Qualcomm TECHNOLOGIES, INC

MSM8974 Android™ Power and Thermal Management

80-NA157-25 B

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

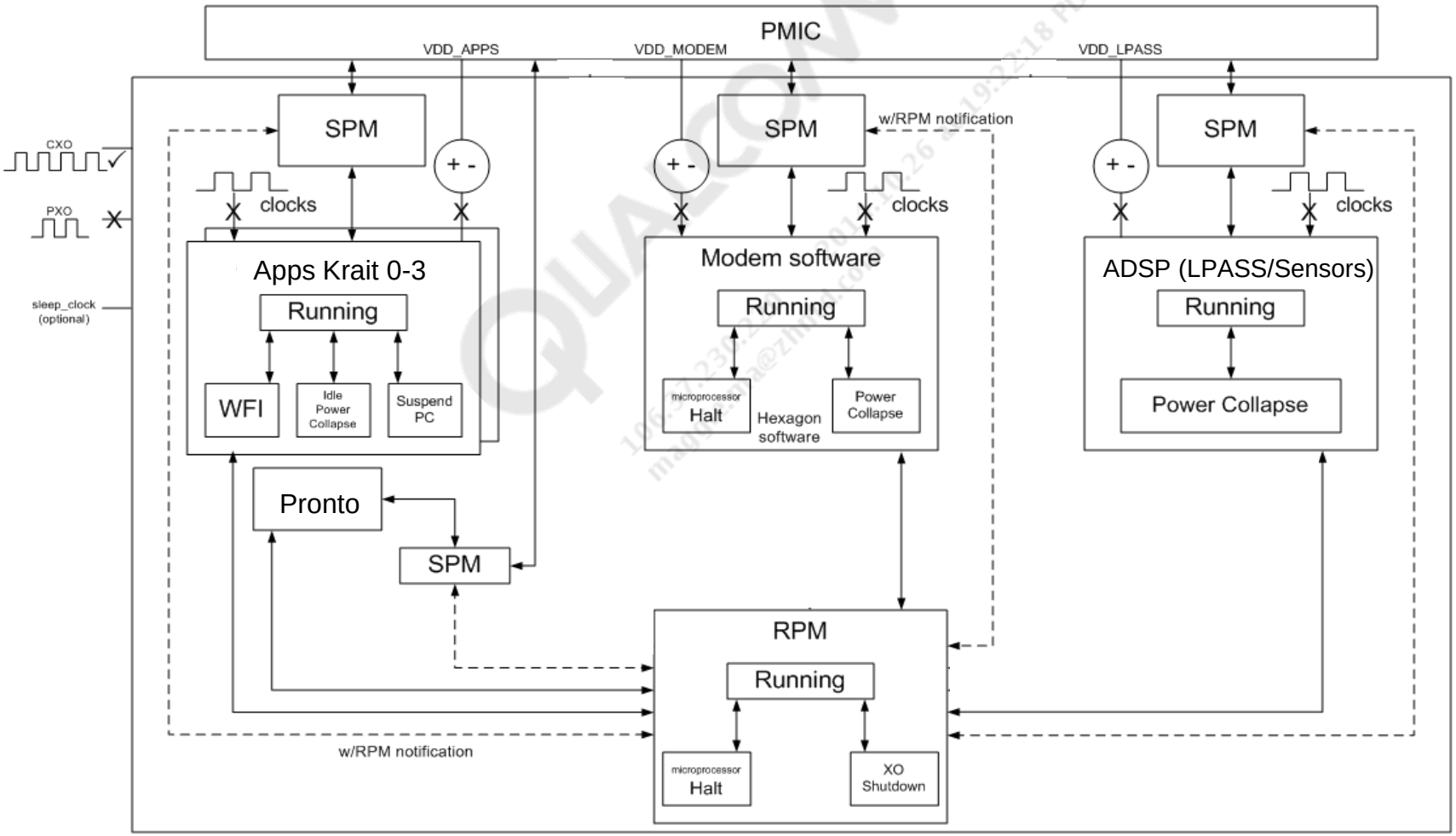
Revision History

Revision	Date	Description
A	Aug 2012	Initial release
B	Dec 2012	Numerous changes were made to this document; it should be read in its entirety.

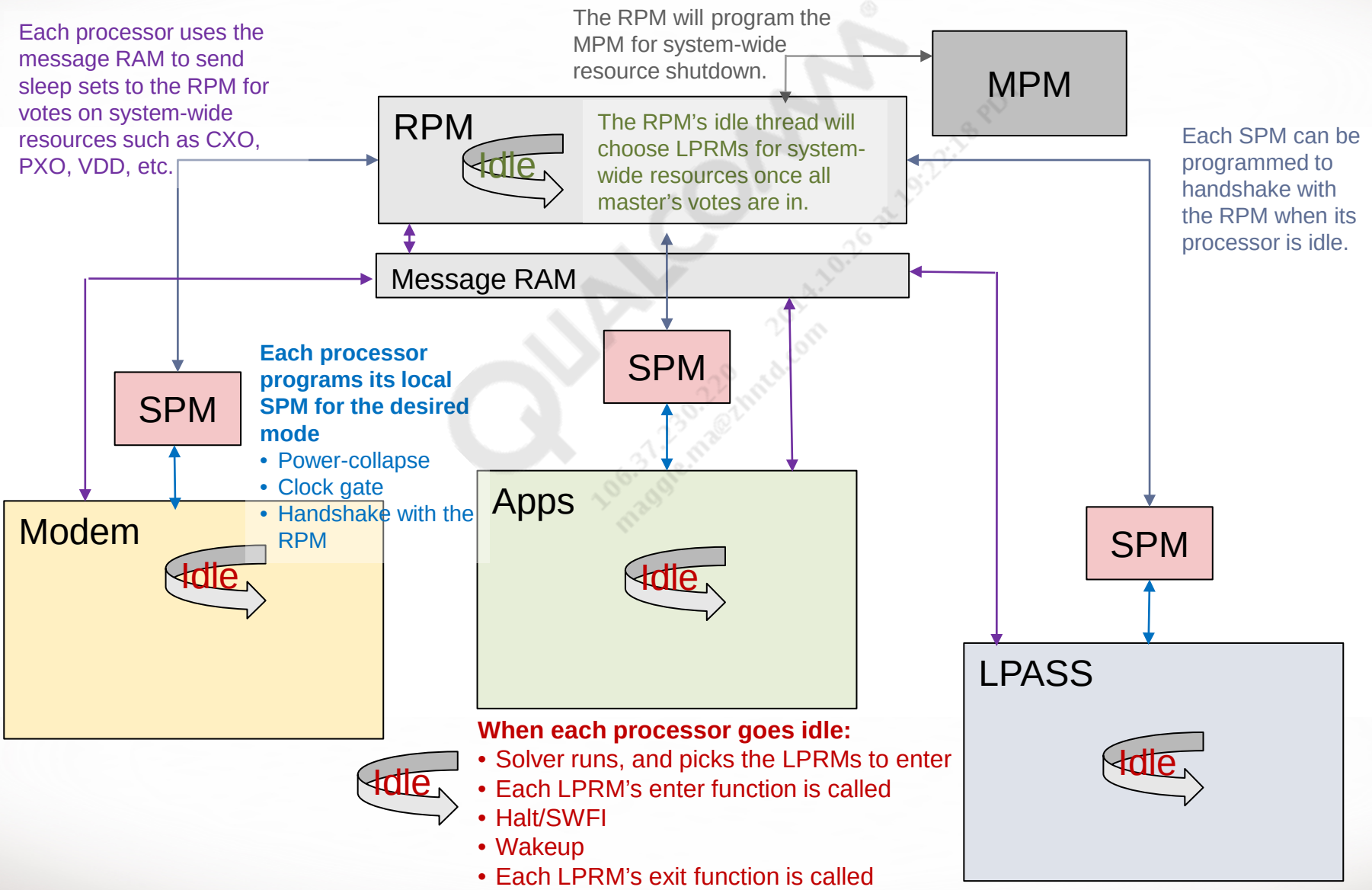
Contents

Contents (cont.)

