Enhancing Verification Efficiency with Garbage-Model Methodology

Stas Yosupov
Smadar Eliyahu-Shulemzon
Xsight Labs, LTD

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

VLSI Verifcation Challenges

Design Complexity

As VLSI designs grow more complex, with billions of transistors per chip, thorough verification becomes increasingly challenging.

Simulation Challenges

Long simulation runtimes and extended regressions are required to achieve comprehensive coverage.

Corner Case Coverage

Rare scenarios are hard to predict, hard to hit, and easy to miss - especially with traditional verification techniques.

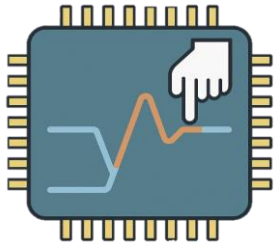
Time-to-Market Pressure

Faster tapeouts demand faster verification - often at the cost of depth and quality.

Garbage Model (GM): Synthetic Stress for Smarter Verification

Novel Approach

GM generates controlled synthetic stress directly inside the DUT by selectively manipulating and releasing internal flow-control signals, **revealing edge-case behaviors** typically missed in standard verification.



How It works

- Built on top of the regular testbench – doesn't replace it
- Manipulates flow-control signals (e.g., ACK, FIFO_FULL, EMPTY, STALL) inside the DUT
- Introduces semi-random, controlled stress to simulate congestion and edge cases
- Works in simulation & emulation

Why it Matters

- Enables earlier detection of critical and rare bugs
- Achieves higher coverage with fewer simulation cycles
- Reduces overall verification effort by minimizing the need for complex stress test development
- Accelerated TO readiness

GM Insertion Strategy

GM Positioning

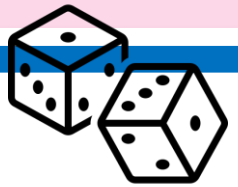
Define unique spots to inject congestion and backpressure to generate challenging verification scenarios

- Focus on flow-control signals (e.g., FIFO_EMPTY, STALL)
- Requires collaboration with designers and architects to identify effective locations

GM Setup

Place GM instances directly into the RTL or TB and configure their behavior

- Configure each GM instance with specific timing and behavior settings
- Supports different durations and frequencies of stress (short/long stalls)

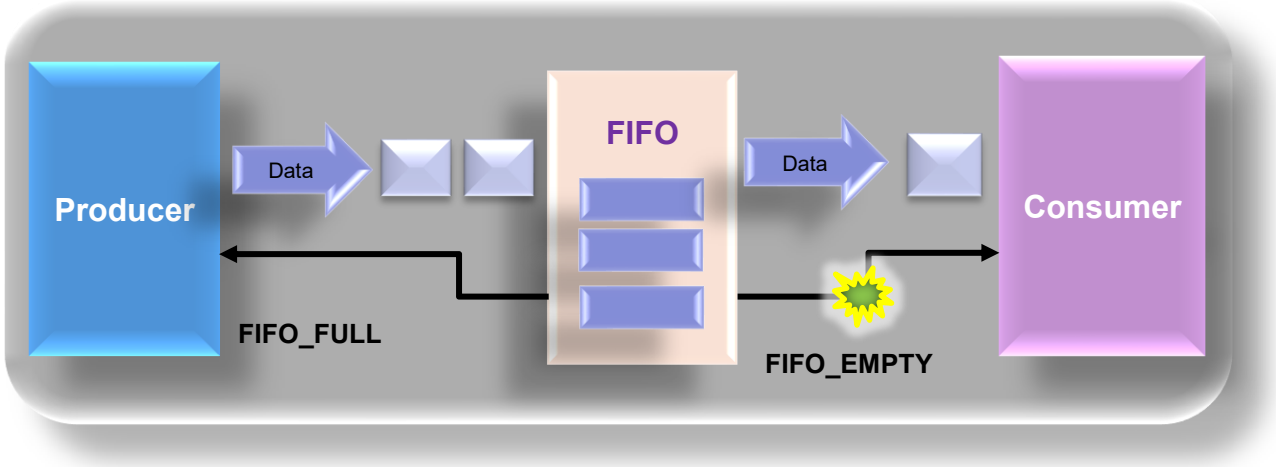


Randomize & Synchronize

Add dynamic behavior and synchronized control across GM instances

- Randomize GM behavior between stress (stop) and natural flow (pass) modes
- Synchronize GM blocks where dependencies exist to preserve system integrity

GM in Action – Uncovering Hidden Scenarios



Scenario:
In typical simulation, the FIFO rarely fills, and stress conditions are hard to reach.

