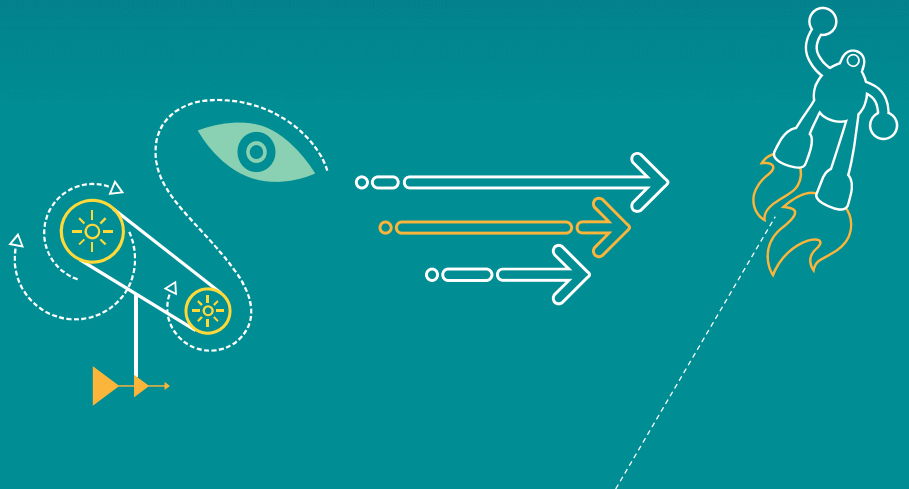
Android Power Overview

Qualcomm Technologies, Inc.



80-P0956-1 A

Confidential and Proprietary – Qualcomm Technologies, Inc.

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

Revision History

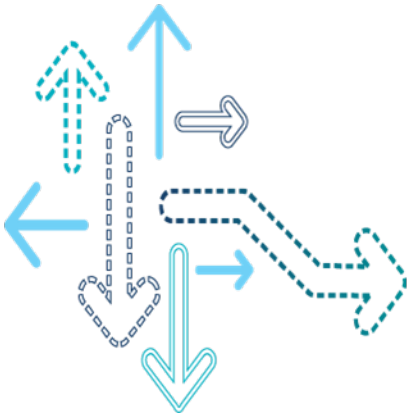
Revision	Date	Description
A	May 2015	Initial release

Contents

Objectives

- At the end of this presentation you will understand
  - Hardware power tree and usage
  - Basic system power features
  - Voltage and clock plans
  - CPU governor
  - Subsystem DCVS
  - Modem and Multimedia use case power

Power Documents



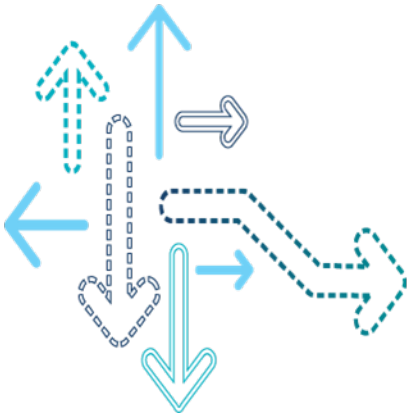
Useful Reference Documents

Document category	Chipset	DCN reference	Comments
Master reference document	MSM8909	TBD	Provides document reference numbers for all power-related documents for a chipset
	MSM8952	TBD	
	MSM8992	TBD	
	MSM8994	80-P0958-1	
	MSM8996	TBD	
Dashboard goals and use case clock plan	MSM8992	TBD	Provides information about the data flow of the dashboard use cases and the frequencies for various clocks
	MSM8994	80-NM328-128	
	MSM8996	TBD	
Power measurement procedure	Common	80-N6837-1	Provides detailed instructions on setup and procedures to be used for measuring power
Power optimization and debug	General	80-P0955-1	Provides general guidelines, tools, logging methodologies, and instructions for debugging power-related issues

Useful Reference Documents (cont.)

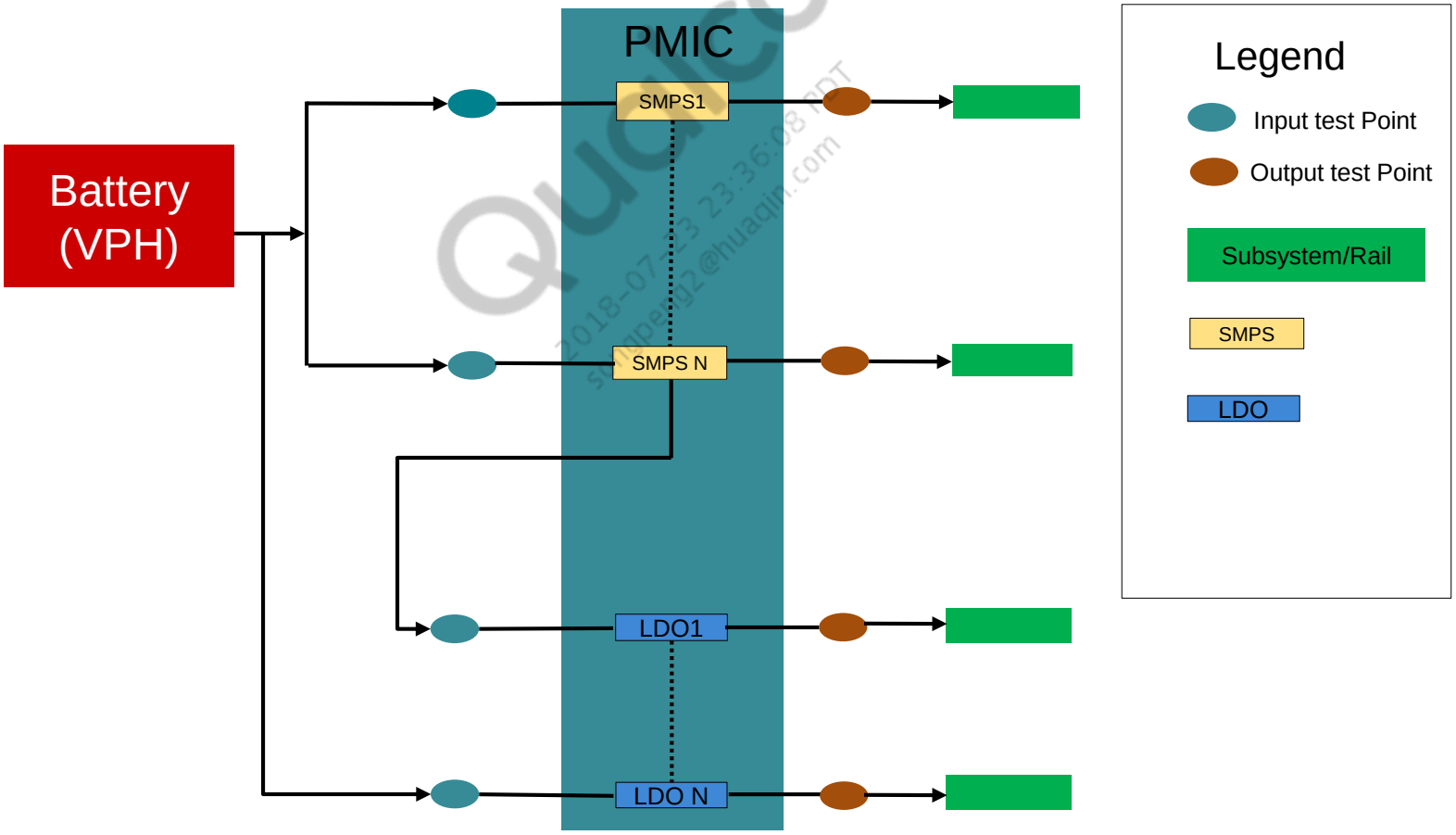
Document category	Chipset/system	DCN reference	Comments
Chipset power overview	MSM8909	80-NR964-5	Advanced system power design and architecture
	MSM8936/MSM8939	80-NM384-2	
	MSM8952	80-NV610-5	
	MSM8992/MSM8994	80-NM328-15	
	MSM8996	80-NV396-15	
Hardware reference documents	LDO FAQ	80-VT310-125	Reference hardware documents
	AVS	80-N8715-14	

Hardware Power Overview



System Power Interconnect Overview

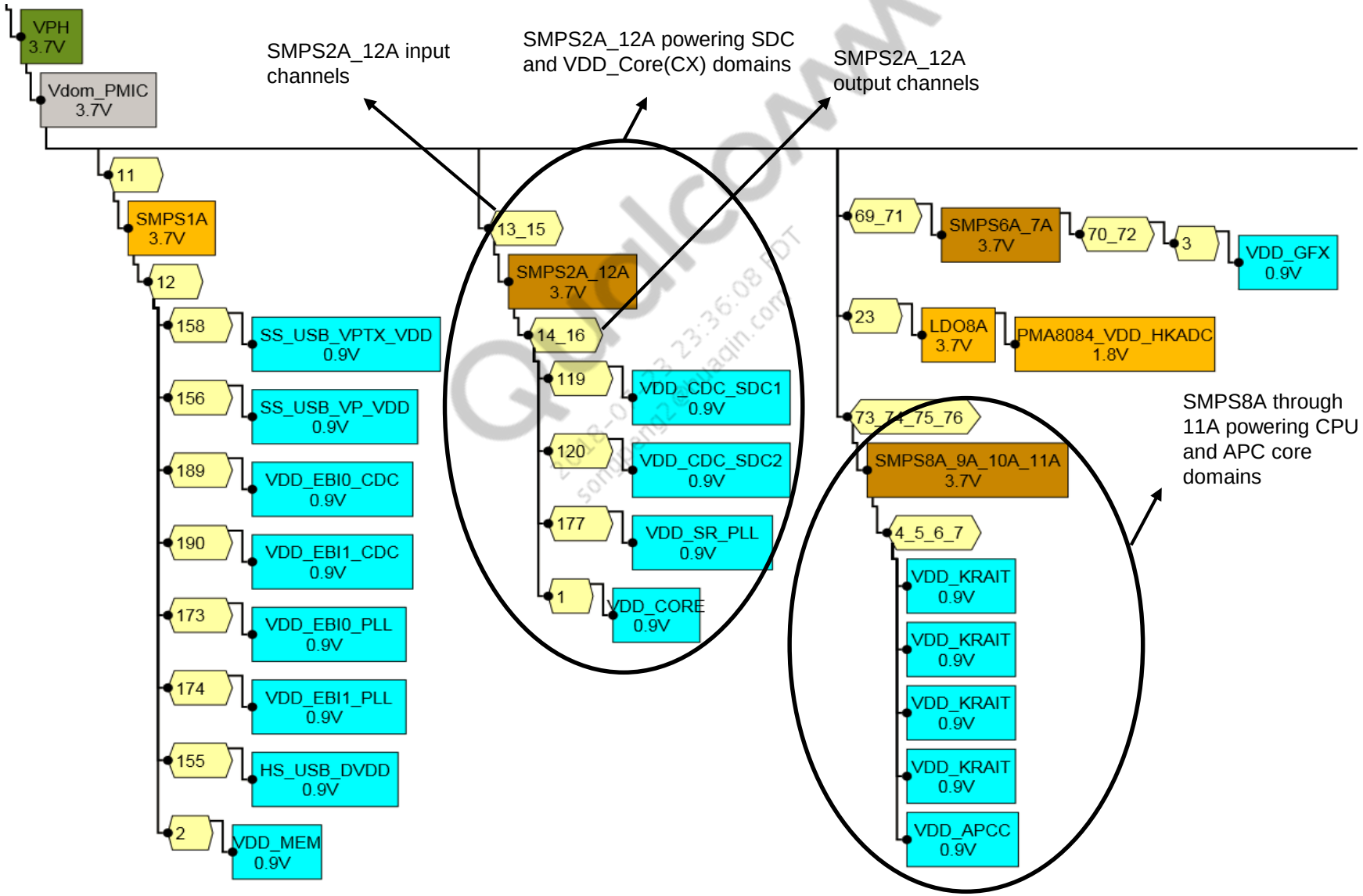
- Subsystem in the figure refers to a module of the device like Display, Camera, etc.
- Rail refers to VDD_CORE,VDD_MEM,VDD_GFX,VDD_APCn, etc.



