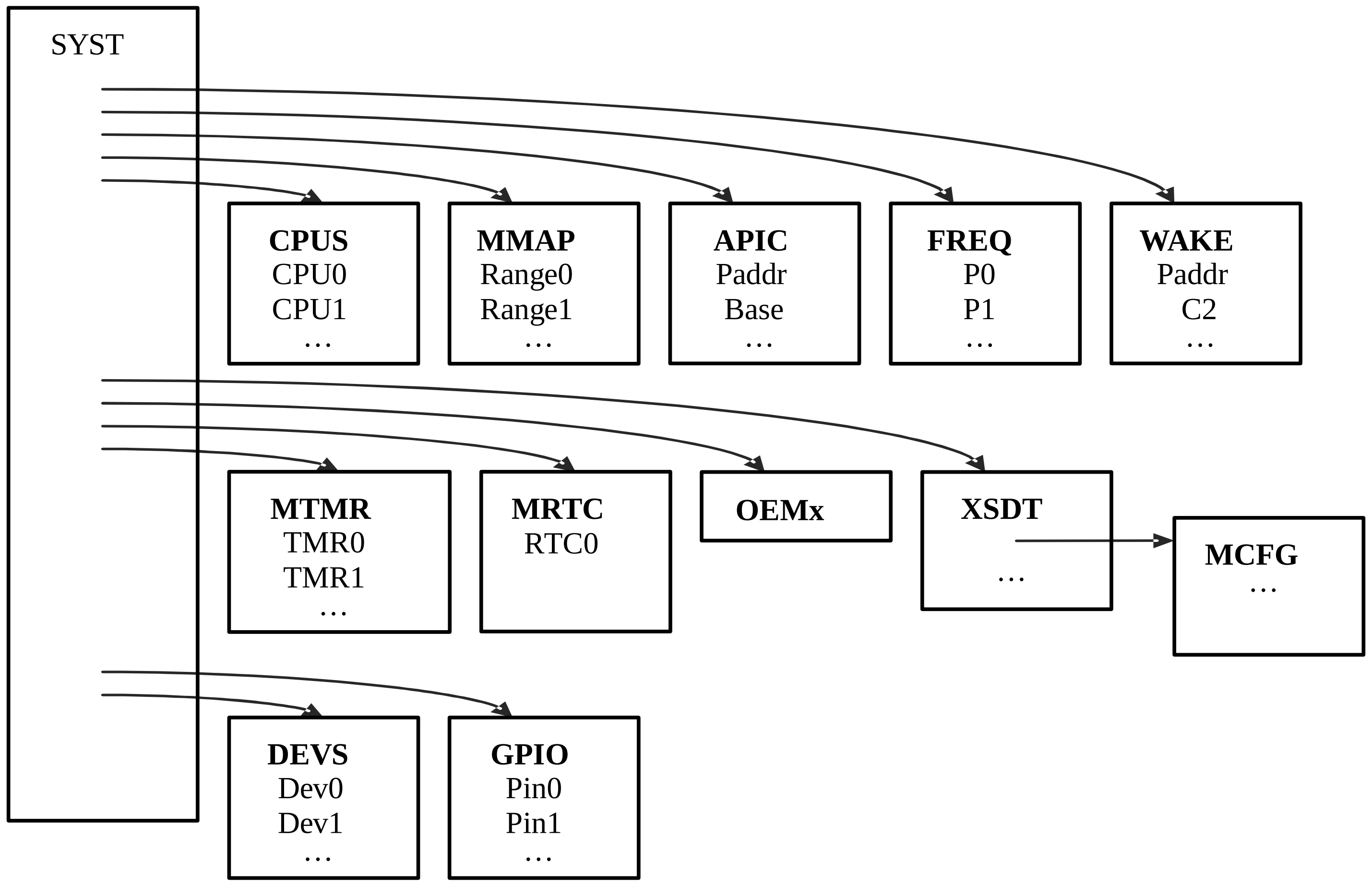
- Highly Integrated Design
- Efficient Power Delivery
- Integrates Components → Lower Power and Cost
- Multi-source availability: Freescale*, Maxim*, Renesas*



PCI on MRST

- Lincroft Graphics is real PCIe
- All Langwell PCI is emulated by “PCI shim”
  - Config space is simply RAM
  - ✗ Requires platform specific config space accessors
  - ✗ PMCSR does nothing
  - ✗ No #PME
- ✓ PCI Device Driver enumeration
- ✗ Not real HW, OS exposed to FW bugs

