

# Synopsys SystemVerilog Support

SystemVerilog Forum  
EDA Interoperability Developer's Forum  
October 21, 2004

Tom Borgstrom



# Synopsys SystemVerilog Status

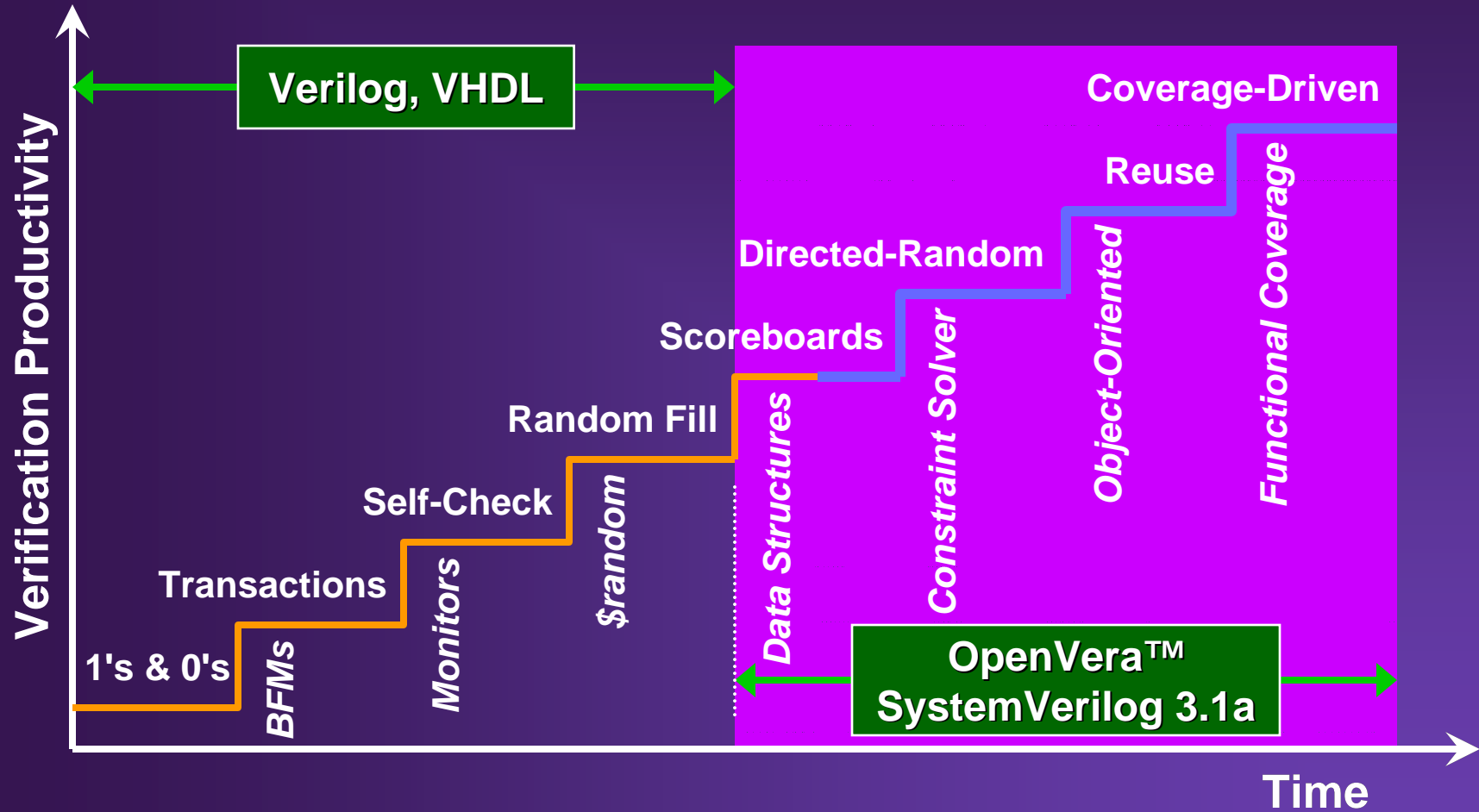
Product	SystemVerilog 3.1a Support	Availability
VCS®	Design Assertions DPI Testbench	Now Now Now (beta) Q4, 2004
Design Compiler® DC FPGA	Design	Now
Magellan™	Assertions	Now (beta)
Formality®	Design	Now (beta)
Leda®	Design	Now

