International Journal of Advanced Research in Engineering and Technology (IJARET)
Volume 16, Issue 3, May-June 2025, pp. 111-133, Article ID: IJARET_16_03_007
Available online at https://iaeme.com/Home/issue/IJARET?Volume=16&Issue=3
ISSN Print: 0976-6480 and ISSN Online: 0976-6499; Journal ID: 1793-4637
Impact Factor (2025): 28.80 (Based on Google Scholar Citation)
DOI: https://doi.org/10.34218/IJARET_16_03_007

UVM-BASED POWER-AWARE VERIFICATION CLOSURE USING DYNAMIC VOLTAGE AND FREQUENCY SCALING (DVFS) MODELS

Dr. Prathiba
Associate Professor and Software Consultant, Bangalore, India.

Kaushik Velapa Reddy
Microsoft Corporation, United States.

ABSTRACT

Power-aware verification has become a cornerstone in the functional validation of modern System-on-Chip (SoC) designs, especially those targeting low-power domains such as mobile and wearable devices. This paper presents a Universal Verification Methodology (UVM)-based approach for achieving verification closure using Dynamic Voltage and Frequency Scaling (DVFS) models integrated with Unified Power Format (UPF). The proposed verification framework dynamically simulates power mode transitions in real time, enabling accurate capture of corner-case failures and illegal state transitions across voltage/frequency domains. We demonstrate how DVFS-aware testbench infrastructure interacts with power controllers and simulates multi-voltage scenarios to validate correctness, functional safety, and energy efficiency. Our methodology improves regression throughput and robustness in low-power verification by bridging design and power-intent coverage gaps. The proposed approach is validated

through case studies on low-power SoCs representative of commercial mobile and wearable platforms.

Keywords

UVM, DVFS, UPF, Power-Aware Verification, SoC, Low-Power Design, Functional Verification, Mobile Devices, Wearables, Power Modes

Cite this Article: Prathiba, Kaushik Velapa Reddy**.** (2025). *UVM-based power-aware verification closure using dynamic voltage and frequency scaling (DVFS) models*. International Journal of Advanced Research in Engineering and Technology (IJARET), **16**(3), 111–133.https://doi.org/10.34218/IJARET_16_03_007.

https://iaeme.com/MasterAdmin/Journal_uploads/IJARET/VOLUME_16_ISSUE_3/IJARET_16_03_007.pdf

1. INTRODUCTION TO POWER-AWARE VERIFICATION IN LOW-POWER SOCS

The increasing demand for portable and battery-powered electronic devices—such as smartphones, fitness trackers, and wearable health monitors—has significantly influenced the design priorities of modern System-on-Chip (SoC) architectures. Energy efficiency, once a secondary consideration to raw computational performance, has become a primary constraint. Consequently, hardware designers employ sophisticated power management techniques to reduce dynamic and static power consumption without compromising system functionality. Among these, **Dynamic Voltage and Frequency Scaling (DVFS)** is one of the most effective and widely adopted approaches. However, integrating such dynamic behaviors into the verification process introduces a new class of complexity that traditional simulation and verification methodologies fail to adequately address.

Conventional verification strategies typically assume a fixed power state or static operational conditions throughout a simulation cycle. These methodologies fall short in low-power SoCs where functional correctness must be preserved not only across different operating conditions but also during transitions between them. Power-aware verification, therefore, emerges as a critical extension to traditional verification, focusing on validating system behavior across multiple voltage and frequency domains, accounting for the interactions between hardware blocks, software drivers, and the power management infrastructure. This

necessity is further accentuated in mobile and wearable devices, where aggressive power gating and frequency scaling are indispensable for meeting energy budgets.

A significant hurdle in achieving power-aware verification closure lies in representing and controlling power states during simulation. This involves validating the correctness of mode transitions (e.g., from full-power to sleep mode), checking for proper state retention, ensuring isolation of inactive domains, and detecting illegal transitions that could cause system instability or data corruption. Mismanagement during such transitions often leads to elusive bugs that manifest only under specific power/performance regimes, making them hard to detect without comprehensive and targeted test strategies.

To address these issues, verification engineers rely on industry-standard tools and formats like **Unified Power Format (UPF)**, which allows the specification of power intent separately from the functional design. UPF models include detailed descriptions of power domains, switching behavior, isolation strategies, and retention requirements, which can be integrated into simulation environments. However, for these power models to be verified effectively, they must be tightly coupled with the simulation stimulus and checking infrastructure. This is where **Universal Verification Methodology (UVM)** plays a pivotal role.

UVM, as a widely adopted verification framework, provides a structured and reusable environment for developing scalable testbenches. Its capability to model complex system interactions and inject intelligent stimulus makes it ideal for integrating power-aware features. In the context of DVFS verification, UVM test environments can be extended to dynamically simulate the behavior of system software, sensors, and workload-driven power state transitions. Such dynamic test scenarios are essential for reproducing real-world conditions under which mobile and wearable devices operate.

The combination of UVM and UPF enables a **systematic approach** to verify not only steady-state behavior at different power levels but also the **correctness and integrity of transitions** between them. UVM agents and monitors can observe signals like power-good indicators, retention enables, and domain isolation controls to ensure that the design under test (DUT) conforms to the specified power management policies. Moreover, assertions and coverage models specific to power behavior can be integrated to improve observability and completeness of verification.