SFI – Simple Firmware Interface



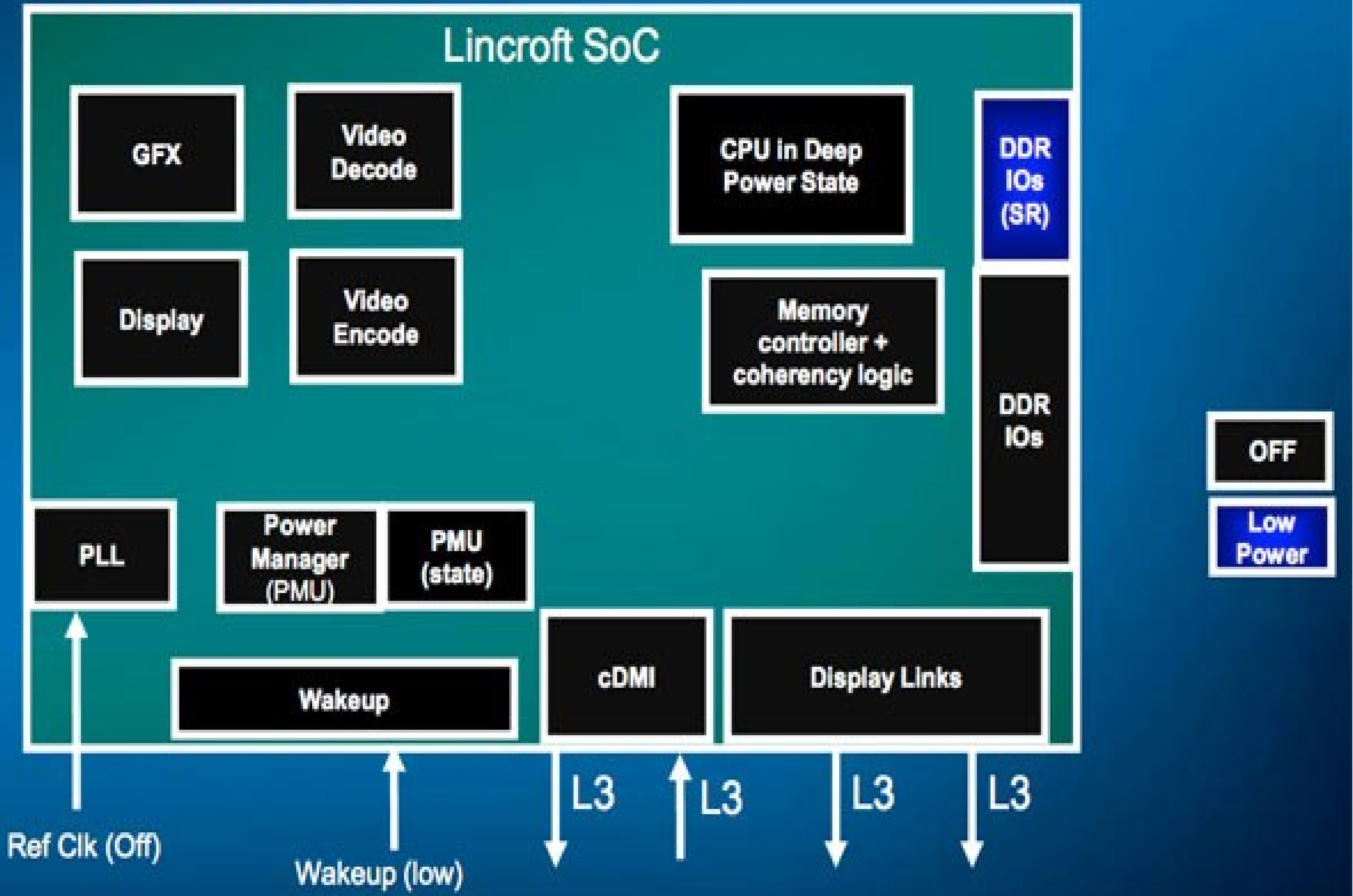
SFI – an 80% solution

- ✔ Simple
  - ✔ Difficult for FW to get it wrong
- ✔ Open and Freely Available
  - ✔ http://simplefirmware.org
  - ✔ Upstream Linux since about 2.6.32
- ✔ Extensible
  - ✔ Vendor specific tables permitted
- ✗ Not Comprehensive
  - ✗ Particularly for device enumeration