# Comprehensive SystemVerilog 3.1a Testbench Support in VCS

- SystemVerilog data types
- Object-oriented programming & encapsulation
- Array support (queue, dynamic, associative)
- Task/function support
- Threading & synchronization
- Random constraints
- Functional coverage
- Testbench debug

# Advanced Verification Methodology

Enables higher productivity



# SystemVerilog Verification Methodology Manual (VMM)

- **By: Synopsys and ARM Experts**

- Janick Bergeron, Synopsys Scientist & Verification Guild moderator
- Eduard Cerny, Synopsys Principal Engineer
- Andrew Nightingale, ARM SoC verification
- Dr. Alan Hunter, ARM verification methodology

- **Describes:**

- Architecture guidelines and industry best practices for more effective and faster functional verification

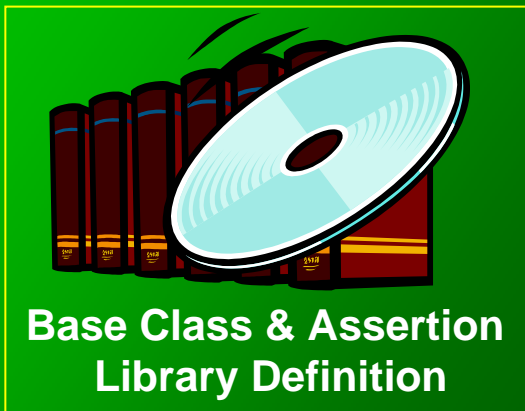
- **When: Q1, 2005**



# SystemVerilog VMM Improves Productivity



Proven Methodology



Base Class & Assertion  
Library Definition

SystemVerilog VMM

- **Proven approach**
  - Lowers risk of adopting SystemVerilog
  - Equivalent methodology ships with Vera today
- **Accelerates testbench ROI**
  - Speeds deployment
  - Propagates best practices
- **Improves productivity**
  - Promotes verification reuse
  - Minimizes code writing
  - Reduces code for individual tests
  - Maximizes use of automation
  - Creates supportable code

# The VMM Environment

## What Does It Provide?

Coding Guidelines

- **Modeling Approach**
  - Transactions
  - Variant Data
  - Transactor Control
  - Transactor Interfacing
  - Simulation Control

Class Library

Base Classes

- **Coding Standards**
  - Data & Transactions
  - Transactors
  - Verification Environments
- **Building Blocks**
  - Transaction Interface
  - Message Service
  - Event Notification

# VMM Base Class Library

- Quickly build a sophisticated, robust environment
  - `vmm_log`
    - Records, filters, promotes & demotes messages
  - `vmm_channel`
    - Passes data between transactors, includes flow control that can suspend threads when full or empty
  - `vmm_env`
    - Instantiates environment, handles high-level flow of the test
  - `vmm_data`
    - Building block for data packets and transactions
  - `vmm_xactor`
    - Used to model all transactors; built-in flow control for starting, stopping and resetting transactors
- Fully documented in the SystemVerilog VMM