Why Power Tree

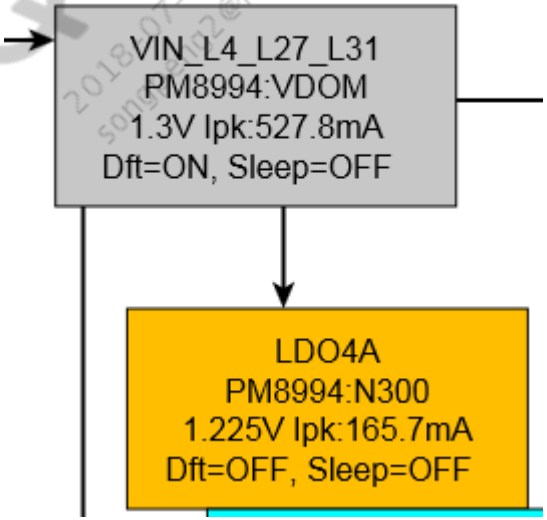
- Power tree shows the current flow from Battery→PMIC Regulators→ Subsystem/rail.
- Power tree shows the channel/test point number for the rail-level power breakdown.
- The rail-level breakdowns are used in debugging power issues to find the subsystem/rail contributing to high power numbers.
- A PMIC is the primary interface between battery and system components.
  - All regulators (SMPSs and LDOs) used to power system components are inside the PMIC.
  - These regulators provide and maintain required voltage and current for the subsystems/rails.

Power Tree



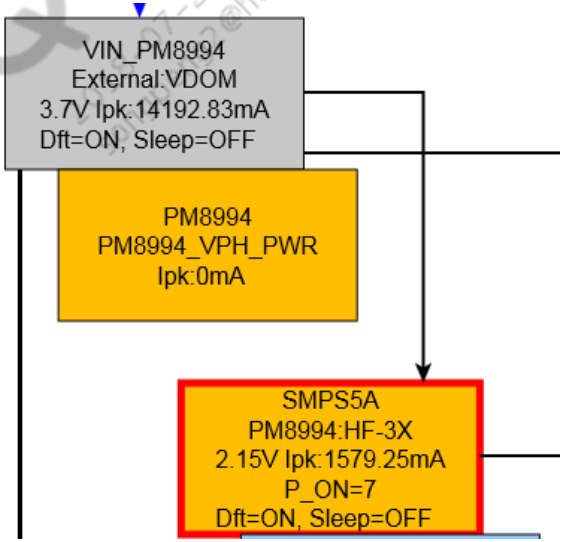
LDO

- A low-dropout, or LDO, regulator is a DC linear voltage regulator that can operate with a very small input–output differential voltage.
- LDOs are suitable for scenarios where the voltage difference across the regulator is low.
- Higher difference implies more loss and more heat dissipation.
- Here we see an LDO with input from the PMIC and the sleep configuration.



SMPS

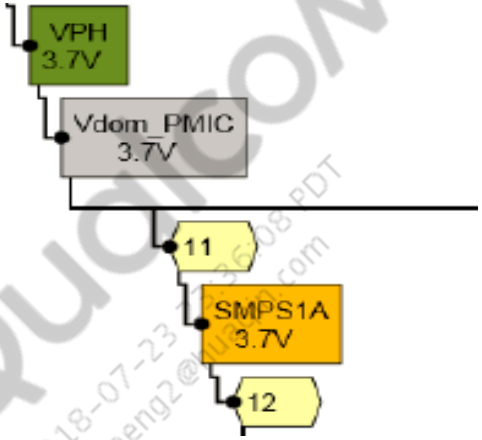
- Switched Mode Power Supply is an electronic power supply that incorporates a switching regulator for higher efficiency for high voltage differential scenarios.
- Advantages of SMPS are:
  - Greater efficiency because the switching transistor dissipates little power when acting as a switch
  - Lower heat generation due to higher efficiency



SMPS with input 3.7 V→Output 2.15 V

Correlating Power Tree to Breakdown

- This is the SMPS1A from the power grid of the APQ8084. Here the input channel is 11 and output channel is 12.

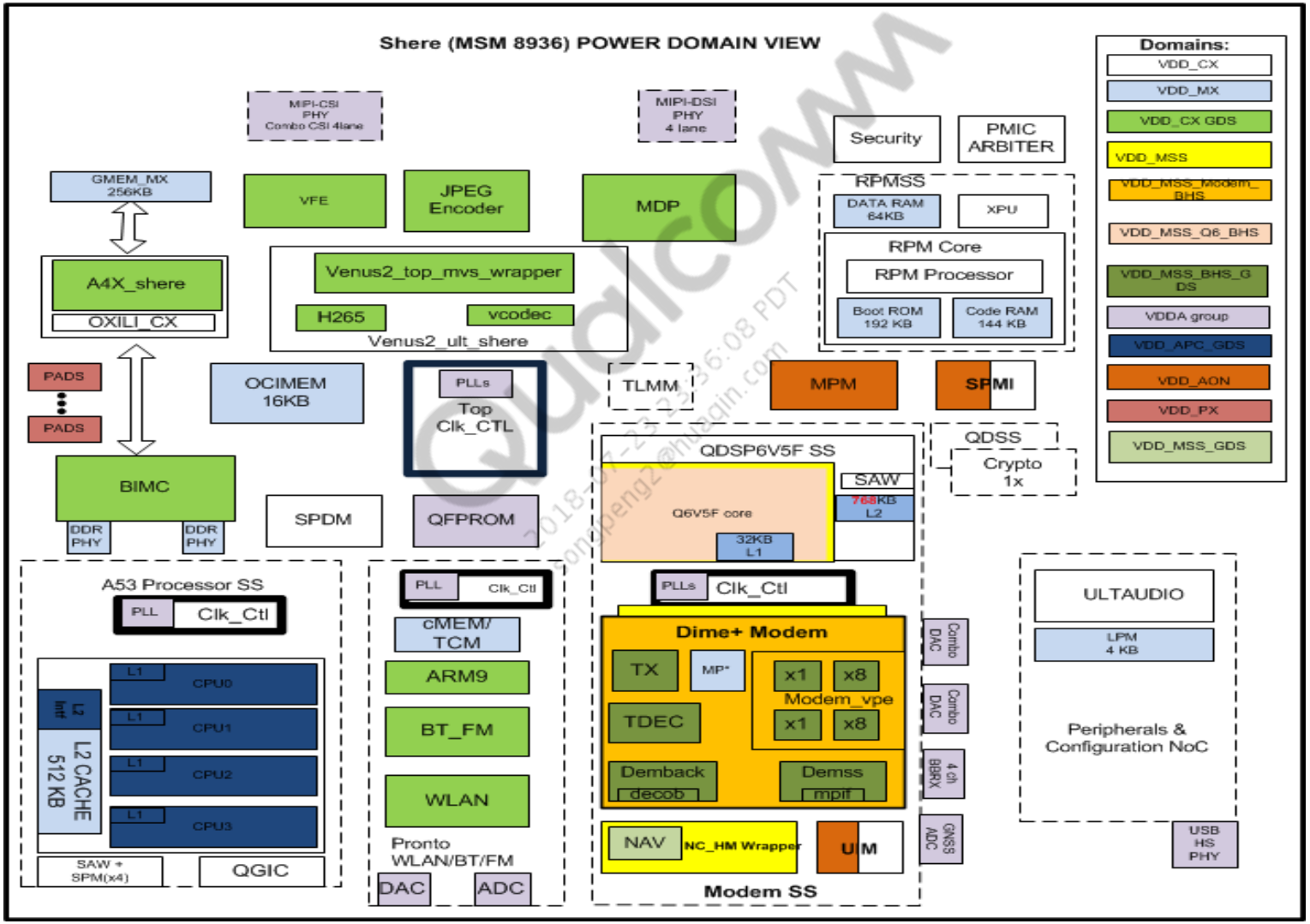


- This is the rail-level breakdown sheet output.

11			PMIC	SMPS1A_input	3.640 V	9.31 mA	9.31 mA	33.88 mW
12			PMIC	SMPS1A_output	1.117 V	25.54 mA	7.82 mA	28.53 mW

Note: If you do not have breakdown boards, contact QTI to obtain SPM boards; see *System Power Monitor Version 4 Overview (SPMv4)* (80-N6594-20) and *System Power Monitor Version 4 Application Note (*80-N6594-16).