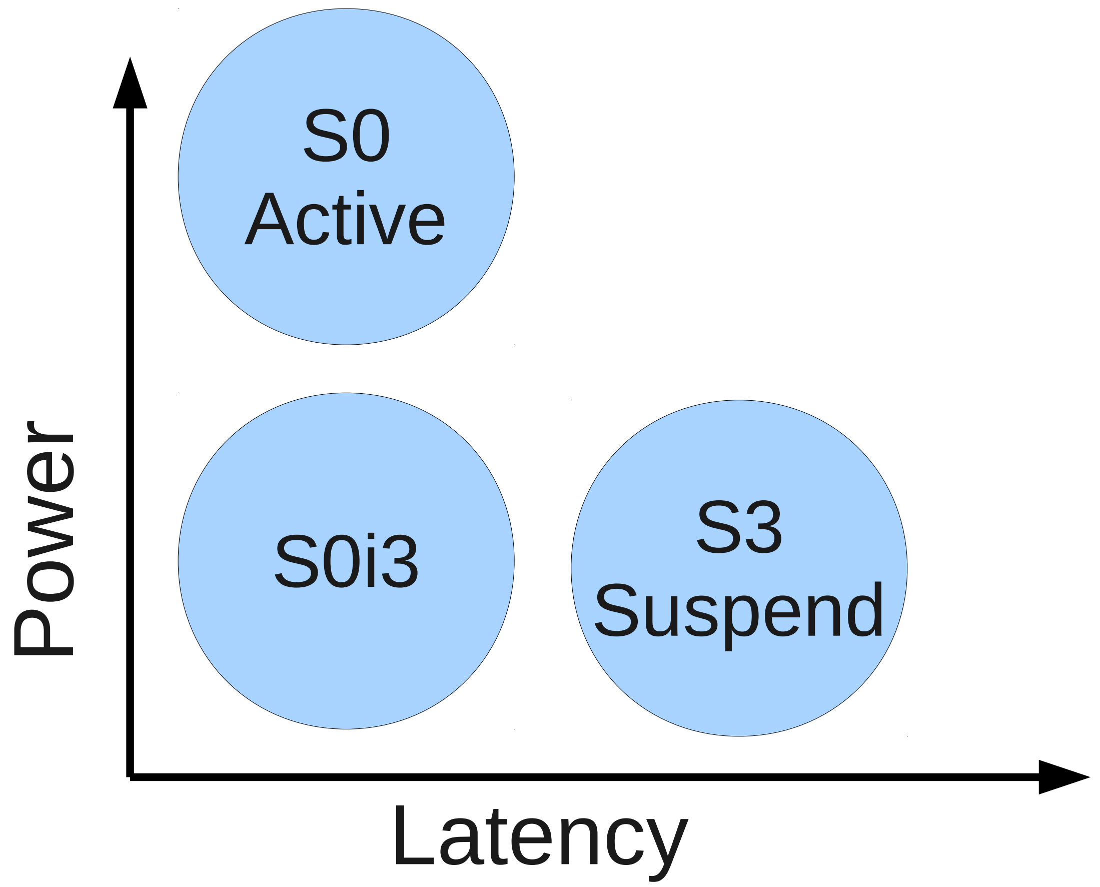
MRST Processor Performance States (P-states)

- Linux sfi-cpufreq cpufreq driver
  - Consumes SFI “FREQ” table
- Linux ondemand cpufreq governor

What is S0i3 state?



