Power Management, Debugging and Optimizations

- **Avinash Mahadeva**
- **Vishwanath Sripathy**

Contents

- Introduction
- Architecture
  - Hardware
  - Software
- Common Problems encountered
- Debugging
- Optimizations

Introduction

Why Power Management

- To maximize the battery life of handheld devices.
- Limit the Power consumption to the minimum without taking a hit on performance.
- Run each usecase with the minimum power consumed and expected performance.

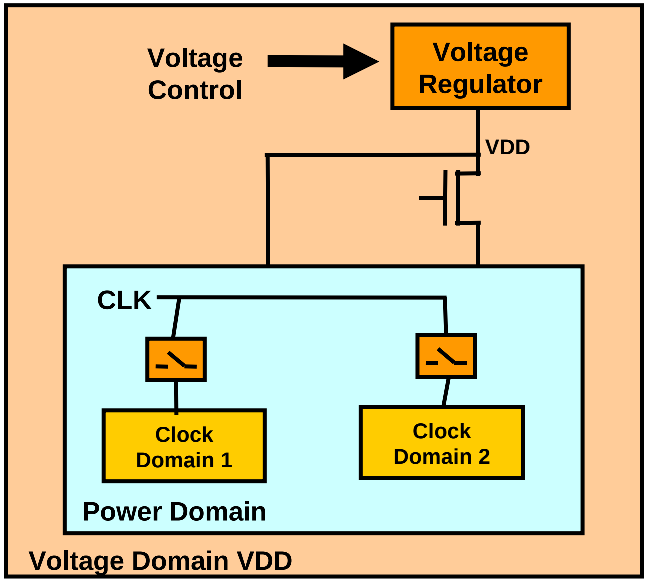
Types of Power Management

- **Active / Dynamic Power Management**
  - When system is active and performing tasks.
  - Clocks are on and processing is going on.
  - Ex: Mp3 playback / AV playback.
- **Standby / Static Power Management**
  - System is idle and no task is performed.
  - Modules are not active.
  - No activity is performed.
  - Ex: Phone left idle, Screen blank and no activity.

Hardware Architecture in SOCs

- How hardware is organized ?
  - Module clocks : ON, OFF
  - Clock domains : ON, OFF
  - Power domains : ON, CSWR, OSWR, OFF
  - Voltage domains : Active, Retention, Off