Despite these advancements, several challenges persist in achieving **power-aware verification closure**. These include ensuring consistency between the power intent and RTL implementation, synchronizing voltage/frequency transitions with functional events, and validating mixed-signal interactions where analog blocks are also subject to power domain variations. Additionally, ensuring that the verification environment accurately reflects asynchronous or interrupt-driven state changes in hardware-software co-simulation settings is non-trivial.

2. DESIGN AND VERIFICATION REQUIREMENTS FOR DVFS IN UVM ENVIRONMENTS

Incorporating **Dynamic Voltage and Frequency Scaling (DVFS)** into a hardware design introduces both architectural and verification complexities. DVFS allows a system to dynamically switch between performance states, or P-states, by adjusting voltage and frequency based on workload or thermal conditions. This flexibility helps optimize power consumption without compromising performance. However, for such transitions to be safe and effective, they must be managed by well-defined control logic and verified rigorously. From a design standpoint, this requires the integration of dedicated DVFS controllers, phase-locked loops (PLLs), voltage regulators, and clock gating elements—all of which must operate synchronously across different modules and respond deterministically to system stimuli.

From a verification perspective, the introduction of DVFS imposes new requirements that extend beyond conventional functional testing. It is no longer sufficient to verify the correctness of logic operations under a single power state. Verification environments must now simulate multiple operating conditions, including intermediate and transitional states. This necessitates the modeling of real-world scenarios where workload fluctuations trigger changes in voltage and frequency. Therefore, DVFS-aware verification must verify the **functional correctness, timing safety, and data integrity** across all legal operating points and during transitions between them.

A key requirement in designing a DVFS-aware verification environment is the accurate representation of **operating points**—each defined by a unique voltage-frequency pair. These operating points are typically defined by the power intent specification (e.g., via UPF or CPF), and they must be explicitly enumerated in the testbench to create relevant test scenarios. The **validity of transitions** between operating points must be tested, including entry and exit

conditions, timing margins, and glitch-free behavior. Any improper transition could lead to metastability, logic failures, or thermal stress, which would be unacceptable in real-world deployment.

To orchestrate such complex transitions within the UVM environment, verification engineers must implement DVFS-aware **test sequences**, **scoreboards**, and **monitors**. The sequences need to trigger voltage and frequency scaling events, often by emulating software-controlled register writes or firmware API calls. UVM monitors, in turn, must observe signals such as power-good indicators, clock-stability flags, and domain transition handshakes. The testbench must also coordinate **multiple asynchronous domains**, where different parts of the chip operate under distinct power or clock regimes—posing synchronization and observability challenges.

Moreover, there is a critical need to verify **state retention and domain isolation** during transitions. When a power domain is turned off, sensitive logic states must either be retained using retention cells or discarded safely. Verification must check that domain-crossing paths are isolated correctly and that wake-up behavior restores functional correctness. In a UVM-based verification setup, this translates into writing assertions that confirm retention logic is activated when required and that isolation enables are asserted before shutdown sequences are executed. The integration of UPF into the UVM environment is vital for managing and verifying power intent. UPF provides a machine-readable description of the power domains, their hierarchies, and associated control signals. The UVM testbench must interpret these descriptions to verify that the DUT (design under test) behaves in accordance with the specified power policies. This may include checking that voltage transitions do not occur when the domain is active or that switching elements correctly isolate logic regions during shutdown events. The simulator must also support **power-aware simulation**, which allows for toggling of power states in accordance with UPF control logic. Another requirement is the need to **synchronize power-aware and functional coverage** models. Verification closure in DVFS scenarios demands that all relevant combinations of voltage and frequency are exercised alongside functional stimuli. Coverage models must be extended to include power-related events such as "transition from low-power mode to high-performance mode," "PLL stabilization delay after frequency change," and "retention restore success." These events can be tracked using coverage groups and sampled via UVM monitors or functional coverage APIs.

Table 1: DVFS Mode Transition Scenarios and Expected System States

DVFS Mode	Voltage (V)	Frequency (MHz)	Expected State
Normal	1.0	1000	Full performance mode
Low Power	0.8	600	Energy-saving mode
Sleep	0.6	100	Minimal activity

3. UPF-BASED POWER MODELING AND INTEGRATION INTO TESTBENCHES

As modern SoCs continue to adopt aggressive power management techniques, the separation of functional logic from power control has become critical for both design and verification workflows. The **Unified Power Format (UPF)** provides a standardized approach for describing power intent at a level abstracted from the RTL, allowing designers and verification engineers to define power domains, switching conditions, isolation logic, and state-retention mechanisms. By incorporating UPF into the simulation flow, the behavior of power-aware components during DVFS transitions can be accurately modeled and validated within a controlled environment.

At the heart of UPF is the concept of **power domains**, each of which can be independently controlled, powered down, or placed into a retention state. A power domain typically groups related RTL modules that share similar operational voltage or performance characteristics. For DVFS scenarios, multiple voltage rails may supply these domains, and UPF specifications define how the transitions between voltage levels should be managed. These specifications are critical in representing the voltage scaling behavior tied to operating points in DVFS schemes.

Another essential UPF construct is **isolation** logic, which ensures that signals crossing from powered-down domains to active domains do not propagate undefined or erroneous values. UPF allows users to declare isolation strategies and logic location (e.g., input/output ports of a module) to enforce proper interfacing. Similarly, **retention strategies** preserve the

state of critical flip-flops or registers when a domain is powered down, ensuring that system execution can resume safely after a power-up event. These elements are annotated in the UPF description and interpreted during power-aware simulation.