Power Management Overview

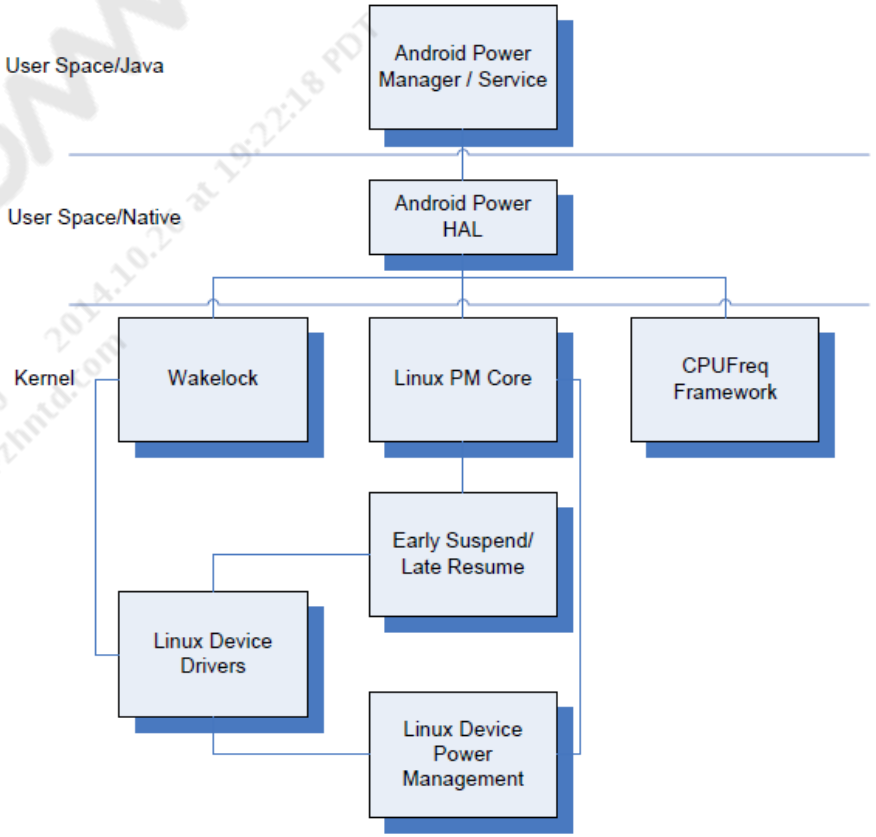


Power Management System Overview



Linux Power Management Framework

- Android-specific PM
  - Wakelock
    - This is used to enable kernel drivers and user space modules to prevent the system from going into low power states.
  - Early suspend/late resume
    - Android power manager/service has an early suspend and late resume mechanism to control devices such as LCD backlight and touchscreen.
    - At wakelock free/expired or user request suspend, the kernel calls the early suspend work queue and executes the registered early suspend handler.
    - When the user (PowerManageService) requests "on" (echo "on" > /sys/power/state), the late resume handlers are called.



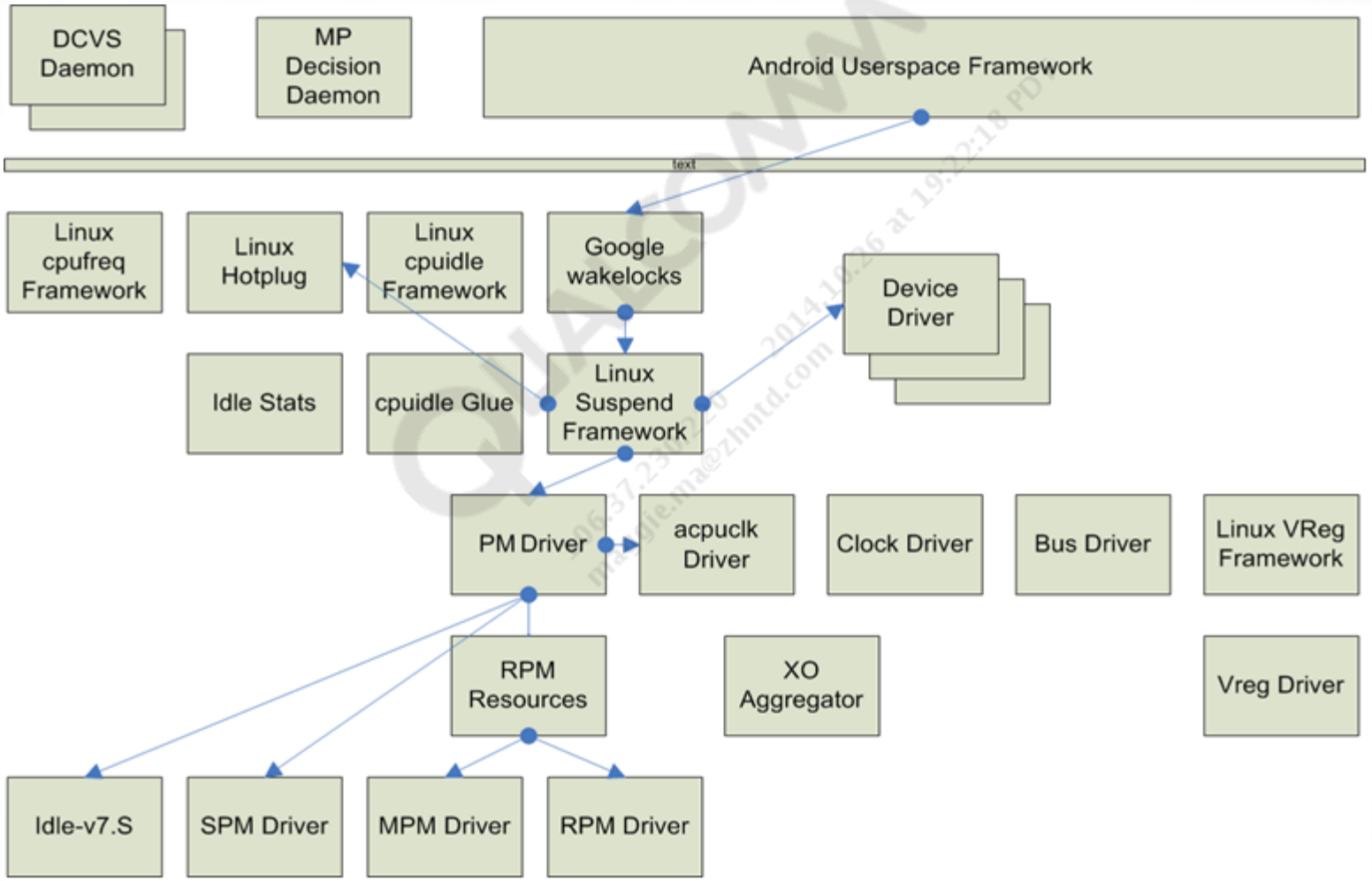
Linux Sleep Model

- In the system sleep model, drivers can enter low power states as part of entering system-wide low power states like Suspend (also known as suspend-to-RAM), or (mostly for systems with disks) hibernation (also known as suspend-to-disk).
- Device, bus, and class drivers collaborate on this by implementing various role-specific suspend and resume methods to cleanly power down hardware and software subsystems and then reactivate them without loss of data.
- Some drivers can manage hardware wakeup events, which make the system leave the low power state. This feature may be enabled or disabled using the relevant /sys/devices/.../power/wakeup file. Enabling the feature may cost some power usage but allows the whole system to enter low power states more often.

Linux Power Modes

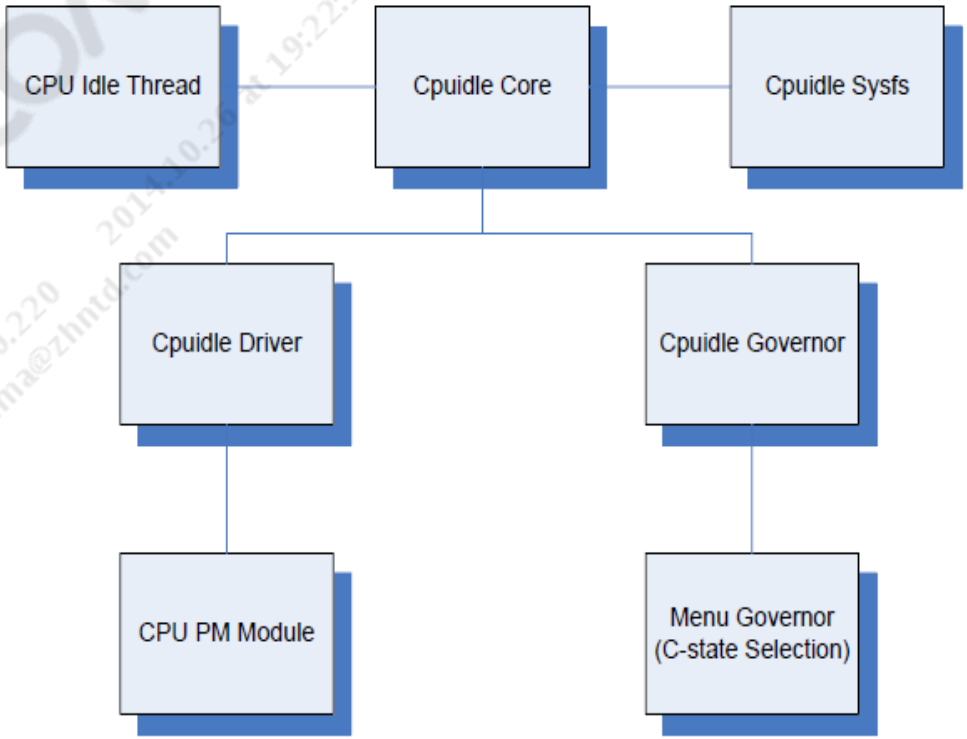
- Supported low power modes
  - SWFI – Clock gating for individual core
  - Retention – Lower voltage in which Krait core is nonfunctional but maintains states along with clock gating for individual core
  - Standalone power-collapse (without RPM notification) – Clock gate and power-collapse for individual core
  - Power-collapse (with RPM notification) – Clock gating and power-collapse for all cores along with LPM on shared resources such as XO, L2 cache, and VDD DIG/MEM, whose combinations result in multiple submodes
- Invocation of low power mode
  - Suspend – Invoked by user or UI and the deepest available low power mode is selected
  - Hotplugging – No need to run a core and a core can be removed or brought back through a file node (/sys/devices/system/cpuX/online) access, and the deepest available low power is selected
  - Idle – A state with no thread to run and proper low power mode is selected based on:
    - Lowest power usage
    - Meet PM_QOS latency requirement
    - Long enough idle residency to break even on energy cost
    - Minimal impact on current system performance

