Pre-Silicon Power Management Verification of Complex SOCs: Experiences with Intel© Moorefield

Rajeev Muralidhar[+], Nivedha Krishnakumar[+], Bryan Morgan[*], Robert Karas[$], Billy Dennie[$], Neil Rosenberg[$]
Intel Corporation
[+] 23-56P, Sarjapur Outer Ring Road, Bangalore, India 560085
[*] 77 Reed Road, Hudson, MA 01749
[$]1300 S. Mo Pac Expy, Austin, TX 78741

***Abstract* - Power management (PM) verification of complex SOCs is a big challenge from hardware as well as from a system level perspective as it spans the entire platform and has become a growing concern with the complexity of SOCs and rapid cadence at which they need to be productized. This paper presents the system level PM verification that was done on Intel© Moorefield platform. A system-level approach to pre-silicon verification was used that included Android OS boot, OS driver power management flows and system and SOC-level low power states verification in pre-silicon models. This yielded significant benefits in overall bug-free silicon/system and faster time to market (TTM). We believe that such a system-level and HW/SW co-design approach to pre-silicon power management verification is crucial and the learning from this paper can be used for other devices/SOCs in the industry.**

I. INTRODUCTION

Power management verification of complex SOCs is a big challenge from hardware as well as from a system level perspective, since power management spans the entire platform. Power management has also become harder with the increase in complexity of today's SOCs and the need to productize them at a rapid cadence. Ideally, each system component (IP/hardware/firmware/software) needs to be verified for its power management capability (both individually as well as how they work in relation to other components), and in addition, system-level power flows (low power idle/standby states, etc.) also need to be verified before silicon tape-in is achieved. Bugs, especially in hardware, can result in subsequent silicon spins, causing both delay in time to market as well as increased costs of additional silicon/system verification. This paper presents the system level power management techniques that were done on Intel© Moorefield platform. Intel© Moorefield is a 22nm based quad core SOC that has been used in several key recent products (Asus Zenfone 2, Google Nexus player, Dell Iconic tablet, etc.).

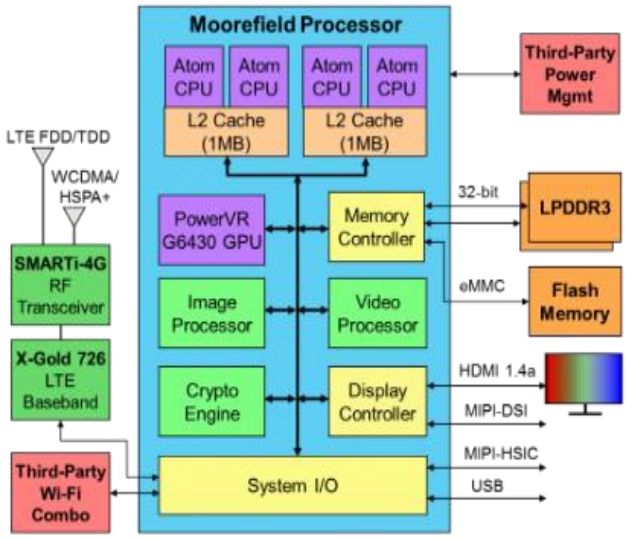
A. Problem Description – Pre-Silicon Power Management Verification of Complex SOCs

The last decade or so has seen an unprecedented explosion in multiple aspects of SOCs – complexity, cost, rapid cadence, etc. that have been primarily fueled by the demand for smart phones, tablets, connected devices, etc. On such increasingly large and complex digital IC and SoC designs, design power closure and circuit power integrity are starting to become one of the main engineering challenges, thereby impacting the device's total time-to-market. For example, in [27], the authors mention that the cost of showstopper bugs after tapeout was a few silicon spins, impacting schedule. However, a similar showstopper bug at revenue release time would cost an estimated $1 million per day, and bugs that show up a year after release would cost over $150 million in recalls, damaging corporate reputation, etc. Hence getting HW/SW right the first time is critical for survival in this industry. Given this, verification of power management features at all levels of design hierarchy has become crucial; at the same time, increasing in complexity and highly compressed time to market puts fundamental limits on how many features can be tested fully before silicon arrival. Bugs, especially in hardware, can result in subsequent silicon spins, causing both delay in time to market as well as increased costs of additional silicon/system verification.

Broadly speaking, the semiconductor industry uses several techniques for low power verification of hardware power management features. Some of these are described in Section II. The focus of this paper is system-level verification of key power management features in pre-silicon environments.

B. Intel© Moorefield Platform Overview

The Intel© Moorefield platform [4] based on 22nm process has all the major ingredients of a modern SOC for high-end smartphones/tablets/devices – quad core Silvermont cores (that can burst at 2.4 GHz), high-end GPU (PowerVR G6430 that can clock up to 533 MHz), high end cameras, 4GB LPDDR3 memory @ 533 MHz, Intel XMM 6360/6260 modem (that supports 150 Mbps downlink, HSPA+ 42Mbps for DSDS configuration), and a minute IA-based Sensor hub that provides an integrated sensor solution using ultra low power sensing modes. It also supports 64-bit Android L-dessert and upwards. The power management architecture on Moorefield platform spans from user space down to kernel/firmware, and finally the underlying SOC components that perform fine grained power management.



Moorefield supports several power features/modes that help in achieving longer battery life for key use cases, which we will describe subsequently – validating these as early as possible are crucial to ensuring silicon health and reduce unnecessary silicon spins.