# VMM Checker Library

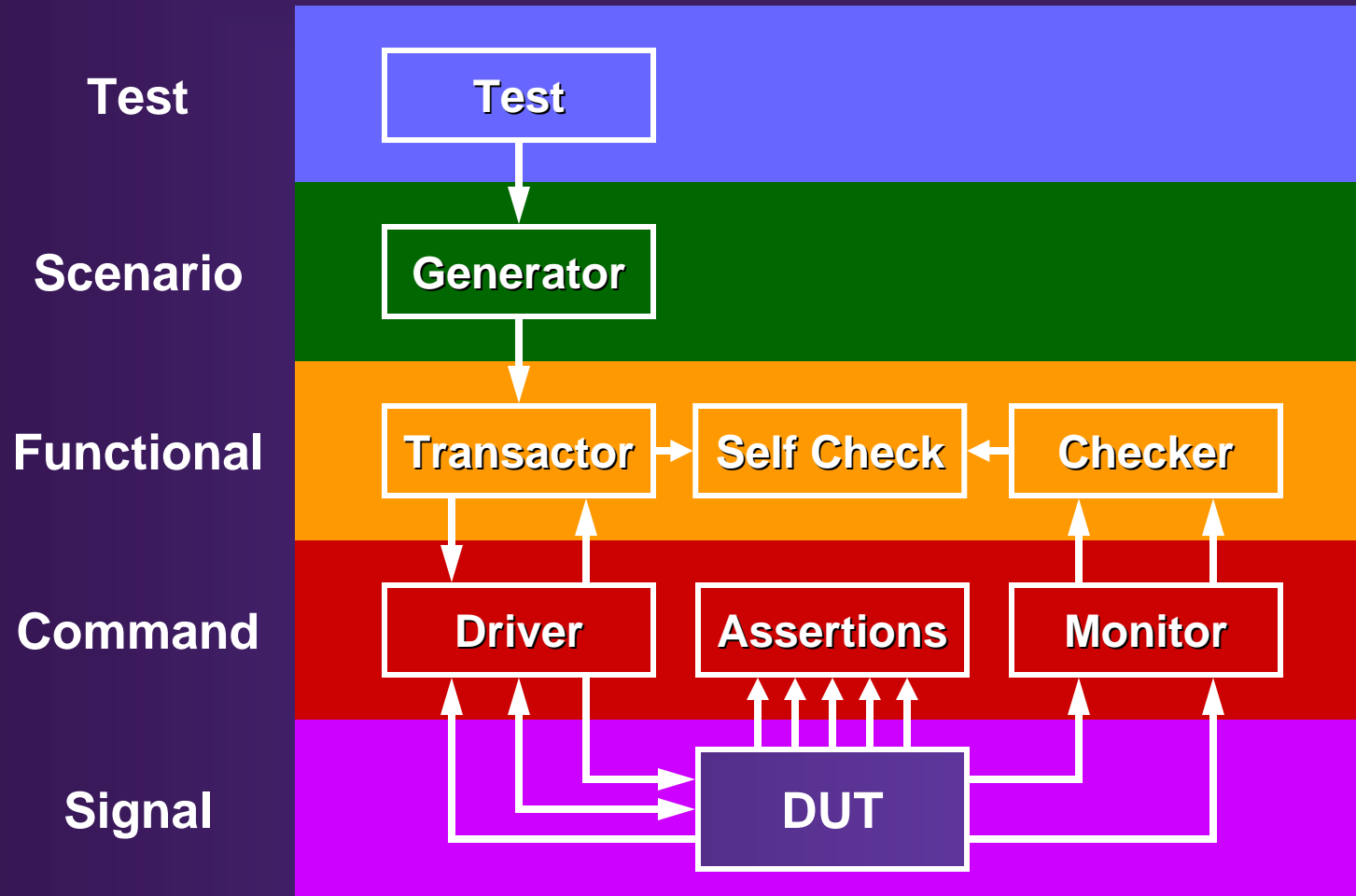
- Collection of advanced SystemVerilog assertions