Suspend Call Flow in Linux



Linux Cpuldle Framework

- The Cpuidle framework is integrated into the kernel with sysfs exposing the Cpuidle states and Cpuidle driver and governor configurations to user space.
- The Cpuidle governor determines the idle states (as defined in C state) to enter at any given time, depending on the idle mode's power consumption
- The menu governor is configured as the default idle governor on Qualcomm Technologies, Inc. (QTI's) Linux PM framework, as its state selection considers not only the PM_QOS_CPU_DMA_LATENCY but also the energy cost and performance impact of getting into that idle state.



C States and CPU Low Power Modes

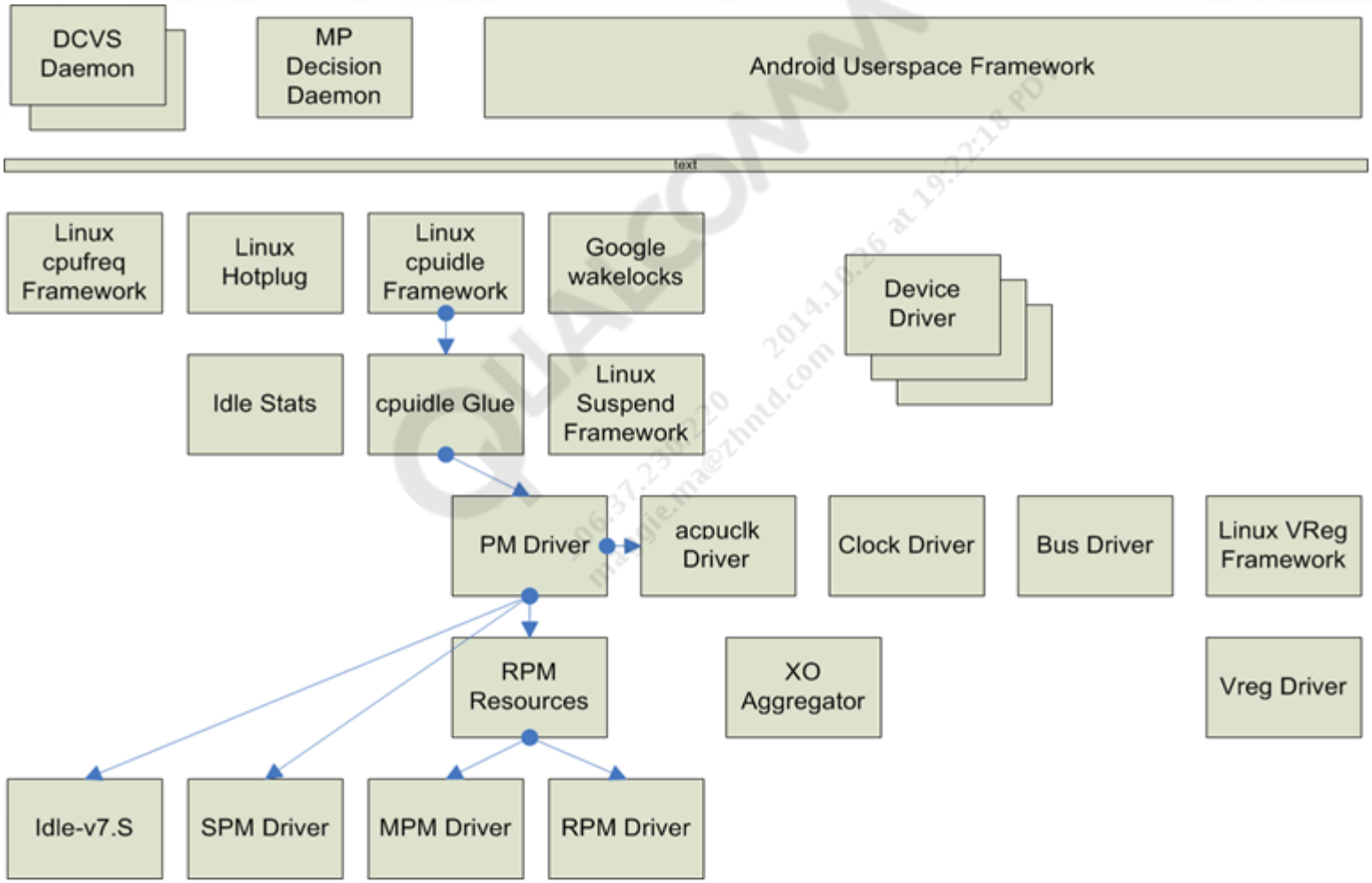
- C states
  - CPU idle states have different power levels.
  - The larger the C-state number, the lower the steady state power level, and the higher the latency and energy cost of entry and exit.
  - The proper state is selected by the menu governor.

C-state	Name	Description	Constraint
C0	Halt/SWFI	CPU Core Clock Domains Off; Power Domains On; QGIC Clock On	None
C1	Standalone power collapse	CPU core clock domains and power domains Off; QGIC Clock/PX0/L2 Cache on; Vdd Mem/Vdd Dig Active	Core0 cannot get into standalone power collapse if any RPM request is pending
C2	Power collapse with RPM notification	CPU clock domains and power domains Off, with different L2 cache, PXO, and Vdd Dig and Vdd Mem configurations	Power collapse with RPM notfication allowed only with other CPU cores hot-plugged out

C2 substate	PXO	L2 cache	Vdd Mem	Vdd Dig
C2_1	On	On	Active	Active
C2_2	On	Off / GDHS	Active	Active
C2_3	Off	On	Active	Active
C2_4	Off	Off / GDHS	Active	Active
C2_5	Off	Off / GDHS	Retention_High	Retention_High
C2_6	Off	Off / GDHS	Retention_Low	Retention_Low

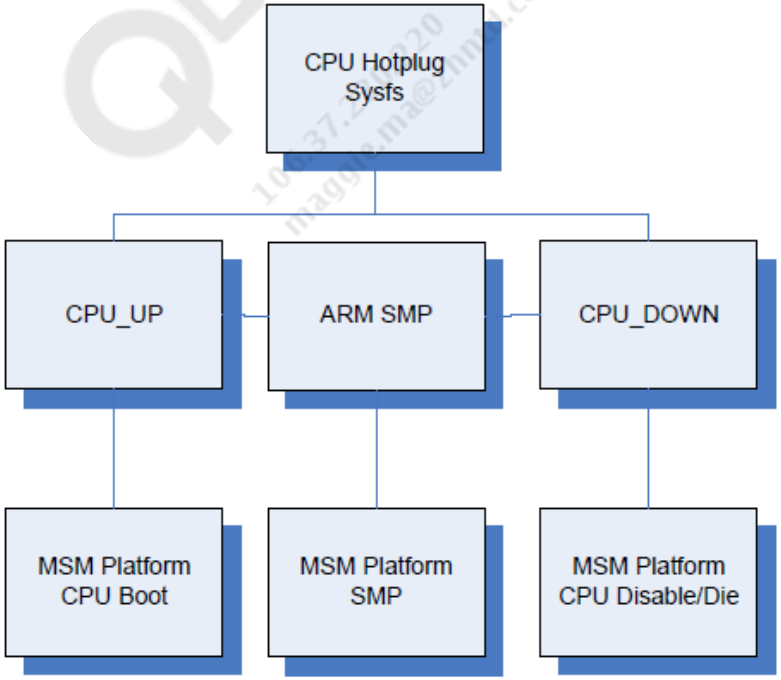
Note: The above table does not reflect the actual MSM8974 C states.

