White Paper

SystemUVM™
Empowering UVM Engineering

Version: 1.0
February 2022

Breker Confidential

Abstract

Test Suite Synthesis holds tremendous promise for semiconductor functional verification, particularly in UVM block-level environments. Today scaling UVM test content for large blocks and sub-systems is problematic. The use of synthesis to solve these issues seems obvious. However, the language learning curve for formats such as PSS is significant. SystemUVM™ is designed to solve this problem by providing a linguistic approach extremely close to UVM/SystemVerilog with the power to easily compose specification models for this process.

Breker Verification Systems, Inc.
www.brekersystems.com

Introduction

Functional Verification Engineers using UVM can enjoy a large number of benefits by synthesizing test content for their testbenches. Abstract, easily composable models, coverage-driven content, deep sequential state exploration, pre-execution randomization for test optimization and configurable reuse are just some examples of the advantages afforded by test suite synthesis.

However, a specification model is required and there are few alternatives that a UVM/SystemVerilog engineer can simply pick up and use. Enter **SystemUVM™,** a UVM class library built on top of Accellera's Portable Stimulus Standard, plus methodology and other improvements, that looks and feels like SystemVerilog with UVM, but enables the level of abstraction, methodology and composability required for this specification model with an almost negligible learning curve.

UVM Today

Accellera's Universal Verification Methodology (UVM) with SystemVerilog has become ubiquitous in modern semiconductor verification, leveraged in almost 80% of development environments (Wilson 2020). The standard provides an excellent framework for structuring verification testbenches and allows for transaction-level abstractions and VIP reuse, among other capabilities. Engineers have adopted the methodology as part of their personal expertise portfolio, and it has enabled them to lead verification processes within their organizations.

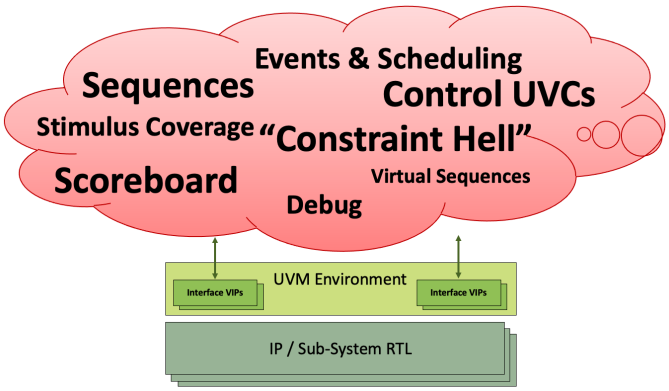

Figure 1: UVM Large Block Scaling Issues

UVM provides an effective framework for verification testbenches (the connectivity between transactional test content and the design under test itself leveraging interface VIPs) and for test content (the actual test sequences that sets up behavior) for small and medium-sized design blocks. However, for large or complex blocks and multi-IP sub-systems (multiple interconnected blocks on a System-on-Chip (SoC) with no processor) composing and executing effective test content is increasingly difficult and requires significant resourcing. UVM cannot scale for these large Designs Under Test (DUTs).

Examining the reason for this issue reveals hard to understand test content modeling, difficulty in describing and measuring effective coverage metrics, a reliance on random test vectors solved during simulation runtime, a lack of reuse configurability,

and other challenges. There is a need for verification services such as resource scheduling, memory management, etc. which has to be recoded for every project. This often results in complex test content architectures, including components such as Control UVM Virtual Components (UVCs), where more effort is expended on test coding than understanding the verification objectives. A synthesis approach to this problem can solve these issues, but an abstract model is required to drive this technology, requiring a purpose-built language.

The SystemUVM Concept

The Accellera Portable Stimulus Standard (PSS) appears to be a viable option. However, PSS as it stands today, is relatively unwieldy for this purpose, does not easily layer into a UVM testbench, and requires a significant learning curve. It is primarily a declarative language, unlike C++ and SystemVerilog which are procedural in nature. What is required is a language approach that is as close as possible to UVM and SystemVerilog, but retains the power of PSS for synthesizing test content, and directly maps into existing UVM environments.