C. System Level Pre-Silicon PM

In this paper, we describe the following important approach we took for system level PM verification:

1. *HW-SW Co-design approach* to aggressively "shift-left" silicon and software development together to de-risk silicon health.
2. Identification, and use of *targeted pre-silicon platforms* specifically tuned/enhanced for power management features at different levels of hardware, firmware and software. A combination of virtual platforms, hybrid and compact SLEs, FPGAs/Hybrid FPGAs were used, and we also had different customized Android OS kernel images booting on these platforms.
3. Multi-pronged strategy to system level power management verification – this included full Android OS boot, OS driver power management flows and system idle/low power states verification in emulation models. We also ensured that real/production PM firmware was developed/validated in cadence with RTL, and OS/driver/system flows were validated with each major version/release of RTL models/production firmware.

We believe that these aggressive and well-planned/executed methodologies yielded significant benefits in overall bug-free silicon/system and faster TTM. We believe that such a system-level and HW/SW co-design approach to pre-silicon PM verification is going to be important for complex SOCs/systems.

D. Organization of this paper

The rest of this paper is organized as follows. This section is in the Introduction. Section II details some of the related work in this area. Section III describes the Moorefield PM architecture in detail, and highlights the key areas that were critical for pre-silicon verification. Section IV describes the Pre-Silicon test methodology that we followed. In Section V we provide some of the key results and offer our conclusions/future areas of work in Section VI.

II. Related Work

Power management verification is a vast area of both academic and industrial relevance. In this section we survey some relevant pre-silicon verification methods at different levels of design abstraction. A more detailed survey can be found in [7]. Power management brings a host of new types of bugs which are not in the class of traditional functional bugs. The table below shows the different classes of bugs and the new verification techniques required.

Power Management Issue	**Verification Techniques required**
Isolation/level shifting bugs	Verify connection, placement, isolation/level shifting
Control sequencing bugs	Include power intent files like UPF
Retention scheme and control errors	Formulate test plan for architectural flows correctly
Electrical problems like memory corruption	Reach good power state coverage
Power/voltage sequencing bugs	Verify design in all states, transitions, sequences

Hardware/software deadlocks	Verify FW/SW control sequences
Power gating collapse/dysfunction, Clock domain/crossover bugs	Verification at each stage of design, not just RTL; verify netlist at each handoff, power switch/rail connectivity
Power-on/reset bugs	Wide coverage of test cases across power-on/reset flows
Thermal runways/cooling inefficiencies	Verify thermal conditions, thermal modeling for different form factors/designs
Bugs due to concurrent access from multiple IPs during end-to-end use cases	Verify end to end system level power sequences, including FW, SW, drivers to uncover race conditions

Table 1: Power Management issues, and different techniques for verification

Some of the above are hard to verify in pre-silicon – for example, voltage sequencing (due to lack of integrated power delivery models into SOC emulation models), thermal runways (usually happens on form factor devices). In this paper we attempted to cover most of the other issues above through a combination of silicon and system level verification methodologies. We cover some important areas for future work towards the end.

It is quite clear that without a carefully planned rigorous methodology in place, correctness guarantees will be hard to come by. Broadly, we can categorize pre-silicon power management verification into the following categories – (1) Low power transformations at gate-level – clock gating, etc. (2) Low power transformations at higher/architectural level – RTL (3) System level low power verification

A. Verification of Low Power transformations at gate level

Formal verification, especially equivalence checking, has achieved considerable success in the context of low power verification. Combinational equivalence checking checks two acyclic, gate-level circuits. Combinational equivalence checkers can also be used to check equivalence of two sequential designs, provided the state encodings of the two designs are the same. Although this technique has widespread use in many commercial tools, the real challenge of sequential verification is in verifying two designs with different state encodings. Sequential satisfiability engines [14], [15] and sequential ATPG engines [12], [13] solve this problem to a large extent by unrolling the circuit until a given time frame. However, these techniques operate at the gate level, where they reason in the Boolean domain.

B. Verification of Low Power transformations at RTL/architectural level

Given the nature of power management and the hardness of the problem at lower levels of design, more verification is usually focused on RTL and higher levels of abstraction. [16] describes methods to verify RTL power gating through transaction level models. Some attempts have been made to apply sequential equivalence checking to the behavioral RTL descriptions of designs. [17] describes a methodology for checking the combinational equivalence between C and RTL is described. [18] and [19] present *dedicated rewriting*, a rewriting methodology to automatically prove the correctness of low power transformations at the RTL-level. They propose a highly automated deductive verification technique which is fine tuned for low power transformations. They prove the equivalence of two Verilog RTL designs, one derived from the other after the application of a low power transformation.

C. Verification of Low Power features at Platform / System level

Industrial designs rely very heavily on ensuring that once the silicon arrives, power management can be validated as soon as possible, and thermal solutions can be built accurately for the specific form factor(s) in consideration. In order to accomplish this, typically companies use complex and costly FPGAs to emulate the entire chip/SoC RTL, and build platform level validation/verification tools that can include the ability to boot entire operating system on such FPGA complexes. SoftSDV from Intel [20] is a pre silicon functional verification tool. However, this does not allow for detailed power estimation/modeling/verification. Several such internal, proprietary validation systems are used typically across the industry for validation power management features. [21] presents a good overview of the different techniques used in system level low power verification and the importance of using power intent specifications like UPF, simulation tools/methodologies that can accurately model power states/sequences, etc. [9] describes System-C based virtual prototyping techniques to perform power intent/sequence validation, and also proposes using system level low power abstractions as possible extensions to UPF. This includes abstract definition of voltage relationships, dynamic aspects such as operating conditions. Addressing power-aware design and verification at higher level abstractions is certainly a very novel research area and is an important emerging area.

[10] talks about the importance of HW-SW co-design early in the product phases including pre-silicon, and the need for understanding and simulating end-to-end use cases in such verification methodologies. [8] discusses the implications of power management on verification of wireless SOCs, especially the impact of verification of wireless/RF components.