S0i3 “Active Idle” System State

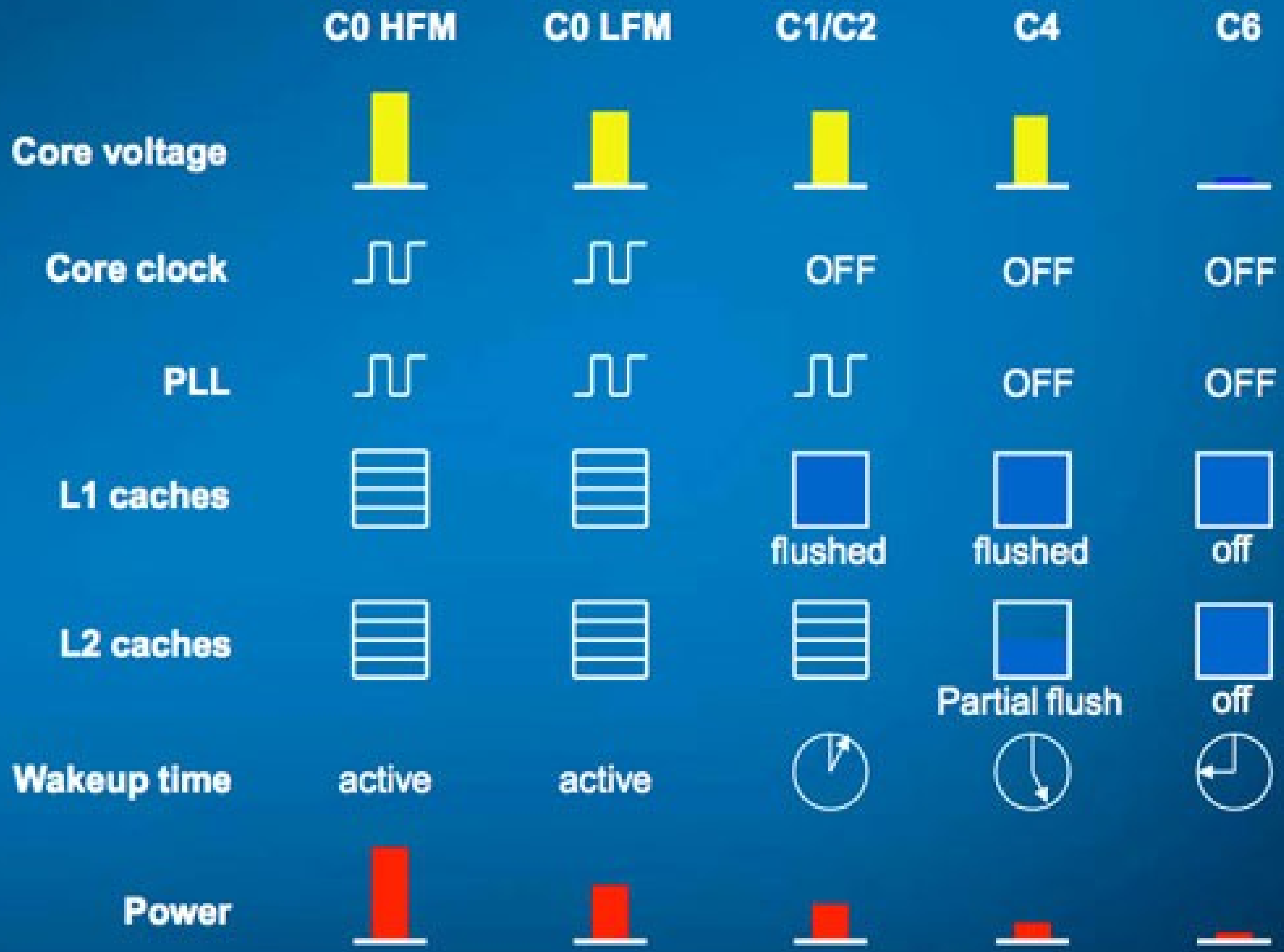


- S0 (C-state) latency
- S3 (system wide) low power

S0i3: C-state or S-state?

- S0i3 can be used for either/both use-cases.
  - Android used S0i3 only for S3 (STR).
  - MeeGo used S0i3 only as a C-state.

CPU C-States



	C0 HFM	C0 LFM	C1/C2	C4	C6
Core voltage	▮	▮	▮	▮	▁
Core clock	⎍	⎍	OFF	OFF	OFF
PLL	⎍	⎍	⎍	OFF	OFF
L1 caches	☰	☰	■ flushed	■ flushed	■ off
L2 caches	☰	☰	☰	■ Partial flush	■ off
Wakeup time	active	active	◷	◷	◷
Power	▮	▮	▮	▮	▁