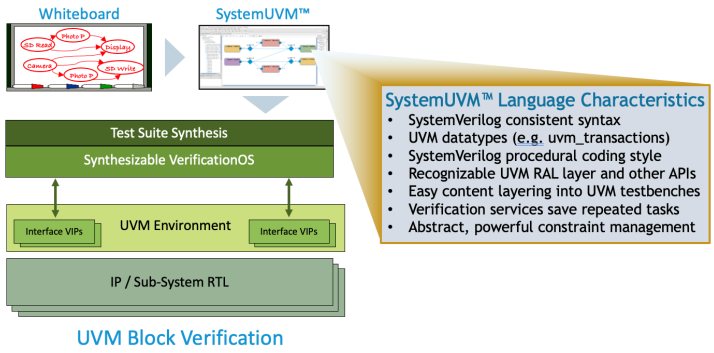

Figure 2: SystemUVM Linguistics and Flow

UVM is a methodology built on top of SystemVerilog. One alternative is to do the same with PSS, that is, provide a UVM-like class library on top of this language. Couple this approach with types that mimic the syntactic feel of SystemVerilog, and use common APIs, such as the UVM Register Access Layer (RAL) and it is relatively easy to build a framework around PSS that makes it look and feel like UVM/SystemVerilog, with a learning curve of just a few hours. This is the approach taken by SystemUVM™.

This linguistic approach that eliminates the disconnect between PSS and UVM, can lead to some significant advantages for UVM test content synthesis with minimal effort.

Test Suite Synthesis with SystemUVM

For example, the single specification model is inherently self-checking, thus eliminating the need to create independent scoreboards. This model may be automatically synthesized into a multi-sequence schedule, thereby providing an auto-virtual sequencer function. SystemUVM™ comes complete with abstract constraints,

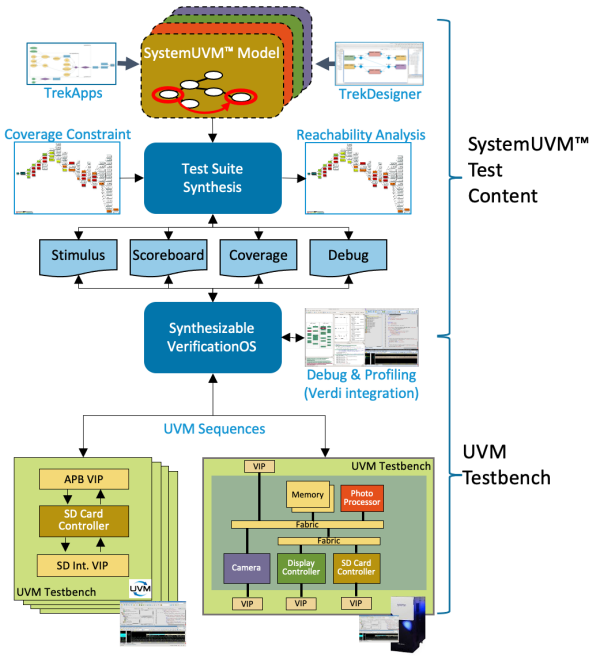

Figure 3: Test Suite Synthesis

or "path constraints", reducing the effect of node level constructs that often result in intermingled constraint layers hard to understand and maintain

The synthesis process is coverage-driven, in that coverage becomes a synthesis constraint that is used to direct test generation based on the specification model. This eliminates post-execution coverage analysis, the arduous requirement to write a coverage model, and the test-edit-to-execution test content respins that are common to fix coverage holes.

The synthesis process explodes the specification model state space, similarly to the way a formal verification tool examines the state space of a Register Transfer Level (RTL) model. Once this state-space is made available, it may be traversed using Planning Algorithms, an Artificial Intelligence (AI) technique that starts from a state in question, or a check, which may be sequentially deep in the state-space and works backwards to the inputs to derive a test set. This approach is different to the classic constrained random method that, if not controlled carefully, can blast the inputs of a block with vectors in the hope that they will trip over the required state. The synthesis method has been proven to detect extremely complex corner-case issues and provide ultra-high coverage. If there is a bug, it will almost certainly be found.