Closer to the topic of this paper, [22] discusses the challenges and experiences of the verification of Intel Atom processors with details on specific verification techniques used. This paper expands the power management verification of these Atom SOCs into system level pre-silicon PM verification.

III. Intel Moorefield Power Management Architecture

A. *General Concepts in Power Management*

Power consumption varies proportional to V^2F, where V is the operating voltage and f is the operating frequency. Total platform power consumption is sum of idle, active and leakage power.

1) Idle Power

In general, power management features are crucial for both idle scenarios (when the device is not being used frequently), and during active scenarios (gaming, etc.). Idle power management refers to how effectively the platform can manage power when idle. For eg. CPU C- states, device D-states and Platform S-states, as defined by ACPI (Advanced Configuration and Power Interface) [3] which is an industry standard that defines several aspects of platform power like active or sleeping. The CPUIDLE framework in Linux guides the CPU idle power management on a per-core basis [1].

2) Active Power

Active power management refers to how effectively the platform can perform work. For example, dynamic voltage frequency scaling (DVFS) is done in the CPU via different P-states (frequency states). P0 refers to the highest frequency state where the core can operate at during its C0 state. There are different levels of operational efficiency in C0 state such as P_1, P_2, P_n. The number of P states differs and depends upon the core. Change in frequency consists of two stages – a frequency transition and a voltage transition. When a transition to higher frequency is requested voltage transition happens first followed by frequency transition. When CPU operates at highest frequency, performance is better and higher the power. CPUFREQ framework in the Linux manages the performance states of the cores [2]. In general, *race to idle* refers to complete its task as fast as possible (high performance) and then shut off. *Crawl-to-idle* refers to delaying the task as long as possible, while consuming as little power as possible. Both get the work done, but the key differentiator will be the energy consumed. An intelligent management system will have to choose correctly whether to race to idle or crawl to idle.

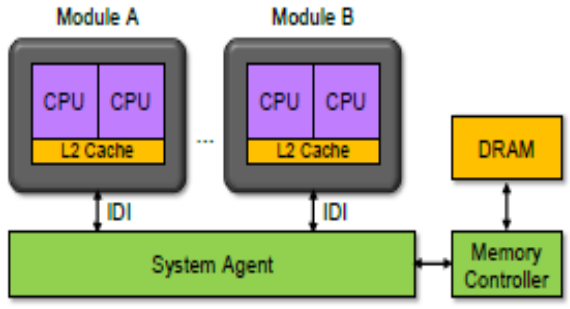
3) Leakage Power

Leakage power depends on the silicon, voltage and temperature. For eg. in some process technologies, for every 10°C change in temperature, silicon leakage varies. Techniques such as LTEC (Low Temperature Effect Compensation) and TDP (Thermal Design Power)-based sharing are used to compensate for low/high temperature scenarios. These are typically implemented in firmware algorithms, with some user configuration (of thermal limits, for example).

B. *Moorefield Power Management*

Moorefield supports several power management features at different levels – IP/hardware, SOC/platform, software/firmware/OS, etc. The power management architecture on Moorefield platform spans from user space down to kernel/firmware, and finally the underlying SOC components that perform fine grained power management.

Moorefield contains the Silvermont CPU cores that are described well in [23] and [26]. Silvermont greatly improves Atom's single-thread performance while still supporting multicore scaling – it scales from dual and quad core versions to 8-core micro-servers. One of the biggest change is its out-of-order (OOO) design; it is to be noted that Silvermont does not support hyper-threading, since the OOO design can cover misprediction penalties by executing subsequent instructions. In addition, Silvermont includes supports for newer x86 instructions like SSE4.1/4.2, is fully 64-bit compatible, supports extended page tables (VT-x2), and also supports real time trace/debug functions. As shown in the figure, two Silvermont CPUs are paired in a module along with a shared L2 cache. The CPUs share no function blocks other than the L2 cache. Each module connects to memory and other units through the system agent.



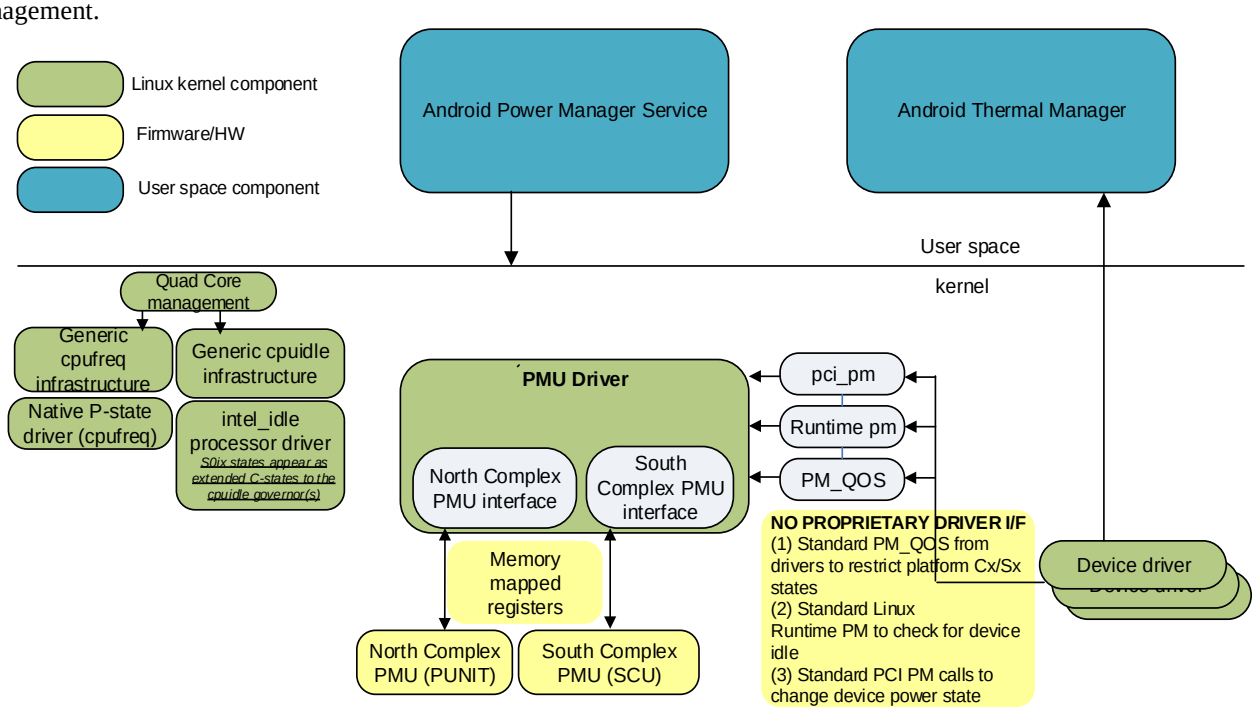
Each module has implemented a number of power-saving modes. The L2 cache, as well as each CPU, has its own voltage plane. The company can place each CPU in the C6 power-off state and can place the cache in an ultra-low-voltage state that can retain memory while the CPUs are inactive. Before the CPU enters a C6 state, the L2 cache can flush partially (but with the PLL still on) to turn off its unused parts, it can retain cache at a lower voltage while the