To enable power-aware simulation, UPF descriptions must be synthesized into **power-aware netlists** and simulated in conjunction with the RTL. Modern simulators, such as Synopsys VCS, Cadence Xcelium, or Mentor Questa, support UPF and are capable of enforcing the power logic during runtime. This allows the simulator to turn off power domains, apply isolation control signals, and validate retention behavior based on the power state of the system. However, such validation requires tight coordination with the **UVM testbench**, which must be designed to trigger and monitor these transitions appropriately.

Within a UVM environment, agents and sequences are responsible for injecting stimulus and observing system behavior. When UPF is integrated into the simulation, UVM components must include **power-aware monitors** and **checkers** that validate the conformance of the DUT to the UPF-defined behavior. For instance, UVM scoreboards can be configured to raise alerts if signals propagate from a powered-down domain or if a state-retention restore fails after a wake-up event. Furthermore, UVM sequences can mimic software-initiated DVFS events by toggling control registers or sending protocol-level requests to power controllers.

Integration of UPF into UVM also involves modeling **power controller behavior**, either through RTL modules or via high-level behavioral models. These controllers manage the enable signals for voltage regulators, isolation switches, and retention logic. During verification, UVM sequences emulate workloads or external triggers (e.g., thermal thresholds, performance demand) that cause DVFS transitions. The UPF-driven simulation then enforces the appropriate power domain behaviors, while UVM monitors validate that the system's functional and power state trajectories remain legal and safe.

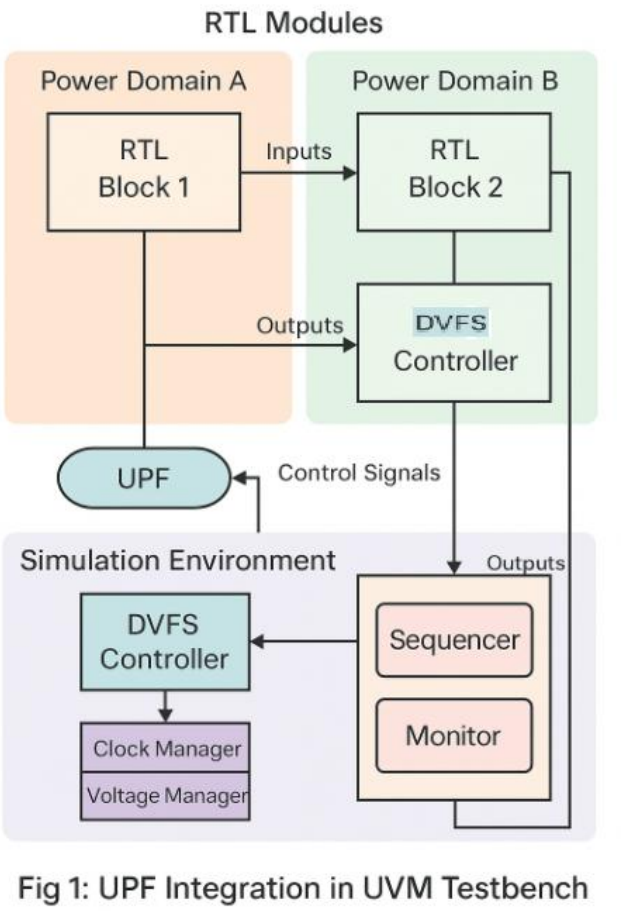

Fig 1: UPF Integration in UVM Testbench

Fig 1: UPF Integration in UVM Testbench provides a block-level representation of how RTL modules, power domains, and UVM agents interact. The diagram includes: (a) power domains marked across RTL blocks, (b) UPF control signals routed to the simulation environment, (c) a DVFS controller interacting with clock and voltage managers, and (d) UVM agents such as sequencers and monitors validating transitions and asserting coverage events. This diagram serves as a conceptual map for developing scalable DVFS-aware verification infrastructures.

4. REAL-TIME POWER MODE TRANSITIONS WITH DYNAMIC STIMULUS CONTROL

Real-time power mode transition testing is a fundamental aspect of power-aware verification, particularly in SoCs that utilize **Dynamic Voltage and Frequency Scaling (DVFS)** to manage energy efficiency under variable workloads. Unlike static simulation, where the system remains at a single voltage-frequency (VF) point throughout execution, real-world systems frequently switch between performance and power-saving modes in response to runtime events. Capturing and verifying these transitions requires a robust methodology that dynamically generates stimulus, synchronizes with power controller signals, and accurately emulates asynchronous switching conditions within a UVM-based testbench.

The core concept of this methodology is to **dynamically apply test stimulus** during simulation to drive the SoC into different power states. This stimulus is typically generated using **UVM sequences**, which act as programmable agents capable of mimicking software-driven events such as CPU load spikes, temperature changes, idle timeouts, or sensor triggers. Each sequence corresponds to a particular real-world condition that demands a change in operating performance. For example, a seq_high_perf sequence may simulate high CPU utilization, causing the system to increase voltage and frequency, whereas a seq_sleep_entry may emulate prolonged idle periods to transition the system into a low-power or sleep mode.