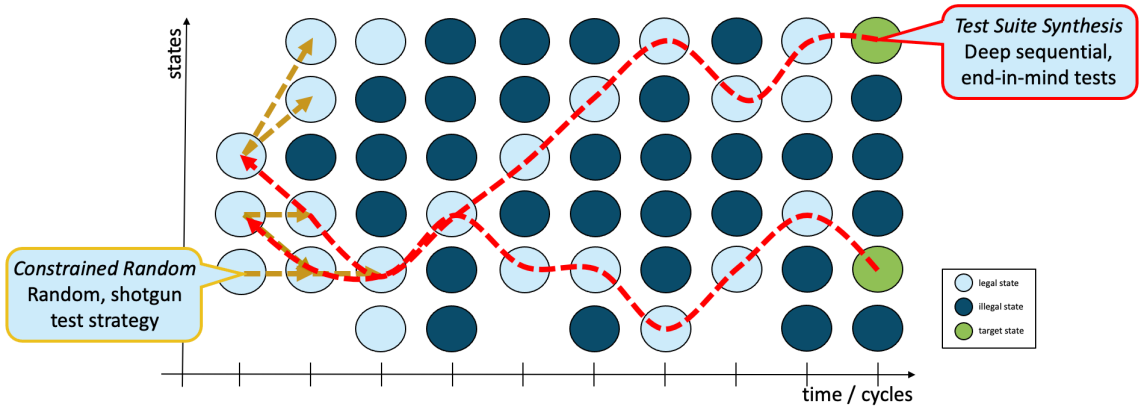

Figure 4: Constrained Random versus AI Planning Algorithm Test Suite Synthesis

The generated test content may be easily layered into existing UVM testbenches with no modification, operate with existing UVM test code and, if required, existing scoreboards and coverage models. The SystemUVM™ code is inherently reusable and can be set up to be easily configured for different scenarios, thus allowing a test content library to be built up, similar to the Breker TrekApps. It is also inherently composable, allowing IP modules to be chained together to quickly create multi-IP test cases.

The synthesis process performs randomization based on the specification model reactivity prior to simulation, although it can also be executed during simulation for RTL reactivity and set up in a hybrid model where randomization can be split between the two. Up-front randomization has a number of advantages, for example it reduces random solver execution during simulation thus accelerating the process significantly. It also enables random test content to be executed inside emulation without the need to run the testbench on a separately interfaced emulator. This dramatically accelerates emulation for UVM block and sub-system acceleration, providing a huge shift-left potential.

Test Suite Synthesis has been shown to improve test composition time by over 5X while also significantly increasing coverage (brekersystems.com/resources/case-studies for a Broadcom Case Study on the subject) at many end-user companies of the technology.

SystemUVM For SoC Reuse

SystemUVM reuse and portability extends beyond the block and sub-system functional verification to full SoC and beyond. Many SoC integration teams have long desired to reuse UVM test content for full system verification but have found it troublesome given the lack of reuse hooks and the inability to run random tests in emulation. SystemUVM solves these issues and allows the UVM teams to supply functional tests with no additional effort.

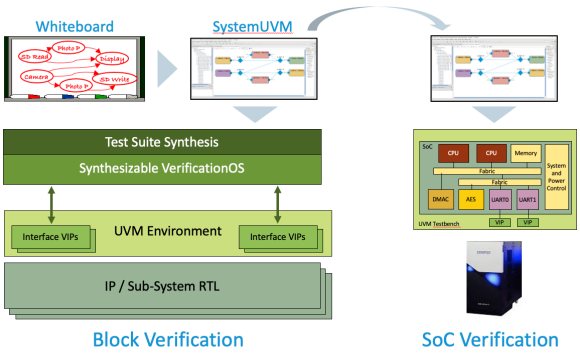

Figure 5: SystemUVM for SoC Reuse

This places these engineers in a strong position as the functional verification test owners for the entire flow, and they are able to do this without impact to their own UVM testbenches. Test Suite Synthesis includes the Synthesizable VerificationOS technology that acts as a layer between the UVM testbench and the SystemUVM test content, which allows the UVM engineering team to provide their test content code without any recoding, or impact to their testbenches.

Summary

SystemUVM could be a significant game changer given that it solves the UVM test complexity problem while minimizing language learning curves. Commentators on the PSS have expressed the idea that the same capabilities could be provided while not requiring a new language. Engineers are excited about the options afforded by PSS, but are concerned by this learning curve. SystemUVM fixes this.

As SystemUVM leverages Accellera PSS, it is based on an industry standard that should operate with multiple vendor tools. Several notable semiconductor companies have worked with Breker on this initiative to drive the reuse approach across the industry.