PLL is off, or it can turn off the entire cache and power it down. Intel calls these power states "subclasses" of the C6 state. It has also implemented a Fast C6 mode in which the CPU state saves to a dedicated on-chip memory.

Moorefield platform is not PC-compatible in several aspects - no BIOS, no ACPI, no legacy devices, no PCI enumeration in South complex, no real IOAPIC, etc. Most of these changes are available in the upstream kernel now. Refer to [5] and [6] for more details of the core kernel changes done for x86-based Intel platforms. The Moorefield power management architecture is built around the idea of aggressively turning off subsystems without affecting the end user functionality and usability of the system. This is enabled by several platform hardware and software changes:

1. On die clock and power gating of subsystems - applicable to all subsystems, fabrics, peripherals, etc.
2. Subsystem active idle states – applicable to all OS/driver controlled components. These states are termed as D0ix states. Traditionally (according to ACPI, for example), subsystems/devices can be in active power state (D0) or in low power state (D1/D2/D3). Most subsystems/platforms implement D0 and D3, however, not many platforms/systems implement really active idle states, where the platform is active, but subsystems, even though are idle are in lower power state. In Moorefield, devices can be in one of the following power states - D0 (Normal operational state), D0i1 (OS-transparent clock gated state), D0i3 (driver directed management of the subsystem with no OS control of the subsystem or D3 (OS directed, device driver is involved in the management of the subsystem and it must perform state retention and restoration in the driver). All devices will be managed through the runtime Linux power management infrastructure. Device drivers must implement D0i3 (driver managed autonomous power management) through the Linux Runtime power management framework, and aggressively (and intelligently) manage the power of their corresponding subsystems. Additionally, device drivers must also support standard Linux suspend/resume callbacks for implementing D3.
3. Platform idle states - extending idleness to the entire platform when all devices are idle. These are termed S0ix states, and are explained subsequently.

The power management architecture on Moorefield platform spans across multiple layers - Android user space, Middleware, Kernel, Firmware, and finally the underlying SOC components that perform fine grained power management.



The key components of power management architecture on Moorefield are:

1. Standard cpuidle-based CPU power management components (native drivers and governors).
2. Platform-specific S0ix extensions to the cpuidle driver (intel_idle-based) for the CPU
3. Power Manager Unit (PMU) driver - This driver interfaces with both North and South Complex Power Management Units (PMUs). It also provides platform-specific implementation of deep idle states to the intel_idle-based processor drive and coordinates with the rest of the platform using standard kernel Power Management interfaces like PM_QOS, Linux Runtime PM, etc.

4. PMU Firmware that coordinates power management between the Platform PMUs: P-UNIT for north complex (CPU, Gfx blocks, ISP), and SCU for south complex (everything else: IO devices, storage, comms, etc.)
5. Multi-core CPU idle power management – Moorefield supports a new power-aware scheduling technique at the OS level with a knowledge of the underlying CPU topology. For more details, see [28].
6. Device idle power management – device low power states, and corresponding kernel device driver support for Linux Runtime Power Management Framework
7. Active power management features
    a. cpufreq-based drivers and governors to support CPU Performance states / DVFS – Moorefield supports quad core burst at 2.3 GHz
    b. Graphics and ISP DVFS – both IPs support DVFS that is controlled by the corresponding device drivers. Drivers support corresponding Linux devfreq framework
    c. Power Sharing between multi-core CPUs and rest of the SOC when the device is performing high activity usages. A power manager (called the PUNIT) allocates the power budget among the CPUs and other modules, such as the graphics unit. This power manager also directly monitors temperature, thus enabling more dynamic and more accurate power allocation. A different power manager, called the System Controller Unit (SCU) coordinates the rest of the SOC/device power management (clock/power gating, voltage rails/shared resource management, etc.). In addition, the SCU also coordinates entry/exit of system idle/standby states.

This is summarized in the table below, and the key Linux/Android components that are responsible for coordinating the different power management states of components and the system as a whole. While some of these terminologies maybe Intel-specific, most complex SOCs today support similar features for power management. Hence the techniques for HW-SW co-design could be readily adopted in other designs as well.