S0i3 as a C-state

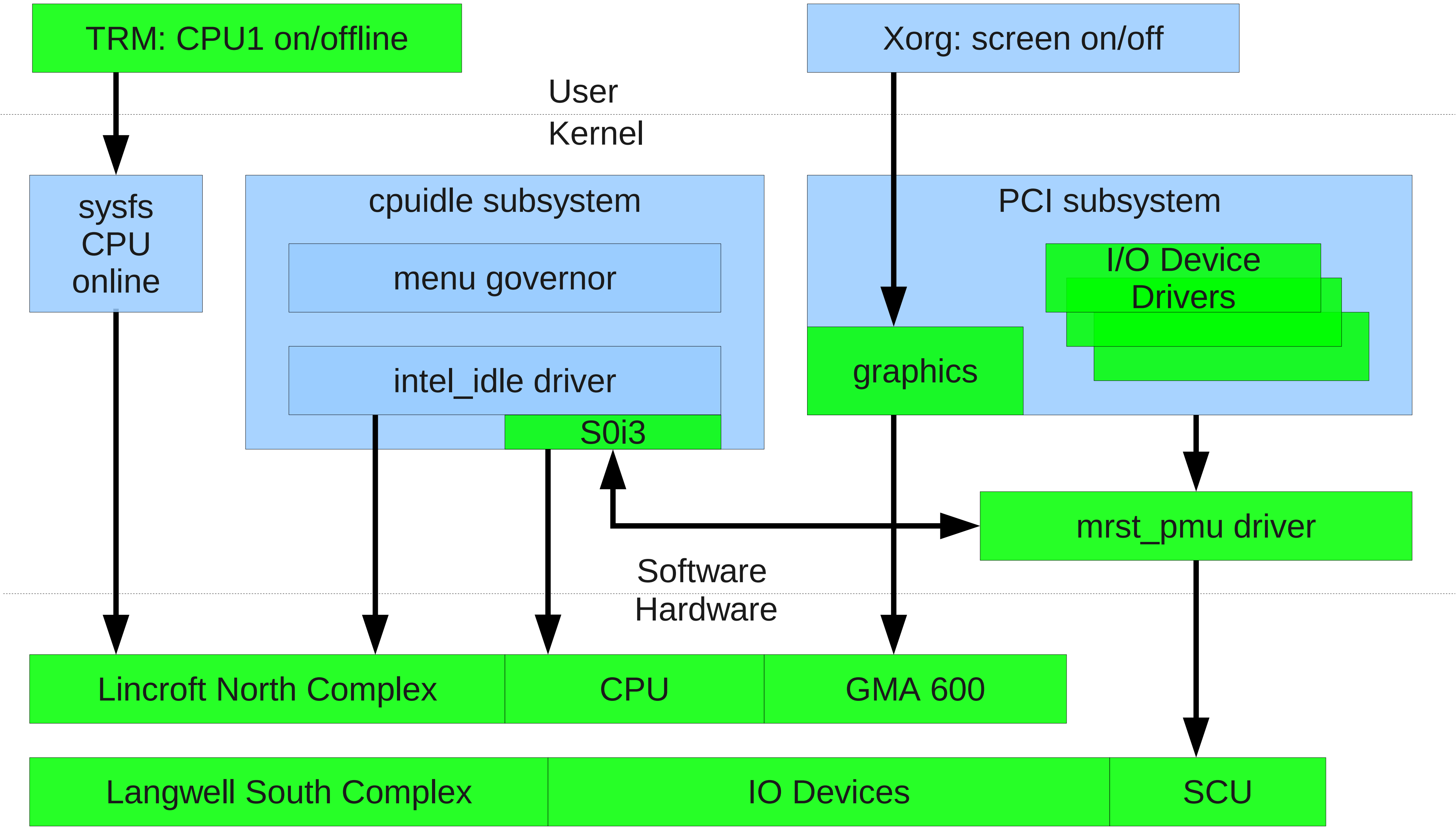
```
           Package   |           CPU 0
ATM-C1       0.5%    | ATM-C1       0.5%    5.1 ms
ATM-C2       1.2%    | ATM-C2       1.2%    7.7 ms
ATM-C4       0.6%    | ATM-C4       0.6%    5.4 ms
ATM-C6      39.0%    | ATM-C6      39.0%   24.9 ms
MRST-S0i3  56.6% | MRST-S0i3  56.6%   70.9ms
```

MRST pmu driver

- Track all PCI D-state requests
  - Translate PCI device to MRST Logical Subsystem
  - Translate PCI D-States to MRST D-states
- Facilitate S0i3 transitions
  - Track system-wide entry criteria
  - Invoked by intel_idle's S0i3 entry point

- Upstream: arch/x86/platform/mrst/pmu.c
  - See initial commit-id for contributors

