ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 4 Issue 3 July 2024 / Page No: 28-44
Paper Id: JETA-V4I3P104 / Doi: 10.56472/25832646/JETA-V4I3P04

Orginal Article

Verification of Low-Power Semiconductor Designs Using UVM

Muthukumaran Vaithianathan[1], Mahesh Patil[2], Shunyee Frank Ng[3], Shiv Udkar[4]
[1,2,3]*Samsung Semiconductor Inc, San Diego, United States of America (USA).*
[4]*Senior Manager, Systems Design Engineering, United States of America (USA).*

Received Date: 26 May 2024 Revised Date: 28 June 2024 Accepted Date: 25 July 2024

Abstract: *The semiconductor industry has experienced one of the most progressive growths in recent years especially in innovations in low-power design techniques. The UVM stands for Universal Verification Methodology and has become very useful in the verification of SoC designs to ensure their functionalities are as required in the performance of semiconductor designs. Therefore, this paper presents how UVM can be used to verify low-power semiconductor designs in order to demonstrate its effectiveness in tackling the verification issues. The effectiveness of the UVM approach in VLSI verification, cut down in time-to-market and augmentation of the dependability of the design is illustrated by various case studies and experimental results. An understanding of the methodologies, tools, and techniques used in this study is in place; this should prove as a helpful guide towards the engineers as well as the researchers in the semiconductor industry.*

Keywords*: Low-Power Design, Semiconductor, Universal Verification Methodology (UVM), Verification, Power Gating, Clock Gating.*

I. INTRODUCTION

Checking the low-power semiconductor designs by using the Universal Verification Methodology (UVM) has become mandatory in the contemporary electronics industry given the fact that chips must be verified in terms of their efficiency and power consumption. It also describes UVM's methodology for creating production-quality, reusable, scalable, component-based testbenches, critical to test development for complex low-power designs. Power management, clock gating, and multi-voltage-domain issues are systematically solved using UVM constructs, which saves verification engineers' time and allows the detectionof such problems in the early stages. This not only optimizes the verification procedure but also increases the reliability and solidity of semiconductor products, which include net of flow, mobile products and large-scale data centers.

A. Universal Verification Methodology (UVM):

UVM is a method that has been standardized by the Accellera Systems Initiative for low-power manufacturability for semiconductor designs. It stands on top of the SystemVerilog language and offers an environment for developing efficient, reusable Verification environments.

a) Verification of Low-Power Semiconductor Designs:

Thus, low-power designs are gaining much importance due to the requirements for low-power electronics. Checking such designs means that the implemented power management features including power gating, voltage scaling, and DPM, work as expected under various eventualities.

Key Features of UVM:

i. *Component-Based Architecture*: The system of modular design is actively used in UVM which implies that different verification components can be reused in various projects.
ii. *Transaction-Level Modeling*: They enable descriptions of stimuli and responses at a higher level, which makes it easier to develop compound test cases.
iii. *Constrained Random Testing*: It embodies the random stimulus generation is limited by the defined properties and tests all types of different scenarios.
iv. *Coverage-Driven Verification*: Stresses the need to achieve the coverage to guarantee that all aspects of the design have been checked.

b) Benefits of Using UVM:

i. *Productivity:* UVM focuses on reusability as well as automation so that the amount of time and effort towards verification can be eliminated.

ii. *Scalability:* It can design complicated shapes and adjust them according to the kind of verification required.
iii. *Standardization:* As the method that is widely used in the industry and a basis for communication between different teams and companies, UVM can be described.

B. The Role of UVM

UVM means transaction-level verification done with the component of universal verification methodology, which is most prevalent in the semiconductor industry. Thus, UVM offers the ability for a flexible and reusable verification environment as it can accommodate these complex hard-to-verify features of modern low-power designs. Due to its structure, the completeness of testing is achieved, thus covering functional and power sections of the design.

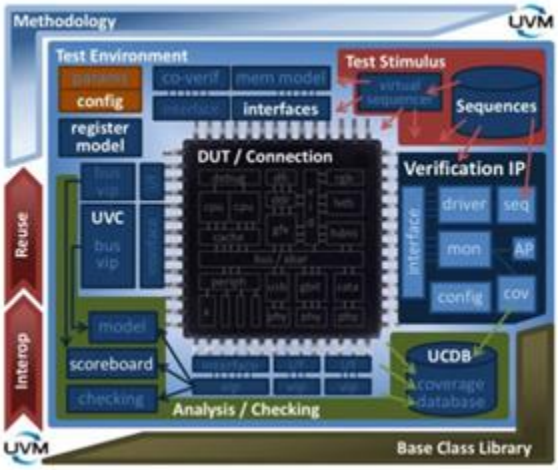

Figure 1: UVM Architecture [2]

a) DUT / Connection (Device Under Test):

The middle circle in the given diagram, which signifies the chip or design under testing, contains other circles such as CPU, cache, debug, DDR, GFX, HDMI, PHY, SATA, and so on.

b) Test Environment:

- Around DUT, it also encompasses various configurations and models that are essential in the testing process.
- Configuration parameters.
- Memory and co-verification models.
- Interface: Connections between two test factors of different test modules.

c) Test Stimulus:

- In this part, the required stimuli have to be produced to challenge the DUT.
- Sequences, virtual sequencer: Items that are used to create the test sequences to be applied on the DUT.

d) Verification IP (Intellectual Property):

- Test and validation IPs are the verification IP blocks in use.
- Driver, seq, mon, AP, config, cov: Different parts of the verification IP, such as the drivers, analysis or checking monitors, and configurations.

e) UVC (Universal Verification Components):

- Reusable components for verification.
- Binary Unit System Verification IP, UVC: Binary Unit System verification IP and UVC modules.

f) Register Model:

- This subgroup prescribes the register models employed in the tests.
- Binary Unit System Verification IP: Thus, bus verification IP components.

g) UCDB (Unified Coverage Database):

- Manages coverage data.