Component	State	Description	Linux / Android framework
CPU Core power states	C0	CPU is busy executing instructions	CPUIDLE Infrastructure in Linux
	C1, C2, C6	CPU is idle, not executing instructions, clocks/power may be taken off	CPUIDLE driver predicts next expected interrupt and provides C-state hint through MWAIT() instruction
Device power states	D0	Active power state of device	Normal operating state, fully ON.
	D1, D2, D3	Devices are in suspended state clocks/power may be taken off	Linux Runtime PM Framework: Driver directed management of the Subsystem. The device driver MUST coordinate and manage the subsystem state, including save and restore of context. Linux Suspend/resume Framework: OS directed management of the subsystem. Device driver is involved in the management of the subsystem and must perform state retention and restoration in the driver.
System power States	S0	Active state of the system, most components fully ON/operational	
	S0i1,	Low latency idle state, can be entered/exited seamlessly transparent to applications. Most of the SOC can be in power-off state, except key peripherals performing low power operations (audio playback, sensing, memory transfer, etc.). Entry + exit latencies are in the order of ~750us - 1ms.	Low latency states with different entry/exit latencies, and power consumption. Completely transparent to OS/drivers/applications. Entry point from OS is a deeper C-state than C6, with a different hint to the CPUs to perform additional PM flow sequencing with PM firmware components/microcontrollers.
	S0i3	Similar to S0i1, with deeper power-off state across entire platform/SOC except for key always-on logic that can monitor the system for wakes. Entry + exit latencies are in the order of ~2ms.	Completely transparent to OS/drivers/applications. Entry point from OS is a deeper C-state than C6, with a different hint to the CPUs to perform additional PM flow sequencing with PM firmware components/microcontrollers.

	S3	System is suspended and the state of the system is stored in memory and all other components are turned OFF	Suspend. Applications are frozen, all system context is lost except system memory. CPU, cache, and chip set context are lost in this state. Hardware maintains memory context and restores some CPU and L2 configuration context.
Special Low power modes	S0ix-display	Entire SOC is in S0i1/S0i3 state except the display, which can be kept ON.	Coordinated by cpuidle and Gfx/display drivers based on the workload and idle prediction. Enabled on both command and video mode display panels.
	S0ix-sensing	Ultra low power sensing mode where everything in the SOC is OFF except some sensors and periodic sensor processing in the minute IA-based sensor hub	Coordinated by cpuidle, ISH and sensor drivers
	S0ix-audio	Low power audio playback, with the audio engine/DSP performing local playback and DMA to/from memory without waking up the rest of the SOC	Coordinated by audio, cpuidle drivers.

Table 2: Idle Power States and their management

IV. Pre-Silicon PM Verification Using System Level Emulation Environments

As it can be seen, the system level power management architecture described earlier requires PM verification spanning hardware, firmware, device drivers, OS kernel and interactions between these components. Hence a multi-pronged strategy was used:

a. Use of targeted pre-si environments for enabling specific PM features
b. "Real" PM firmware developed/validated in cadence with RTL
c. Customized Android kernel on major RTL + FW release for SW development
d. Device driver PM validation on major RTL + FW release and Android release
e. System Level PM flows/sequences validated on each major RTL + FW release

A detailed project plan was created for each of these steps, along with a master project plan, calling out the most important silicon tape-in gate items, key items that were crucial to be done before silicon power-on, PM checklist for power-on, etc.; all these were tracked across different teams, assigned the right priorities/resources and continuous progress was ensured.

In this section we will describe the different environments/techniques used in Intel, and call out the specific methodologies/environments that were used for effective HW-SW co-design for Intel© Moorefield platform.

A. Pre-Silicon Environments

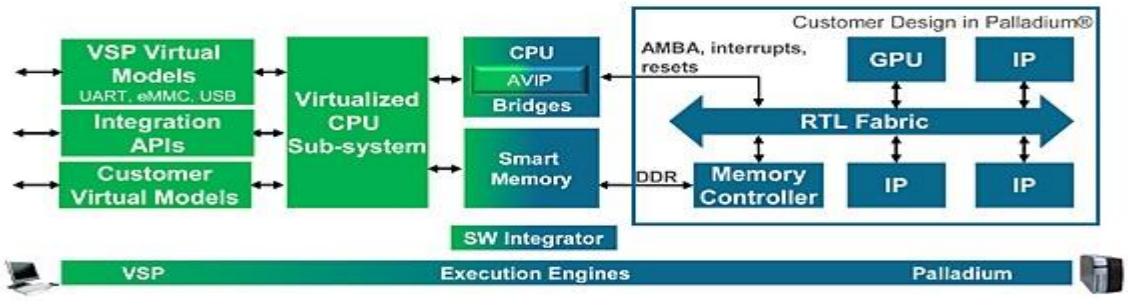
Targeted verification of each IP block, including CPU cores, Graphics, etc. were done using traditional silicon verification (SV) techniques through a combination of random, targeted and functional PM tests. Since the SOC integrated third party IP blocks, specific PM related SV tests were also designed for such IPs; all these were done completely by in-house silicon verification teams. Given the mix of different kinds of IPs, sources (third-party vs internal), etc., the silicon verification teams used a combination of SV, Verilog, VHDL test cases/methodologies.

As described in [24] and [25], Intel uses a combination of different platforms/environments for different aspects of pre-silicon verification. These include Virtual Platforms, FPGAs, Hybrid Virtual platforms (VP + FPGA), System Level Emulation (SLE) platforms, Hybrid SLEs, Full Chip SLEs, Hybrid Gen-1/FPGA (this refers to using the previous generation of silicon along with a FPGA prototype). Each environment is best suited for a specific set/category of pre-silicon verification. Some of them can support production OS boot in reasonable times for SW development/co-design/debug. Also, not all of these can be used for effective power management verification. For example, Hybrid FPGA + VP platforms include high speed with RTL accuracy, reuse of RTL models, mitigating the lack of a 3rd party virtual platform model, enabling pre-silicon systems with at-speed real-world devices, and "true" pre-silicon hardware/software co-validation. However, what goes into the FPGA and what goes into the virtual platform is an important decision that determines the efficacy of the environment.