Idle Call Flow in Linux



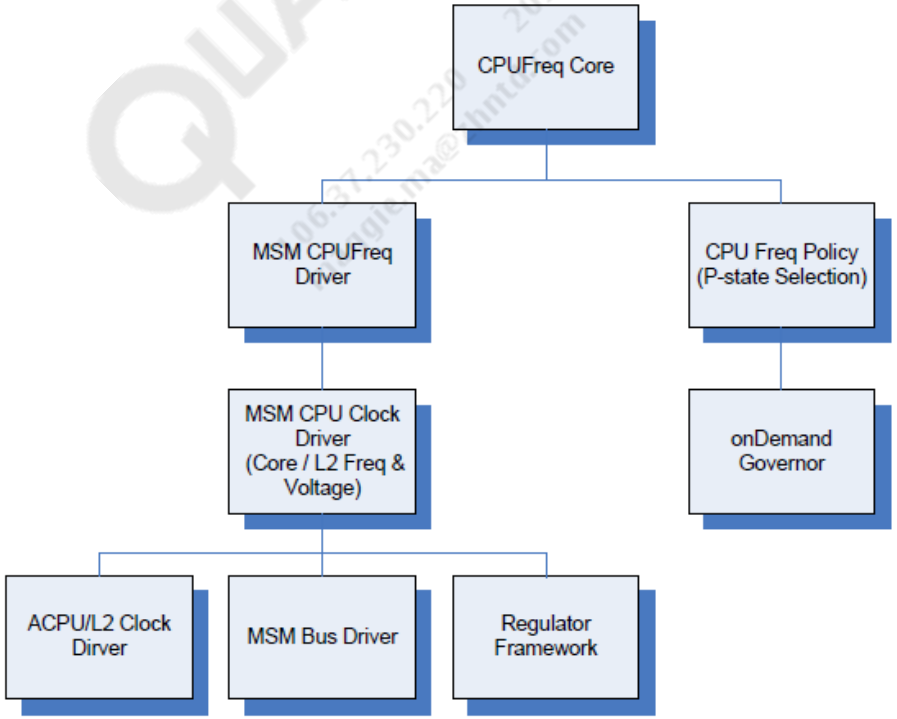
Linux CPU Hotplug Framework

- Linux CPU hotplug provides to the user space processes the capability to put nonboot CPUs online or offline.
- The Linux CPU hotplug framework consists of CPU sysfs APIs (/sys/devices/system/cpu/ cpuX/online).
- All CPU cores can be hotplugged except for CPU0.



Linux CPUFreq Framework

- The frequency to use for a certain CPU (P-state) is determined by the corresponding CPUFreq scaling governor which, in turn, may use an algorithm on its choice to make the decision.
- Governors are the CPUFreq driver components that decide the frequency at which a CPU should be running based on algorithms of their choice.



Krait DCVS/MP-Decision

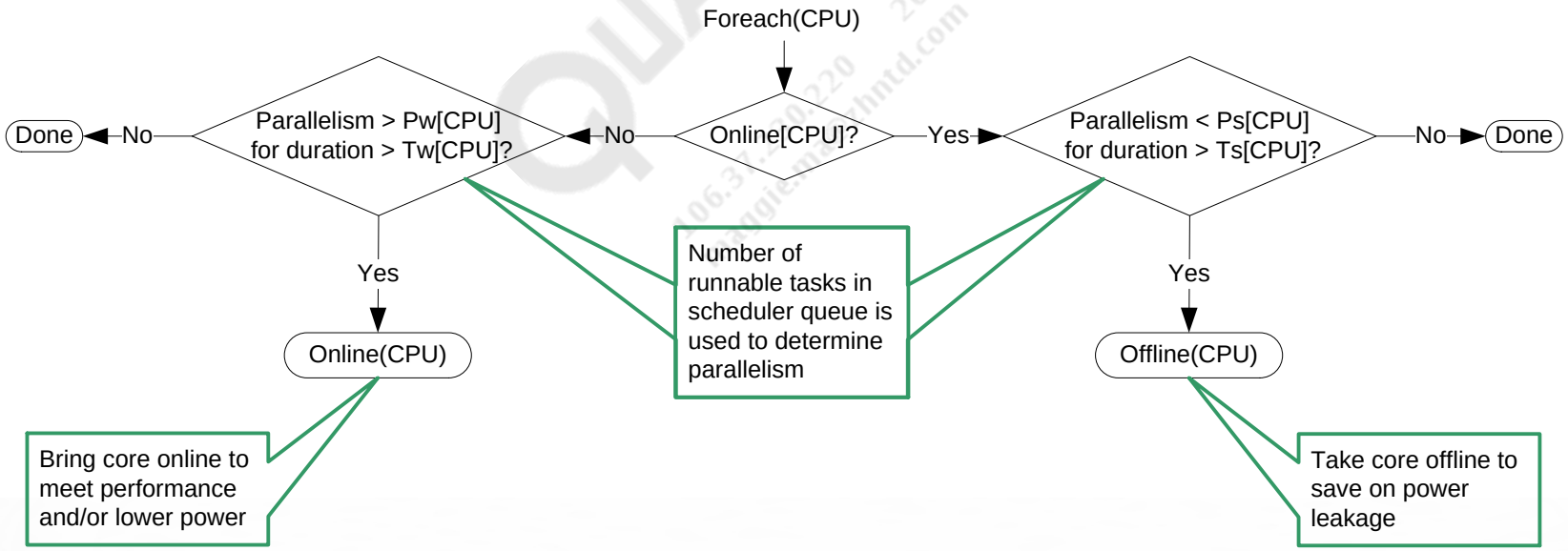
- CPUfreq governor – Diverse governors such as on-demand, performance, and interactive governors are presently supported. This determines at what frequency each CPU runs.
  - Krait DCVS table – Defines frequency levels Krait core supports along with corresponding voltage levels
  - On-demand Governor – Dynamically picks proper frequency level from the DCVS table based on CPU utilization level change
- MP-Decision – This determines the number of active cores based on overall system load and utilizes the hotplug mechanism to remove or bring back cores. The current MP-Decision algorithm takes as input the runQ depth defined as the average of the sum of all the tasks in the active CPU runQs in a fixed time interval.
  - decision_ms – Defines the decision interval in user space or how often the runQ average is read from kernel
  - poll_ms – Defines the polling interval in kernel or how often the runQs are polled to determine the number of tasks in them

MSM8974 Krait DCVS Table (ES build)		
Perf level	**Frequency**	**VDD Kx**
0	300000	0.9500
1	384000	0.9500
2	460800	0.9500
3	537600	0.9500
4	576000	0.9500
5	652800	0.9500
6	729600	0.9500
7	806400	0.9500
8	883200	0.9500
9	960000	0.9500
10	1036800	0.9500
11	1113600	1.0500
12	1190400	1.0500
13	1267200	1.0500
14	1344000	1.0500
15	1420800	1.0500
16	1497600	1.0500
17	1574400	1.0500
18	1651200	1.0500
19	1728000	1.0500
20	1804800	1.0500
21	1881600	1.0500
22	1958400	1.0500
23	1996800	1.0500