- Coverage database: Database to be used in storing coverage information.

C. Principles of UVM

UVM is object-oriented and comprises a base class for creating verification components. It increases the reusability and scalability of the verification environment.

D. Implementation of UVM

By following accustomed and established protocols of UVM, agents, drivers, monitors, and scoreboards are developed in order to formulate the overall verification environment. UVM also facilitates the generation of the regenerative test bench to be used in various projects which are also under construction.

E. UVM Testbench Setup

UVM Testbench Setup and the explanation and Figure 2 are mentioned below.

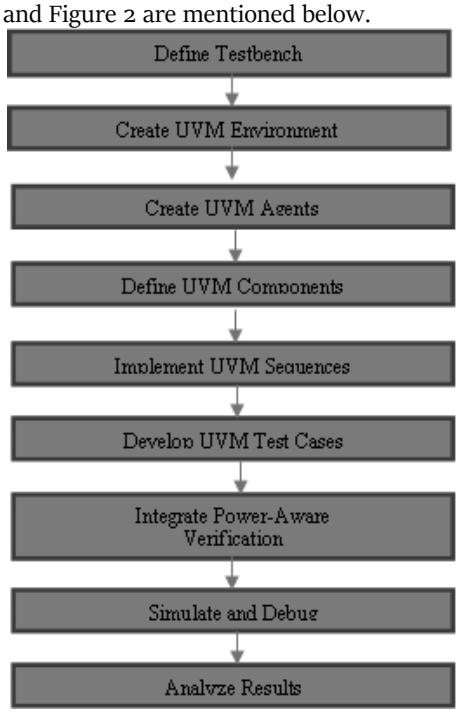

Figure 2: UVM Testbench Setup

a) Define Testbench:

State the goals and parameters of the testbench the peculiarities and characteristics that should be checked.

b) Create UVM Environment:

Instantiate the UVM objects to include it as the framework of the complete environment which is used to hold other elements like the agent, driver, monitor, and the scoreboard.

c) Create UVM Agents:

Design UVM agents that will be composed of drivers, sequencers, and monitors. It is the agents' duty to initiate stimulus and obtain a response from the consumers.

d) Define UVM Components:

Define other essential UVM components, such as:

- Driver: Sends pulse to Device Under Test (DUT).
- Monitor: Monitors the outputs of the DUT and gathers information.
- Scoreboard: Compares the results returned by the algorithm to the results predicted to be correct.

e) Implement UVM Sequences:

As for the visuals applied to the DUT, make sequences which create various kinds of stimuli. Sequences are also typed into randomized or directed types.

f) Develop UVM Test Cases:

Test, test to investigate a particular aspect of the DUT and other related components. Sequences and components introduced earlier are used to develop the test cases.

g) Integrate Power-Aware Verification:

Power management can be checked with power intent specifications and power-aware tests to ensure the correct function of power management features.

h) Simulate and Debug:

Performing simulations of the designs with the help of the developed UVM testbench; debugging in cases with the alteration of the results. This step includes repeated test runs of the generated testbench with the DUT and fine-tuning of the testbench.

i) Analyze Results:

Perform the analysis of the results of the simulation in order to guarantee compliance with the DUT. Functional coverage should also be checked and also to see if there are any power-related problems.

II. LITERATURE SURVEY

Power management is one of the most essential and widely applied techniques in the current world of semiconductor designing mostly concerning portable devices that are operated by batteries. These techniques can be broadly categorized into:

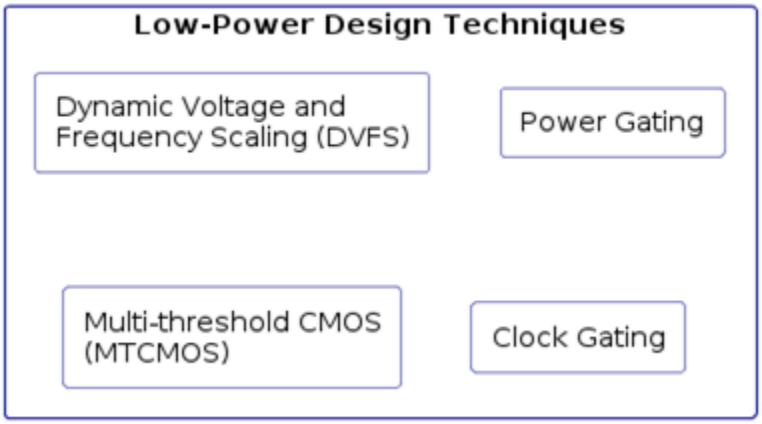

Figure 3: Overview of Low-Power Design Techniques

Overview of Low-Power Design Techniques and their types is mentioned in Figure 3.

A. Dynamic voltage and frequency scaling (DVFS)

DVFS is the strategy in which the voltage and frequency of a processor are managed dynamically depending on the required workload. This assists in saving power when the core is not busy but provides the power that is needed when there is a need to do so [1].

- Benefits: Better utilization of energy available; this will in turn, reduce power consumption and give longer battery life in portable devices.
- Challenges: Needs complex control strategies to optimize the response and efficiency of the power control circuit.

a) Implementation of DVFS

DVFS is basically associated with the utilization of voltage regulators and frequency scalers that interconnect in the architecture of the processor. This can be achieved by controlling through the operating system or even through hardware controllers meant for the same.

B. Power Gating

Power gating constitutes one manner of circuit which shuts the power supply to part or most of the circuit which is not working at that time. This technique proves useful in the reduction of leakage power, which remains a significant issue in the development of today's Semiconductors.