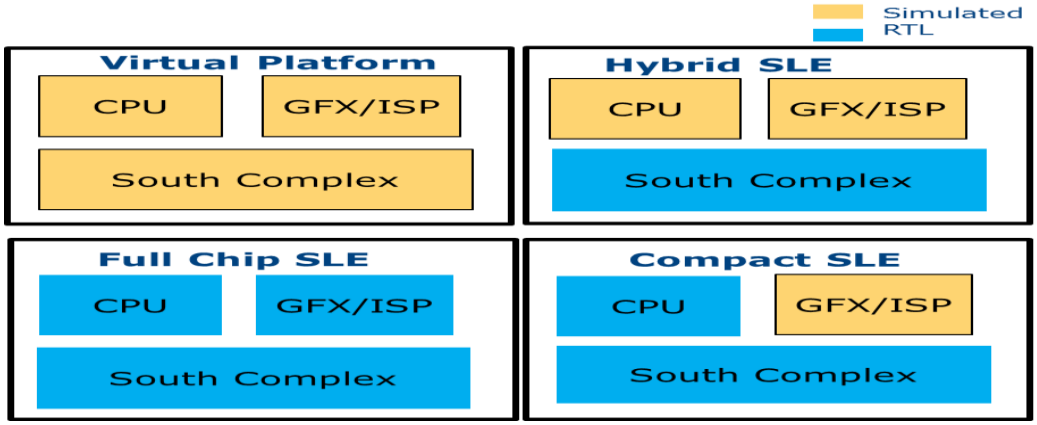
B. Pre-Silicon Environments for PM Verification

From a pre-silicon environment perspective, the industry is moving towards a continuum of connected hardware/software development engines, including virtual platforms, simulation, emulation, and FPGA prototyping.

While there are many types of hybrids, some have "TLM-emulation" [transaction-level modeling] hybrids for early OS and software bring-up. An example of this include the Cadence Palladium/Virtual System Platform hybrid solution depicted below. Virtual models, CPUs, and "smart memory" reside on the virtual platform side. Other system components run in the emulator. Users need to decouple the two domains so the virtual platform side can run at high speeds.



Historically, PM verification involved verification/test cases at the logic, RTL and focused/synthetic tests level. However, as SOCs became more complex with respect to the system level use cases, these traditional methodologies were not sufficient to uncover PM related bugs that spanned the whole system. This is fundamentally because there can be multiple modes of operations of the entire SOC involving hardware, firmware and software interactions across different applications, leading to complex dynamic power profiles of those applications. Hence, a system-level approach was adopted to cover different aspects of PM. We partitioned the testing across the following environments:



1. Virtual platforms – Here, the CPU cores are modeled in Simics, and had specific PM related messages/handshakes simulated with the underlying CPU power management controller. This was used for basic PM related OS driver development and very simple functional testing of CPU C-states, etc. This had the benefit of very fast full Android OS boot (within a few minutes), hence was ideal for SW driver developers to ensure their code is kept in alignment with the latest internal Android/Linux kernel codebase.
2. Hybrid SLE – This had the quad core CPUs simulated in Simics, and most of the SOC in RTL (except some specific complex IPs that needed separate emulators like Graphics, Imaging). These models also had several customized IP transactors for key IPs (like eMMC, USB, SDIO, I2C, etc.) that enabled PM touch-points (specific PM registers, handshakes, wake sequencing, interrupts, etc.). This could also boot a customized Android OS within 15-20 minutes, hence could be used for most driver PM flows verification. Hence this could be used to validate PM flows of most IPs, using the latest driver codebase mentioned above.
3. Compact SLE – This had the quad core CPUs and most of the SOC in RTL, hence could be used to validate most of the system level SOC PM flows/sequences – quad core CPU idle/performance states, SOC standby modes, etc.
4. The Graphics/Imaging IPs were modeled separately in full RTL emulation models, and corresponding PM registers/interfaces were accessible through memory-mapped registers, thus allowing SW drivers to enable PM flows/transactions.
5. Full Chip SLE – this had full SOC RTL, and was very slow for OS level debug; hence this was used only for ensuring OS boot prior to power on.

Additionally, in the above models, special clocking schemes were introduced going beyond normal fixed clocking for emulators/FPGAs; the infrastructure simulated the entire SOC clocking scheme with variable clock speeds, thereby we could validate changing core/bus speeds on the fly. Each environment above greatly helped PM testing.

C. System Level Pre-Silicon Methodologies PM Verification

In this section, we show how a system-level approach can benefit complex SOC pre-silicon power management verification. Broadly, we used the following coverage techniques/methodologies – (1) Silicon Verification test content focused on power management, (2) Customized Android OS that boots on different emulation environments, (3) System level, OS and driver PM flows verification with Android stack that complements synthetic silicon verification. Each of these are described subsequently.

Focused Silicon Verification (SV) for PM

Typically, SV comprises of focused IP-level testing (clock/power gating, isolation, firewalls, etc.) and synthetic tests (multiple IPs simultaneously – for eg., eMMC doing DMA to memory). Exhaustive SV test cases touched all SOC components - quad core CPU Power/Performance states with different fuse settings, DVFS of CPU, graphics, camera (through simulated workloads like Dhrystone), wake sequences (wakes due to snoops from devices, platform GPIO wakes, USB/modem wakes, etc.), graphics PM flows along with simulated/stubbed driver, PM flows involving new IPs like SSIC (voice call and PM sequences therein for modem components), etc. SV PM testing thus covered key HW PM scenarios.