MSM8936 Power Domain View Diagram

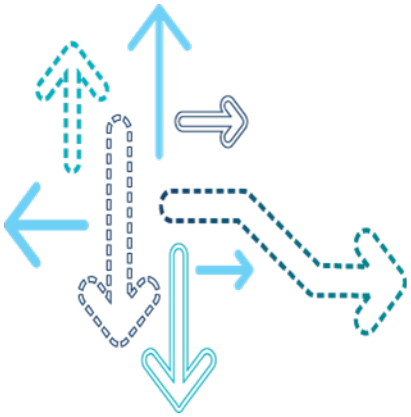


*Available in the *MSM8936/MSM8939 System Power Overview* (80-NM846-2)

MSM8936 Voltage Domains Table

Voltage domains on MSM8936		
Subsystem	**Block**	**Domain**
MPM	—	VDD_AON (Always on)
Memories (L2 Cache, OCMEM)	—	VDD_MX
Multimedia	MDP	VDD_CX
	VFE	VDD_CX
	Venus	VDD_CX
	Q6V5F SS	VDD_CX
	Oxili	VDD_CX
	CAMSS	VDD_CX
	MMSSNOC	VDD_CX
Modem	NAV	VDD_MSS
	Q6V5F Core	VDD_MSS
	Tx	VDD_MSS
	TDEC	VDD_MSS
	MEMPOOL	VDD_MSS
	MPIF/Demss	VDD_MSS
	DemBack	VDD_MSS
APC	CPU0-4	VDD_APC
QD	—	VDD_CX
RPM	RPM	VDD_CX
Peripherals and Config NoC	ULT Audio	VDD_CX
BIMC	BIMC	VDD_CX
PLLs	—	Analog
QFPROM	-—	Analog
DDR_PHY	-—	Analog
ADC/DAC	—	Analog
WCNSS	ARM9, BT_FM, WLAN	VDD_CX

Power Feature Overview



Basic Power Processor/System Design

No.	Power features	Implementation	Description
1	Adaptive Voltage Scaling (AVS)	Hardware	▪ AVS is a closed-loop mechanism used to set the operating voltages for a chipset. ▪ There are two types of AVS, Type I and Type II.
			AVS Type II includes both closed-loop and open-loop techniques. Operating voltages are dynamically adjusted by a CPU AVS controller to the PMIC, based on an internal AVS sensor. AVS Type II is applicable for early chipset families like MSM8960 and APQ8064. See *Application Note: Adaptive Voltage Scaling (AVS)* (80-N8715-14) for details.
			AVS Type I, also referred to as Core Power Reduction (CPR), is similar to AVS Type II; operating voltages are dynamically adjusted using the AVS controller via RPM, based on internal sensors. Used in latest chips MSM8974 and later. See *Application Note: Adaptive Voltage Scaling (AVS)* (80-N8715-14) for details.
2	Clock gating	Hardware and software	Clock gating reduces the active power for a circuit. Shutting off the clock for a circuit prevents switching activity of the clocks and hence eliminating switching current consumption. Here, only leakage currents are incurred due to VDD supply being on.
3	Power Gating/Collapse	Hardware and software	System modules that are not in use can be shut down individually to eliminate leakage current. This is called power gating/power collapsing a module.
4	Block Head Switch (BHS)	Hardware	Master switch used to power down cores/modules that share a common power domain, e.g., LPASS, Q6, GPU
5	Globally Distributed Head Switch (GDHS)	Hardware	Used to individually power collapse modules that are not on a common power domain. GDSC is essentially multiple switches spread across the entire chipset to control individual modules compared to BHS, a single block switch, that turns off an entire subsystem, e.g., submodules of a block like VFE in CAMSS, GFx OCMEM under Graphics module.
6	NoC	Hardware	Network-on-a-chip (NoC) is a bus architecture that facilitates the higher peak frequency support to match DRAM bandwidth.
7	OCMEM	Hardware	Shared on-chip SRAM that helps improve performance and power by reducing DDR traffic.

Basic Power Processor/System Design (cont.)

No.	Power features	Implementation	Description
8	XO Shutdown	Software	Disables the clock supply to the MSM. This is triggered when each master processor (APPS, GPU…etc.) is in Power-Collapse mode, dedicated clocks should be off, and shared clocks should have “off” vote from their clients.
9	VDD Minimization	Software	▪ VDD minimization is the deepest low power mode for the system. ▪ Though all hardware and software states are maintained, the chip is not operational in this state (except for detecting wakeup interrupt/timer expiration for VDD min).

10	CPU low power modes	Software	▪ Depending on the duration of idle time for CPU, it can enter:
			  ▪ WFI (clock gating)
			  ▪ Retention (clock gated and set to low retention voltage)
			  ▪ Standalone power-collapse (clock and power collapse without RPM notification)
			  ▪ Power collapse (RPM assisted)
			  ▪ Clock gating and power collapse for all cores

11	Modem low power modes	Software	MSS supports various low power modes, such as MSS PC, WFI, XO shutdown, VDD minimization, L2 nonretention, etc.; the decision to enter these various LPMs is based on latency and power penalty to enter/exit these modes.
12	LPASS (Hexagon) power modes	Software	Hexagon supports SVS mode (setting of minimum required voltage to support the requested frequency); voting for ADSP power collapse depends upon its aggregated client requests. ADSP power collapse and resumption from power collapse are triggered by notifications from the Sleep task.
13	Memory controller (BIMC) power collapse	Software	The Thin-Apps FABRIC and memory controller, which is referred to as Bus Interface Memory Controller (BIMC), can be power collapsed to remove leakage. Once BIMC is collapsed, DDR is not accessible. This could be done while portions of the chip are operational; those portions must ensure that DDR will not be accessed.
14	PMIC Auto mode	Hardware	▪ SMPS operates in Pulse Width Modulation (PWM) mode during active operation (higher voltages) because SMPS efficiency is better when operated in PWM mode at higher voltages. ▪ SMPS should operate in Pulse Frequency Modulation (PFM) mode during low power modes (lower voltages) for better efficiency. ▪ The PMIC Auto mode feature makes sure that SMPS shifts its mode of operation between PFM and PWM automatically, based on operating conditions to maximize efficiency.

Basic Power Processor/System Design (cont.)

No.	Power Feature	Implementation	Description
15	Voltage Scaling	Software	▪ Voltage levels are changed per usage ▪ SVS2, SVS, SVS+, Nominal, Nominal+, Turbo, Super Turbo are in the increasing order of voltage/clock levels and power consumption **Note:** The availability of these modes changes based on chipsets and also within the same chipset. A component (CX, MX, CPU.. etc.) may have part of the voltage modes applicable, e.g., on MSM8994, CX only has LowSVS, SVS, Nominal, Turbo applicable.
16	CPU DCVS	Software	cpufreq governor changes CPU frequency depending on the CPU utilization. Multiple governors such as Interactive, On-Demand, Performance, Use space are available.
17	GPU DCVS	Software	Dynamically scales the GPU frequency based on the load and reduces system power consumption for the active GPU use cases without negatively impacting performance.
18	Modem Subsystem (MSS) DCVS	Software	Modem DCVS is also referred to as CPU dynamics; it scales the CPU clock based on measured CPU utilization and MIPS request from clients.
19	CPU Bus DCVS	Software	CPU DDR (AB/IB)vote is scaled based on the expected traffic on the BUS. This in turn enables DDR to operate at lower frequency and therefore power saving can be observed.
20	GPU Bus DCVS	Software	Based on the bandwidth usage history of GPU, GPU Bus DCVS votes for System Memory Bandwidth dynamically. This enables DDR to operate at lower frequency and hence power savings can be observed.

Low Power Mode Details – Processor Lower Power Modes

- Each processor has its own Low Power modes that it can enter to save power. The selection of the specific Low Power mode is latency dependent. Depending on the amount of time available before the next scheduled task needs to run, the processor's sleep algorithm decides the appropriate Low Power mode for the processor to enter.
- Processors generally have the following Low Power modes that they can enter:
  - Software Wait For Interrupt (SWFI) – SWFI is the least power saving mode and involves only the clock supply of that processor to be gated. There is no RPM intervention and the processor is halted.
  - Retention – In this mode, the processor's clock is already gated and the supply voltage to the processor is lowered to retention voltage level, which helps in saving leakage.
  - Automatic power collapse and restore (APCR)/fast power collapse – APCR was introduced from MSM8994 for Hexagon processors and is extended to the apps processor for MSM8996. This mode provides a mechanism to power collapse and restore the CPU core logic in a manner that is transparent to software execution. APCR provides a lower power leakage alternative to the traditional clock gating Low Power mode.
  - Standalone power collapse – Power collapse refers to gating the power supply to the processor. The processor's clock and the power supply is cut off without any assistance from the RPM. Therefore, the RPM is not notified of the processor's power collapse, and the processor can wake up without involving the RPM.
  - Power collapse – This mode involves notifying to RPM and getting assistance from RPM for power collapsing the resources. The processor can wake up only when the RPM brings it up from power collapse.

Low Power Mode Details – System Low Power Modes

- The MSM can go to Low Power modes when all the processors are already power collapsed and notify to RPM for system sleep via voting.
- Depending on the processor votes and the time available from current time to next scheduled wakeup, the RPM decides to go to one of the following system Low Power modes to save power:
  - XO shutdown – XO is the source to the clocks supplied to different functional blocks on MSM SoC. XO shutdown basically disables the clock supply to MSM for power saving purpose. CXO is always-on, only its buffer is turned off when XO shutdown mode is exercised. Clock generation of CXO is still kept on at the PMIC and is used to generate the 32 kHz sleep clock, which is required for the always-on parts like the MPM block.
  - VDD minimization – VDD minimization is the deepest low power mode that can be achieved for VDD CX and VDD MX. When VDD minimization is achieved, the chip is not operational, except for detecting wakeup interrupt/timer expiration for VDD minimization; however, all hardware (supplied by VDD CX/MX) states are still maintained. These two power domains cannot be power collapsed, so they are put to a lower voltage that can sustain the contents in memories, etc. Lowering the voltage still saves some leakage current and, therefore, power. XO shutdown conditions must be met before VDD minimization, as XO shutdown must happen after VDD minimization.

PMIC Auto Mode

- PMIC operates in PWM mode during active operation (higher voltages) because SMPS efficiency is better when operated in PWM mode at higher voltages.
- Instead, SMPS should operate in Pulse Frequency Modulation (PFM) mode during low power modes (lower voltages), as the SMPS efficiency is better when operated in PFM mode at lower loads.
- The PMIC Auto mode feature makes sure that SMPS shifts its mode of operation between PFM and PWM automatically, based on operating conditions to maximize efficiency.