Linux/MRST Power Mgmt. Architecture



PCI devices -> MRST Logical Subsystems (LSS)

- PCI device has a driver
- MRST LSS is a HW clock/power domain

✔ Usually a simple 1:1 mapping

✗ Sometimes LSS shared

PCI -> MRST Device-States: Imperfect Mapping

<u>PCI</u>	**<u>Lincroft</u>**
D0	**D0 – full on**
	D0i1 – auto clock gate
D1, D2	
D3hot, D3cold	**D0i2 – hard clock gate** **D0i3 – hard clock gate**

/sys/kernel/debug/mrst_pmu

```
0800 dw_spi_pci         LSS 10 D0i1 PCI-D3hot D3hot 94 0 0 47 0
0802 i2c-designware-p LSS 10 D0i1 PCI-D3hot D3hot 135034 0 0 67517 0
0803 i2c-designware-p LSS 10 D0i1 PCI-D3hot D3hot 4 0 0 2 0
0804 i2c-designware-p LSS 10 D0i1 PCI-D0   D0 3110 0 0 1554 0
0806 ehci_hcd           LSS  7 D0i3 PCI-D3hot D3hot 2 0 0 1 0
0807 sdhci-pci          LSS  1 D0i3 PCI-D3hot D3hot 10 0 0 5 0
0808 sdhci-pci          LSS  2 D0i1 PCI-D0   D0 65944 0 0 32971 0
0809 pci                LSS  3 D0i3 PCI-D0   D3hot 0 0 0 1 0
080A intel_sst_driver LSS  9 D0i3 PCI-D3hot D3hot 8 0 0 4 0
080B mrstisp            LSS  4 D0i3 PCI-D3hot D3hot 2 0 0 1 0
080C pci                LSS  5 D0i3 PCI-D0   D3hot 0 0 0 1 0
080D pci                LSS  6 D0i3 PCI-D0   D3hot 0 0 0 1 0
080E intel_scu_ipc                 PCI-D0   D0 2 0 0 0 0
4102 pvrsrvkm                      PCI-D0   D0 14722 0 0 12661 0
0814 Intel MID DMA     LSS  9 D0i3 PCI-D3hot D3hot 2 0 0 1 0
0815 pci                LSS  9 D0i3 PCI-D0   D3hot 0 0 0 1 0
084F sdhci-pci          LSS 12 D0i1 PCI-D0   D0 160702 0 0 80350 0
```

Linux Run Time Device PM

- 10 observations from deployment on MRST

Run Time Device PM #1

Understand the concept of runtime power management.

**** you sleep when you are not busy, not when someone tells you to! ****

Transparent to user space. Initiated by kernel.

Run Time Device PM #2

Know what your <u>subsystem</u> does for you.

Review your subsystem power management code - what does it really do?

For example, on PCI, the amount of setup work you have to do in the driver is limited to allowing runtime pm and then decrementing the usage count. Similarly, in the   pm_runtime_suspend/resume code paths in the driver, you will not need to duplicate core functionality like sending power state transitions.

Run Time Device PM #3

Understand all the possible <u>entry points</u> to your driver.

sysfs counts!

Run Time Device PM #4

Understand when you are actually <u>touching hardware</u>.

Run Time Device PM #5

Understand the <u>context</u> in which you are being called during these entry points.

Are you called in interrupt context?

Run Time Device PM #6

For interrupt context, ensure the hardware is available without sleeping, otherwise use a workqueue.

Run Time Device PM #7

Refcounting during the <u>irq handler</u> does cost cycles. this may be ok depending on your device.

On mrst, we had <u>no #PME</u>, so wakeups would come as regular interrupts.

This required that we do ref counting at irq time, and that we use threaded irqs as much as possible.

Run Time Device PM #8

Have the correct granularity for your ref counting. Depending on your hw, it likely has a performance penalty for entering a suspended state. You may evenlose device state. Restoring can take time, so you want to make sure you are appropriately idle before allowing a suspend.

Run Time Device PM #9

Unbalanced ref counting occurs when you do not pay attention to <u>error paths</u>.

Run Time Device PM #10

Unbalanced ref counting can occur when you try to do something tricky with the refcounting to work around overly complex code paths.

eg. setting your pm usage count to zero without decrementing.

This will cost you hours and hours of debug time

Don't do it! Clean up your driver and stop trying to hack your way out of it.

Thank you!