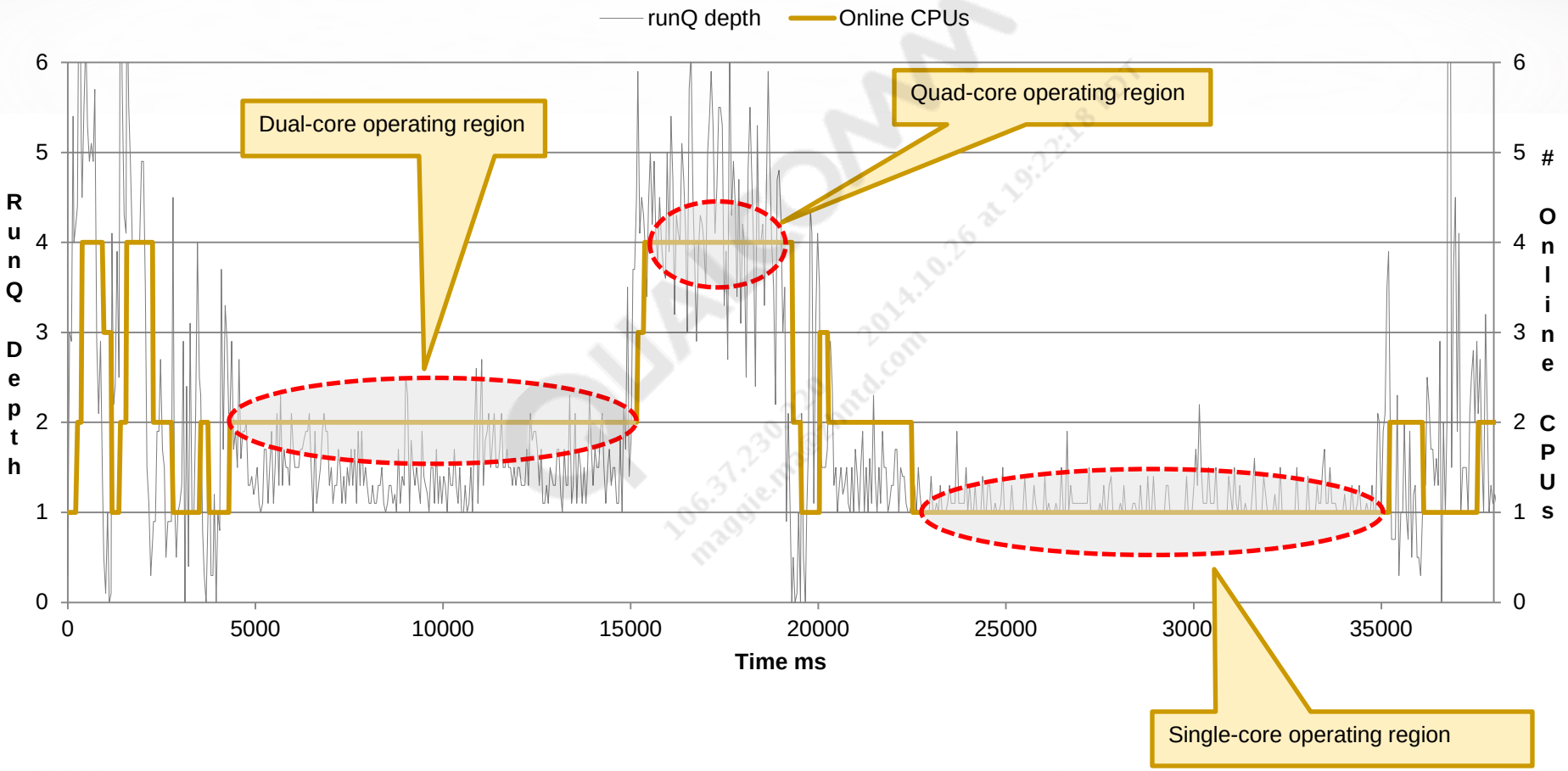
Note: Above table is subject to change for software updates

MP-Decision V1 Algorithm

- Task parallelism is currently determined by monitoring the number of ready-to-run tasks across all OS scheduler queues.
- Parallelism is independent of CPU utilization.
  - 100% CPU utilization does not imply that another core should be powered up.



MP-Decision v1 Algorithm – Parallelism Monitoring



The power and performance benefit of bringing up a core is dependent on:

- The percentage of time that the CPUs would be simultaneously active
- The percentage of time that the CPUs would go into idle power-collapse or SWFI
- The overhead of SWFI, idle power-collapse, and hotplug (power and time)

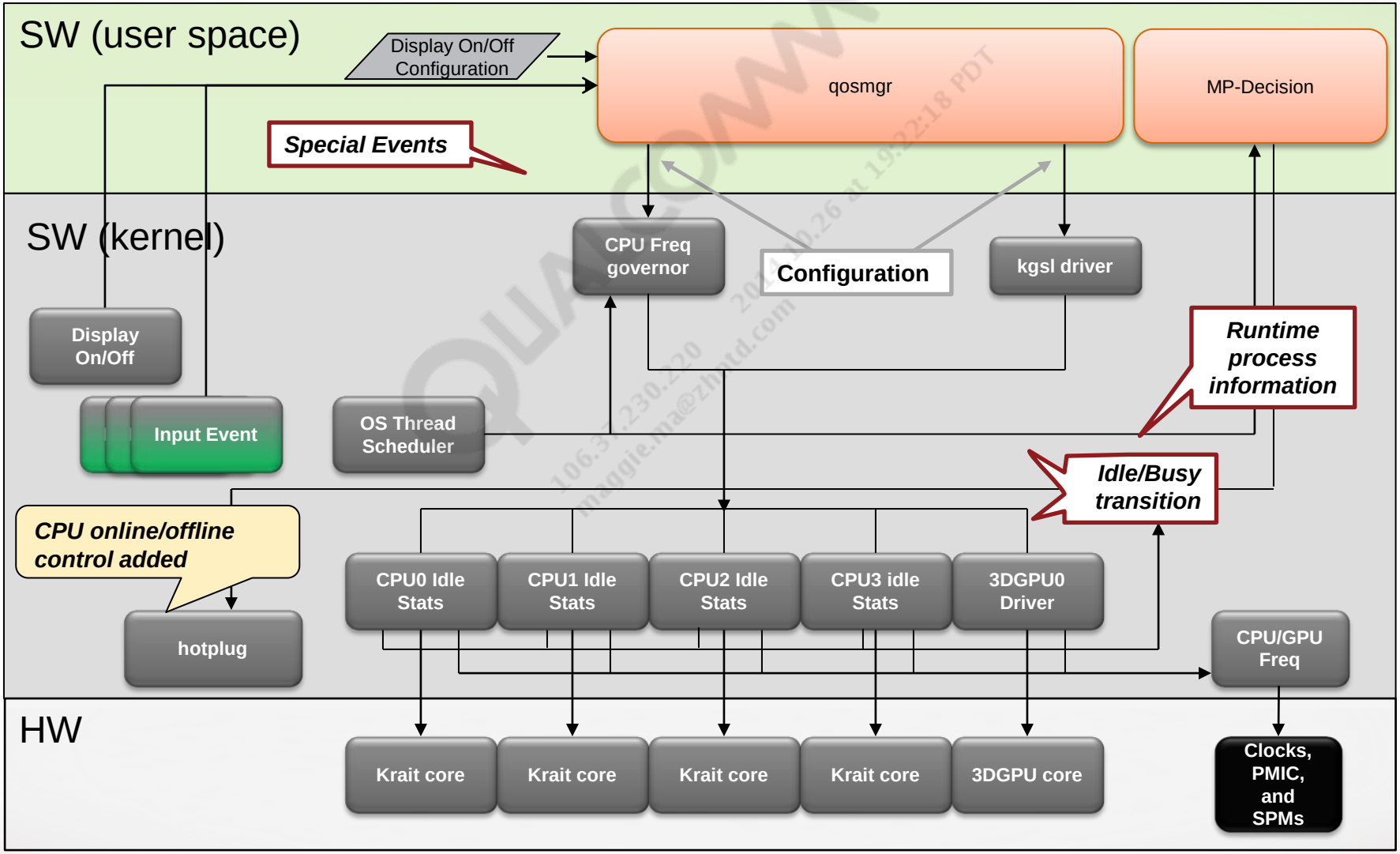
Overheads (ballpark MSM8960):

- SWFI − < 1 ms
- Idle power-collapse − 3 ms to 6 ms
- Hotplug with internal QTI patches <10 ms

qosmgr

- This is used to customize the power/performance of a device on a specific use case basis. It consists of a program for parsing rules and setting assignments and a configuration file that defines those various rules. These rules can be modified or new rules can be added in the configuration file to change/enhance the desired behavior of a device without changing the program.
  - qosmgrd – User space daemon that does all the processing
  - qosmgr_rules.xml – Configuration file that defines the rules