Voltages/Clocks and Their Importance

- Clock plan provides the frequency-voltage level mapping for all subsystems
- Sample clock plan for MSM8939

	SVS	Nominal	Turbo
System NoC	100	200	266
BIMC	166	333	400

  - In this table, 150 MHz is SVS for BIMC where as it is Nominal for System NoC
  - Hence, it is very important to understand that each subsystem could have different voltage level for the same frequency value.
- Clock dump provides frequencies of all active components for a use case
  - Below is the BIMC clock state and frequency from a clock dump.

```
      Clock          State      Frequency (MHz)
-------------------------------------------------------------------------------
gcc_bimc_clk       ON         100.001525
```

  - Compare the value listed in the clock dump to the value from a reference log from a good device or the value in the use case details document.
  - If a component has a higher frequency/voltage level than the reference value, it results in higher power consumption.

Sample Voltage/Clock Plan Based on MSM8936

Subsystem	Frequency (MHz)	Voltage Level
Modem	384.00	SVS
	576.00	Nominal
	691.20	Turbo
GPU	220.00	SVS
	400.00	Nominal
	550.00	Turbo
BIMC	166.8	SVS
	333.6	Nominal
	400	Turbo

CPU Frequency Governor

- Diverse governors such as interactive, on-demand, and performance governors are supported. They determine the current CPU frequency from the CPU DCVS table.
- On-Demand Governor
  - This governor sets the CPU frequencies depending on the current usage.
- Performance Governor
  - This governor sets the CPU to the highest frequency within the borders of min and max frequencies defined.
- Interactive Governor
  - Similar to On-Demand governor, this dynamically selects the proper frequency from the DCVS table based on CPU utilization.
  - It is used for latency-sensitive workloads.
  - The only difference between On-demand and Interactive governor is the set of parameters that are used to determine the CPU frequency.

Interactive Governor Parameters and Tuning

Parameters	Description	Usage
target_loads	▪ The CPU load value is used to adjust CPU frequency to bring the current CPU load close to the value that this parameter is set. ▪ The purpose of target_loads is to calculate the next frequency. ▪ The format is a target load, optionally followed by pairs of CPU frequencies and CPU loads to target at or above those frequencies. ▪ Colons can be used between the frequencies and associated target loads for readability.	Example: 85 1000000: 90 1700000:99 System targets ▪ 85% CPU load for frequencies below 1 GHz ▪ 90% load at or above 1 GHz ▪ 99% load at or above 1.7 GHz Frequencies and loads are specified in ascending order. The default target load is 90% for all frequencies.
min_sample_time	The minimum amount of time to spend at the current frequency before ramping down	Default is 80000 µs
hispeed_freq	An intermediate "high speed" at which to initially ramp when CPU load hits the value specified in go_hispeed_load; if load stays high for the amount of time specified in above_hispeed_delay, the speed may be bumped higher.	Default is the maximum speed allowed by the policy at governor initialization time
go_hispeed_load	The CPU load at which to ramp to hispeed_freq	Default is 99%
above_hispeed_delay	Time to wait at hispeed_freq before raising the speed	

Interactive Governor Parameters and Tuning (cont.)

Parameters	Description	Usage
timer_rate	Sample rate for reevaluating CPU load when the CPU is not idle; a deferrable timer is used such that the CPU will not be woken from idle to service this timer until something else needs to run. (The maximum time to allow deferring this timer when not running at minimum speed is configurable via timer_slack.)	Default is 20000 µs
timer_slack	Maximum additional time to defer handling the governor sampling timer beyond timer_rate when running above the minimum speed	If timer_rate is 20000 µs and timer_slack is 10000 µs, then timers will be deferred for up to 30 ms when not at lowest speed; a value of -1 means defer timers indefinitely at all speeds; default is 80000 µs
boost	If nonzero, immediately boost speed of all CPUs to at least hispeed_freq until zero is written to this attribute; if zero, allow CPU speeds to drop below hispeed_freq according to load as usual	Default is 0
boostpulse	On each write, immediately boost speed of all CPUs to hispeed_freq for at least the period of time specified by boost pulse_duration, after which speeds are allowed to drop below hispeed_freq according to load as usual	
boostpulse_duration	Length of time to hold CPU speed at hispeed_freq on a write to boostpulse before allowing speed to drop according to load as usual	Default is 80000 µs

Note: The above parameters are available at /kernel/drivers/cpufreq/cpufreq_interactive.c.

Scheduler Tunables – Load Tracking

Name	**sched_ravg_window**
Location	At kernel command line argument
Default value	10000000 (10 ms, units of tunable are nanoseconds)
Description	Specifies the duration of each window in window-based load tracking. By default, each window is 10 ms long. This quantity must currently be set at boot time on the kernel command line (or the default value of 10 ms can be used).

Name	**RAVG_HIST_SIZE**
Location	Compile time only (see RAVG_HIST_SIZE in include/linux/sched.h)
Default value	5
Description	Specifies the number of windows the window-based load tracking mechanism maintains per task. If default values are used for both this and sched_ravg_window, a total of 50 ms of task history is maintained in five 10-ms windows.

Name	**sched_window_stats_policy**
Location	/proc/sys/kernel/sched_window_stats_policy
Default value	2
Description	Controls the policy in how window-based load tracking calculates an overall demand value based on the windows of CPU utilization it has collected for a task. Possible values for this tunable are: 0 – Use only the most recent window sample of task activity when calculating task demand. 1 – Use the maximum value of all the window samples of task activity. 2 – Use the maximum of (the most recent window sample, average of all samples). 3 – Use the average of all samples. **Note:** The number of window samples maintained per task, and the size of those windows, are controlled by sched_ravg_window and RAVG_HIST_SIZE tunables listed above.

Note: Tunables are subject to change.

Scheduler Tunables – Task Load Characterization

Name	**sched_init_task_load**
Location	/proc/sys/kernel/sched_init_task_load
Default value	15
Description	Percentage. When a task is first created it has no history, so the task load tracking mechanism cannot determine a historical load value to assign to it. This tunable specifies the initial load value for newly created tasks.

Name	**sched_small_task**
Location	/proc/sys/kernel/sched_small_task
Default value	10
Description	Percentage. If a task consumes this much or less of a CPU, it is considered a small task. The scheduler does not attempt to find an idle CPU for small tasks; they may be awakened on busy CPUs.

Name	**sched_mostly_idle_nr_run**
Location	/proc/sys/kernel/sched_most_idle_nr_run
Default value	4
Description	If a CPU has this many runnable tasks (or more), it cannot be considered mostly idle. A mostly idle CPU is a preferred destination for a waking task. To be mostly idle, a CPU must have fewer than sched_mostly_idle_nr_run runnable tasks and be less than sched_mostly_idle_load percent busy.

Name	**sched_mostly_idle_load**
Location	/proc/sys/kernel/sched_mostly_idle_load
Default value	20
Description	Percentage. If a CPU is this busy, it cannot be considered mostly idle. A mostly idle CPU is a preferred destination for a waking task. To be mostly idle, a CPU must have fewer than sched_mostly_idle_nr_run runnable tasks and be less than sched_mostly_idle_load percent busy.

Note: Tunables are subject to change.

Scheduler Tunables – Task Migration Thresholds

Name	**sched_upmigrate**
Location	/proc/sys/kernel/sched_upmigrate
Default value	80
Description	Percentage. If a task consumes more than this much of a CPU, the CPU is considered too small for the task, and the scheduler tries to find a bigger CPU in which to place the task.

Name	**sched_downmigrate**
Location	/proc/sys/kernel/sched_downmigrate
Default value	60
Description	Percentage. When a task previously ran on a big CPU and the scheduler evaluates whether it can now fit on a little CPU, this tunable specifies the threshold used to determine where the task is placed. This value is used instead of sched_upmigrate to allow the system to guard against a task frequently migrating back and forth between the little and big cluster.

Name	**sched_spill_nr_run**
Location	/proc/sys/kernel/sched_spill_nr_run
Default value	10
Description	Maximum number of concurrently runnable tasks that a CPU can accommodate during task placement. When a task cannot be placed on any CPU in a cluster without causing that CPU to exceed this number of runnable tasks, the task is eligible for inter-cluster migration.

Name	**sched_spill_load**
Location	/proc/sys/kernel/sched_spill_load
Default value	100
Description	Percentage that represents the maximum load a CPU can accommodate during task placement. When a task cannot be placed on any CPU in a cluster without causing that CPU to exceed this load threshold, the task is eligible for inter-cluster migration.

Note: Tunables are subject to change.

CPU Bus DCVS

- Before CPU Bus DCVS
  - BIMC frequency was a straight map with CPU Frequency.
    - This implies that the BIMC frequency will be high if CPU frequency is high irrespective of the DDR traffic, which can imply that the DDR power will be high even when DDR is not being used.

- With CPU Bus DCVS
  - BIMC frequency changes depending on bus traffic and not directly with CPU frequency.
    - The number of packets that are transferred between BIMC and L2 Cache are recorded by a counter in BIMC. This number is monitored at a constant sampling rate and the bus frequency is adjusted per any changes.
    - To determine if CPU bus DCVS is enabled, from adb shell type the following command and check the node:

      `/sys/class/devfreq/qcom, cpubwxx/governor`
      - If cpufreq, it means DDR is statically mapped to the CPU frequency.
      - If bw_hwmon, it means CPU bus DCVS is enabled.

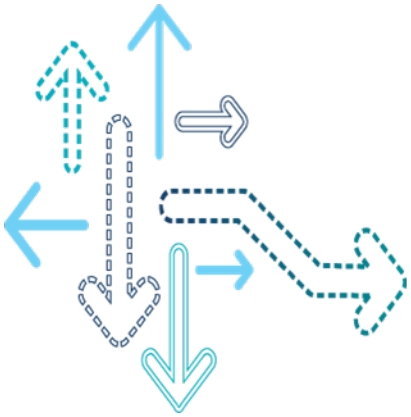
  - Algorithm
    - DDR AB/IB votes are computed based on expected traffic on the bus.
      - Interrupt is generated when the DDR usage exceeds the previous measured value and the new throughput, and hence new AB/IB values are computed.
      - Certain tunable parameters are used to decide when to trigger an interrupt and when the throughput needs to be recomputed.
    - **Note:** Low-end chipsets have a static bus frequency mapping to the CPU frequency. APQ8084, MSM8994 support Bus DCVS.