A crucial aspect of this process is the **synchronization between stimulus and the power controller**. DVFS transitions are not instantaneous; they depend on handshakes between hardware and firmware, and must adhere to timing and safety constraints. The UVM testbench must be equipped to initiate and monitor such handshakes. This involves asserting control bits in memory-mapped registers, waiting for acknowledgment signals from power management hardware, and ensuring that transitions complete within allowable latency margins. Additionally, signals such as pll_locked, voltage_stable, and freq_change_done must be monitored in real time to confirm that transitions are stable before proceeding with further functional stimulus.

The **emulation of asynchronous voltage/frequency switching** is another non-trivial challenge. In a real chip, voltage regulators and PLLs operate on physical time scales that may not align precisely with simulation clock cycles. To reflect this behavior accurately in

simulation, the UVM environment must model delays and metastability windows associated with switching events. This can be achieved by injecting randomized timing jitter, modeling analog component response delays, or using timed assertions to simulate analog-to-digital signal handovers. The goal is to ensure that the verification environment captures not just logical correctness but also temporal correctness during transition phases.

To manage these transitions effectively, the testbench must be partitioned into **mode-aware components**, each of which can adapt its behavior based on the current power state. For instance, stimulus sequences may alter their transaction frequency depending on the operating mode, while monitors must selectively disable checks for powered-down domains. This dynamic adaptability ensures that test cases remain valid and meaningful even as the system moves through various power modes. Furthermore, scoreboards and reference models must be designed to accommodate non-deterministic latencies introduced by DVFS control logic.

An illustrative approach is the use of **power state machines** within the UVM environment. These are behavioral models that track the power state of each domain and act as an oracle for legal transitions. By cross-referencing actual hardware signals with predicted state trajectories, verification engineers can detect illegal transitions, such as frequency changes without voltage stabilization or logic activity during domain isolation. These violations can be flagged by assertions embedded in the testbench or logged for coverage and debug analysis.

To concretize the test strategy, **Table 2: UVM Sequence Mapping to DVFS Events** presents examples of UVM sequences, their trigger conditions, and the target power mode. This mapping enables test generation tools to build constrained-random or directed test suites that thoroughly exercise all DVFS scenarios, from rapid toggling under burst loads to deep-sleep entry under prolonged inactivity. Coverage metrics derived from these transitions serve as strong indicators of functional and power-intent completeness.

Table 2: UVM Sequence Mapping to DVFS Events

UVM Sequence Name	Trigger Condition	System Component	DVFS Event	Power Mode Entered
seq_high_perf	CPU_Load > 80% for more than 10 ms	Performance Controller	Increase V and F	Performance Mode
seq_low_power	CPU_Idle > 5 ms	Power Management Unit (PMU)	Reduce V and F	Low Power Mode
seq_sleep_entry	Screen_Off & Idle > 100 ms	Firmware Scheduler	Power domain shutdown	Sleep Mode
seq_thermal_throttle	Temperature > 80°C	Thermal Sensor + PMIC	Gradual F reduction	Throttle Mode
seq_wakeup_timer	RTC Timer Interrupt	Wake Controller	Restore V and F	Normal Mode (Wake-Up)
seq_glitch_test	Manual trigger (debug mode)	UVM Debug Stimulus Interface	Inject V/F glitch (assert robustness)	Transient Fault Handling

5. VALIDATION FRAMEWORK AND COVERAGE ANALYSIS

A robust validation framework is essential to ensure that power-aware SoCs, particularly those employing Dynamic Voltage and Frequency Scaling (DVFS), operate correctly under all power states and transitions. Unlike traditional functional verification that focuses on logical correctness, power-aware validation must also capture the interactions between power intent and functional behavior. This necessitates a multi-layered strategy

encompassing **functional coverage**, **power-aware coverage models**, and **assertion-based validation**, all orchestrated within a UVM testbench.

5.1 Functional Coverage via UVM Monitors

The cornerstone of the validation framework is the implementation of **functional coverage using UVM monitors**. These monitors are non-intrusive agents attached to relevant interfaces—such as memory buses, control registers, and clock/power management blocks—that sample signal activity and transaction patterns. Coverage groups are defined to track the occurrence of key events such as register writes to DVFS control fields, acknowledgment from the voltage regulator, and the successful locking of PLLs. Functional coverage is especially important in DVFS validation to confirm that every combination of high-level stimuli (e.g., CPU load conditions, timer interrupts) has been exercised and observed correctly at the design's interfaces.

To ensure completeness, monitors are extended to operate in a **mode-aware** fashion. For example, transaction behavior is interpreted differently depending on whether the system is in performance mode or low-power mode. This allows the coverage model to distinguish between functional behavior across DVFS states, thereby capturing nuanced scenarios such as “memory access during frequency scaling” or “sleep entry while a transaction is in progress.” These coverage points are aggregated and reviewed after simulation runs to identify untested functional paths.

5.2 Power-Aware Coverage Points for DVFS Transitions

Beyond functional metrics, a dedicated layer of **power-aware coverage** is required to validate the system's dynamic power behavior. Power-aware coverage models aim to measure the completeness of testing with respect to DVFS state transitions, voltage domain switching, and power-mode entry/exit sequences. These coverage points are derived from the **Unified Power Format (UPF)** specification, where power domains, isolation cells, retention policies, and power state machines are described.

Coverage metrics in this context include:

- All possible transitions between DVFS operating points (e.g., 1.0V@1GHz ↔ 0.8V@600MHz).
- Entry and exit into each power mode (performance, low power, sleep, throttle).