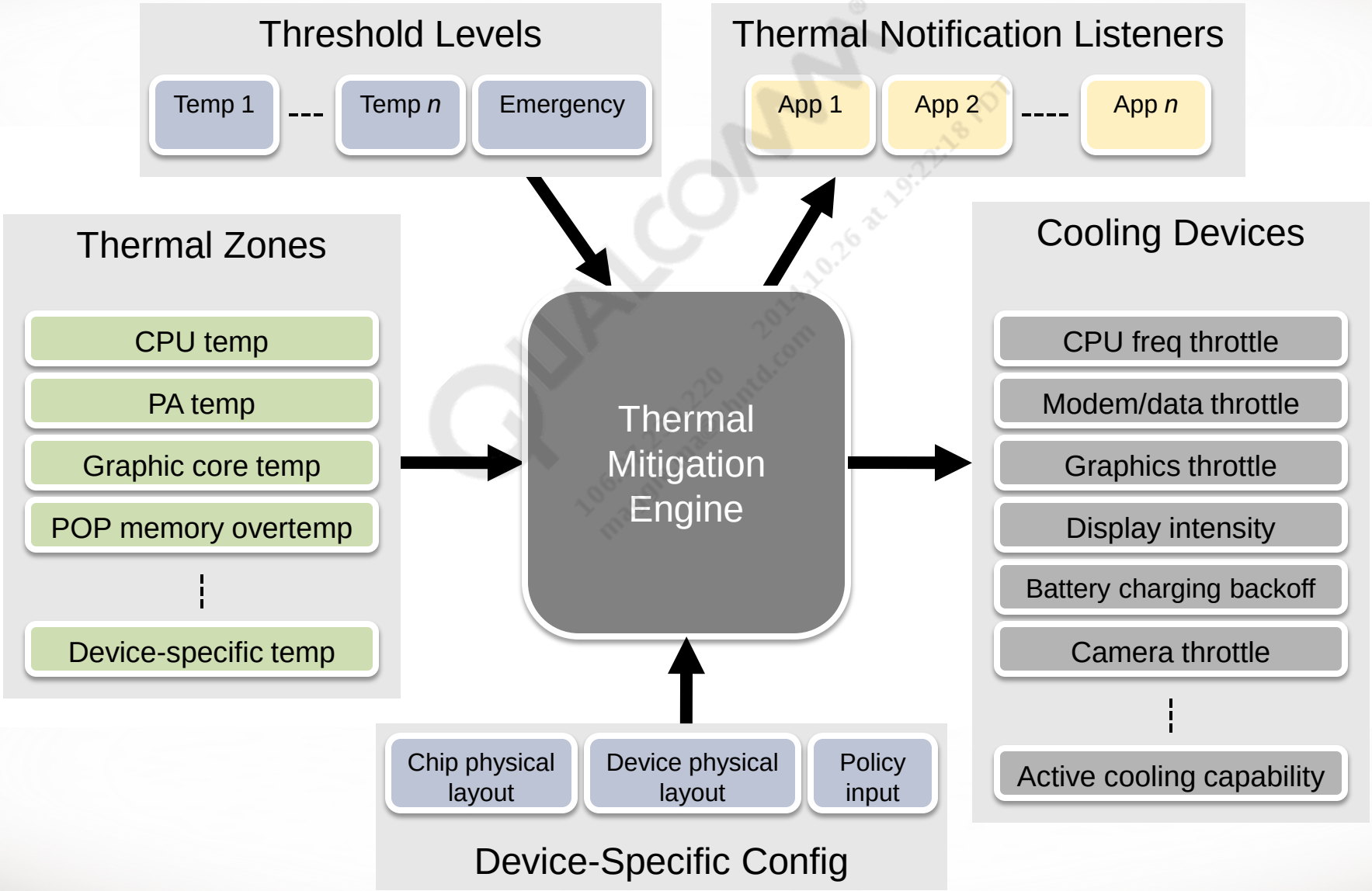
Linux Runtime Power Management Overview



Krait Power Software Change in MSM8974 from MSM8960

- CPU Retention mode
  - Minimize core voltage, core nonoperational but state maintained
- HFPLL inside KPSS
  - HFPLL switch as part of SPM powerup/down sequence
  - Linux apps will not use HFPLL switch in SPM; apps will mostly run at 384 MHz, and that runs global PLL; hence, using an SPM sequence to turn off/on HFPLL is unnecessary and power inefficient
- Quad cores
  - More combo of Krait and L2 low power modes
- RPM communication
  - SMDlite transport instead of legacy RPM message transactions

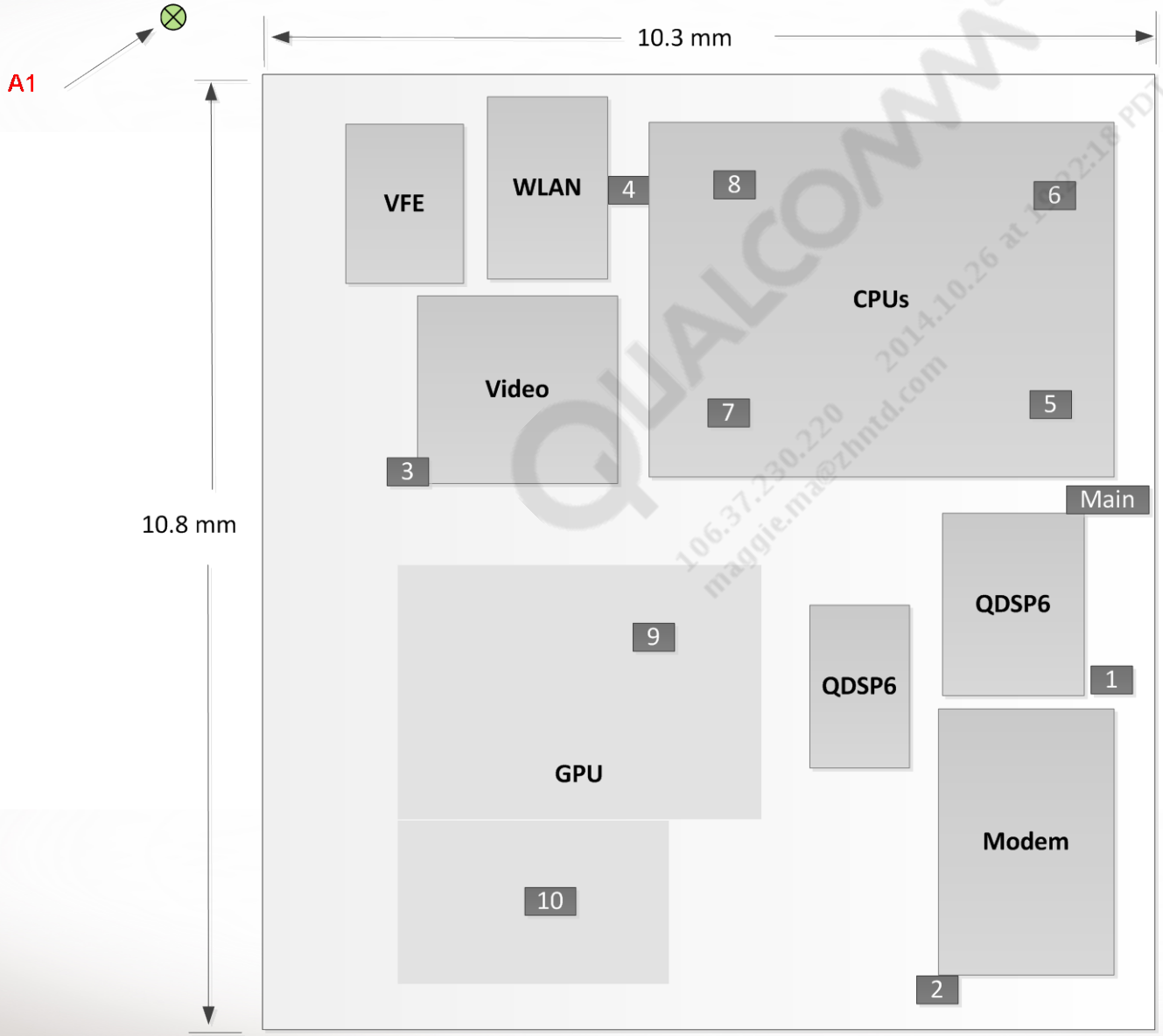
Thermal Mitigation Software Concept Architecture



Temperature Sensors − Placement and Increased Coverage

- Sensors are placed near the hot spots of the Si die
  - On-die sensors were first available on MSM8660 (1 sensor)
  - MSM8960 has 5 on-die sensors
  - MSM8930 has 10 on-die sensors
  - MSM8974 and APQ8064 have11 on-die sensors (see slide)
    - Increases capability to identify and target thermal problems
- Additional sensors (thermistors) are *needed* on the PWB − Near power amplifiers
- On-die sensor accuracy had greatly improved due to new hardware architecture
  - MSM8974 will approach <1.5ºC accuracy