Tunable CPU Bus DCVS Parameters

Parameter	Description	Usage
tolerance_parameter	Percentage of increase in throughput compared to previous measured throughput for an interrupt to trigger	Decides the DDR/Bus throughput change at which an interrupt needs to be triggered; default is 10% `threshold_bytes = (100 + tolerance_percent) * measured average throughput * sample_ms / (100 * 1000)`
guard_band_mbps	Additional MBps to account for additional data that could have been transferred while DDR/Bus is at lower frequency before the frequency is increased to the new required value	Decides how much additional MBps needs to be added to account for the delay between calculation of new throughput requirements; default is 100 MBps `adjusted_throughput = measured_throughput + guard_band_mbps`
decay_rate	Percentage of decrease from the previous throughput value when the throughput decreases	Default is 90% `new_throughput = ((100 – decay_rate) * previous_throughput + decay_rate * adjusted_throughput) / 100`
Io_percent	Percentage of time CPU is allowed to spend accessing DDR	Decides the instantaneous bandwidth vote (IB) for CPU/bus; default is 16% `ib = eff_throughput * 100 / io_percent`
bw_step	Step to increase the AB vote	Default is190 MBps `ab=((effective_throughput+bw_step–1)/ bw_step) *bw_step`

Note: The above parameters are in kernel/drivers/devfreq/governor_bw_hwmon.c.

Subsystem DCVS

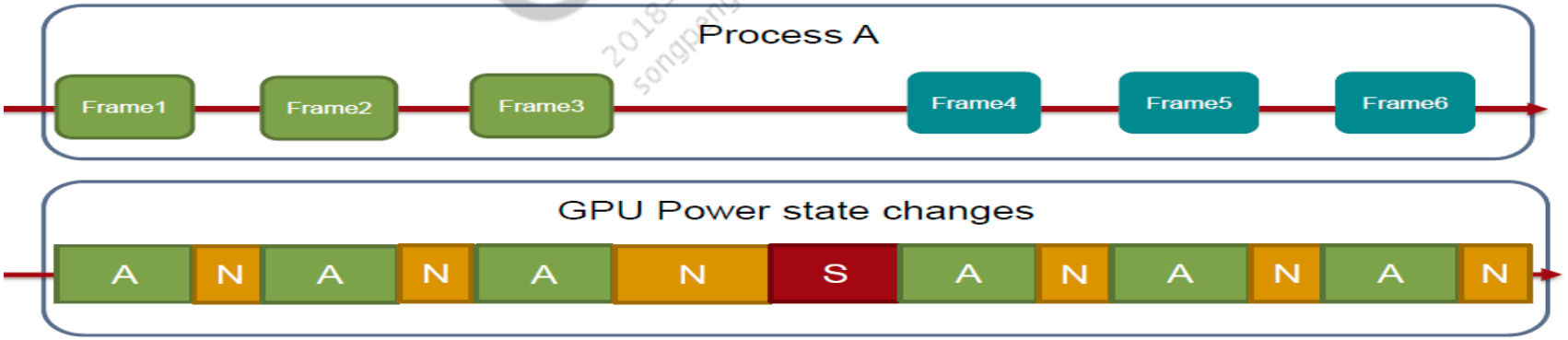


GPU DCVS/GPU Bus DCVS Terminology

GPU DCVS terms	Description
GPU Power Levels	A set of GPU's operational power levels that include GPU clock and default bus level for the clock; typically, Power Level 0 is mapped to maximum GPU clock
GPU DCVS	Dynamic Clock and Voltage Scaling for GPU clock
GPU Bus DCVS	Dynamic Clock and Voltage Scaling for memory bandwidth voting
Initial GPU Power Level	Default power level used when GPU is initialized or resumed after suspension
Default Bus Level	Default bus level defined inside each GPU power level to be set when GPU power level changes
Static Bus Bandwidth Mapping	No GPU Bus DCVS; bus bandwidths are tied to GPU power levels
Dynamic Bus Bandwidth Mapping	Many-to-one mapping of bus levels to each GPU power level

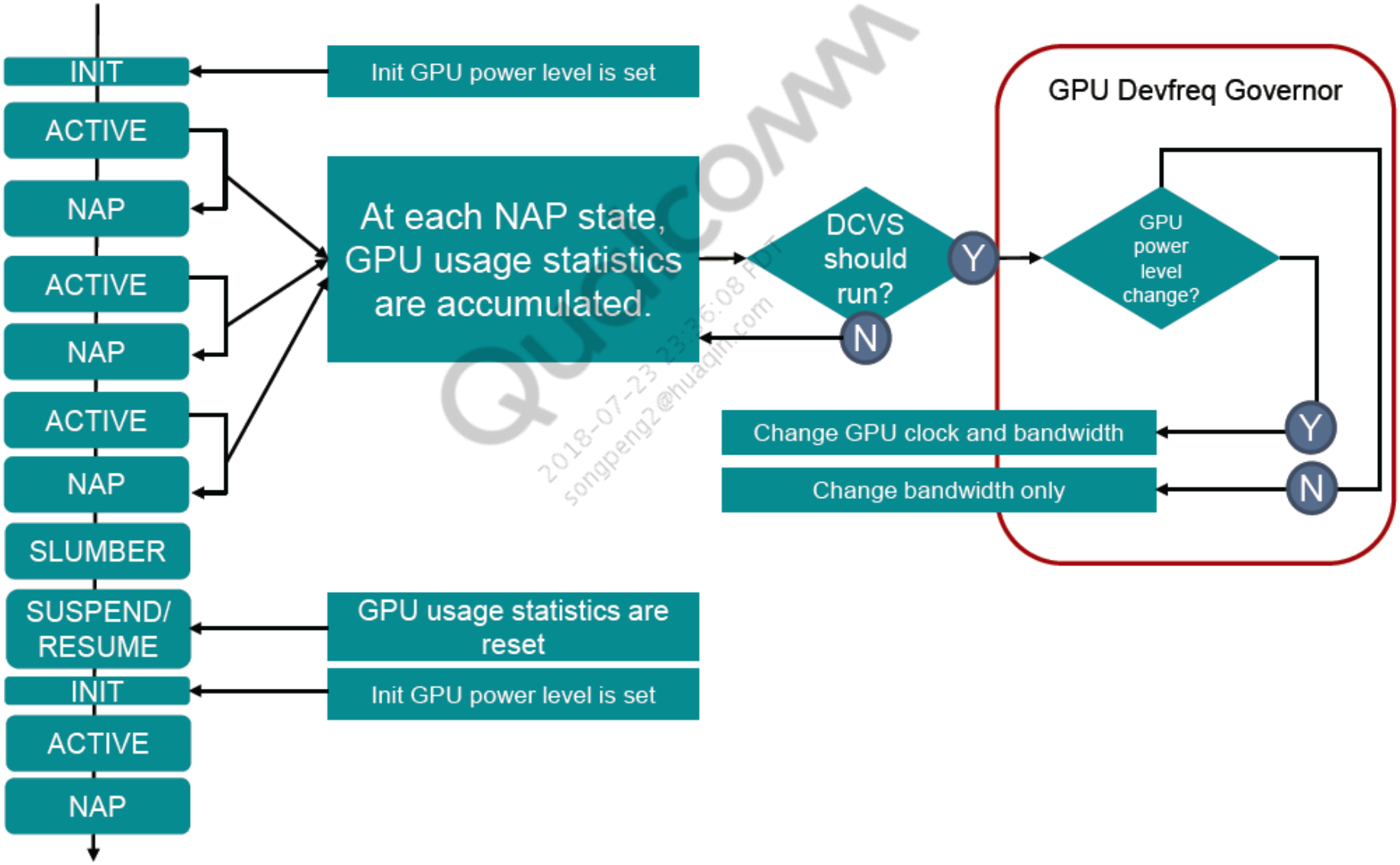
GPU Bus DCVS

- GPU requires system memory access for its external resources such as textures. Based on the bandwidth usage history, GPU Bus DCVS votes for System Memory Bandwidth dynamically.
- GPU utilizes a part of OCMEM for fast GPU memory operations.
- GPU Bus DCVS is not the same as GPU DCVS. Even though there is one governor controlling both GPU clock level and GPU bus bandwidth level, running GPU clock and bus bandwidth vote are decoupled within the same power level.



A	**Active State**
N	**Nap State. Low latency resources like clocks and irqs are off**
S	**Sleep/Slumber. All power resources are off. System enters this state when sleep timer expires in NAP state**

GPU Power State Transitions



GPU DCVS vs GPU Bus DCVS

- GPU DCVS is for GPU clock.
- GPU Bus DCVS is for bus bandwidth voting.
- GPU Bus DCVS is not available on all platforms. Low-tier chipsets have static mapping to GPU frequency.
  - Checking /sys/class/kgsl/kgsl-3d0/bus_split
  - 1 – GPU Bus DCVS is enabled→dynamic bus bandwidth mapping to GPU clocks
  - 0 – GPU Bus DCVS is not enabled→static bus bandwidth mapping to GPU clocks

MSM8916 GPU levels	
Voltage level	DCVS frequency (MHz)
SVS	19.2
	200
Nominal	310
Turbo	400

MSM8936 GPU levels	
Voltage level	DCVS frequency (MHz)
SVS	19.2
	220
Nominal	400
Turbo	500

GPU Levels for MSM8994

MSM8994 GPU levels		
Voltage level	GPU frequency (MHz)	GPU Bus DCVS frequency (MHz)
SVS2	190	300
		460*
		547
SVS	305	460
		547*
		691
Nominal	390	547
		691*
		777
	450	691
		777*
		1036
	510	777
		1036*
		1296
Turbo	600	1036
		1296*
		1555

***Default frequency**. For each voltage level, the bus frequency is set to the default level and then adjusted to lower or higher level based on traffic between DDR and Cache.

For more details on GPU/GPU Bus DCVS, see *Presentation: Graphics Power and Performance Overview* (80-NP885-1).