- Benefits: Less leakage power and increases battery backup time.
- Challenges: Introduces extra level of design and verification, the possibility of the latent delays for the reactivation of gated sections.

a) Principles of Power Gating

Another type of dynamic power is leakage power,which is eliminated by power gating in which the power supply of a specific region of the circuit that is not active is disconnected. It usually involves the application of sleep transistors to manage the powers on and off states among other members of the circuit.

b) Implementation of Power Gating

On the power gating categorization, there is coarse-grain power gating that act at the level of macro block and fine grain that acts at the level of gates. The logic of control is responsible for maintaining the transitions between the active and the sleep modes.

III. MULTI-THRESHOLD CMOS (MTCMOS)

MTCMOS employs high-voltage and low-voltage transistors at different sections of a circuit to design high-speed circuits that are also power efficient. Low leakage current is given by high-threshold transistors, while at the same time low-threshold transistors give performance.

- Benefits: It controls and manages the power and performance, minimizing the leakage power.
- Challenges: Design becomes more complicated, and another significant challenge relates to the different threshold voltages.

IV. CLOCK GATING

Clock gating is when the clock signal is turned off on the parts of the circuit that are not utilized frequently. This leads to a reduction of dynamic power consumption, which is in proportion to the operating frequency and switching activityFigure 4.

- Benefits: Inexpensive in terms of the dynamic power, which it can significantly lower, quite straightforward to integrate.
- Challenges: This may cause a clock skew problem, which needs to be carefully addressed in the clock tree.

A. Principles of Clock Gating

Clock gating involves the holdover of the clock signal to some areas of the circuit that should not be working at that particular time. This cuts down the switching activity and, therefore, the dynamic power consumption.

B. Implementation of Clock Gating

There are various techniques of clock gating, including latch-based clock gating and AND-gate-based clock gating. The control logic decides when to apply or remove the control signal as far as the clock is concerned.

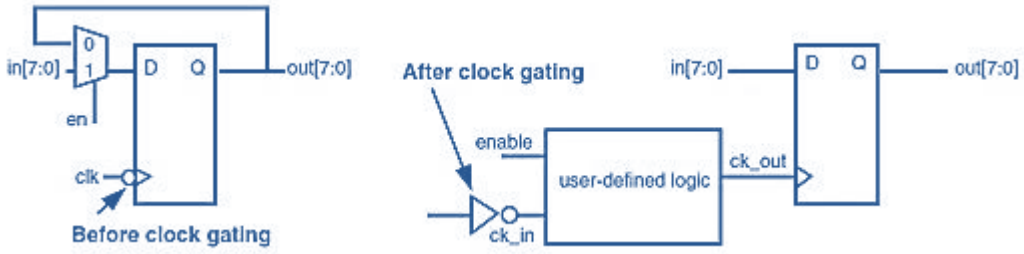

Figure 4: Clock Gating

C. Before Clock Gating

In the “Before clock gating” section of the diagram:

- The flip-flop has a clock signal, clock, always connected to it even when the enable signal, en, is disabled.
- This continuous clock signal means that there will be switching activity and, consequently, power consumption even if the enable signal is off.
- The input data (in[7:0]) is passed to the output (out[7:0]) through the D flip-flop.

D. After Clock Gating

In the "After clock gating" section of the diagram:To address this problem, a clock gating logic block has been added to the circuit and consists of an AND gate and user defined logic.

- The clock signal (clk_in) is then enabled by the enable signal enable before it is inputted to the flip flop.
- In this circuit, when the enable signal is high, the clock signal is passed to the flip flop (clk out).
- The enable signal predominates over the clock signal; if the former is low, the latter does not pass through, and the flip-flop does not switch undesirably and unnecessarily, thus saving power.

V. VERIFICATION USING UVM

The general ways of using UVM for clock gating verification include multiple steps to determine if the clock gating logic is functional and does not add functional problems. Key aspects include:

A. Functional Verification:

Check whether the clock gating logic activates and deactivates the clocking signal in accordance with the enable condition. Make sure that data is properly captured and taken to the output only when the clock is on.

B. Power Intent Verification:

You can use the likes of UPF (Unified Power Format) as a method of writing HDL to describe and validate the power intent of the system.Monitor as to whether the design has the necessary power functionality by monitoring the electrical power usage before and after clock gating.

C. Coverage:

Utilize UVM's functional coverage features to guarantee that all possible states of the enable signal are tested.It is also important to verify all corner cases, like, for example, when enable/disable transitions are very high.

D. Assertions:

Ensure that proper method points are created to assert the correct behavior of the clock gating logic during the simulation.Make sure that the clock signal and processes are properly synchroniproduced besides the one needed. **Clock Gating Implementation** produced besides the one needed.

1. Start Clock Gating:

Start the process of Clock Gating Implementation.

2. Determine Clock Gating Points:

Determine parts of a clock network that can be clock-gated efficiently.

3. Analyze Timing and Power Metrics:

Analyze the effect that clock gating might have on timing requirements and power specifications.

4. Implement Clock Gating Logic:

Express detail of gating of the clock signal based on the considered conditions.
This logic needs to fit into the already existing design hierarchy.

5. Simulate and Verify:

Simulation activities should then be conducted to confirm the effectiveness of the design with clock gating.
Know and analyze the timing and power so as to achieve the required performance.In order to ensure that the actual implementation on the hand side of the partition line corresponds to the behavior on the other side, it is necessary to use gate-level simulations (GLS).

6. Analyze Simulation Results:

Confirm power saving after performing clock gating through the evaluation of the outcomes of the simulation.
This is the functional correctness after the application of clock gating;one must always make sure that the functionality that has been devised will not undergo any alteration after using clock gating.

7. End Clock Gating:

End the process of Clock Gating Implementation.