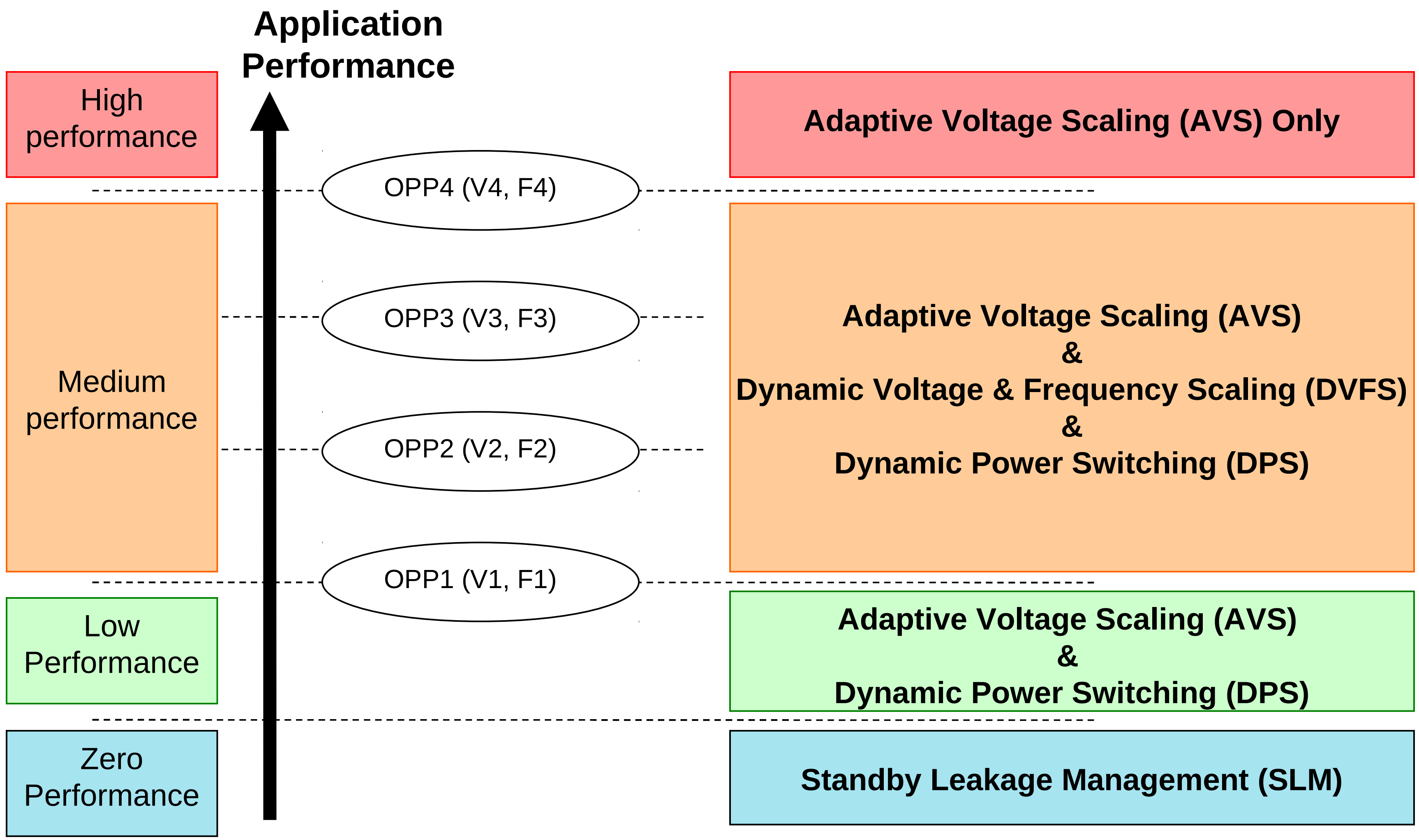
- Ex: In OMAP
  - VDD_CORE_L
  - PD_L4_PER
  - CD_L4_PER
  - GPIO, GPTIMER, MMC



Software Power Management Techniques

- Standby Power Management
  - Suspend Resume
  - CpuIdle
    - Dynamic Clock Switching
    - Dynamic Power Switching.
    - Aggressive clock cutting.
  - Turn off as many devices as possible when not used.
- Active Power Management
  - Dynamic Frequency and Voltage Scaling.
    - Multiple Operating Performance Point (V, F)
    - Ex: CORE OPPs in OMAP 4430
      - OPP1 (0.962V, 100MHz)
      - OPP2 (1.127V, 200MHz)
  - Adaptive Voltage Scaling.

How does it look together



NOTE: OPP is “Operating Performance Point”

Common Problems Encountered

- Functionality after Low power mode. (Retention, OFF)
- Performance drop with Power Mangement.
- Aborts and Crashes.
- Random Hangs and Reboots.
- Regression with PM enabled.
- And the list continues ☺

Debugging in Software

- Prints
  - Simple to use.
  - Difficult to put code in recurring code, gets flooded and may not be able to print all times (when UART clocks cut).
- Spinloops
  - while(1) during boot, after waking from OFF.
  - Needed when onchip breakpoints are lost. (many times from OFF)
  - Attach, Break using Lauterbach and print register dumps
- Persistent Memory can be used to log counters.
  - SAR memory in OMAP is not lost in OFF mode.
  - Persistent memory tracing.
- Sysfs and debugfs entries
  - cat /sys/devices/system/cpu/cpu0/cpufreq/*
  - cat /sys/devices/system/cpu/cpu0/cpuidle/state*/*
  - cat /debug/pm_debug/*

