Manickam Muthiah, International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 15, Issue 5, May 2025, pp 135-139

RESEARCH ARTICLE OPEN ACCESS

An Adaptive Simulation Termination Controller for a Test Environment

Manickam Muthiah*
**(Principal Engineer, ARM Inc., Chandler, Arizona, USA.)*

ABSTRACT
The Adaptive Simulation Termination Controller is a methodology-independent, standalone module for pre-silicon functional simulation/verification using object-oriented programming. It dynamically predicts when to end a simulation by analyzing packet-level latency statistics for each DUT data path, rather than relying on a fixed wait time. The controller maintains Active, Complete, and Inactive queues per DUT data path, along with a Latency Record that tracks expected, minimum, maximum path latencies, sample count, and timeout values per DUT data path. Timestamp Packets are inserted into the expected DUT output packets by the scoreboard to measure actual latencies; these measurements feed an adaptive algorithm that updates the expected latency. A Sim End Timer uses these predictions to raise and drop simulation objections automatically, minimizing idle cycles and manual timing adjustments. The approach improves efficiency across any industry-standard verification methodology (OVM, UVM, etc.) and can issue fatal errors for excessive time-outs, ensuring both robustness and adaptability.

Keywords - Adaptive Simulation Termination, Dynamic End Time Prediction, Latency Record per DUT Data Path, Packet-Level Latency Statistics, Sim End Timer

---

Date of Submission: 10-05-2025 Date of acceptance: 21-05-2025

I. INTRODUCTION

Pre-silicon functional simulation is a cornerstone of modern hardware verification, leveraging object-oriented methodologies and languages like SystemVerilog under the Universal Verification Methodology (UVM) or Open Verification Methodology (OVM) frameworks [2], [6]. Accurate determination of simulation end-conditions is critical: terminating too early risks undiscovered errors, while over-waiting wastes compute resources and extends project schedules [1]. Efficient simulation termination is critical to optimize verification cycles and resource utilization under IEEE Std 1800-2017 SystemVerilog flows [1]. Traditional approaches use a constant safety margin after applying all test stimuli, but this cannot adapt to variations in DUT data-path delays, pipeline depths, or conditional behavior. Automated, latency-aware termination promises to optimize verification cycles without manual intervention.

II. PROBLEM STATEMENT

In functional simulations, the interval between input stimulus and observed output can vary widely across tests based on DUT data paths and DUT configurations. If a DUT data path involves different delays for various input packets, using a fixed, conservative wait time for all tests often leads to either premature simulation termination (risking undetected errors) or unnecessarily long idle cycles, both of which degrade verification productivity. Manual recalculation of per-test wait intervals upon changes in DUT behavior or test scenarios incurs further overhead and is prone to human error [3].

- **Fixed Wait-Time Drawbacks**: A single conservative wait time may cover worst-case delays but leads to idle cycles when most transactions complete faster [3].

- **Manual Tuning Overhead**: Recomputing per-test wait times after DUT changes or new scenarios is labor-intensive and error-prone.

- **Variable Input Rates**: Low stimulus rates or bursty traffic can leave occasional packets in flight long after others complete, complicating static timeout selection.

These factors degrade simulation efficiency and increase verification turnaround time, motivating a dynamic, self-adjusting mechanism.

III. PROPOSED SOLUTION

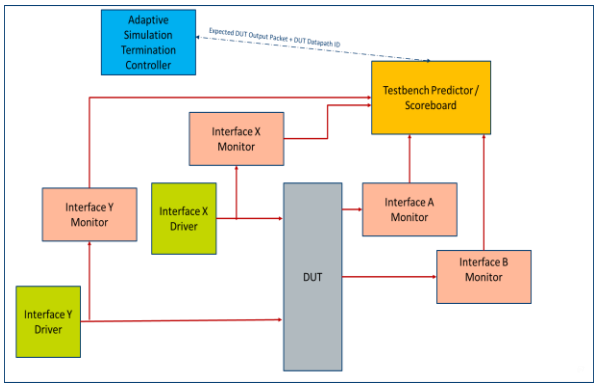

Figure 1: Adaptive Simulation Termination Controller in a OVM/UVM Test Environment.

The standalone Adaptive Simulation Termination Controller added to a standard UVM / OVM test environment as shown in Fig. 1 predicts and controls simulation end-conditions based on on-the-fly latency measurements, obviating manual timing adjustments [4], [5].