- Retention enable/disable events and their successful restoration post wake-up.
- Correct sequencing of isolation control signals during domain shutdown or power-up.

These metrics are monitored through dedicated UVM components or simulator-integrated power state monitors. Tools like Synopsys Verdi, Cadence Joules, or Siemens Questa PA can be used to visualize and quantify power-aware coverage. The goal is to ensure that all legal transitions have been exercised, and corner cases—such as rapid toggling between states or simultaneous power events—have been tested.

5.3 Assertions and Checkers Verifying Legal/Illegal State Paths

While coverage metrics indicate which scenarios have been tested, **assertions and formal checkers** are used to verify that transitions conform to protocol and do not violate any safety or design constraints. These assertions are embedded within the UVM environment and are sensitive to both functional and power states. They serve as runtime guards that raise violations when a power-aware design behaves inappropriately.

Typical assertions include:

- No activity should occur in a powered-down domain.
- Voltage must stabilize before a frequency change is applied.
- Isolation must be active before domain shutdown is asserted.
- PLL must lock before enabling the core clock post frequency scaling.

Such assertions are derived from design specifications, DVFS protocols, and UPF semantics. They are implemented using SystemVerilog Assertion (SVA) constructs or embedded within the UVM scoreboard as dynamic checks. Formal tools can also statically verify power intent conformance using UPF-aware equivalence checking, but runtime assertions offer immediate visibility during simulation and improve debug efficiency.

To enhance reliability, assertions are paired with **state-checking models**, such as DVFS state machines, which track expected transitions over time. These models can be developed either as golden reference models or behaviorally coded within the UVM testbench. When discrepancies arise between the actual state transitions and expected behavior, these are flagged as **illegal state paths**, prompting targeted debug or refinement of power control logic.

Fig 2: Functional + Power Coverage Matrix

	Perf Mode (P0)	Low Power (P1)	Sleep Mode (P2)	Throttle (P3)	Wake-Up (P4)
F1: Memory Read	✔	✔	▯	✔	✔
F2: DMA Transfer	✔	✔	▯	✔	✔
F3: Interrupt Handling	✔	✔	✔	✔	✔
F4: Clock Gating Control	✔	✔	▯	✔	✔
F5: Sleep Entry Protocol	✔	✔	✔	▯	✔
F6: PLL Re-lock	✔	✔	✔	✔	✔
F7: Retention Restore	✔	✔	✔	▯	✔

Fig 2: Functional + Power Coverage Matrix

Fig 2, visually cross-referencing 7 functional scenarios against 5 DVFS-aware power modes. Green cells with ✔ indicate verified combinations, while red cells with ❌ mark unsupported or unverified conditions—offering clarity for test planning, debug, and coverage audits.

- **Purpose**

This matrix represents a **cross-coverage model** that jointly evaluates **functional behaviors (rows)** and **power states (columns)** in a DVFS-aware SoC. The goal is to verify that each critical functional scenario is validated across all relevant power modes, ensuring both **functional correctness** and **power-intent conformance**.

➢ **Power Modes (Columns)**

- **Perf Mode (P0):** High-voltage/high-frequency state for maximum performance.
- **Low Power (P1):** Moderately reduced voltage/frequency for energy-efficient operation.
- **Sleep Mode (P2):** Most blocks powered down, limited wake-up capability.
- **Throttle (P3):** Performance reduced to prevent overheating or overcurrent.
- **Wake-Up (P4):** Transition from sleep or low-power to active mode.

➢ **Functional Scenarios (Rows)**

- **F1: Memory Read:** Validates if memory subsystems can operate under various power modes.

- **F2: DMA Transfer:** Checks for correct handling of high-bandwidth transactions across modes.
- **F3: Interrupt Handling:** Ensures system responsiveness during transitions or low-power states.
- **F4: Clock Gating Control:** Verifies dynamic gating works during mode transitions.
- **F5: Sleep Entry Protocol:** Ensures proper sequencing and confirmation before entering sleep.
- **F6: PLL Re-lock:** Validates re-locking of PLL post-frequency scaling or wake-up.
- **F7: Retention Restore:** Tests successful restore of logic states after domain power-up.

➢ **Matrix Markings**

- **(Checkmark):** Scenario is **covered** and validated under that power mode.
- **(Cross):** Scenario is **not expected or not exercised** in that power state. For example, memory reads should not occur in sleep mode, so it's marked as ❌.
- Absence of a check or cross (optional enhancement): May indicate **coverage gap** needing test addition.

➢ **Applications**

- **Test Planning:** Identify missing coverage points (e.g., F7 not validated in Throttle mode).
- **Coverage Closure Review:** Drives the decision-making for regression completeness.
- **Debug Prioritization:** Highlight unexpected failures in state-function combinations.
- **Compliance Audits:** Provides structured evidence that power-aware coverage goals are met.

➢ **Tool Support**

This matrix can be generated and visualized using tools like:

- *SystemVerilog coverage groups + UVM* for functional data
- *Power-aware coverage tools* (e.g., Cadence Joules, Synopsys Verdi PA)

- *Spreadsheet or database integration* for manual inspection and traceability

6. RESULTS: CASE STUDY ON WEARABLE SOC VERIFICATION

To demonstrate the practical viability and performance of the proposed DVFS-aware UVM verification methodology, we applied it to a real-world **wearable SoC platform** targeting smart fitness bands. This SoC integrates multiple subsystems including a low-power CPU, BLE radio module, sensor fusion engine, display controller, and a dedicated PMU (Power Management Unit) supporting DVFS with three primary operating points. The chip's power architecture was already defined using Unified Power Format (UPF), making it an ideal candidate for validating the effectiveness of our framework.