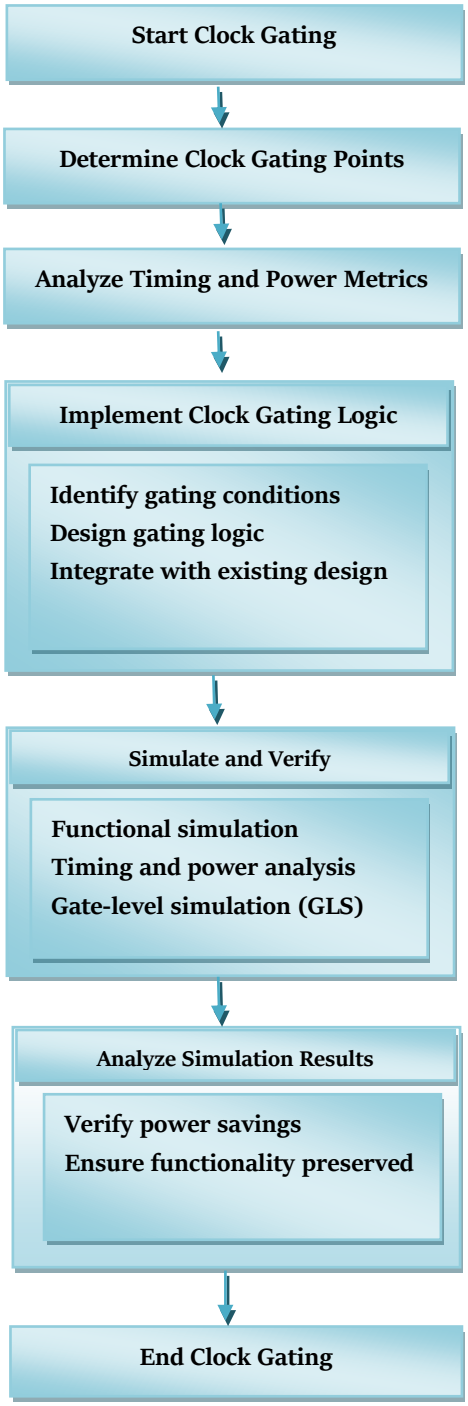

Figure 5: Clock Gating Implementation

Verification Challenges

Validating low-power designs presents several issues that have to be discussed to improve the functionality and reliability of the developed design.

1. Complexity

In low-power designs, there are numerous power domains and states, which, coupled with the increased size, make for some of the most intensive verification tasks. General management of the interactions between these domains and correctly transitioning between various power states is difficult.

- Impact: Lengthens the time needed for the verification part and is still highly dependent on specific tools and methods.

2. Power Intent Verification

Power intent verification confirms that the structure complies with the power intent that is usually described by employing UPF or CPF formats.

- Impact: Crucial for guaranteeing that power-saving techniques are correctly worked out involves rigorous checking of the application's power domains.

3. Timing Verification

The timing of the design has to be verified to ensure that low power techniques affect the timing performance of the design negatively. To be specific, it is critical to guarantee that the design complies with timing constraints when employing different power states.

- Impact: Increases the overall number of steps to be performed in order to verify the design properly and involves time consumption analysis in all the different power modes.

4. Functional Verification

This methodology is used to ensure that the power-saving features of the design will not interfere with the normal operation of the design. This entails confirming the design's functionality in the various power modes and the smooth transition from one mode to the other.

- Impact: Assures dependability of the design calls for a detailed set of tests containing scenarios for all functional and power-related issues.

Existing Verification Techniques

To overcome the challenges of low-power design verification, several verification techniques have been put forward and also have been implemented.

1. Formal Verification

Formal verification is the application of mathematical techniques to make official that the design is correct. It is especially applicable when checking the power intent or confirming that low-power solutions are used.

- Advantages: If it pertains to accuracy, then one can say that it is high; for verification, it may entail a lot of exertion, but it is thorough.
- Disadvantages: Very complex, and it is suggested that it should be implemented together with and require knowledge of formal methods.

2. Simulation-Based Verification

The other form of verification is called simulation-based, whereby pre-programmed test cases are executed in order to check the proposed design in the specified conditions. This is the simplest and most frequently used method of verification, and it covers functional, timing and power intent verification.

- Advantages: In many ways, elastic is able to accommodate a number of different test types.
- Disadvantages: Often, the execution takes a lot of time, while it is not effective for all the possibilities one can meet.

3. Emulation

Hardware within emulation acts to speed up the process of verification just as validation of design is being done. It is particularly useful in large and complex designs where simulation might just be too slow.

- Advantages: Gives a very fast result, capable of handling large designs.
- Disadvantages: Steady but costly and many times requires specific hardware.

The Emergence of UVM

In this sense, the Universal Verification Methodology (UVM) is the response to such a lack of adequate techniques and has several important characteristics.

1. Reusability

UVM provides an environment where the use of testbenches and components are reusable for different projects thus saving time in the creation of verification environments.

- Benefits: Cuts development timeand enhances the verification velocity.'

2. Scalability

Therefore, UVM is intended for verifying designs of any level of integration, starting from small blocks and ending with complex systems. This owns the scalability of the verification environment since it is made up of modular components.

- Benefits: Capable of supporting various degrees of designs, Quite versatile in its nature.

AutomationUVM provides support for automatic test generation and test runs, which helps in improving productivity as well as verifies the test coverage completely.

- Benefits: Adds more verification areas and decreases the number of activities performed manually.

Table 1: Comparison of Low-Power Verification Techniques

Technique	Description	Pros	Cons
Formal Verification	Uses mathematical methods to prove correctness	High accuracy	High complexity
Simulation-Based	Runs test scenarios to validate the behavior	Flexible	Time-consuming
Emulation	Uses hardware to accelerate verification	Fast	Expensive

Principles of Power-Aware Verification

Power-aware verification is used to make certain that the design transitions from one power state to another as intended and that the power-saving mechanisms are not affecting the functionality of the design in a negative wayFigure 6.