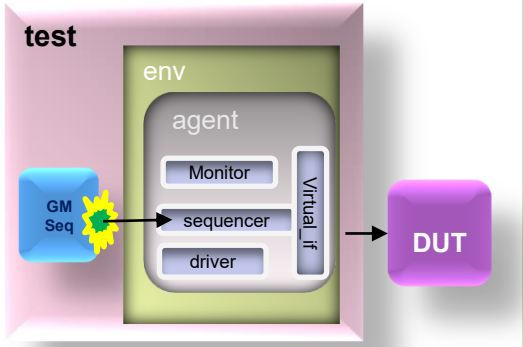
With GM:
- Force FIFO_EMPTY=1 -> Consumer stalls
- FIFO fills -> backpressure propagates to producer
- Release FIFO_EMPTY -> Consumer pops data continuously

Impact: Quickly reveals congestion, starvation and recovery logic – **scenarios that are typically hard to hit**

Two Approaches, One Methodology

UVM Based

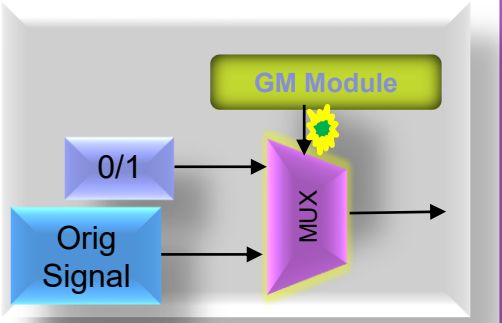
- Integrated as standard UVM components
- Test Controlled and easy to enable
- No Impact to RTL code
- Configurable
- Synchronized




```
while (GM_enable) begin
  dly = randomize_dly();
  uvm_hdl_force("gm_if.fifo_empty", 1
);
  repeat (dly) @(posedge clk);
  uvm_hdl_release("gm_if.fifo_empty");
end
```

RTL Based

- Synthesizable RTL module inside DUT
- Configured through registers
- Protected by DEFINE
- Emulation-ready
- Controllable




```
`ifdef GM_ENABLE
    gm_ctrl u_gm_ctrl (
      .clk(clk),
      .rst_n(rst_n),
      .gm_active(gm_active),
    );
  `endif
assign fifo_empty = gm_active ? 1'b1:
orig_fifo_empty;
```

Comparing GM Approaches

	UVM Based	RTL Based
Ownership	Verification Engineers	RTL Designers
Where It's Used	Simulation (cluster / full-chip)	Simulation & Emulation
Hierarchy & Reuse	Easily reused from cluster to FC	RTL-level instantiation
Configurable	Highly configurable via TB	Configurable via design registers
Flexibility	Can be added anytime, even post-RTL freeze	Integrated before RTL freeze
Enable / Disable	Controlled via UVM / testbench config	Controlled via RTL Define and Registers
Impact on Silicon	No impact, exists only in TB	Wrapped with IFDEF - not synthesized

Both variants use the same logic and philosophy – **enabling seamless** migration between **simulation and emulation flows** while maintaining consistent **stress behavior.**

A Few Things To Consider

1 Careful Selection

Careful selection of GM signals prevents false failures

3 Highlight Errors

GM may increase regression failures by revealing RTL bugs and environment issues

5 Artificial Coverage

Coverage is driven by artificial stress, but the design's behavior is real — interpret results with that in mind.