Value Integrity	Protocol	State Integrity	Sequence
assert_bits assert_value assert_odd_parity assert_one_cold assert_one_hot assert_even_parity assert_always_on_edge assert_decrement assert_delta assert_zero_one_hot assert_no_underflow assert_no_overflow assert_no_underflow assert_increment assert_range	assert_arbiter assert_data_used assert_dual_clk_fifo assert_fifo assert_memory_sync assert_memory_async assert_no_contention assert_req_ack_unique assert_stack assert_valid_id assert_fifo_index assert_proposition assert_frame assert_handshake	assert_code_distance assert_driven assert_mutex assert_next_state assert_no_transition assert_proposition assert_quiescent_state assert_never assert_next assert_always	assert_hold_value assert_reg_loaded assert_time assert_transition assert_unchange assert_width assert_win_change assert_win_unchange assert_window assert_implication assert_cycle_sequence assert_change

- Fully documented in SystemVerilog VMM

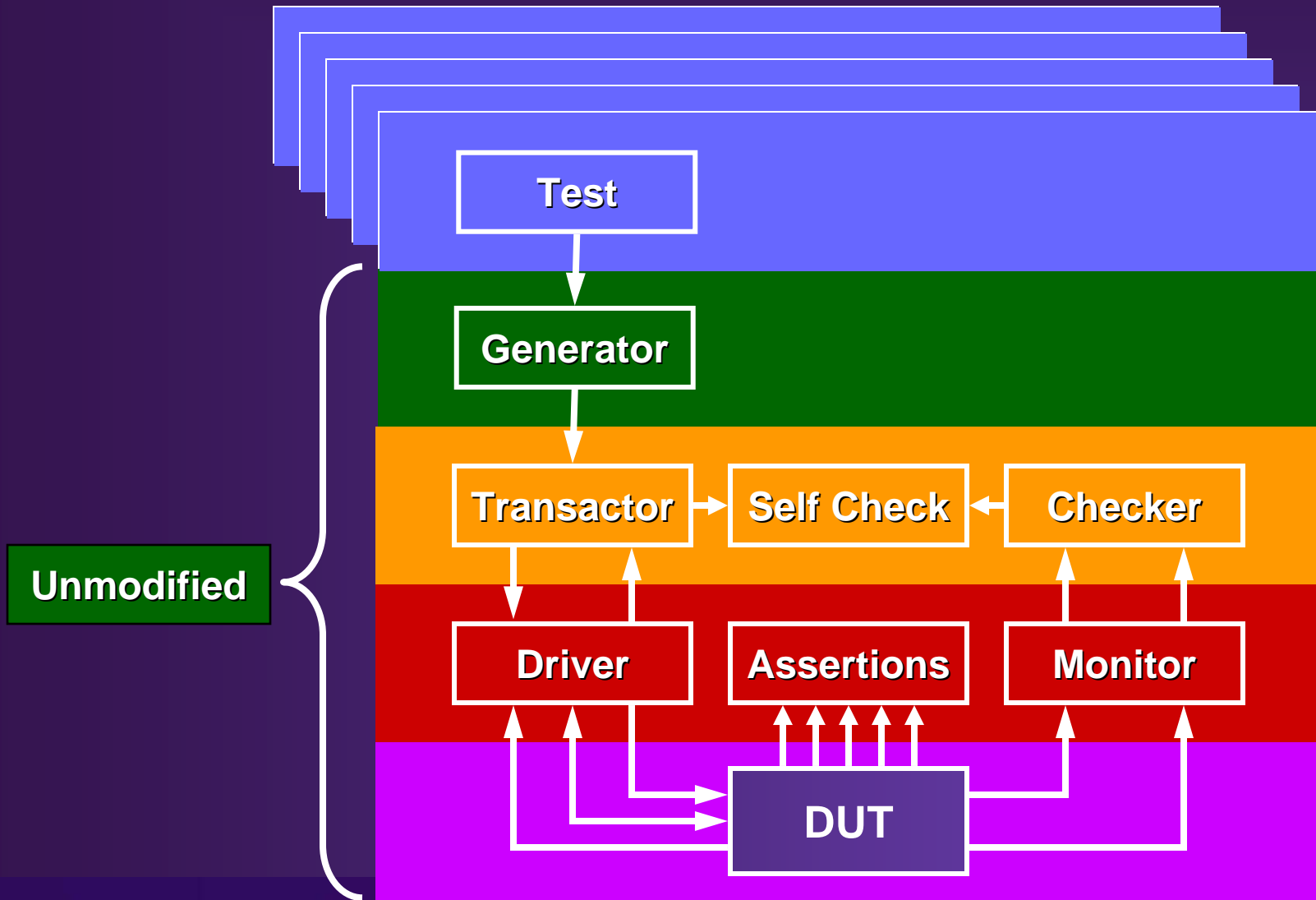
# VMM Environment

## Layered Testbench Architecture



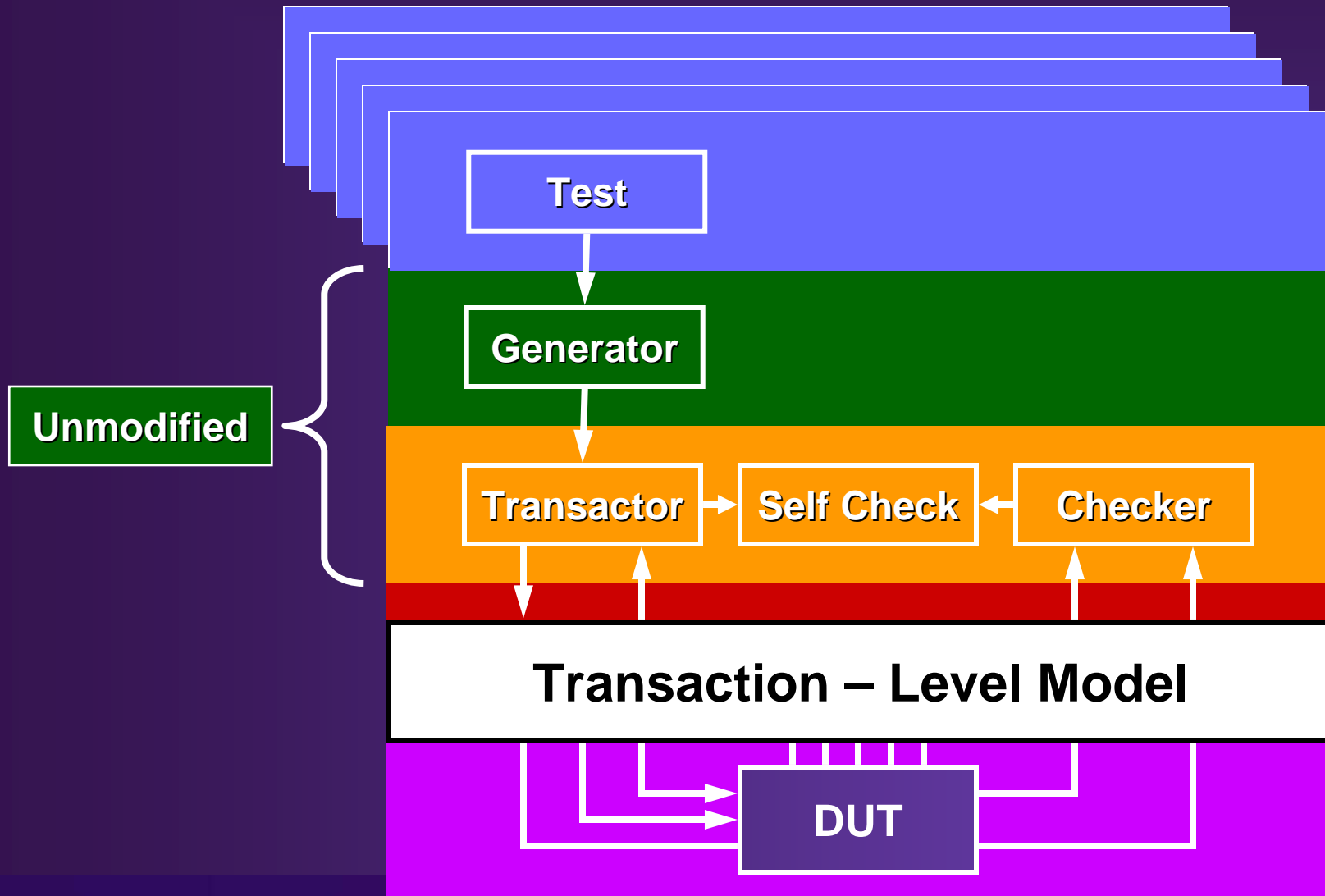
# Verification Environment

## Reused Across Tests



# Verification Environment

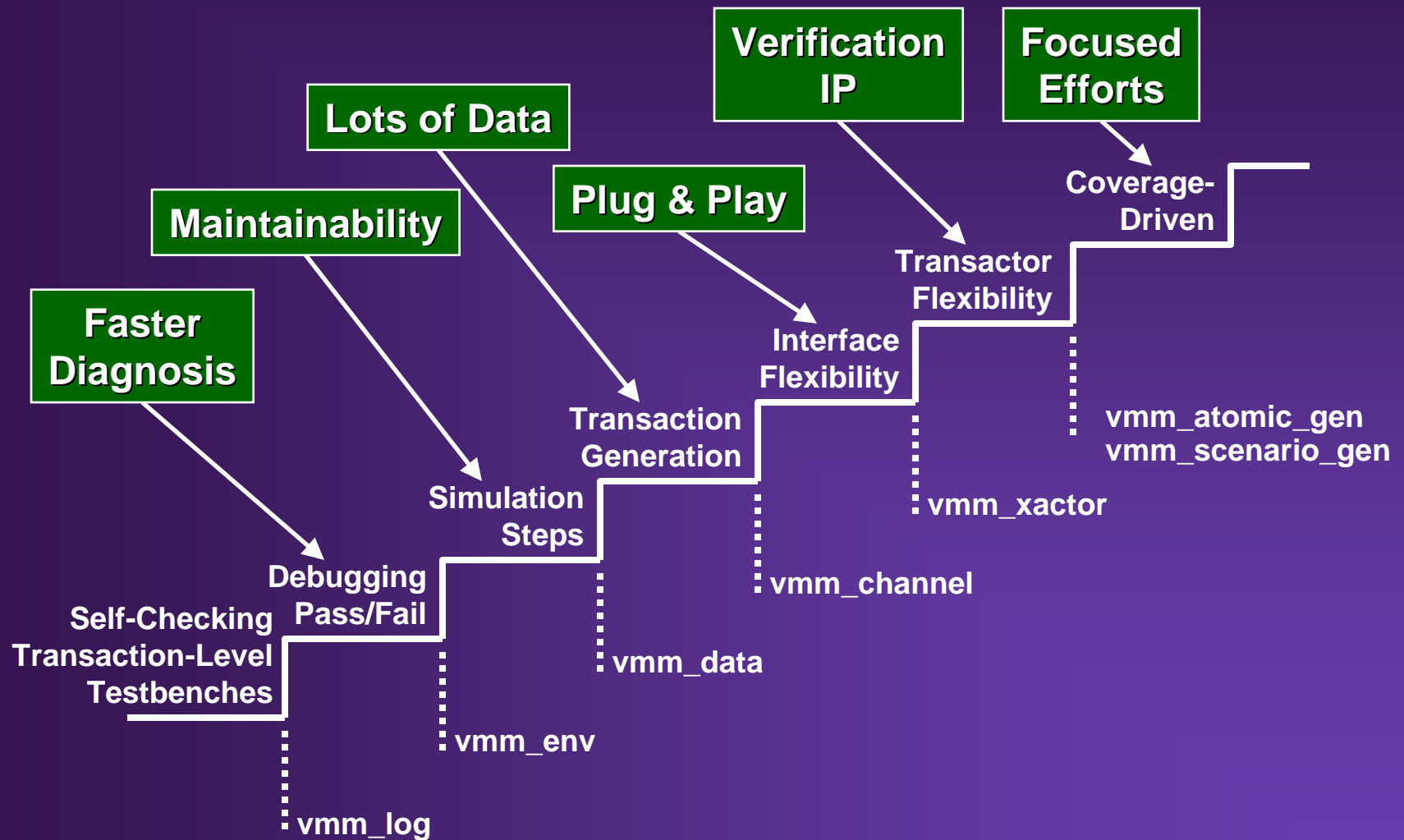