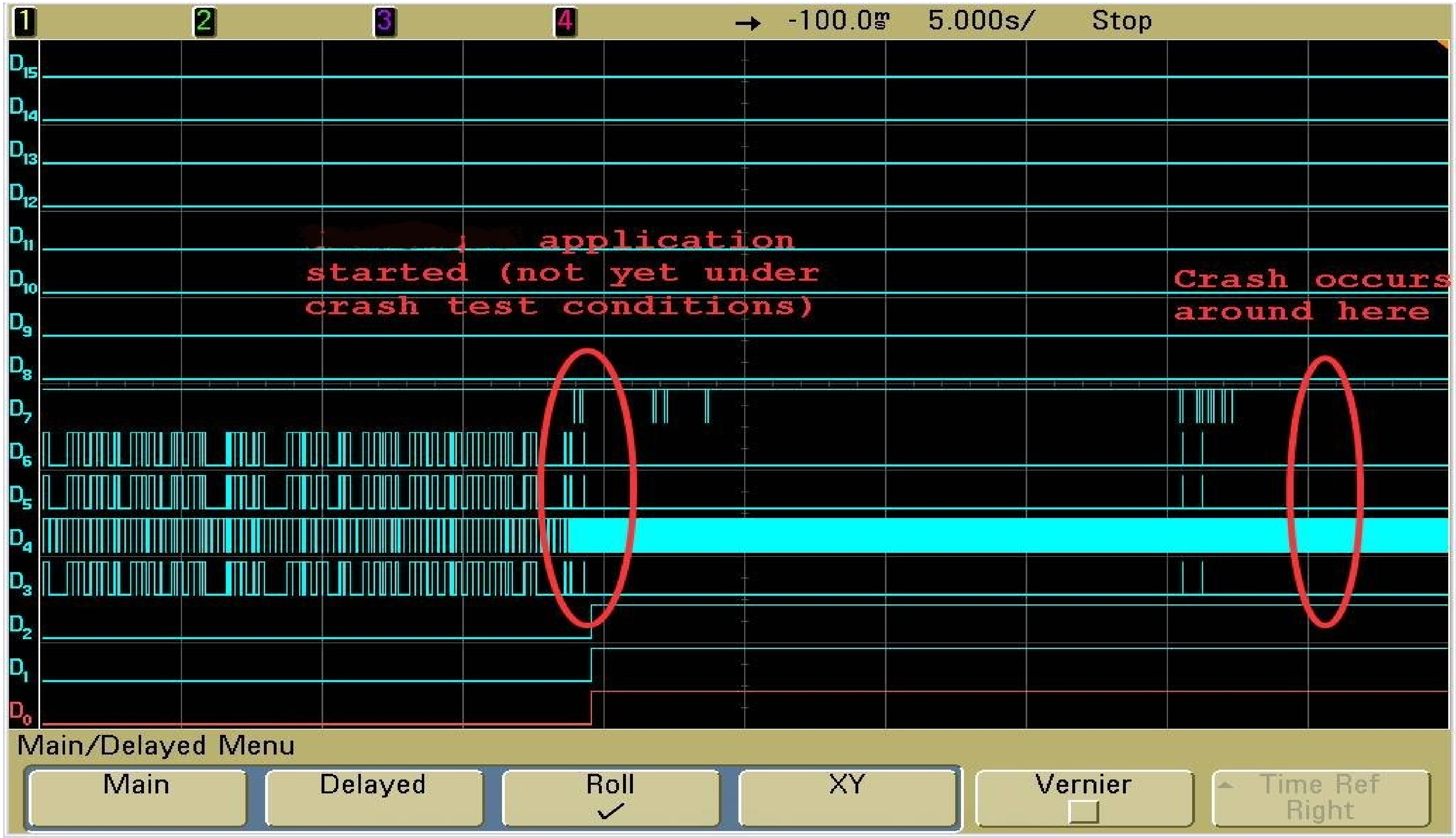
Debugging in Hardware

- Probing using Oscilloscope, LA
  - Voltages.
    - Useful to check if appropriate voltage is supplied as desired.
    - Helpful to capture any shootups or variations.
  - Clocks.
    - Can check if Clocks are turned on or not.
    - Can verify the rate of the clocks.
- Triggers in Oscilloscope can be very useful.
- Few of the signals are available at Testpoints, others may need mod.

HWOBS1: What it is ?