GPU Power Tuning

- The power-related attributes/settings of GPU are configured statically by kernel and managed dynamically by the GPU DCVS governor.
- Static attributes
  - Initial power level
  - Idle timeout
  - Maximum power level
  - Minimum power level
  - Thermal power level
- Dynamic attribute
  - Active (running) power level
- Power levels – A group of predefined GPU clock levels that have different power source levels, i.e., SVS, Nominal, and Turbo
- Idle Timeout – A set timer limit for GPU not to power collapse

GPU DCVS Parameter Check Through Sysfs

Sysfs operation	Description	Command
Turn off/on GPU DCVS	Use the performance governor for max GPU frequency or the power governor for min GPU frequency command sequence	cd /sys/class/kgsl/kgsl-3d0/devfreq echo performance > governor cd /sys/class/kgsl/kgsl-3d0/devfreq echo powersave > governor
Available GPU power levels	To ensure you request a supported frequency, check availability	cat /sys/class/kgsl/kgsl-3d0/gpu_available_frequencies
Set the min/max power level for GPU	To test GPU power levels power consumption or have a fixed GPU clock profiling	cd /sys/class/kgsl/kgsl-3d0 echo 1 > max_pwrlevel echo 1 > min_pwerlevel //this will fix GPU clock to be at power level 1's GPU Frequency
GPU busy statistics	The first value is the GPU busy time, and the second one is the total system time (~1 sec)	cd /sys/class/kgsl/kgsl-3d0/ cat gpubusy //(first_value/second_value)*100 gives percentage of the last sec the GPU core was //busy
Checking running GPU clock	To ensure GPU is running at optimal clock frequency for given use case	cd /sys/kernel/debug/clk/oxili_gfx3d_clk/ cat measure

Checking GPU bus vote	To ensure GPU is voting at optimal bus bandwidth	cd /sys/kernel/debug/msm-bus-dbg/client-data/
		cat grp3d
		//master 26 slave 512 is for system bus (BIMC)
		//master 89 slave 604 is for OCEM
		//example output
		// ab: 120000000 0 → Bus at 1.2 Gbps ab/ OCMEM at 0 Gbps ab
		// ib: 245600000 5280000000 → Bus at 2.456 Gbps ib/ OCMEM at 5.28 Gbps ib

Modem DCVS

- Modem DCVS
  - DCVS on non-HLOS processors is referred to as CPU dynamics.
  - CPU dynamics scales the processor clock based on utilization and MIPS request from clients. Instantaneous bus bandwidth requests are made to the bus arbiter for CPU data and instruction access for CPU to DDR.

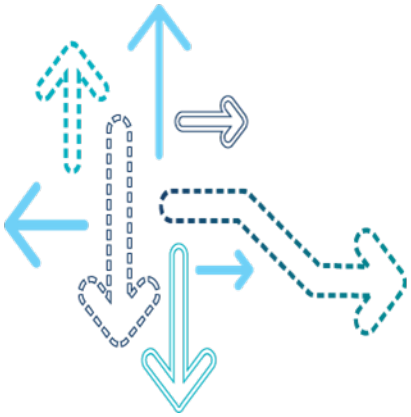
Voltage level	Frequency (MHz)
SVS	19.2
	115.2
	144.00
	230.40
	288.00
	384.00
Nominal	576.00
Turbo	691.20

Venus DCVS

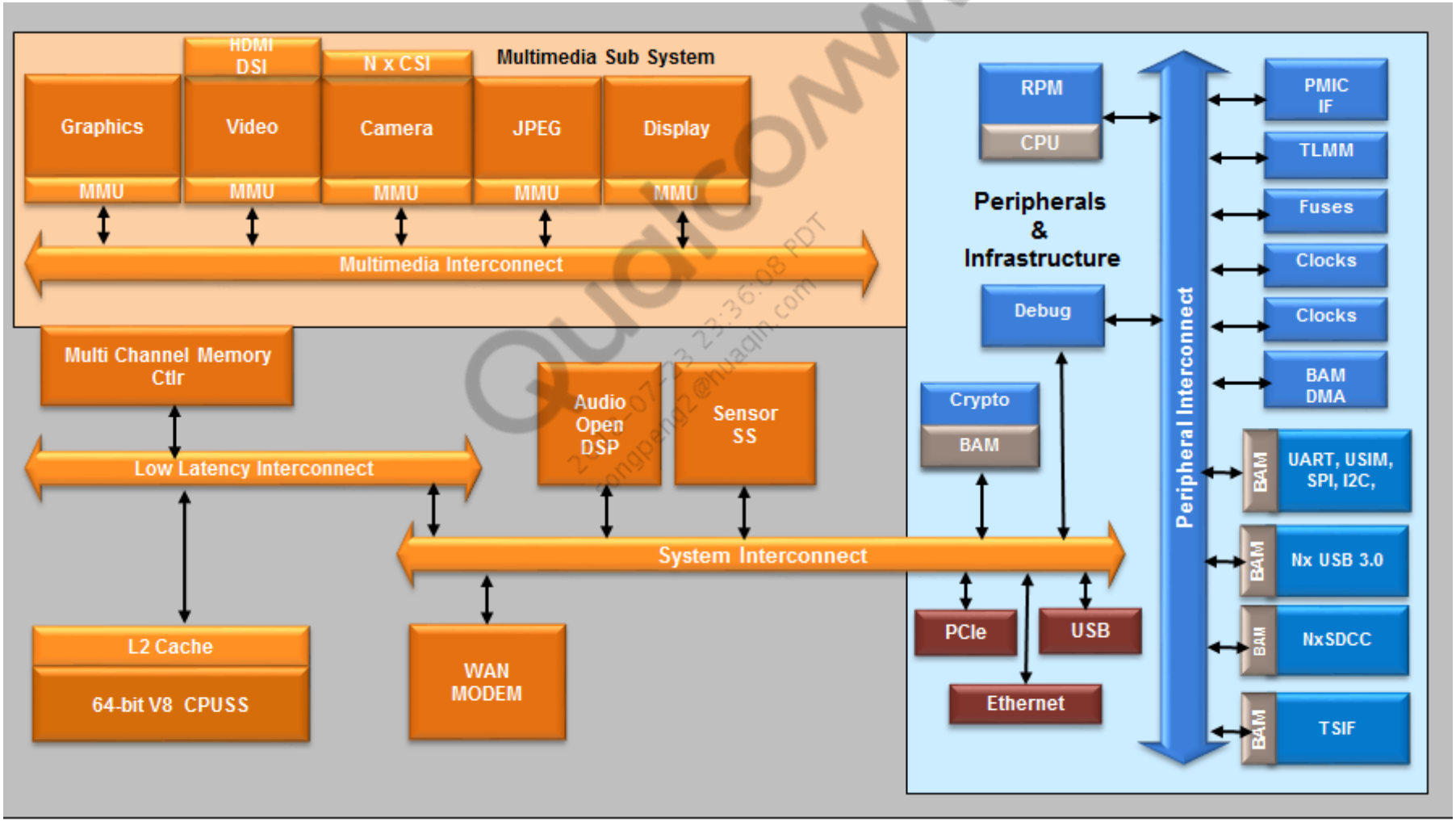
- Before DCVS, Venus clock was decided based on static mapping between frequency and resolution of the video.
- But in real-world scenarios, the video frame complexity changes and Venus clock can be varied accordingly to save power.
- On MSM8994, DCVS is implemented on Venus for 4K Video Decode use case. The table shows the levels currently supported for Venus DCVS.

Voltage level	Frequency (MHz)
SVS	150
Nominal	320
Turbo	510

Multimedia and Modem Power



System Modules in MSM8994



Factors Effecting Multimedia Power

- Multimedia use cases comprise the majority of daily usage of smartphones
- Optimizing these is essential to ensure good DoU for a customer.
- Factors contributing to Multimedia power are:
  - Display panel power
  - Camera sensor power
  - Headset/speaker power
- The power consumption of a particular multimedia use case is dependent on the components involved, e.g., LPASS, DDR, CPU, Codec for Audio playback use case.

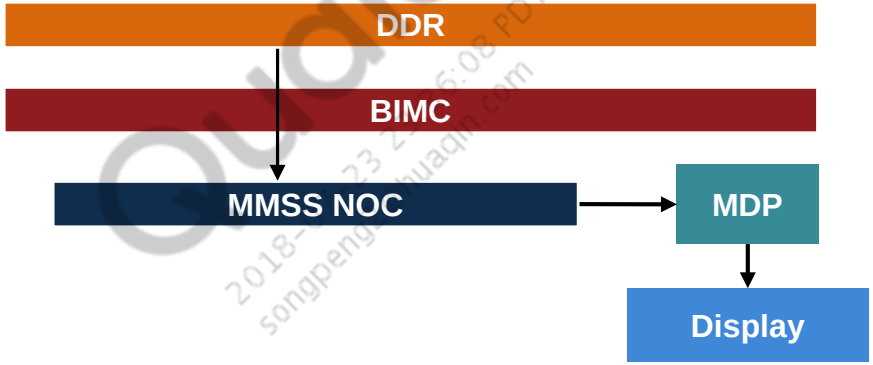
Multimedia Use Case Power

- The following slides provide the usual data flow for QTI standard multimedia use cases*.
- Clocks for all th subsystems mentioned in the data flow must be collected using a clock dump.
- These clocks need to be verified by comparing them to the specific chipset dashboard goals and use case clock plan document.
- It is recommended that customers optimize the standard QTI dashboard use case before trying any other, e.g., for Video playback, customers should verify the power consumption on QTC77A (720p Video Playback) and QTC88A (1080p Video Playback) before trying any other videos.

*The data flow and some components involved might vary based on platform. Refer to the specific chipset power overview for platform-related differences.

Static Image (LCD04A)

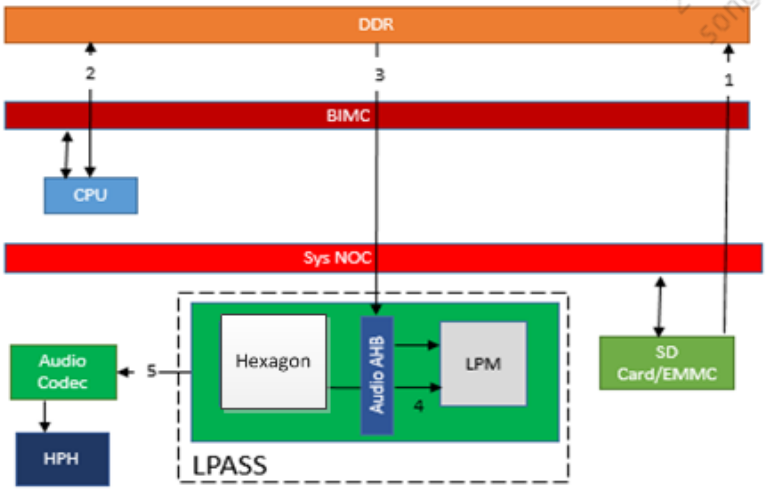
- Static image is the basis of all display active use cases.
- This use case needs to be optimized before optimizing video, camera, graphics, browser, or any use case where the screen is on.
- This is a basic data flow for the static image use case for a dumb panel.



  - The display subsystem updates the frame on the display at the panel refresh rate from DDR.
- For a smart panel, the image when unchanged is refreshed from the internal panel memory, and hence the system can enter power collapse.