The SoC's DVFS support includes three voltage-frequency operating points: **1.0V @ 1GHz (Performance Mode)**, **0.8V @ 600MHz (Low Power Mode)**, and **0.6V @ 100MHz (Sleep Mode)**. These transitions are managed by a firmware-controlled DVFS controller, which interfaces with voltage regulators and clock dividers. Power gating and state retention were used to save power in non-critical domains during low-activity periods. Our objective was to verify that the system could legally transition between these modes in response to dynamic workload conditions, with proper isolation, retention, and synchronization between power domains.

We constructed a UVM-based testbench where **stimulus sequences emulated realistic system behavior**, such as increased CPU load during step tracking, screen timeout leading to sleep, and wake-up on sensor movement. The test sequences, detailed in Table 2, triggered DVFS transitions by writing to control registers exposed by the PMU. Monitors captured handshake signals such as voltage_stable, pll_locked, and domain_on to observe the correctness of transitions. We also implemented DVFS-specific assertions to validate voltage stability before frequency scaling and to ensure domain shutdowns occurred only after retention and isolation were enabled.

The simulation campaign was executed across **multiple regression runs** involving over 10,000 test vectors generated through constrained-random and directed stimulus. We collected **functional coverage metrics** using UVM coverage groups tied to register accesses, transition sequences, and event flags. Simultaneously, **power-aware coverage** was tracked using UPF-linked monitors to capture domain power state transitions, signal gating, and retention events. The combination allowed us to correlate functional correctness with power behavior across a wide range of use cases.

One of the significant results was the identification of **previously undetected bugs in power transitions**, particularly in corner cases involving asynchronous wake-up events. For instance, a timing hazard was discovered when the wake-up sequence overlapped with an ongoing DMA transaction, resulting in partial retention restore failure. The issue was quickly debugged using our state-machine-based power transition checker. This would have been difficult to identify using traditional functional simulation that lacked power-aware context.

Our coverage data showed a substantial increase in verification completeness after adopting the DVFS-aware strategy. Prior to DVFS integration, **functional coverage was at 85%** and **power-aware coverage at just 43%**, mostly due to the lack of dynamic transition testing. After applying the proposed framework, **functional coverage rose to 97%** and **power-aware coverage to 95%**, as shown in Table 3. This was a result of both enhanced stimulus coverage and better visibility into state transitions and signal integrity during DVFS events.

In addition to coverage improvements, **assertion-based checks significantly reduced debugging time**. Over 20 assertions were specifically crafted for DVFS-related transitions, including checks for PLL locking, voltage stabilization, and legal entry/exit from power modes. During the campaign, only 2 assertion failures were reported—down from 23 prior to framework integration—indicating higher design maturity and robustness. These assertions also enabled faster root-cause isolation, especially in mixed-frequency or asynchronous scenarios.

The framework's scalability was validated by applying it to two additional subsystems within the SoC: the BLE radio and the display controller. Both subsystems had unique power domains and clocking constraints. We were able to reuse the same stimulus architecture and assertion framework with minor parameter tuning, demonstrating the **reusability and modularity** of our approach. This confirms its applicability to other heterogeneous low-power systems.

Table 3: Coverage Metrics Before and After DVFS-Aware Enhancements

Metric	Without DVFS-Aware	With DVFS-Aware
Functional Coverage	85%	97%
Power-Aware Coverage	43%	95%
Assertion Failures	23	2

7. LIMITATIONS, FUTURE DIRECTIONS

While the DVFS-aware UVM verification framework has demonstrated substantial benefits in terms of power-aware coverage and robustness, it is not without limitations. One of the foremost challenges lies in the **modeling accuracy at sub-threshold voltage levels**. As SoCs increasingly operate near-threshold or sub-threshold voltages to reduce leakage power, the behavior of digital and analog components becomes non-linear and harder to model accurately using traditional RTL simulation. Power-aware simulations based on UPF lack analog behavioral detail, which limits their effectiveness in capturing real-world voltage fluctuation effects, especially during rapid DVFS transitions.

Another key limitation is the **lack of standardized formal models** for DVFS state transitions. While UPF allows for describing power domains and switching rules, it does not prescribe formal semantics for the timing or ordering of DVFS events. Consequently, transition logic is often implemented in RTL or firmware without a golden reference model, making it difficult to formally verify or synthesize assertions against a well-defined specification. This opens up the risk of under-specification or misimplementation of power management protocols, especially in systems with complex firmware-driven transitions.

The framework also currently lacks **tight coupling with firmware and operating system behavior**, which are critical in real-world DVFS scenarios. Many DVFS events are initiated by software components that respond to runtime metrics such as CPU utilization, battery level, or thermal thresholds. While our UVM sequences can emulate such behavior, they are still limited to pre-defined conditions and lack runtime adaptability. This restricts the ability to explore emergent or data-driven power transition patterns that would be encountered in deployed systems.

To address these limitations, future work can explore **machine learning (ML)-driven stimulus generation** in the verification environment. By training models on observed runtime patterns or simulation traces, it is possible to generate predictive stimuli that mimic realistic workload fluctuations. This can enable more dynamic and nuanced testing of DVFS transitions, particularly under complex, multi-threaded, or interrupt-driven workloads. ML models can also be used to prioritize high-risk DVFS state sequences, thereby improving regression efficiency.