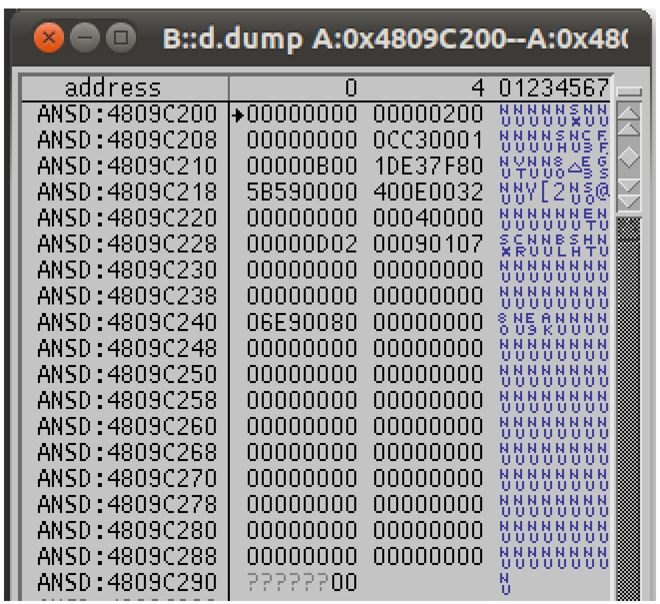
- Hardware and Observability signals in OMAP.
- Nearly 500 internal omap signals can be brought out on 32 pins.
  - hw_obs0 to hw_obs31
  - settings
    - padmux to bring hw_obs signals
    - mux configuration to bring out appropriate signals at pads.
  - Tie High, Tie Low settings are available – helps to verify the setup and wirings.
  - Various signals including
    - Clocks and DPLL outputs.
    - Reset signals
    - Standby, IdleRequests, IdleAcks for various modules
    - Powerdomain FSMs for almost all powerdomains
    - IRQs
    - EMIF, Cache controller signals.

HWOBS2: signals during a crash

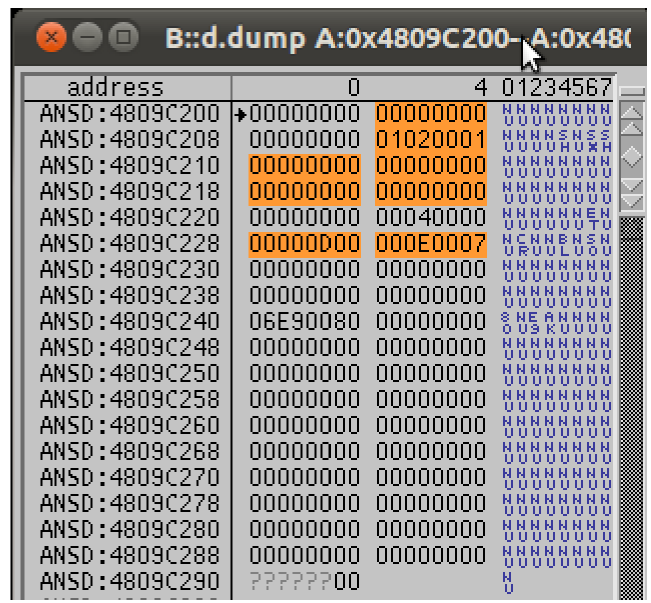


Lauterbach

- Very Powerful JTAG debuggger.
- Step, run through the code with viewing stack.
- Has Linux Awareness
- Can get Register dumps (CPU, IO mapped device)
- SpotLight : Can be used to find memory corruptions; SAVE – RESTORE comparision
  - **data.dump 0x4809C200--0x4809C290 /SpotLight /WIDTH 2** (MMC1 registers)
  - Stop once after saving and once after Restoring.
  - Before Off mode. After Off mode.


B::d.dump A:0x4809C200--A:0x48(

address	0	4
ANSD:4809C200	00000000	00000200
ANSD:4809C208	00000000	0CC30001
ANSD:4809C210	00000B00	1DE37F80
ANSD:4809C218	5B590000	400E0032
ANSD:4809C220	00000000	00040000
ANSD:4809C228	00000D02	00090107
ANSD:4809C230	00000000	00000000
ANSD:4809C238	00000000	00000000
ANSD:4809C240	06E90080	00000000
ANSD:4809C248	00000000	00000000
ANSD:4809C250	00000000	00000000
ANSD:4809C258	00000000	00000000
ANSD:4809C260	00000000	00000000
ANSD:4809C268	00000000	00000000
ANSD:4809C270	00000000	00000000
ANSD:4809C278	00000000	00000000
ANSD:4809C280	00000000	00000000
ANSD:4809C288	00000000	00000000
ANSD:4809C290	??????00	