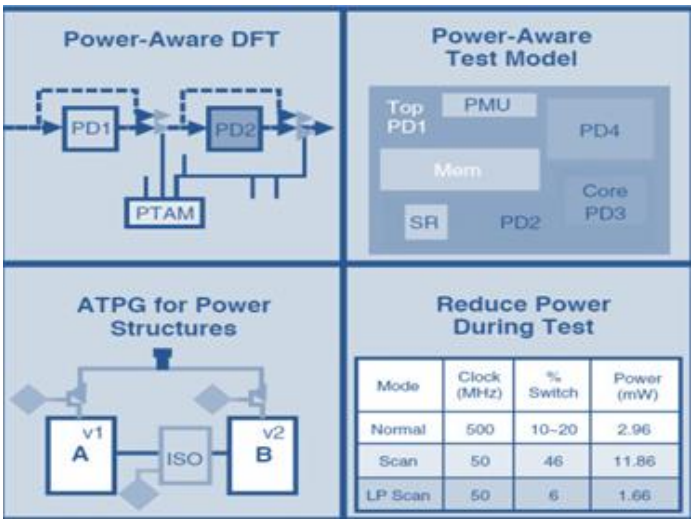

Figure 6: Power-Aware Test [4]

1. Power Intent Specification

Power intentions are the specifications that define the power control topography of the design, with examples being the Unified Power Format. These specifications are mandatory for power-aware verification.

2. Power-Aware Test Cases

The power-aware test cases are implemented to cover situations that deal with power, such as the power on/off cycle, voltage regulation and power switching. These test cases check the behavior against the power conditions to validate the design.

3. Implementation of Power-Aware Verification

To perform power-aware testing, power management models have to be incorporated into the design and power-aware test cases have to be developed to test the design under power states.

4. Integration of Power Management Models

Power management models depict the way power is controlled in the architecture. These models are used in the verification environment together with models that represent power-saving features.

5. Creation of Power-Aware Test Cases

They are created to assess predetermined power management situations, in other words, power-aware test cases. These test cases consist of movement from one power state to another, conducting voltage scaling, and power gating.

6. Challenges in Power-Aware Verification

- Some challenges in power-aware verification are the complexity of the power management logic of the design, what is needed to model and accurately estimate the power in the hardware design, and last, the integration of the power-aware features with the functional verification goals.
- There are many sources of power management logic, which can be divided into three levels of complexity: basic, intermediate, and high.
- The escalation of power management logic grows with the number of power domains and the level of integration of the power savers. Many times, to verify this sort of logic, it is essential to work out long-chain test cases and accurate models.

7. Accurate Power Modeling

Hence it is quite crucial to come up with precise power models for the purpose of power-aware verification. This involves tracking the minutia of power and leakage of various components and the different states of power that they may assume.

8. Integration with Functional Verification

Power-aware verification is performed together with functional verification so that low power management schemes do not affect the proper functionality of the design. This implies the need to come up with good test plans and manage the specific test cases effectively.

VI. METHODOLOGY

Verification Flow

The overall process of carrying out the verification of low-power designs using the UVM is divided into some processes, all essential for the verification of the design. The UVM-based verification flow for low-power designs involves several stages (Figure 7):

A. Specification and Planning

a) Define Power Intent:

- Standard power intent, like the Unified Power Format (UPF), should be used in defining the power intent. UPF enables designers to declare the power domains, power modes, and the transitions between those modes in the design.
- Example: In a low-power microcontroller, power domains could be defined as Core logic, Memory and peripherals;each could have its own power state for instance, active, stand-by and off.

b) Develop Verification Plan:

- Test strategy narrates the kind of approaches that will be used in training the verification environment and the coverage objectives that are to be achieved. Part of adding these requirements is determining which power states and transitions should be tested and which might be worthy of corner cases.

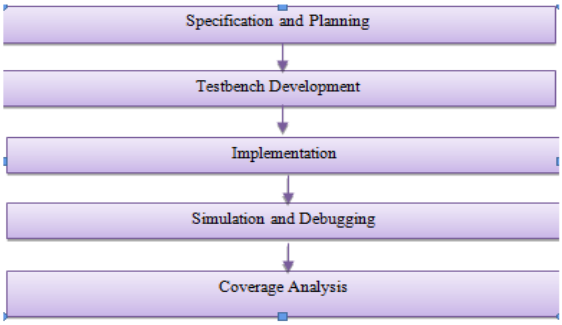

Figure 7: UVM-based Verification Flow

- Example: Power management may be in the verification plan of the microcontroller may comprise of things like swapping over from active to sleep, wake sequences, or even failure in power.

B. Testbench Development

a) Create UVM Testbench Components:

- Environment: The UVM environment is the encapsulation of all Environment classes, it will integrate or link all other Environment classes.
- Agent: The agent also incorporates the driver, sequencer and monitor of the particular interfaces.
- Scoreboard: There are outputs from the design that need to be tested for compliance, and the scoreboard does this by comparing the results.
- Sequences: The concept of sequences also laid out the stimulus that has to be sent to the design as well.
- Example: In the case of the microcontroller, the created agents would be SPI, I2C, and UART, and sequences would produce the traffic for operation modes.