MP3 Audio in Tunnel Mode (AU04A)

- Audio power is influenced by:
  - Postprocessing methodologies like Dolby
  - Speaker vs. Headset
  - Tunnel mode vs. Non-Tunnel mode
  - Custom audio player
- This use case needs to be optimized before optimizing any use case with audio, like video playback.
- The following is the data flow for Audio playback in Tunnel mode.



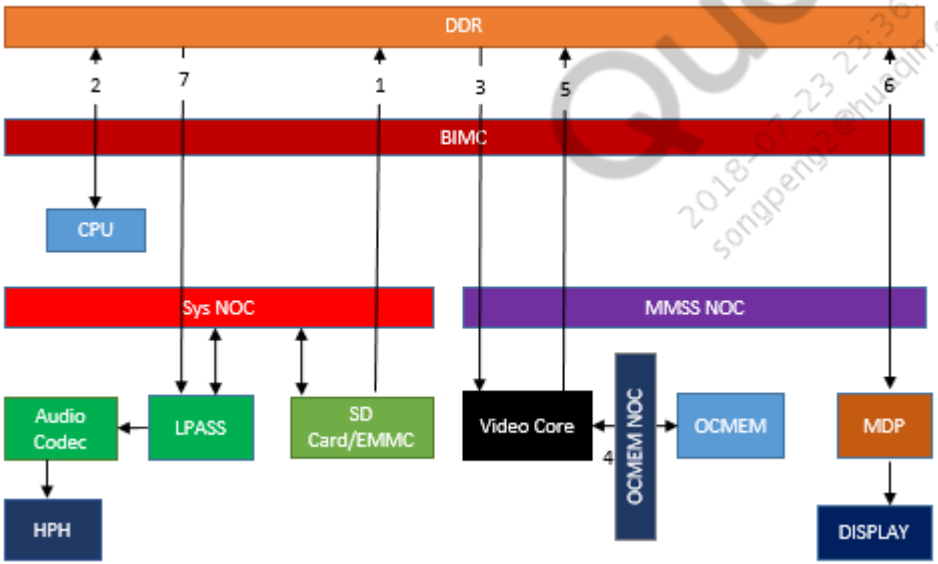
1. CPU reads 1 MB of data from SD card/eMMC to the DDR.
2. CPU parses the data and puts it in the DDR.
3. LPASS sets up DM-Lite to transfer 32 Kb of bit stream from DDR to LPM.
4. LPASS performs decoding and post processing and puts a PCM sample into LPM.
5. LPASS sets up the DMA to render PCM sample to the audio output device.

MP3 Playback Modes

- Compress offload playback (Tunnel mode)
  - CPU transfers large buffers of compressed bit streams to the ADSP and goes to sleep.
  - The ADSP then decodes and outputs the rendered PCM data to the physical sound device.
  - Before the ADSP decoder input runs out of data, it interrupts the CPU to wake up and send the next set of buffers.
  - Examples – Local audio playback (MP3, AAC)
- Deep-buffer playback (Non-Tunnel mode)
  - The CPU continuously fetches bit streams, decodes, and hands the PCM samples to ADSP.
  - The ADSP then renders the PCM data and sends output to sound device.
  - Use cases – Ringtone, audio video playback, audio streaming, YouTube streaming, etc.

Video Decode (QTC77A, QTC88A)

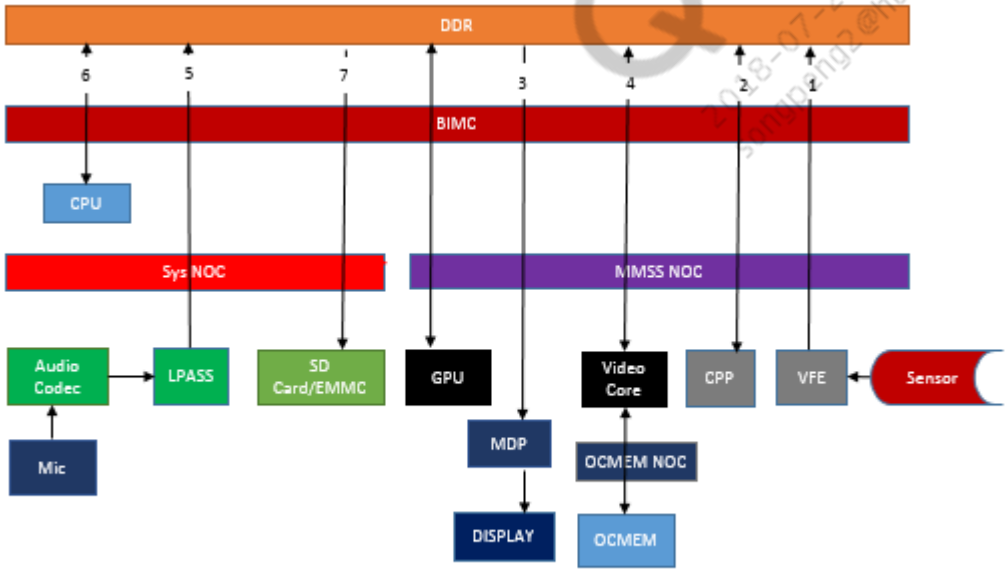
- Video playback power is influenced by:
  - Content and resolution of the video
  - Factors that influence audio playback power
  - Factors influencing display
- Basic data flow for Video Decode use case is shown here.



1. CPU reads multiplexed A/V from SD card/eMMC into DDR input buffers.
2. CPU reads multiplexed data from DDR input buffers and audio and video bit streams, writes back to DDR raw audio and video streams.
3. Video bit stream is read in chunks by Venus core.
4. Venus core uses local scratch memory (IMEM/OCMEM) to reconstruct YUV pixels; this reduces DDR traffic and, therefore, the overall power consumption.
5. The decoded data is put back in DDR by Venus core.
6. MDP reads the YUV frame from DDR, rotates it, and sends it back to DDR. Then it fetches the rotated frame, converts to RGB, and displays it on display.
7. While Venus is doing the video processing, LPASS decodes the raw bit stream for audio and generates a PCM sample, postprocesses, and passes them to the audio codec and audio peripheral devices (headphones/speakers).

Video Encode (QMC31A)

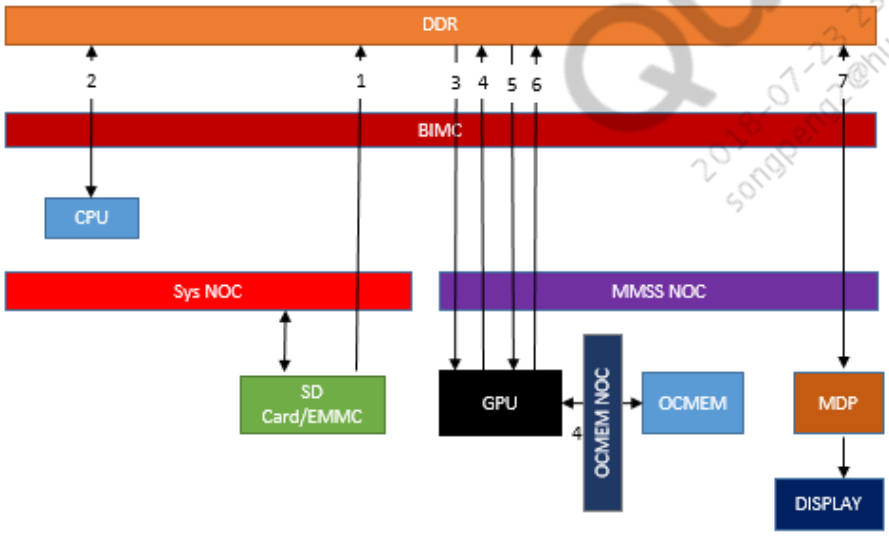
- Video Encode/Camera power depends on:
  - Camera postprocessing algorithms
  - Image enhancement algorithms (AWB, AEC, etc.)
  - Video recording resolution and FPS
  - Camera features like ZSL
  - Factors influencing display power
- Basic data flow for Video Encode is shown here.



1. VFE hardware simultaneously outputs two frames, one for preview and one for video encoding, to DDR.
2. CPP post processes the preview and video encode streams and outputs it back to the DDR.
3. MDP reads preview frame from DDR and outputs to the display.
4. In parallel, Venus core encodes the video stream and writes the video bit stream to DDR in YUV format. Venus core uses OCMEM as scratch memory.
5. In parallel to video encoding, audio LPASS receives PCM samples from the mic and writes encoded audio into DDR.
6. CPU reads the audio and video bit stream from DDR, multiplexes them into the final file format, and writes back to DDR.
7. CPU transfers the bit stream data from DDR to the SD card or flash where the recorded file is stored.

Graphics (QGC23A and QGC26A)

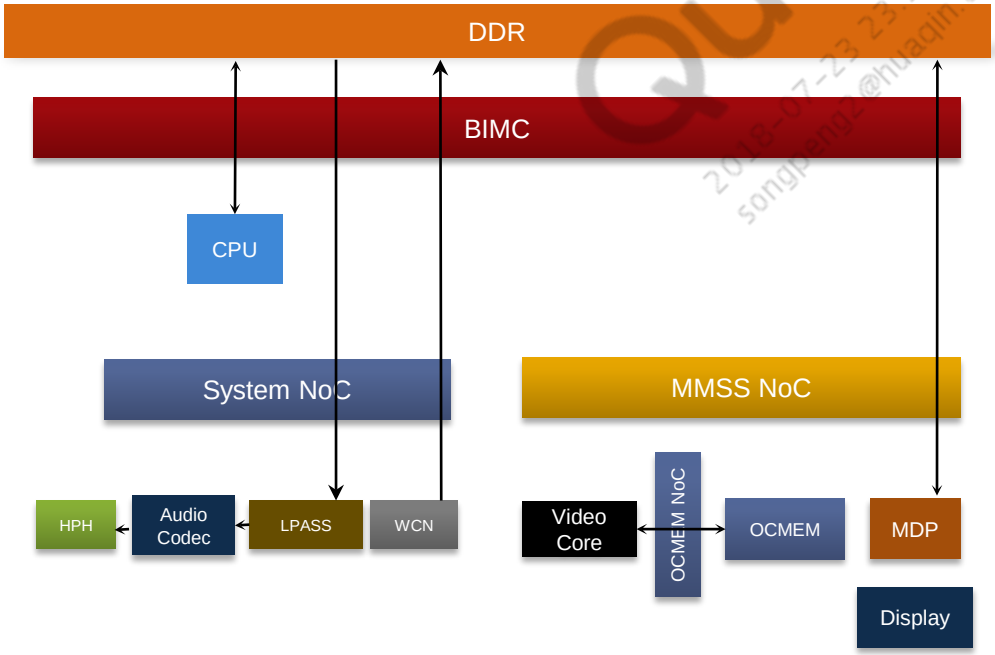
- Graphics power is influenced by:
  - Intensity of content
  - Factors influencing audio, if there is audio
  - Factors influencing display power
- Basic data flow for graphics use case is shown here.