B::d.dump A:0x4809C200--A:0x48(

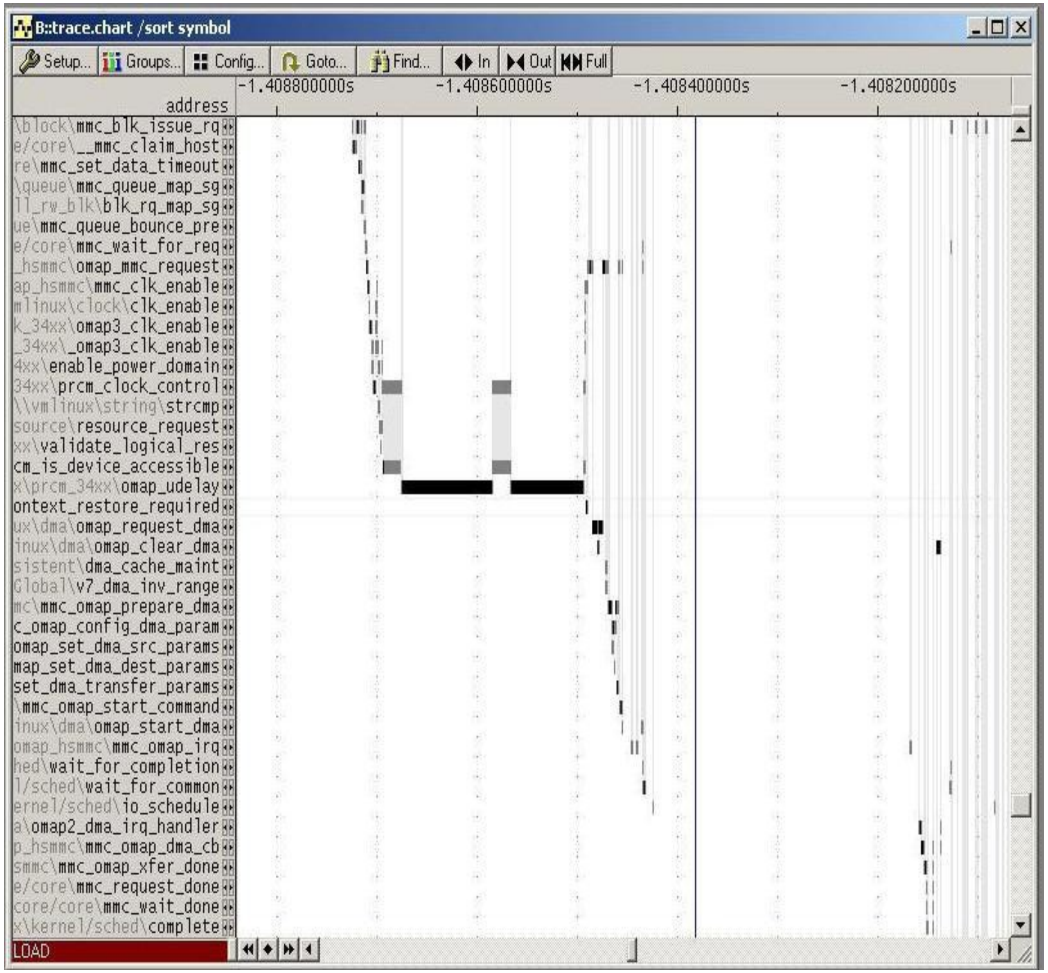
address	0	4
ANSD:4809C200	00000000	00000000
ANSD:4809C208	00000000	01020001
ANSD:4809C210	00000000	00000000
ANSD:4809C218	00000000	00000000
ANSD:4809C220	00000000	00040000
ANSD:4809C228	00000D00	000E0007
ANSD:4809C230	00000000	00000000
ANSD:4809C238	00000000	00000000
ANSD:4809C240	06E90080	00000000
ANSD:4809C248	00000000	00000000
ANSD:4809C250	00000000	00000000
ANSD:4809C258	00000000	00000000
ANSD:4809C260	00000000	00000000
ANSD:4809C268	00000000	00000000
ANSD:4809C270	00000000	00000000
ANSD:4809C278	00000000	00000000
ANSD:4809C280	00000000	00000000
ANSD:4809C288	00000000	00000000
ANSD:4809C290	??????00	