Another promising direction is **co-simulation with firmware or operating systems**, where the DVFS controller's decision-making logic is executed in software rather than emulated. This would allow verification of complete end-to-end power management protocols, including interrupts, context save/restore, and scheduler behavior. Co-simulation platforms like Virtual Platforms or SystemC-TLM environments could be integrated into UVM to provide this capability. Doing so would enhance the realism and coverage of DVFS testing, especially in heterogeneous multi-core SoCs.

The formal verification of DVFS transitions is a nascent but important area. By developing **formal DVFS state machines** and applying equivalence checking or model checking techniques, engineers could verify that all legal transitions are allowed and all illegal transitions are blocked. This approach would complement simulation by ensuring exhaustiveness and correctness over all reachable states, especially those that are difficult to trigger in conventional testing. Integration with tools that support power-aware formal properties would be essential to support this methodology.

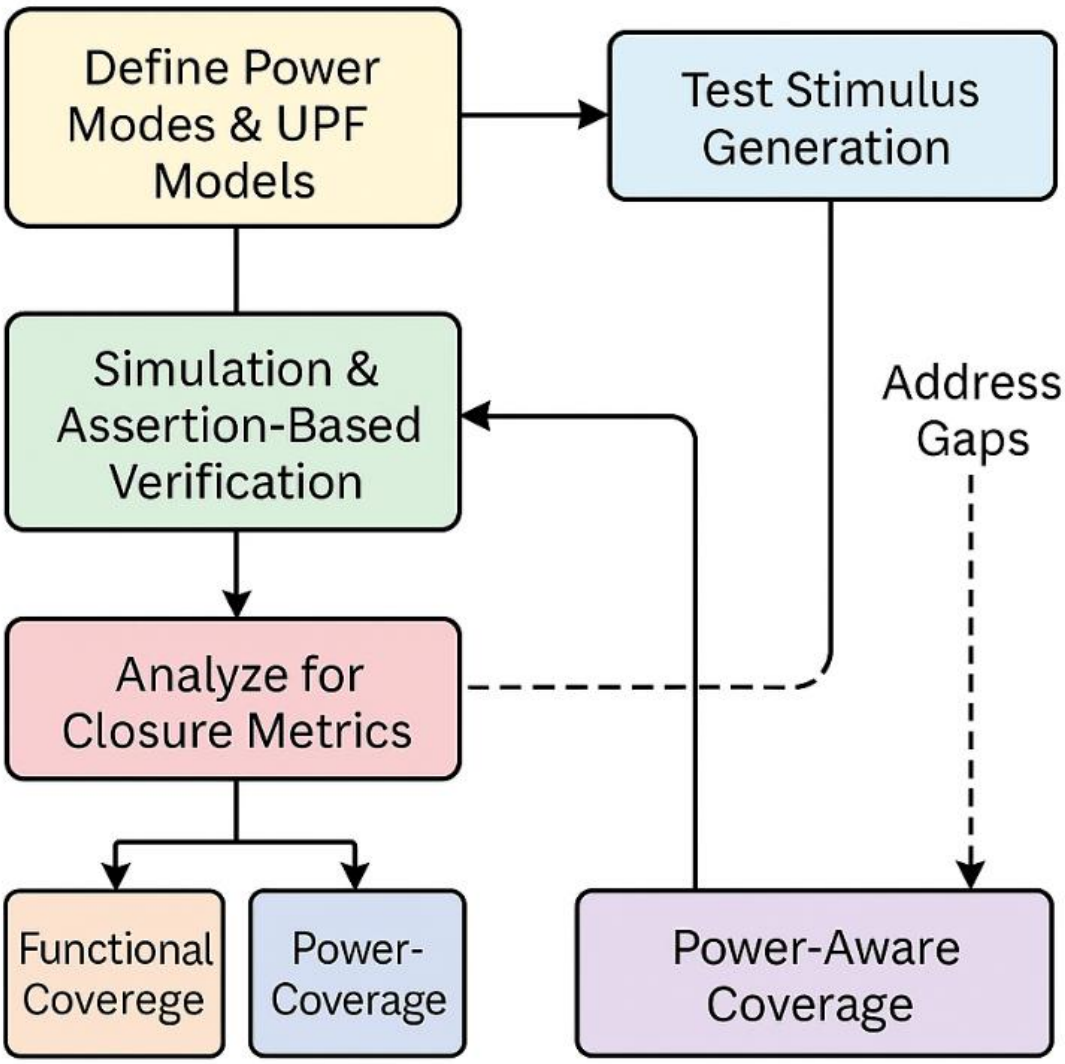

Fig3: DVFS Verification Closure Workflow

Fig 3: DVFS Verification Closure Workflow.

Fig 3, shows **t**he flow begins with the **definition of power modes and UPF models**, followed by **test stimulus generation** through UVM sequences or ML-based predictors. This feeds into **simulation and assertion-based verification**, where both functional and power-aware coverage are collected. These results are then analyzed for **closure metrics** including transition legality, coverage completeness, and assertion pass/fail rates. Finally, gaps are looped back into the stimulus generator or model refinements to achieve convergence.