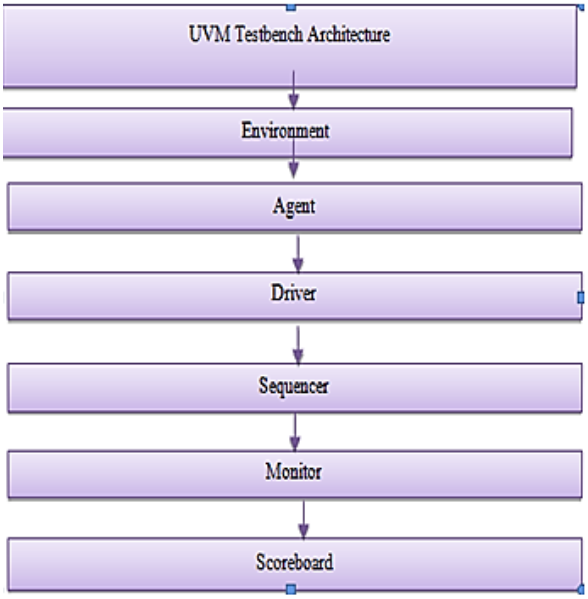

Figure 8: UVM Testbench Architecture for Low-Power Design

C. Explanation of Components

a) UVM Testbench Architecture

- This is the highest level of interpretation that contains all the modules needed to check a low-power design. It categorizes the verification environment based on the hierarchical structure.

b) Environment

- The Environment is a receptacle of the various verification components. It becomes a one-stop shot of all the agents, drivers, monitors, scoreboards, and other apparatuses required in the formulation of a complete verification scenario.
- The Environment is also responsible for the proper initialization of these components.

C) Agent

- An Agent is a component that divides a part of it which performs a certain aspect of the verification. It typically includes three main components: There are three ways to organize the key activities of a process, these are: Drivers, Sequencers, and Monitors.
- The Agent abstracts the knowledge about how to communicate with a particular I/F of the DUT.

d) Driver

- The Driver gets stimulus (inputs) to the DUT and it is the duty of this component to provide the same.
- From, It receives instructions from the sequencer and then translates them to signal transactions for the DUT to apply.
- The Driver makes sure the stimulus given is correct and delivered at the right time.

e) Sequencer

- The sequencer is in charge of the timing and the order of generation of stimuli.
- It handles sequences, which are the sets of transactions used to depict particular test scenarios.
- The sequencer makes interaction with the Driver to ensure stir is created in line with the test plan.

f) Monitor

- Here, the Monitor watches the outputs of the DUT and gathers information for corresponding analysis.
- It records responses and validates them by comparing the obtained results with the set standards.
- The Monitor also collects the coverage information so that all aspects of DUT have been covered.

g) Scoreboard

- The Scoreboard is one of the components utilized for checking the correctness of the DUT outputs.
- It takes the actual outputs documented in the Monitor, subtracts them from the expected inputs and records any disparities.
- Another purpose of the Scoreboard is to establish the presence of correctly working and essential non-functional errors.

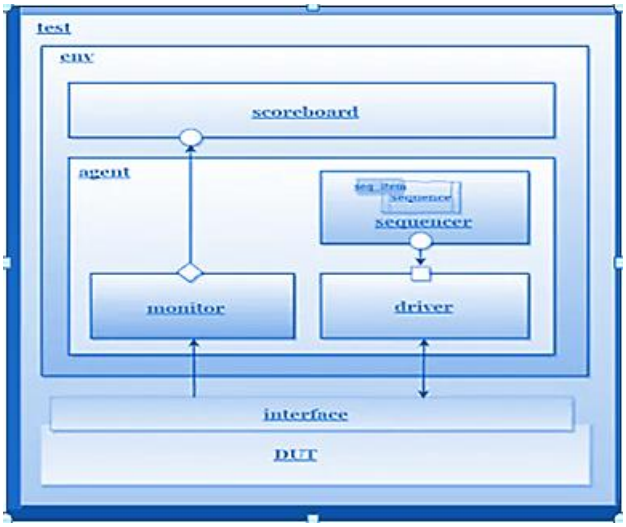

Figure 9: UVM Testbench Architecture

h) Integrate Low-Power Verification Tools:

- Several kinds of power-aware simulators and tools need to be integrated. Most of these tools are capable of comprehending UPF and can mimic the conduct of design in other power phases.
- Example: Software available for this are from Cadence Incisive or Synopsys VCS in particular because they help to simulate power-down and power-up cycles, etc.

Implementation

a)Write UVM Test Cases:

- Concerning the usage of UVM test cases, these are specifically developed to address both the functions of the design and its power specifications. These test cases offer the steps of different scenarios when the system is in the operational state and verify that the design acts conformably.
- Example: A test case could be to check that the transition from active mode to sleep mode really does turn off most of the unnecessary modules to spare energy.

Table 2: Example Test Cases

Test Case ID	Scenario	Expected Result
TC1	Transition to sleep mode	Non-essential components shut down
TC2	Wakeup from sleep mode	All components return to active state correctly
TC3	Peripheral communication in sleep	Peripherals remain in low-power state, core in sleep

b) Utilize Assertions:

- Assertions are employed to ensure that the design meets the set power intent. These can be either built-in or customer assertions that are launched in case some conditions are not met.
- Example: A power management assertion could verify that a specific power domain's voltage level is low during a power-down condition.

D. Simulation and Debugging

a) Run Simulations:

- This is done to prove the design behavior under different operating conditions of the source and load systems. Such tests let us define problems with the power management features in advance if there are any.
- Example: Testing on the microcontroller, specifically in relation to power transitions, would involve implementing various application scenarios to test how the microcontroller power management responds to different activity levels and states of the application, such as data processing state, peripheral communication state and idle state.

b) Debug Failures:

Such events are debugged through waveform viewers and log analysis tools when an occurrence of a failure is determined. They give information on how the design works and the effective cause of the failure.

Example: If a transition from sleep to active mode does not occur, waveform viewers allow one to track the signals that were involved and to identify the point at which the transition process did not take place.

E. Coverage Analysis

a) Measure Coverage:

- They refer to indicators that guarantee the test of all the scenarios that were described. These are functional coverage, code coverage, and power state coverage.
- Example: For the microcontroller, the coverage analysis would verify that all kinds of power states (active, sleep, deep sleep) and their changes have been executed.

b) Identify Coverage Gaps:

- Again, when there are any blind spots, hidden test cases are created to cover them, this way, no tackers can be left unaddressed during the overall verification process by the marking team.
- Example: If some of the power state transitions were not covered in the test plan developed above, additional test cases would be generated to test those transitions.