Lauterbach - 2

- Very useful to find regressions when PM enabled.
- Comparing register dumps in both working and non working cases gives a good hint.
  - printer.file ~/working.txt
  - winprint per.view peromap4430.per
  - Enable PM (reproduce regression)
  - printer.file ~/non.working.txt
  - winprint per.view peromap4430.per
  - diff -U 1 ~/working.txt ~/non.working.txt

Embedded Trace Macrocell

- Very Powerful Hardware Tracer.
- Useful for profiling and debugging random lockups, crashes.
- On a 512M ETM, can capture upto 40s of activity. Can increase by limiting the code section which needs to be monitored.



Things are not that simple sometimes:

- Sometimes extremely difficult
- Logical debugging is the key.
- Eliminate the suspects to narrow down.
  - Disable CpuIdle or Disable DVFS to check issue
  - If CpuIdle
    - Narrow down C states
    - only in 1 or across all
    - RET or OFF.
  - If DVFS
    - Eliminate OPPs
    - In 1 OPP or across All.
    - Disable smartreflex

Optimizations:

- Even with Lowpower support, Power numbers are not better. Why ?
- May be you are entering Lowpower very often.
- Profiling needs to be done for few critical sections:
  - GPIO toggling
    - Set and Clear gpios. (Ex: Request in driver for clk enable; Till clock enabled)
    - Accurate since no SW delay involved and Non Intrusive.
    - But difficult to average and get min, max, avg values.
  - ktime_t, ktime_get
    - Lot of processing APIs present. (ktime_sub, ktime_to_timespec)
    - Based on kernel ticktimers (GPT1), which is lost in OFF mode. Hence may be inaccurate sometimes and intrusive
  - 32KHz counter in wkup domain
    - Retained in offmode.
    - Can read 32K counters at appropriate instances.
    - Overflows in 36hours (2^32 / 2^15 / 60 / 60) and intrusive

All modules need not be active at all time:

- Modules involved in Mp3 playback scenario
  - MMC, IVA, sDMA, McBSP/McPDM, TWL audio codec.
- Stages
  - Data Fetch : MMC
  - Data processing (decoding) : IVA
  - Data transfer (sDMA fills McBSP FIFO) : sDMA, McBSP
  - Data send out MsBSP/McPDM to TWL audio codec) : McBSP, Codec
- Do the best you can in what you have: Ex – MMC in Mp3
  - The inactivity time is less.
  - Turning off the LDOs is not possible. (latency in seconds).
  - Just cut the clocks.
  - Will have significant savings.

Idle scenarios

- See to it that appropriate C states are hit as expected.
- Ex: OSIdle (Phone idle, Screen on) – 5 seconds
  - **[bb]root@android $ cat /sys/devices/system/cpu/cpu0/cpuidle/state*/desc**
    - CPU WFI
    - CPUs OFF, MPU + CORE INA
    - CPUs OFF, MPU + CORE CSWR
    - CPUs OFF, MPU + CORE OSWR
  - **[bb]root@android $**
  - **[bb]root@android $ cat /sys/devices/system/cpu/cpu0/cpuidle/state*/usage ; sleep 5 ; cat /sys/devices/system/cpu/cpu0/cpuidle/state*/usage**
    - 320183
    - 1495
    - 22738
    - 50017

    - 321292
    - 1501
    - 22785
    - 50118
  - **[bb]root@android $**

Interrupts:

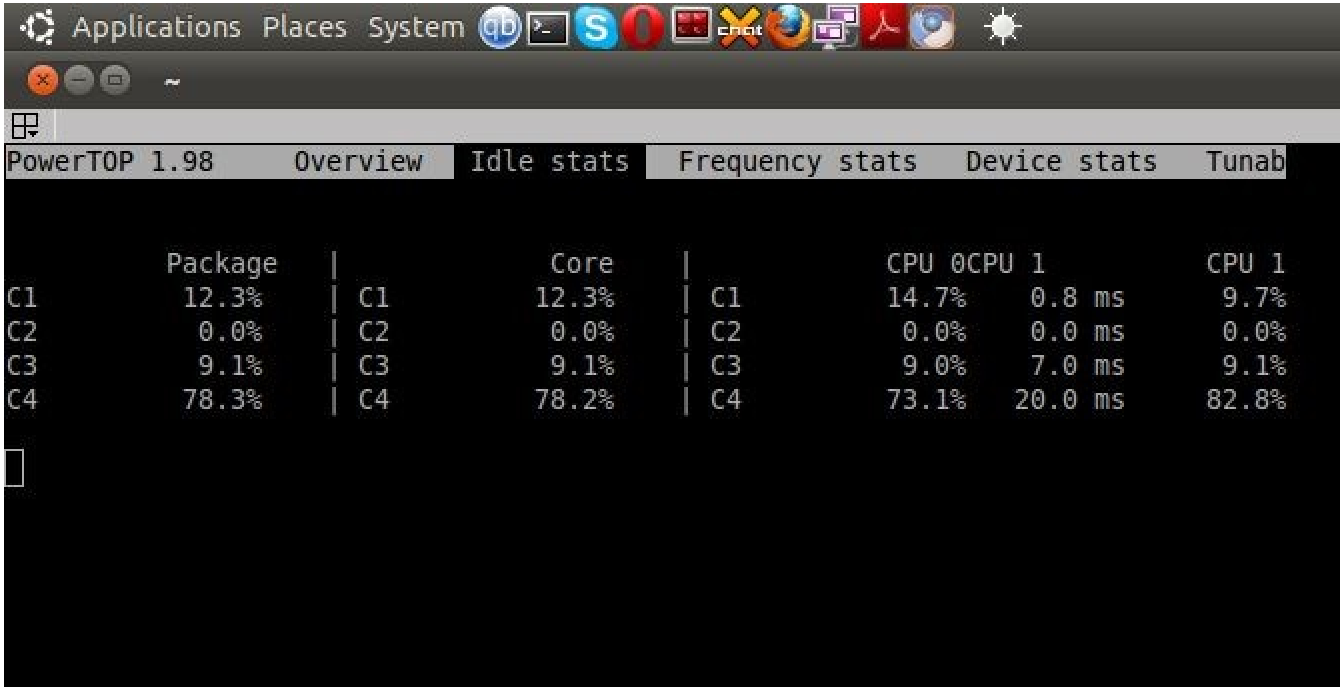
- Too many interrupts wakes the system often
- Might spend less time than expected in Low power state. (More power consumption)
- Ex: During OSIdle (Phone Idle, Screen ON)
  - $
  - $ **cat /proc/interrupts | grep i2c ; sleep 5 ; cat /proc/interrupts | grep i2c**
    - 88: 1595 0 GIC omap_i2c
    - 89: 891 0 GIC omap_i2c
    - 93: 927 0 GIC omap_i2c
    - **94: 1545736 0 GIC omap_i2c**
    - 88: 1595 0 GIC omap_i2c
    - 89: 891 0 GIC omap_i2c
    - 93: 927 0 GIC omap_i2c
    - **94: 1547174 0 GIC omap_i2c**
  - $
- ~1500 interrupts in 5 seconds. Is this expected ?

Tools:

- Powertop
- Powerdebug

Powertop

- Nice tool from www.lesswatts.org
  - git://git.kernel.org/pub/scm/status/powertop/powertop.git
  - git://git.linaro.org/tools/powertop.git
- Can get Idlestats (C state stats) and Freq stats(P state stats)
  - CPU_FREQ_STAT, CPU_FREQ_STAT_DETAILS should be enabled



Powerdebug

- Nice tool from linaro for clock, regulator, sensor.
  - http://git.linaro.org/gitweb?p=tools/powerdebug.git;a=summary
- Settings:
  - export TERM=xterm
  - export TERMINFO=/system/etc/terminfo
- Ex1: find partents of a clock

```
[bb]root@android $ powerdebug -p mmc4_fck
Parents for "mmc4_fck" Clock :
/
 `-- virt_38400000_ck (flags:0x0, usecount:1, rate: 36 MHZ)
    `-- sys_clkin_ck (flags:0x30611000, usecount:5, rate: 36 MHZ)
      |-- dpll_per_ck (flags:0x0, usecount:1, rate: 732 MHZ)
      |    `-- dpll_per_x2_ck (flags:0x815020, usecount:1, rate: 1 GHZ)
      |       |-- dpll_per_m2x2_ck (flags:0x815000, usecount:2, rate: 183 MHZ)
      |       |   |-- func_48m_fclk (flags:0x810800, usecount:2, rate: 45 MHZ)
      |       |   |   |-- mmc4_fck (flags:0x0, usecount:0, rate: 45 MHZ)
[bb]root@android $
```