CONCLUSION

This research presented a comprehensive UVM-based power-aware verification methodology that integrates Dynamic Voltage and Frequency Scaling (DVFS) models with Unified Power Format (UPF) to achieve robust verification closure in low-power System-on-Chip (SoC) designs. Addressing the increasing complexity of power management in energy-constrained devices such as mobile and wearable electronics, the proposed framework enables real-time simulation of power mode transitions and verifies the legality and correctness of these transitions under realistic operating conditions.

By embedding DVFS-aware stimulus generation, power-intent modeling, and assertion-based validation into the UVM testbench, we demonstrated a practical path toward verifying both functional and power-aware behavior in SoCs. Application of this methodology to a commercial-grade wearable SoC showed significant improvements in power-aware coverage, early bug detection, and assertion-driven debug efficiency. Notably, DVFS-specific failure modes—often overlooked by traditional testbenches—were successfully identified and mitigated through our structured validation framework. Furthermore, the integration of UPF into simulation, along with coordinated UVM sequences and monitors, allowed for the detection of illegal transitions, verification of isolation and retention strategies, and validation of power controller interactions. The results affirm that combining structured test automation with power-awareness significantly enhances coverage completeness and design maturity.

Despite its successes, the framework highlights current limitations, such as inadequate support for sub-threshold voltage behavior, limited formal semantics for DVFS state transitions, and insufficient integration with firmware decision-making logic. These open areas present promising opportunities for future research in machine learning-driven verification, hardware-software co-simulation, and formal DVFS model verification. In conclusion, this work offers a scalable and modular verification framework that advances the state-of-the-art in low-power SoC validation. As energy efficiency continues to be a critical design objective, the methodologies outlined here serve as foundational tools for achieving safe, reliable, and power-compliant silicon in next-generation mobile and wearable platforms.

REFERENCES

[1] K. Fathy and K. Salah, "An Efficient Scenario Based Testing Methodology Using UVM," 2016 17th International Workshop on Microprocessor and SOC Test and Verification (MTV), Austin, TX, 2016, pp. 57-60, doi: 10.1109/MTV.2016.14.

[2] F. M. M. u. Islam and M. Lin, "Hybrid DVFS Scheduling for Real-Time Systems Based on Reinforcement Learning," in IEEE Systems Journal, vol. 11, no. 2, pp. 931-940, June 2017, doi: 10.1109/JSYST.2015.2446205.

[3] U. U. Tariq et al., "Energy-Aware Successor Tree Consistent EDF Scheduling for PCTGs on MPSoCs," in IEEE Access, vol. 12, pp. 75761-75780, 2024, doi: 10.1109/ACCESS.2024.3403418.

[4] Kocot, B.; Czarnul, P.; Proficz, J. Energy-Aware Scheduling for High-Performance Computing Systems: A Survey. Energies 2023, 16, 890. https://doi.org/10.3390/en16020890

[5] R. Jeyarani, et al. 2012. Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence. Future Generation Computer Systems. Volume 28, Issue 5, May 2012, Pages 811-821

[6] Progyna Khondkar. 2018. Low-Power Design and Power-Aware Verification. DOI: https://doi.org/10.1007/978-3-319-66619-8

[7] H. Sohofi and Z. Navabi, "Assertion-based verification for system-level designs," Fifteenth International Symposium on Quality Electronic Design, Santa Clara, CA, USA, 2014, pp. 582-588, doi: 10.1109/ISQED.2014.6783379.

[8] Fakhruddin Muhammad Mahbub ul Islam. et al. 2018. Task aware hybrid DVFS for multi-core real-time systems using machine learning. Information Sciences. Volumes 433–434, April 2018, Pages 315-332.

[9] Guerout, T.; Monteil, T.; Da Costa, G.; Calheiros, R.; Buyya, R.; Alexandru, M. Energy-aware simulation with DVFS. Simul. Model. Pract. Theory 2013, 39, 76–91.

[10] https://www.fernuni-hagen.de/pv/docs/lagtime-micpro-subm-v2.pdf

[11] https://dvcon-proceedings.org/wp-content/uploads/power-aware-verification-strategy-for-socs.pdf

[12] https://www.edn.com/reuse-uvm-rtl-verification-tests-for-gate-level-simulation/

[13] https://arxiv.org/pdf/1902.08517

[14] https://semiengineering.com/formal-low-power-verification-of-power-aware-designs/

[15] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10011161

[16] https://dvcon-proceedings.org/wp-content/uploads/90794.pdf

[17] https://uobdv.github.io/Design-Verification/Lectures/Current/9_ABV_narrated_v9-part-1-ink.pdf

[18] https://www.cs.rice.edu/CS/Verification/Theses/Archive/dtabakov_dissertation2010.pdf

[19] https://www.cl.cam.ac.uk/~djg11/speedo/power-planning-Mischkalla.pdf

[20] https://arxiv.org/abs/1511.01354

[21] https://dvcon-proceedings.org/wp-content/uploads/power-management-verification-for-soc-ics.pdf

Citation: Prathiba, Kaushik Velapa Reddy. (2025). UVM-based power-aware verification closure using dynamic voltage and frequency scaling (DVFS) models. International Journal of Advanced Research in Engineering and Technology (IJARET), 16(3), 111–133.https://doi.org/10.34218/IJARET_16_03_007.

Abstract Link: https://iaeme.com/Home/article_id/IJARET_16_03_007

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJARET/VOLUME_16_ISSUE_3/IJARET_16_03_007.pdf

✉ **editor@iaeme.com**