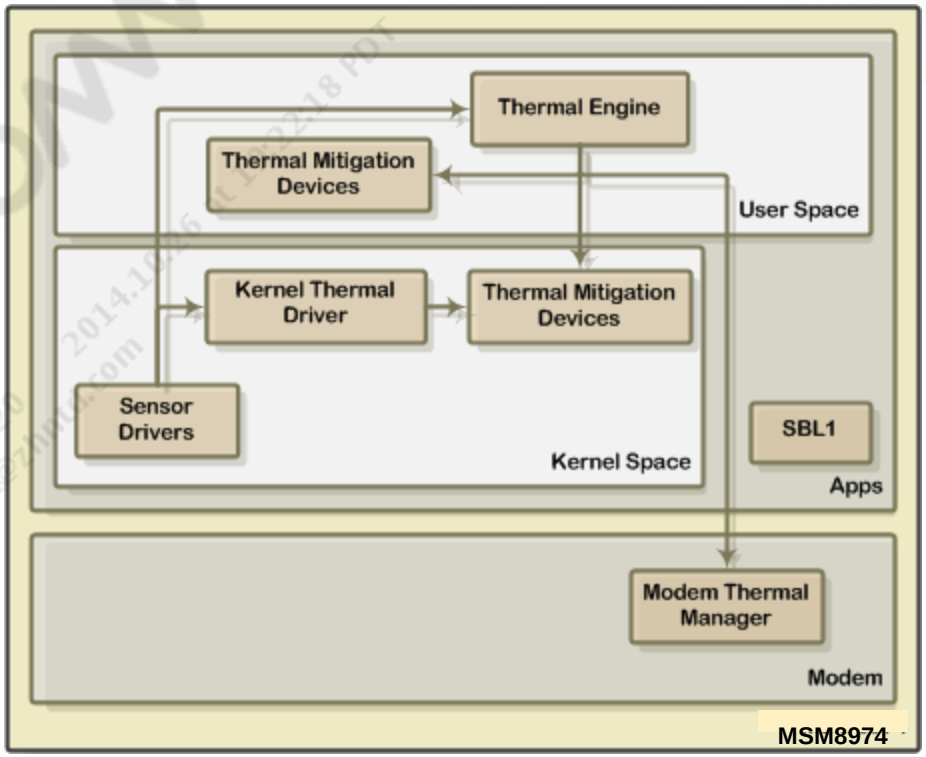
Placement of MSM8974 On-Die Temperature Sensors



- 11 on-die temperature sensors
- Sensors represented by text boxes with proper tsens IDs

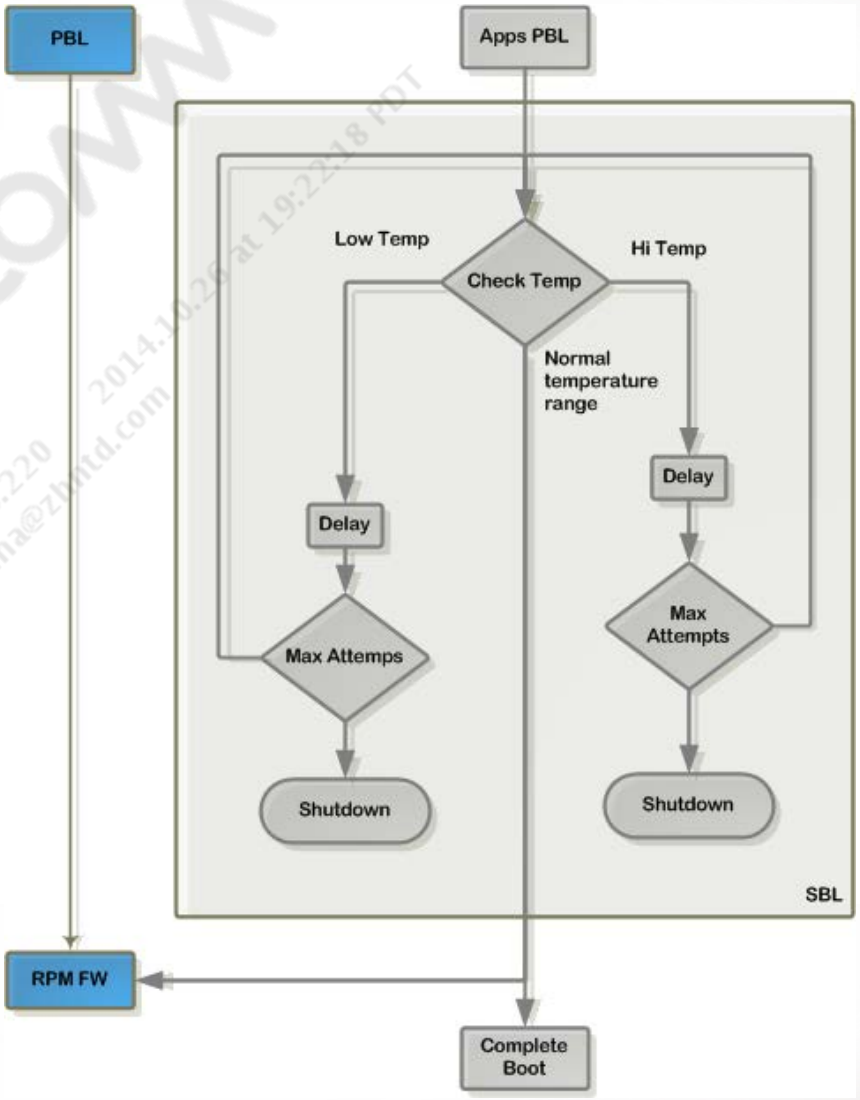
Thermal Management Software Overview

- Motivation for Thermal Management software
  - Manage memory case temperature limits 85C
  - Manage external device skin temperature limits UL 55C for metal and OEM/Carrier typically 45C
- Sensors
  - 11 sensors on the MSM™ die on MSM8974
  - Board thermistors – PMIC, PA, XO, etc.
- Management devices
  - Passive cooling applied by reducing performance
  - Device selection and threshold are configurable by OEM to tune for ID variability through configuration file



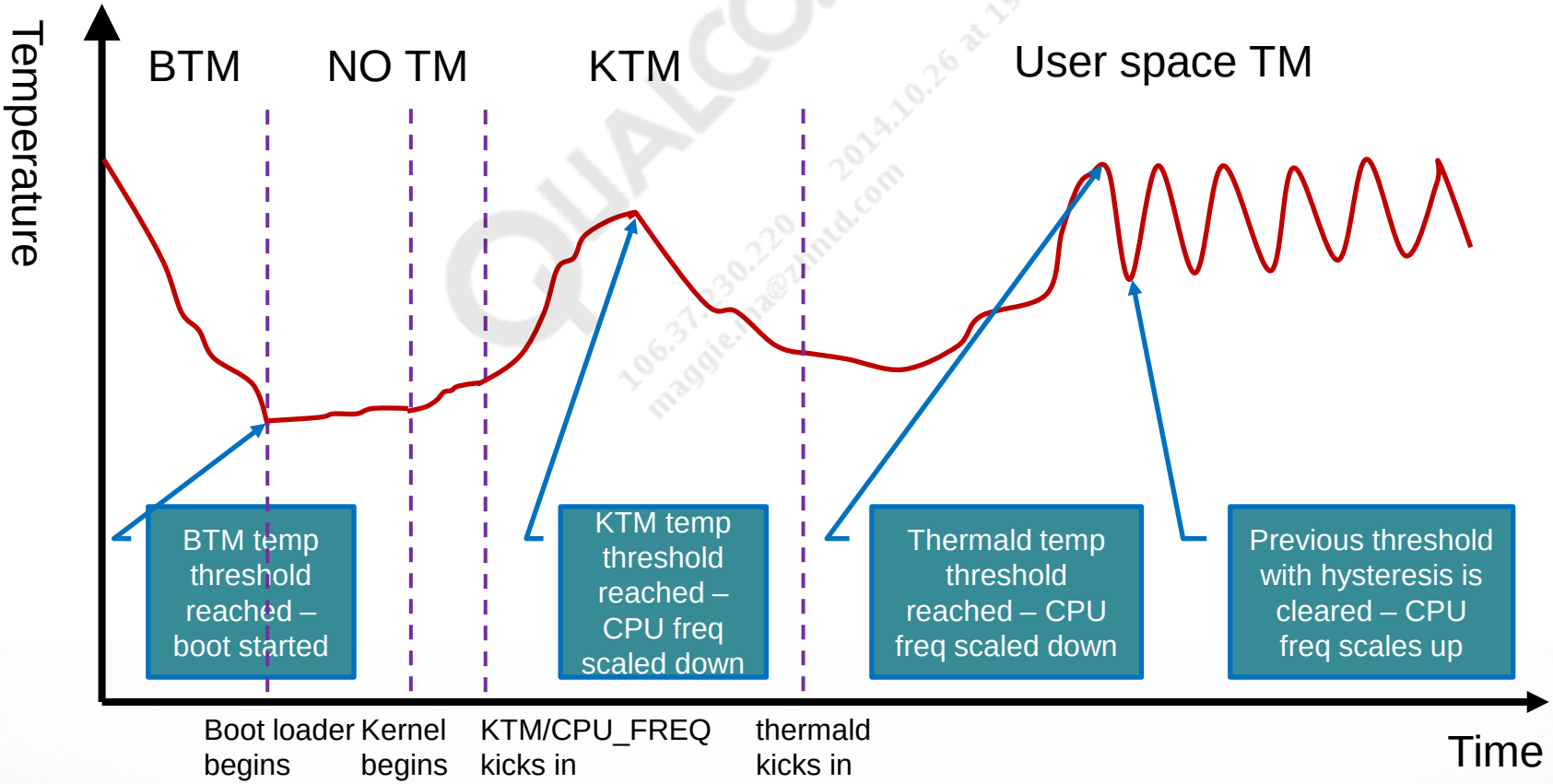
Boot Thermal Management Algorithms

- Boot loader thermal protection
  - Ensures device is in normal temperature region before allowing boot
  - Backs off for a predefined time if temperature is out of normal range
  - Delays and thresholds are configurable