## Reused Across Models





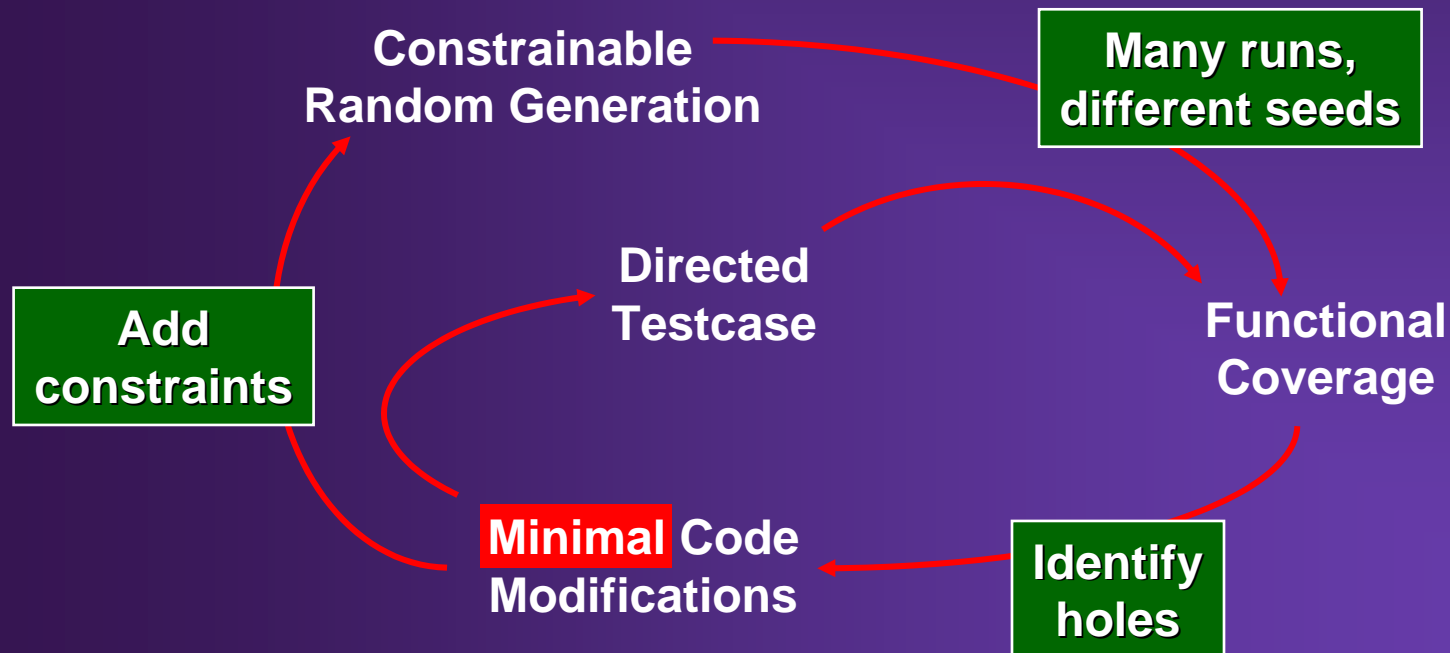
# Step-by-Step Advances

## Incremental VMM Adoption



# Coverage-Driven Verification

- Focus on uncovered areas
- Trade-off authoring time for run-time



# Summary

- **Synopsys SystemVerilog Status**
  - **Complete tool flow available now**
- **SystemVerilog Verification Methodology**
  - **Get the most out of SystemVerilog**