1. CPU transfers graphics content from storage (SD/Flash) to DDR.
2. The application uses standard graphics APIs for changing graphics state, draw paths, textures, and other inputs for GPU. These are queued as commands into the input buffer of GPU.
3. GPU reads command/data descriptor data from DDR for frame rendering. GPU renders a frame bin by bin. Splitting the frame into bins improves the rendering performance and minimizes power consumption by eliminating unnecessary data traffic to DDR.
4. GPU transfers the rendered frame to DDR.
5. GPU then reads command and frame data for frame composition from DDR, one bin at a time.
6. GPU transfers the composed frame to DDR, one bin at a time. Steps 5 and 6 are repeated until all bins in the frame are done.
7. MDP reads the composed frame from DDR, converting pixel format if necessary, and transfers the frame to the display.

Video Streaming over Wi-Fi (VS06A)

- Factors influencing power are:
  - Video power factors
  - Wi-Fi power factors
  - Display power factors
  - Application used like YouTube
- Basic data flow for video streaming over Wi-Fi is shown here.



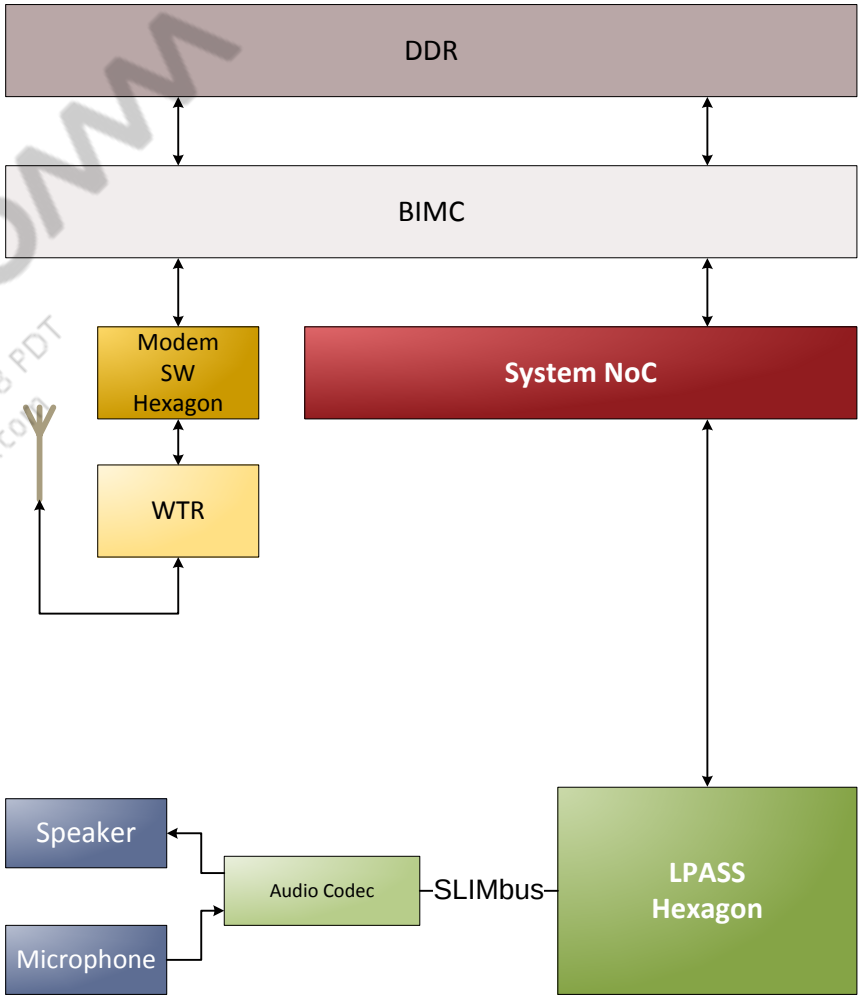
1. WLAN module (WCN) gets the input bitstream from the network and puts it in DDR.
2. CPU reads multiplexed data from DDR input buffers and audio and video bitstreams, writes back to DDR raw audio and video streams.
3. Video bitstream is read in chunks by video core.
4. Video core uses OCMEM as scratch memory and reconstructs YUV pixels.
5. The decoded data is put back in DDR by video core.
6. MDP reads YUV frame from DDR, rotates it, and sends it back to DDR; fetch's rotated frame, converts to RGB, and displays it on the LCD.
7. In parallel to video processing, audio LPASS decodes the raw bitstream and generates a PCM sample, post processes, and passes them to the audio hardware and headphones.

Factors Affecting Modem Power

- Modem power is the power consumption incurred while running modem use cases, i.e., voice call, LTE data call, HSDPA DC data call, modem standby, and other modem technology use cases.
- Optimizing modem power is essential for a good DoU and passing carrier specifications.
- Factors contributing to modem power are as follows:
  - Modem processor power
  - RF/WTR power
  - PA power
- The power consumption of a particular modem use case is dependent on the components involved during that use case, e.g., DDR, CPU, modem processor, RF hardware, PA, etc.

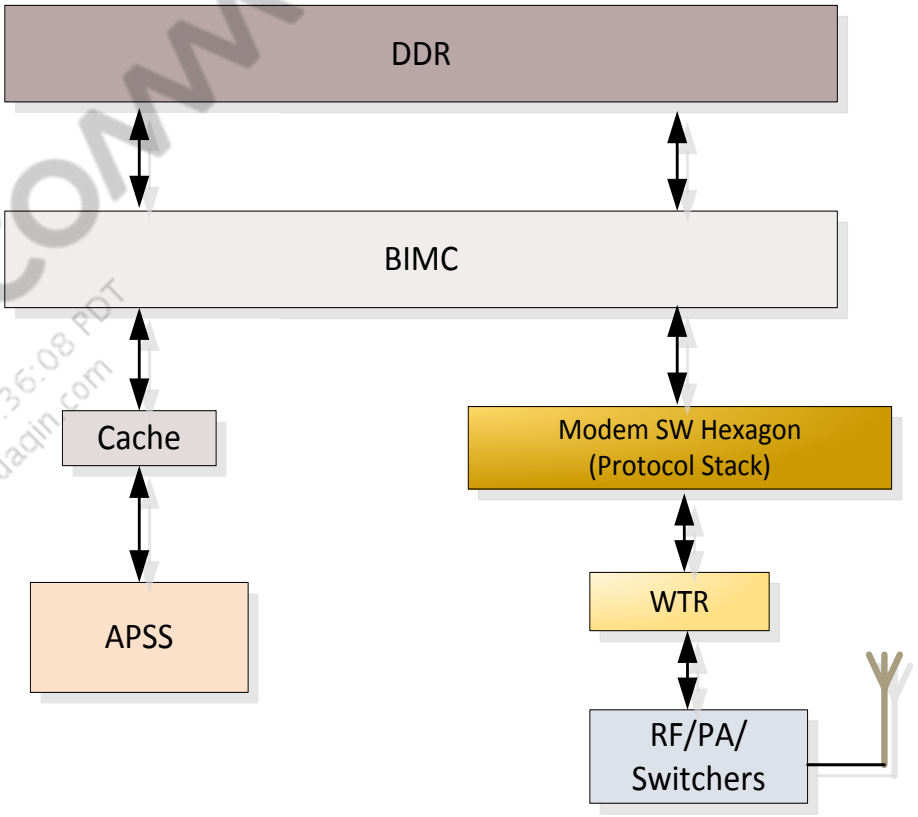
Modem Use Cases –Talk/Voice Call Flow Diagram

- Rx
  - Antenna receives the signal, which gets processed by the modem hardware for algorithmic processing and Hexagon software modem for the protocol stack; modem demodulates the received signal and produces digital data.
  - Digital data is stored in DDR. LPASS takes this digital data and processes it. Convolutional decoding (speech decoding) is done by LPASS, and the data is converted into raw digital data.
  - Raw digital data is converted to analog signals in the analog codec.
  - Audio codec feeds the analog signal to the speaker.
- Tx
  - Analog voice signals are received through the microphone and sent to the audio codec.
  - The audio codec does the sampling/quantization and converts it into digital raw data.
  - Digital raw data is processed by LPASS. Voice/speech encoding is done in LPASS; encoded packets are stored in DDR.
  - Modem hardware and Hexagon software modem modulates the data from DDR and adds the protocol headers before feeding it to the Tx antenna.



Modem Use Cases – Data Call Flow Diagram

- Rx
  - The data packets are received over the air and are demodulated and processed by the Hexagon modem software processor for the protocol stack. Packet data conversion takes place and the data is then moved to cache via the system NOC, BIMC, and DDR.
  - The apps processor picks up the data from cache for processing, e.g., TCP/IP stack processing, and then stores the data in DDR for real-world use (web browsing, file download, etc.).
- Tx
  - The data from the application running on APSS flows in exactly the opposite way, as mentioned above, and is sent over the air to the base station.



Recap

- Useful power-related reference documents
- System voltage domains
- Understanding the power tree and how it is useful in debugging
- Overall system power features
- Voltage and Clock Plan
- DCVS for system components
  - Governor tunables
  - CPU bus
  - GPU
  - GPU bus
  - Modem
  - Venus
- Modem and multimedia power use cases

References

Documents	
Qualcomm Technologies, Inc.	
Application Note: Software Glossary for Customers	CL93-V3077-1
Power Consumption Measurement Procedure for MSM (Android™-Based)/MDM Devices	80-N6837-1
Presentation: Customer Engineering Power Optimization Process	80-NB739-1
Application Note: Understanding Low-Dropout (LDO) Regulators	80-VT310-125
Application Note: Adaptive Voltage Scaling (AVS)	80-N8715-14
Presentation: Graphics Power and Performance Overview	80-NP885-1
Presentation: Android™ Multimedia Power Debugging Guidelines	80-NT-615-1
System Power Monitor Version 4 Overview (SPMv4)	80-N6594-20
System Power Monitor Version 4 Application Note	80-N6594-16
MSM8936/MSM8939 System Power Overview	80-NM846-2

Questions?

https://support.cdmatech.com