Overall AP TM Mechanism

- Three distinct TMs are provided
- No TM for SBLs and first few seconds after kernel started



Thermal Daemon Configuration Example

```
sampling 1000

[tsens_tz_sensor0]
sampling          1000
thresholds        90            93          96          99        102         105
thresholds_clr    87            90          93          96         99         102
actions             cpu           cpu         cpu         cpu       cpu       shutdown
action_info       1296000 1188000 918000 756000 648000 5000

[pa_therm0]
sampling          1000
thresholds         70           80          90
thresholds_clr     65           75          85
actions              modem    modem   modem
action_info        1             2           3
```


Badger Thermal Management Feature Updates in MSM8974 from MSM8960

- Sensors
  - More accurate sensors
  - Always-on TSENS hardware
  - Independent die sensor and ADC thresholds provide full interrupt support – No more complex Hybrid Polling mode
- Thermal Management controls
  - Port from A family − CPU freq, CPU core control, GPU freq, battery charging, WLAN, backlight, modem (LTE, GSM, UMTS, C2K)
  - New for B family (see FRs) – Camcorder, 0ºC voltage restriction (apps and boot), speaker coil calibration, modem (TD-SCDMA)
- LA thermal daemon updates
  - Rearchitect for new algorithm adoption
  - Dynamic control using only setpoint-based thermal control
  - Threshhold-based control still supported

Thermal Daemon Updates

- Thermal daemon initially commercialized on MSM8660
- Rework thermal daemon to enable integration with Sensor Manager and allow for multiple algorithms – Thermal Management Engine (TME)
  - TME supports legacy and advanced algorithms running in parallel. OEMs will be able to choose the existing algorithm or the new algorithm in the config file.
  - The new algorithm is expected to reduce tuning effort, improving average DMIPS.
- TME configuration examples

```
Sampling        1000

[PMIC_THERM_MON]
algo_type       monitor
sensor          PMIC_THERM
sampling        5000
thresholds      40200       45000       50000
thresholds_clr  38000       43000       48000
actions         cpu+report  cpu         cpu
action_info     1188000+0   368640      245760
```

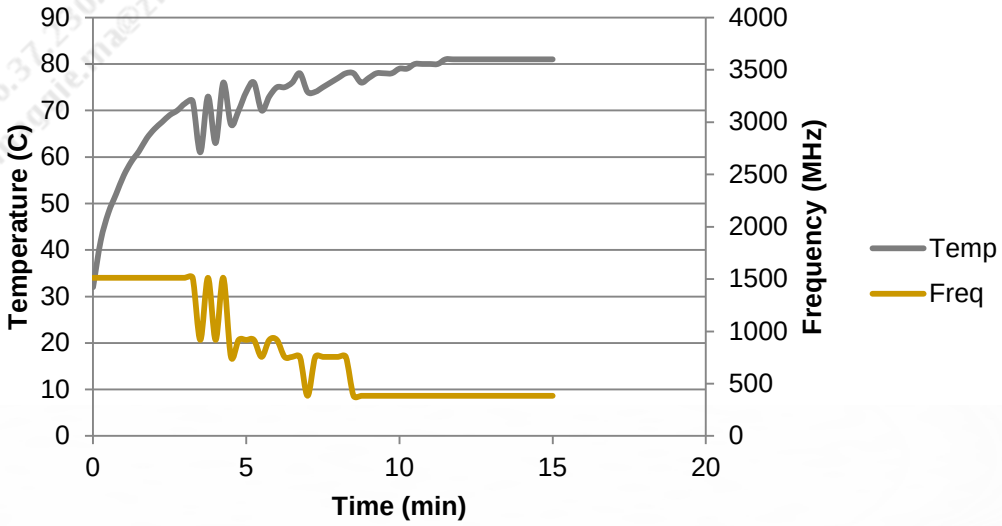
```
debug

[TEST_PID]
algo_type       pid
Sensor          tsens_tz_sensor0
Device          cpu
Sampling        1000
set_point       85000  // threshold for PID mitigation and associated set point
set_point_clr   65000  // threshold at which to stop PID mitigation calculation
p_const         1.0
i_const         1.0
d_const         1.0
i_samples       10
dev_units_per_calc          10000
```

Existing Algorithm

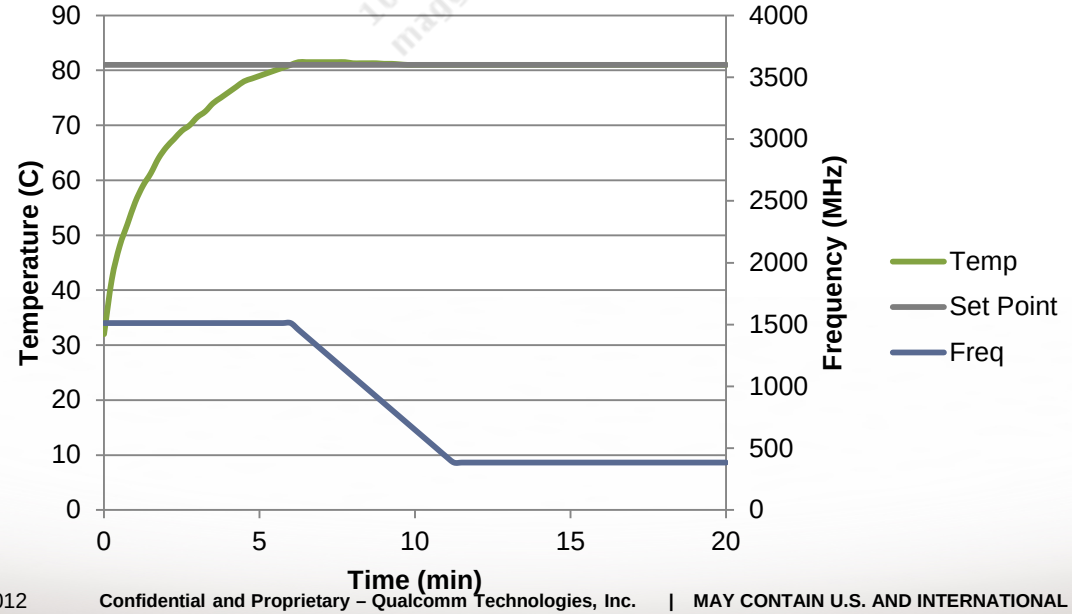
- Measured temperature crosses a predefined threshold then sets predefined mitigation level
  - When sensor temp reaches 72ºC, CPU frequency is reduced from 1512 to 918 MHz, etc.
  - Final sensor temperature is maintained at 81ºC
  - Average DMIPS in this example is 2487 (3.1 DMIPS/MHz)

	Level 1	Level 2	Level 3
Threshold set	72 C	75 C	78 C
Threshold clear	65 C	72 C	75 C
CPU frequency	918 MHz	756 MHz	384 MHz



Real-Time Algorithm Example (Estimated Response)

- Setpoint-based control
- When temperature crosses setpoint, reduce performance until temperature stabilizes
- Setpoint in this example is 81ºC
- Reduce performance one step in DCVS table at each sampling interval while above threshold
  - Final sensor temperature is maintained at 81ºC
  - Mitigation starts later 6 min vs 3.5 min
  - Average DMIPS in this example is 3200 (3.1 DMIPS/MHz)



References

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1

Questions?

https://support.cdmatech.com