Powerdebug - 2

- **Ex2: To find change in power states after an event.**
  - After an MMC insertion
    - **powerdebug -d > before.mmc.txt**
    - **Insert MMC card**
    - **powerdebug -d > after.mmc.txt**
    - **[bb]root@android $ diff -U 1 before.mmc.txt after.mmc.txt**

```
--- before.mmc.txt
+++ after.mmc.txt
@@ -28,7 +28,7 @@
        name: VMMC
-       status: off
-       state: disabled
+       status: normal
+       state: enabled
        type: voltage
-       num_users: 0
-       microvolts: 1800000
+       num_users: 1
+       microvolts: 3000000
        max_microvolts: 3000000
```

Powerdebug - 3

- Ex2: continued

```
@@ -295,3 +295,3 @@
        |      |    `-- gpu_fck (flags:0x922000, usecount:0, rate: 146 MHZ)
-       |      |-- dpll_per_m2x2_ck (flags:0x815000, usecount:1, rate: 183 MHZ)
+       |      |-- dpll_per_m2x2_ck (flags:0x815000, usecount:2, rate: 183 MHZ)
        |      |  |-- func_12m_fclk (flags:0x0, usecount:0, rate: 11 MHZ)
@@ -299,3 +299,3 @@
        |      |  |-- hsi_fck (flags:0x933800, usecount:0, rate: 183 MHZ)
-       |      |  |-- func_96m_fclk (flags:0x810800, usecount:0, rate: 91 MHZ)
+       |      |  |-- func_96m_fclk (flags:0x810800, usecount:2, rate: 91 MHZ)
        |      |  |  |-- mcasp2_fclk (flags:0x0, usecount:0, rate: 91 MHZ)
@@ -309,7 +309,7 @@
        |      |  |  |   `-- mcbsp4_fck (flags:0x94e000, usecount:0, rate: 91 MHZ)
-       |      |  |  |-- mmc1_fck (flags:0x932800, usecount:0, rate: 91 MHZ)
-       |      |  |  `-- mmc2_fck (flags:0x933000, usecount:0, rate: 91 MHZ)
+       |      |  |  |-- mmc1_fck (flags:0x932800, usecount:1, rate: 91 MHZ)
+       |      |  |  `-- mmc2_fck (flags:0x933000, usecount:1, rate: 91 MHZ)
        |      |  |-- func_24mc_fclk (flags:0x0, usecount:0, rate: 22 MHZ)
        |      |  |  `-- slimbus2_fclk_0 (flags:0x0, usecount:0, rate: 22 MHZ)
-       |      |  |-- func_48m_fclk (flags:0x810800, usecount:2, rate: 45 MHZ)
+       |      |  |-- func_48m_fclk (flags:0x810800, usecount:3, rate: 45 MHZ)
        |      |  |  |-- mcspi1_fck (flags:0x0, usecount:0, rate: 45 MHZ)
@@ -322,3 +322,3 @@
        |      |  |  |-- ocp2scp_usb_phy_phy_48m (flags:0x0, usecount:1, rate: 45 MHZ)
-       |      |  |  |-- uart1_fck (flags:0x0, usecount:0, rate: 45 MHZ)
+       |      |  |  |-- uart1_fck (flags:0x0, usecount:1, rate: 45 MHZ)
        |      |  |  |-- uart2_fck (flags:0x0, usecount:0, rate: 45 MHZ)
```

[bb]root@android $

Winding up:

- Thanks to:
  - Texas Instruments – for employing me and sponsoring me ☺
  - Wonderful Engineers at TI and in the community, from whom I keep learning everyday.
- Questions ??
- Contact
  - avinashhm@ti.com
  - vishwanath.bs@ti.com
- Thank you All.