Too Much Garbage 2

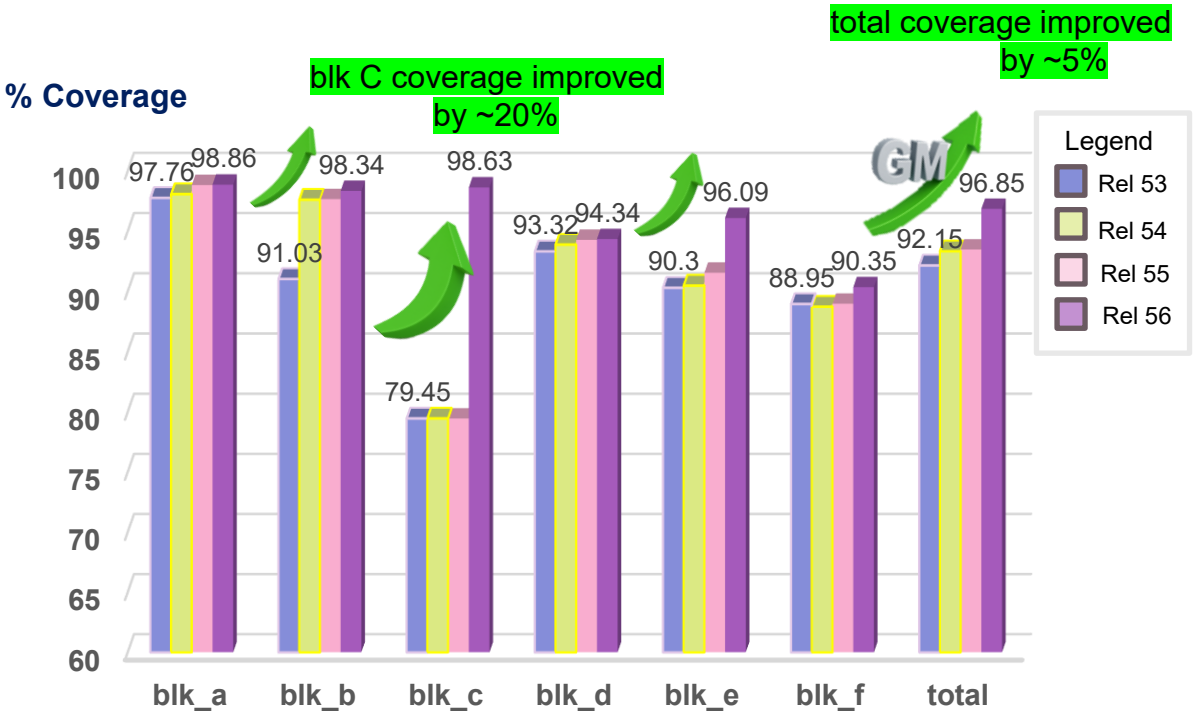
Too many GM points may stall traffic and reduce test effectiveness – control garbage intensity

Simulation time 4

Simulation time may grow, use with consideration for resources and scheduling impact

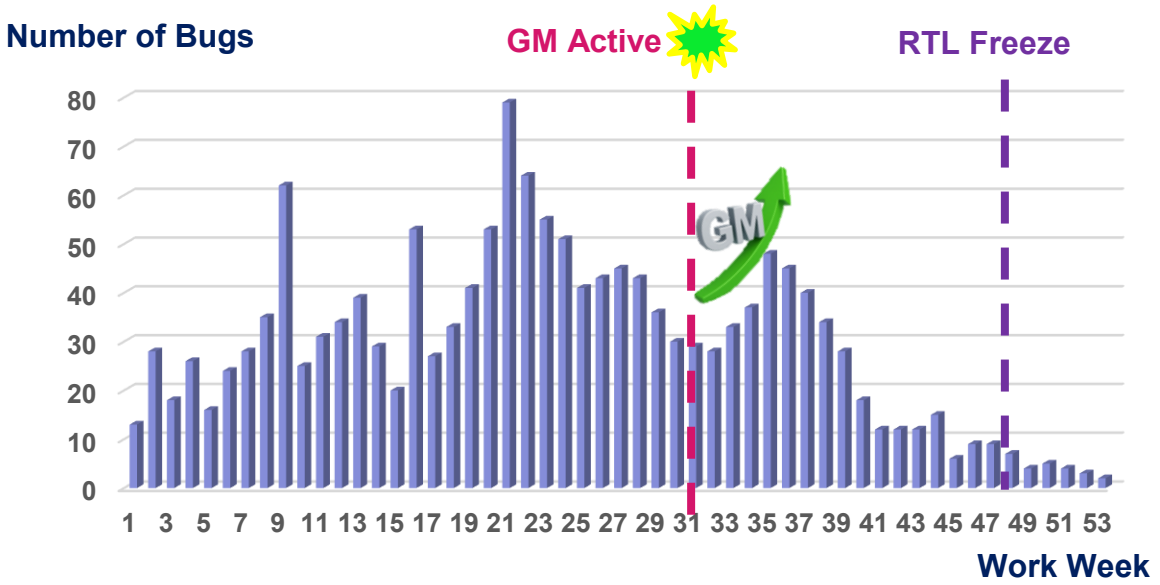
Evidence

Coverage vs. Project Time



Overall functional coverage increased by 5%
Time-to-Tapeout reduced through faster coverage convergence

Bug Detection vs. Project Time



Increased bug detection trend in Full-Chip (FC) environment
7 critical bugs were uncovered early using GM —
all in features previously considered stable

Summary

Faster Bug Detection

Uncover critical design issues

Flexibility

Applicable in both simulation and emulation env

Increased Coverage

Reach coverage milestones faster

Impact

Shortening time-to-market

Conclusion

GM : Transformative solution to modern VLSI verification challenges, providing a comprehensive approach to achieving high-quality designs.

AI

Security

Systems

EDA

Design

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation

sigda acm