Customized OS/fast boot techniques for SLE environments

In order to have easily usable Android/Linux kernel for SW/kernel developers, and specifically for power management drivers, a customized Android kernel was forked off the internal development mainline. The Android boot path was optimized for boot time by bypassing the kernel and initial file system decompression steps during initial kernel boot since memory-intensive operations (decompression) are slow in emulation. Tradeoffs were made with respect to the kernel configuration to provide the necessary components for PM validation in order to again speed up boot times. We optimized boot times further by injecting code into the boot loader to bump up the core frequency. In this manner, a reduced kernel configuration was created to match the emulation/firmware components thus providing an Android kernel that could boot on emulation models in all the environments described above. Once this methodology was established, each release of SOC RTL and firmware components had accompanying releases of Android kernel that could be used for extensive PM verification to ensure maximum coverage as the emulation models matured towards silicon tape-in and power on.

OS/driver PM flows verification

Extensive OS PM testing was then added to the above. On Compact SLE models, we enabled full Quad core CPU power/performance states – multi-core PM functionality such as individual core/module and package C-states were verified with OS by enabling the Linux cpuidle idle driver and menu governor. CPU DVFS was verified by enabling the Linux cpufreq infrastructure, with ondemand governor and CPU intensive workloads (like Dhrystone). Additionally, SOC standby (S0ix) flows and corresponding SW/FW/HW interactions/wakes were validated with thousands of back-to-back S0ix sequences on these models. In parallel, most IP PM flows were validated on hybrid SLE model with corresponding Android/Linux device drivers where Linux Runtime PM sequences tested out IP level clock/power gating sequences through the HW PM capability exposed by each IP transactor. Thus, significant OS power management validation was done on different pre-silicon environments.

V. RESULTS

The above infrastructure and methodologies enabled validating significant system level PM flows/sequences. Most of the important power management features were functional with full Android stack within a few days of Power On of the first silicon stepping. This aggressive use of pre-silicon testing helped to uncover and fix several bugs in the pre-silicon phase itself that led to overall quicker TTM.

PM Feature	**Power States**	**Verification performed before Silicon power-on/bring-up**	**Post Silicon PM enabling results**
CPU Core power, perf states	C0-C6	All C-states entry/exit sequences validated on bare-metal as well as with full Android stack. Exit sequences verified included	All CPU PM states, including DVFS and turbo features enabled with full Android OS stack within a week of silicon arrival

		interrupts, wakes, timers, etc. CPU DVFS including quad core DVFS validated with real benchmarks (like Dhrystone) on full compact SLE models	
Device power states	D0, D0i1, D0i3, D3	All power states of devices validated with driver code based on latest Android/Linux codebase/mainline. Driver power flows including power on, clock/power gating, and use of Linux Runtime PM for autonomous power management validated.	All device PM states (D0ix/D3) enabled with full Android OS stack within a week of silicon arrival
	S0i1, S0i3	Hundreds of S0i1, S0i3 cycles (entry and exit/wake flows) validated on compact SLE models	Basic S0i1/S0i3 states enabled with full Android stack within a couple of weeks of Silicon arrival
	S3	System is suspended and the state of the system is stored in memory and all other components are turned OFF. Firmware flows were validated in pre-silicon	Android Suspend-to-RAM enabled fully within a couple of weeks of Silicon arrival
Special Low power modes	S0ix-display	Not validated in Pre-silicon testing	With basic S0ix functional, enabling advanced features took much less time
	S0ix-sensing	Not validated in Pre-silicon testing	With basic S0ix functional, enabling advanced features took much less time
	S0ix-audio	Basic low power audio playback sequences validated in pre-silicon	With basic low power audio enabled, enabling advanced features took much less time

Table 3: Results of aggressive system level PM verification in pre-silicon

VI. CONCLUSIONS, FUTURE WORK

Power management verification of complex SOCs is a big challenge from hardware as well as from a system level perspective, since power management spans the entire platform. Ideally, each system component (IP/hardware/firmware/software) needs to be verified for its power management capability (both individually as well as how they work in relation to other components), and in addition, system-level power flows (low power idle/standby states, etc.) also need to be verified before silicon tape-in is achieved. Bugs, especially in hardware, can result in subsequent silicon spins, causing both delay in time to market as well as increased costs of additional silicon/system verification.

This paper described a system-level approach that we took to enable power management features early on in the silicon design phase, adopting an aggressive HW/SW co-design approach. Different pre-silicon environments were used for targeted PM test cases/sequences to ensure broad coverage of PM features as much as possible.

While this helped uncover a lot of bugs involving silicon/firmware/SW, and interactions across these layers, there are still a lot of improvements that can be done for future complex SOCs. Some of these (in no specific order of priority) are:

1. Some features are hard to validate in pre-silicon like low temperature effect compensation, power sharing between CPU and rest of SOC, since it is very hard to get dynamic current/power consumption data in pre-silicon environments.
2. Integration of power delivery components into SOC environments is still in its early stages since this involves integration of analog and digital components; this is an important area for the industry to collaborate on.
3. Non-logic issues like analog/RF/digital integration, process, etc. have become dominant aspects of post silicon verification cycles; doing this in pre-silicon is currently a very hard problem.
4. Pre-silicon environments today can at best do functional PM flows/sequences; power modeling/estimation is still done in a different domain using complex architectural models. Hence there is very limited capability to estimate power consumption during dynamic use cases that are run in pre-silicon models (for example, while we can run Dhrystone on multi core CPU models, it is very hard to estimate the power consumed in pre-silicon). This is an important area for future SOCs.

ACKNOWLEDGMENT

We would like to thank Ron Zinger, Vijay Bhimarao, Arvind Mandhani, Hari Seshadri, Terry Fletcher, Suresh Chemudupati, Edic Chyle, Maulik Kapuria, Shiraz Saleem, Bharadwaj Ramanujam, Miguel Vadillo, Sun Yi and

several others in the Atom SOC design/emulation, firmware and validation teams, and OS/Android device driver teams that were involved in this effort.