A. System Architecture

- Queues per Data Path (shown in Fig. 2): The Adaptive Simulation Termination Controller has three queues for each DUT data path ID (string or integer identifier):

    1. Active – pending transaction handles

    2. Complete – successfully timed/completed transactions

    3. Inactive – timed-out transactions

- Latency Record (shown in Fig. 3): For each DUT data path ID, the Adaptive Simulation Termination Controller stores the following fields in a Latency Record Object:

    - Expected Path Latency
    - Minimum Path Latency
    - Maximum Path Latency
    - Sample Count
    - Path Latencies Sum
    - Timeout Value (user-configurable safety margin)

The Scoreboard inserts a Timestamp packet (shown in Fig. 4) into each expected DUT output packet that it usually (shown in Fig. 4): upon injection into DUT, the Input Timestamp is set in the Timestamp Packet in the respective expected DUT output packet and the Timestamp packet is sent to the Adaptive Simulation Termination Controller where its handle is enqueued in to the respective DUT data path Active queue; upon DUT output, in a similar fashion, the Output Timestamp is set in the Timestamp packet in the respective expected DUT output packet by the Scoreboard.

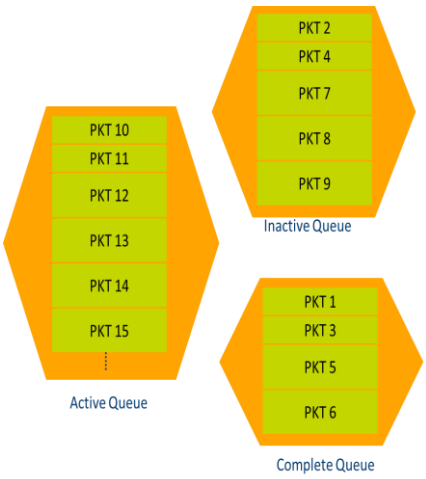

Figure 2: Active, Inactive, Complete Queue Representations

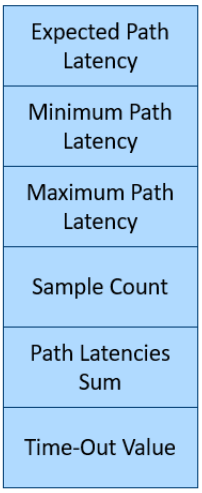

Figure 3: Latency Record Fields per DUT Data Path ID

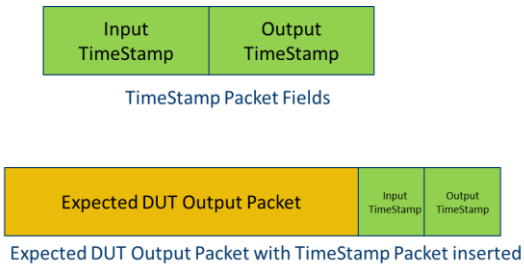

Figure 4: Timestamp Packet Fields and Expected DUT Output Packet with Timestamp Packet inserted.

B. Active Queue Processing & Latency Update

The Simulation Termination Controller performs the following functions for the Active Queue:

1. Access the next timestamp packet handle from the Active queue (FIFO).

2. If the Output Timestamp is populated in the respective timestamp packet:
   a. Compute Δ = Output – Input Timestamp.
   b. Move the handle to the Complete queue.

3. If not populated:
   a. Compute Δ = Current Time – Input Timestamp.
   b. If Δ exceeds Timeout Value from Latency Record for the DUT data path, move the handle to the Inactive queue.

4. Update Latency Record [7] for this DUT data path:
   a. If Sample Count = 0, set Path Latencies Sum, Min, and Max to Δ.
   b. Else, add Δ to Path Latencies Sum; update Min/Max if Δ is outside current bounds.
   c. Increment Sample Count.
   d. Compute var1 = (Path Latencies Sum / Sample Count), var2 = (Min + Max)/2; set Expected Path Latency = max(var1, var2).

5. Repeat the steps above for all the timestamp packet handles in the Active queue.

C. Inactive Queue Monitoring

The Simulation Termination Controller periodically checks the size of each DUT data path ID Inactive queue; if it exceeds a user-configured threshold, it issues a fatal error and terminates the simulation.