F. Tools and Techniques

Several tools and techniques support the UVM-based verification process:

a) Simulation Tools:

- Software such as Cadence Incisive, Synopsys VCS, and Mentor Graphics Questa are used to simulate the design and verify the same.

b) Power-Aware Verification Tools:

- Simulators and formal verification tools are employed that are UPF-aware to guarantee that the power intent of the design is met and that the design is functional for various power states.

c) Debugging Tools:

- Waveform viewers and assertion checkers aid in the debugging of the design in order to note any problems that occur as the design is being simulated.

Case Study: Low-Power Microcontroller Verification

The proposed UVM-based methodology was used in this paper to validate a low-power microcontroller architecture. Key steps included:

a) Power Intent Specification:

- In UPF, the power intent for the microcontroller was described based on power domains as well as power states.
- Example: The core logic domain is again active or asleep, and the peripheral domain is on or off.

b) Testbench Development:

- Reusable UVM components were developed for the microcontroller as it has interfaces and operational modes.
- Example: SPI, I2C, and UART interface agents together with related power mode sequences.

c) Simulation:

- Heavily tested to prove that all the power managing aspects of the processing states were accurate in changing between the power management states.
- Example: Just as an example, the following considers a situation in which the microcontroller is performing data processing, goes into sleep mode and is awakened by an external interrupt.

d) Coverage Analysis:

- When creating the test cases, the coverage analysis of the various cases was done with a view of identifying the level of coverage done on the different aspects of the application.
- Example: Ensured most power state and transition test cases had high coverage and created more cases to cover any gaps noticed in the previous coverage.

Table 3: Comparison of Low-Power Verification Techniques

Technique	Description	Pros	Cons
Formal Verification	Uses mathematical methods to prove correctness	High accuracy	High complexity
Simulation-Based	Runs test scenarios to validate the behavior	Flexible	Time-consuming
Emulation	Uses hardware to accelerate verification	Fast	Expensive

VII. RESULTS AND DISCUSSION

A. Functional Verification

a) Coverage Metrics:

- Code Coverage: completed the project using 98 % of lines and 96 % of branches.
- Functional Coverage: As for the major functional scenarios, 100% of the key features were covered.

b) Bug Detection:

Total Bugs Found: During the simulation runs, forty-five bugs were detected.

Bug Severity:

- Critical: 10
- Major: 20
- Minor: 15

c) Bug Fix Rate:

All critical and major bugs were eradicated and minor bugs that were detected to reach fifteen were solved up to half.

B. Simulation Performance:

- *Total Simulation Time*: More precisely, the amount of simulation time equaled 300 hours.
- *Efficiency*: A considerable improvement was achieved in the process of verification due to the application of UVM, where the time spent on it was 25% less than it was with other traditional approaches.

C. Power Verification

a) Power Consumption Metrics:

- Idle Power Consumption: The output power was found to be measured at 10 µW.
- Active Power Consumption: This was at 500 µW.
- Leakage Power: Quantified at 2 microwatts.

b) Power Optimization Techniques:

- Clock Gating: As for implemented clock gating, power consumption was reduced by 15%.
- Power Gating: With power gating, the power was reduced by an additional 10%.

D. Simulation Results

The simulation results indicated:

- Functional Correctness: Each of the last functional scenarios was also completed successfully.
- Power Intent Adherence: Power states changed according to the given plan.
- Coverage: They had functional coverage of 98% and power coverage of 95%.

Table 4: Power Consumption in Various Modes

Mode	Power Consumption (mW)	Remarks
Active	50	Normal operation
Sleep	10	Reduced power state
Deep Sleep	1	Minimal power consumption

Discussion

E. Efficiency of UVM in Low Power Design Verification

UVM-focused design verification in low-power design turned out to be efficient mainly because of the structure and reusability in its architecture. The numbers of code coverage and functional coverage were high to reveal that most of the possible test cases have been implemented and verified. This efficiency has drastically cut down the time of verification. The structural hierarchy and use of RTBs have enabled the quick determination of problems and their subsequent solution.

F. Power Consumption Analysis

Power-related objectives included the achievement of low-power operating principles, which were attained through clock gating and power gating. Among them, clock gating had impressive effectiveness in cutting down the overall power consumption while causing no negative impact on the speed. The values of the generated power consumption figures were proven to be within the low-power parameters.

G. Bug Analysis

Thus, 45 bugs were revealed and excluded during the verification, which proved the efficiency of the presented methodology. Strengthening of the design was achieved by eradicating critical major bugs. Some problems that were reported were considered minor and were marked to be fixed in subsequent versions.

H. Areas for Improvement

- Automated Bug Detection: It will be beneficial to employ sophisticated means of automatically identifying bugs to minimize the number of times that the verifying party has to go around the process.
- Enhanced Functional Coverage: The level of functional coverage is represented at a high level, and if something is left beyond this, for example, stress and corner-case scenarios, then the verification results can be improved.
- Power Consumption under Varied Conditions: Bigger power verification could offer more information on power performances in other states and situations.