REFERENCES

[1]. V. Pallipadi, A. Belay, S. "*cpuidle: doing nothing, efficiently*". Ottawa Linux Symposium 2007.
[2]. V. Pallipadi, A. Starikovskiy. "*The Ondemand Governor: Past, Present and Future*", OLS 2006
[3]. ACPI Standards: http://www.acpi.info/
[4]. http://www.anandtech.com/print/7789/intel-talks-merrifield-moorefield-and-lte-at-mwc-2014
[5]. Jacob Pan. *Porting the Linux kernel to x86 MID Platforms*. Embedded Linux Conference, 2010.
[6]. Rajeev Muralidhar et. al. *Experiences with Power Management Enabling on the Intel Medfield Phone*. Proceedings of Ottawa Linux Symposium (OLS), 2012.
[7]. Vinod Viswanath, Rajeev Muralidhar, Hari Seshadri, Ananth Narayan. *Power Management Methods: From Specification and Modeling, to Techniques and Verification.* Journal of Low Power Electronics (JOLPE), 2012.
[8]. J. Scott Runner. "*Verification of Wireless SOCs: No Longer in the Dark Ages*", http://www.dvclub.org/events/109-verification-of-wireless-socs-no-longer-in-the-dark-ages
[9]. Fabian Mischkalla, Wolfgang Mueller. "*Advanced SOC Virtual Prototyping for System-Level Power Planning and Validation*", Intl. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS) 2014.
[10]. Swaminathan Venkatesan, Ashwin Menon. "*HW-SW Co-Verification: A Constrained Random Approach*". http://www.dvclub.org/events/78-hw-sw-co-verification-a-constrained-random-approach
[11]. http://www.linleygroup.com/mpr/article.php?id=11179
[12]. Jacob A. Abraham, Vivekananda M. Vedula, and Daniel G. Saab. *Verifying properties using sequential ATPG*. International Test Conference, pages 194–200, 2002.
[13]. Shi-Yu Huang, Kwang-Ting Cheng, and Kuang-Chien Chen. *Verifying sequential equivalence using ATPG techniques*. ACM Trans. Des. Autom. Electron. Syst., pages 244–275, 2001.
[14]. Feng Lu, Madhu K. Iyer, Ganapathy Parthasarathy, Li-C. Wang, Kwang-Ting Cheng, and Kuang-Chien Chen. *An efficient sequential SAT solver with improved search strategies*. In DATE, pages 1102–1107, 2005.
[15]. Mukul R. Prasad, Armin Biere, and Aarti Gupta. *A survey of recent advances in sat-based formal verification*. STTT, 7(2):156–173, 2005.
[16]. George Sobral Silveira, Alisson Vasconcelos Brito, and Elmar U. K. Melcher. *Functional verification of power gate design in systemc RTL*. In Proceedings of the 22nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes, SBCCI '09, pages 52:1–52:5, New York, NY, USA, 2009. ACM.
[17]. Luc Smria, Renu Mehra, Barry Pangrle, Arjuna Ekanayake, Andrew Seawright, and Daniel Ng. *RTL C-based methodology for designing and verifying a multi-threaded processor*. In DAC '02: Proceedings of the 39th conference on Design automation, pages 123–128, 2002.
[18]. Vinod Viswanath, Jacob A. Abraham, and Warren A. Hunt, Jr. *Automatic insertion of low power annotations in RTL for pipelined microprocessors*. In DATE '06: Proceedings of the conference on Design, automation and test in Europe, pages 496–501, 2006.
[19]. Vinod Viswanath, Shobha Vasudevan, and Jacob A. Abraham. *Dedicated rewriting: Automatic verification of low power transformations in RTL*. Journal of Low Power Electronics, 5(3):339–353, 2009.
[20]. Richard Uhlig, Roman Fishtein, Oren Gershon, Israel Hirsh, and Hong Wang. *SoftSDV*. http://download.intel.com/technology/itj/q41999/pdf/softSDV.pdf.
[21]. Bhanu Kapoor, S. Hemmady, Shireesh Verma, Kaushik Roy, M.A. D'Abreu. *Impact of SoC Power Management Techniques on Verification and Testing*. Proceedings of Intl. Conference: Quality of Electronic Design, 2009.
[22]. Shahram Salamian. *Intel Atom Processor Pre-Silicon Verification Experience*. http://dvclub.org/events/113-intel-atom-processor-pre-silicon-verification-experience
[23]. Linley Group, Microprocessor Report. *Silvermont Energizes Atom*. http://www.linleygroup.com/mpr/article.php?id=11041
[24]. EDPS 2015: "*Why "Hybrid" Platforms are Needed for Pre-Silicon Hardware and Software Development*". http://community.cadence.com/cadence_blogs_8/b/ii/archive/2015/04/29/edps-2015-why-hybrid-platforms-are-needed-for-pre-silicon-hardware-and-software-development
[25]. EDPS 2014 Keynote: "*What Intel Needs from Pre-Silicon Prototyping*". http://community.cadence.com/cadence_blogs_8/b/ii/archive/2014/04/21/edps-2014-keynote-what-intel-needs-from-pre-silicon-prototyping
[26]. AnandTech report on Silvermont - http://www.anandtech.com/show/6936/intels-silvermont-architecture-revealed-getting-serious-about-mobile
[27]. Jai Kumar, Sun Microsystems. "*HW Emulators: Does it belong in your verification tool chest*". http://www.dvclub.org/events/89-hw-emulators-does-it-belong-in-your-verification-tool-chest
[28]. Yuyang Du et. al. "*A new CPU load metric for power-efficient scheduler: CPU ConCurrency*" https://lwn.net/Articles/603504/