IV. SIMULATION TERMINATION CONTROL

The Adaptive Simulation Termination Controller maintains a Sim End Timer, an End Sim flag, and variables Current Wait Time and Previous Wait Time. It periodically executes:

1. For each DUT data path ID, count Active-queue timestamp packet handles without Output timestamps.

2. Compute Wait Time per DUT data path = (Expected or Maximum Path Latency) × (count of unresolved handles from #1 above). The user can select whether to use Expected or Maximum Path Latency for this calculation.

3. Determine the maximum Wait Time across all DUT data paths as Current Wait Time; store the old value in Previous Wait Time.

4. If the Sim End Timer is not running or Current Wait Time ≠ Previous Wait Time, (re)start it with Current Wait Time and clear the End Sim flag (e.g., raise a UVM objection) [8], [2].

5. Upon Sim End Timer expiry, set End Sim flag (e.g., drop the UVM objection).

6. Continue monitoring by repeating steps #1 through #5 above, allowing the simulation to

extend if new stimuli arrive after Sim End Timer expiry.

V. IMPLEMENTATION AND INTEGRATION

A sample implementation can be that the adaptive simulation termination controller is packaged as a SystemVerilog class extending UVM component with the following sample methods in the scoreboard and data structures in the Adaptive Simulation Termination Controller:

- task add_timestamp(handle, path_id) – For adding input timestamp
- task update_output(handle, time) – For adding output timestamp
- Associative arrays indexed by DUT data path ID for Active, Inactive and Complete queues.
- UVM events and objections for Sim End Timer control.

The following is a sample integration plan for the Adaptive Simulation Termination Controller:

- Instantiate the Adaptive Simulation Termination Controller in the UVM test environment.
- Connect Adaptive Simulation Termination Controller to scoreboard / testbench predictor via TLM ports [2], [8] for receiving timestamp packet handles for each DUT data path.

VI. DISCUSSION

The Adaptive Simulation Termination Controller offers several advantages:

- **Resource Efficiency**: Eliminates wasted run time, adapting to real latency distributions [3].
- **Scalability**: Supports multiple concurrent DUT data paths with minimal overhead.
- **Robustness**: Fatal-error thresholds prevent runaway simulations when DUT behavior deviates unexpectedly.
- **Methodology Independence**: Interoperates with any OOP-based environment (UVM, OVM, or proprietary) [6].

Potential limitations include sensitivity to path-identification granularity and the need to calibrate timeout thresholds for highly bursty traffic. Future enhancements might incorporate confidence intervals or machine-learning models for latency prediction.

VII. CONCLUSION

The Adaptive Simulation Termination Controller automates end-condition determination in pre-silicon functional simulation / verification. By learning per-path latencies through queue-based timestamp tracking and statistical updating, it dynamically adjusts simulation wait times and manages UVM objections without manual tuning. The controller enhances verification efficiency, reduces idle cycles, and enforces robustness via error thresholds. Integration into existing UVM/OVM flows is straightforward, making it a practical addition to verification environments. Future work will explore advanced prediction techniques and mixed-signal support.

REFERENCES

Standards & Manuals:

[1]. IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language, IEEE Std 1800-2017, Dec. 2017.
[2]. Accellera Systems Initiative, Universal Verification Methodology (UVM) Class Reference Manual, Version 1.2, May 2017.
[3]. Accellera Systems Initiative, Open Verification Methodology (OVM) Reference Manual, Version 2.1, Apr. 2013.

Books:

[4]. J. E. Bergeron, Writing Testbenches: Functional Verification of HDL Models, Springer, 2003.
[5]. G. Tumbush and C. Spear, UVM Cookbook, DeviceRealization, 2014.

Journal Papers:

[6]. A. Chandra and M. Rudin, "Adaptive Simulation Termination in UVM Environments," IEEE Design & Test of Computers, vol. 35, no. 2, pp. 50–58, Apr.–June 2018.

Conference Proceedings:

[7]. K. Magel and K. Robertson, "Dynamic Termination Controllers for SystemVerilog Simulations," in Proc. IEEE Int. Symp. Quality Electron. Design (ISQED), Santa Clara, CA, USA, Mar. 2016, pp. 125–130.

[8]. C. Faanes and A. Bhattacharya, "Queue-based Latency Analysis for Simulation Acceleration," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), San Diego, CA, USA, Nov. 2020, pp. 25–32.