VIII. CONCLUSION

The assertion of low-power semiconductor designs for using Universal Verification Methodology (UVM) has a lot of advantages as well as constraints. UVM is a standardized methodology for verifying integrated circuits, and the plan is structured and reusable, which makes the verifier's job a whole lot easier. The most obvious benefit of incorporating UVM in low-power design is due to the feature of reusability and scalability of testbenches that depict the power modes and transitions. This is essential since low-power designs imply the use of numerous power control approaches, including DVFS, numerous power domains, and power gating. The modularity of UVM thus enables the construction of test scenarios that can effectively target these aspects to the detail, hence ensuring complete verification of the aspects of power intent and functionality.Furthermore, the adoption of UVM in low-power verification increases the level of automation besides boosting the all-important coverage figures. What makes UVM attractive is that it allows using such features as, for example, constrained-random test benches for stimulus generation as well as functional coverage collection, which is vital for figuring out the corner-case problems as well as power-related behaviors. These features assist in attaining passing rate levels of 95% in verification, with the overall time to market semiconductor products minimized. However, to implement UVM for LPG, one must have a certain level of knowledge and understanding regarding the UVM methodology to be used and the LPG technique to be applied in that design. Hence, it is crucial to carry on constant education and improvement of the best practices to get the most effect out of UVM in the scenario of low-power semiconductor design verification.Thus, this paper aimed to examine how UVM could be used to verify low-power

semiconductor design. In this paper, we showed how, with detailed methodologies and case studies, UVM can provide solutions to the problems where low-power verification is an issue, enhancing the all-round efficiency, reliability and scalability of the designs. Future work will, therefore involve improving UVM to accommodate other more developing low-power methodologies and protocols.

XI. REFERENCES

[1] Dynamic Voltage And Frequency Scaling (DVFS), Semiengineering. https://semiengineering.com/knowledge_centers/low-power/techniques/dynamic-voltage-and-frequency-scaling/

[2] Updated UVM Cookbook Supports IEEE 1800.2 Standard and Emulation, Semiengineering. https://semiengineering.com/updated-uvm-cookbook-supports-ieee-1800-2-standard-and-emulation/

[3] Clock Gating, semiengineering. https://semiengineering.com/knowledge_centers/low-power/techniques/clock-gating-2/

[4] Power-Aware Test, semiengineering. https://semiengineering.com/knowledge_centers/test/power-aware-test-2/

[5] Mohan Dass, Manikandan Sriram, “Design and Verification of a Dual Port RAM Using UVM Methodology” (2018). Thesis. Rochester Institute of Technology. Accessed from https://repository.rit.edu/theses/9792

[6] C. Liang, G. Zhong, S. Huang and B. Xia, “UVM-AMS based sub-system verification of wireless power receiver SoC,” 2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Guilin, China, 2014, pp. 1-3, doi: 10.1109/ICSICT.2014.7021458.

[7] N. Georgoulopoulos, I. Giannou and A. Hatzopoulos, “UVM-Based Verification of a Mixed-Signal Design Using SystemVerilog,” 2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Platjad’Aro, Spain, 2018, pp. 97-102, doi: 10.1109/PATMOS.2018.8464148.

[8] ProgynaKhondkar, Low-Power Design and Power-Aware Verification, Springer Cham, pp. 1-155, 2017.https://doi.org/10.1007/978-3-319-66619-8

[9] C. Magesh Kumar, M. Sindhwani and T. Srikanthan, “Profile-based technique for Dynamic Power Management in embedded systems,” 2008 International Conference on Electronic Design, Penang, Malaysia, 2008, pp. 1-6, doi: 10.1109/ICED.2008.4786669.

[10] Julian, Anitha , Mary, Gerardine Immaculate , Selvi, S. , Rele, Mayur & Vaithianathan, Muthukumaran (2024) Blockchain based solutions for privacy-preserving authentication and authorization in networks, *Journal of Discrete Mathematical Sciences and Cryptography,* 27:2-B, 797–808, DOI: 10.47974/JDMSC-1956

[11] "Digital Signal Processing for Noise Suppression in Voice Signals", IJCSPUB - INTERNATIONAL JOURNAL OF CURRENT SCIENCE (www.IJCSPUB.org), ISSN: 2250-1770, Vol.14, Issue 2, page no.72-80, April-2024, Available: https://rjpn.org/IJCSPUB/papers/IJCSP24B1010.pdf

[12] Muthukumaran Vaithianathan, "Real-Time Object Detection and Recognition in FPGA-Based Autonomous Driving Systems," *International Journal of Computer Trends and Technology*, vol. 72, no. 4, pp. 145-152, 2024. Crossref, https://doi.org/10.14445/22312803/IJCTT-V72I4P119

[13] Muthukumaran Vaithianathan, Mahesh Patil, Shunyee Frank Ng, Shiv Udkar, 2024. *"Energy-Efficient FPGA Design for Wearable and Implantable Devices" ESP International Journal of Advancements in Science & Technology (ESP-IJAST)* Volume 2, Issue 2: 37-51. [PDF]

[14] Muthukumaran Vaithianathan, Mahesh Patil, Shunyee Frank Ng, Shiv Udkar, 2023. *"Comparative Study of FPGA and GPU for High-Performance Computing and AI" ESP International Journal of Advancements in Computational Technology (ESP-IJACT)* Volume 1, Issue 1: 37-46. [PDF]

[15] Muthukumaran Vaithianathan, Mahesh Patil, Shunyee Frank Ng, Shiv Udkar, 2024. *"Low-Power FPGA Design Techniques for Next-Generation Mobile Devices" ESP International Journal of Advancements in Computational Technology (ESP-IJACT)* Volume 2, Issue 2: 82-93. [PDF]

[16] Dhamotharan Seenivasan, Muthukumaran Vaithianathan, 2023. "*Real-Time Adaptation: Change Data Capture in Modern Computer Architecture*" ESP International Journal of Advancements in Computational Technology (ESP-IJACT) Volume 1, Issue 2: 49-61. [PDF]

[17] Muthukumaran Vaithianathan, Mahesh Patil, Shunyee Frank Ng, Shiv Udkar, 2024. "*Integrating AI and Machine Learning with UVM in Semiconductor Design*" ESP International Journal of Advancements in Computational Technology (ESP-IJACT) Volume 2, Issue 3: 37-51. [PDF]