Glossary

Functional Verification

Amplification
The use of dynamic formal verification to expand exploration of design behavior beyond that performed in simulation.

Assertion
A directive to a tool telling it what to do with a property. Assertions are properties that are evaluated within an execution engine: a simulator, emulator, or formal analysis tool. An assertion provides a monitor that ensures a property holds once, always holds, or never holds during verification of the design.

Assertion-Based Verification (ABV)
A verification methodology in which assertions are a key element of verification, specifying design behaviors (see Property) and critical scenarios (coverage points) to be modeled. ABV uses simulation and formal analysis to ensure that no properties are violated and all coverage points are reached.

Assertion Coverage
A subclass of Functional Coverage. Since assertions can include flexible sequence specifications (“a followed 1 to 4 cycles later by b”), it is useful to keep track of how often (and if) a particular sequence was satisfied by each possible temporal behavior. The recording and analysis of this information is Assertion Coverage.

Assertion Density
A measure of the quality of the assertions in the design.

Assertion Library
A collection of common assertions in a reusable format.

Assertion Macro
A high-level compiler directive for the automatic generation of assertions based on supplied arguments.

Assertion Synthesis
Converting assertion macros into assertions.

Black-Box Verification
Functional verification based on the specification of the output behaviors in response to the inputs applied.

Bounded Proof
An assertion that cannot be violated within the specified proof-radius and supplied constraints.

CheckerWare
A product that is a collection of assertion macros that check common RTL structures and industry standard protocols.

Clock Domain Crossing (CDC)
Situation in which a signal crosses between asynchronous clock domains. Also, a product (0-In CDC) that automates CDC verification.

Constrained-Random Simulation
A methodology that allows a single test to check multiple features. Each test describes a set of possible scenarios, and the simulator chooses a specific scenario for each invocation. Stimulus scenarios are described in terms of constraints, which are mathematical relationships limiting the set of legal values for signals that drive the design. The simulator generates random values for stimuli and the constraints ensure that the scenarios thus generated are valid.

Control-Oriented Functional Coverage
A method of data collection and sub-class of functional coverage where specific temporal behaviors that occur are recorded. Control-oriented coverage can be specified using assertions, and therefore is sometimes referred to as Assertion Coverage.

Corner Case
Verification scenario of a design that is difficult or not commonly reached in testing.

Counter-Example
Stimulus sequence from a legal design state that violates an assertion.

Coverage-Driven Verification (CDV)
A methodology where functional coverage metrics are used automatically to record and analyze information to ascertain whether (and how effectively) a particular test verified a given feature; then feeding that information back into the process to target additional verification efforts more effectively.

Data-Oriented Functional Coverage
A method of data collection and sub-class of functional coverage where sets of variable values at a particular point during simulation are recorded. Used during the stimulus generation process to ensure that all transaction types to all address spaces or other analogous scenarios have been created. Also used during the results checking process to record that the device responded with the correct results.

Dynamic Formal Verification
Exhaustive formal verification that leverages simulation states as starting points for mathematical analysis to consider alternative sequences to existing simulation tests.

Equivalence Checking
Functional comparison of a reference design against a modified design (for example, RTL vs. gate-level netlist). Note: This is what the FormalPro tool does.

Firing
An assertion violation.

Formal Model Checking (FMC)
Uses detailed mathematical analysis of the design to exhaustively examine all possible states of a design to determine if any of them violate a specified set of properties. The properties themselves, often expressed as assertions, represent a precise description of sequential (“this happens after that”) or invariant (“this will never happen,” or “that will always happen”) behaviors about the design, with each property being considered a “piece of the specification.” Also referred to as Property Checking.

Formal Verification (FV)
The use of mathematical models and analysis to functionally verify design behavior. Also, a product (0-In Formal Verification) that uses formal techniques for model checking.

Functional Coverage
A collection of user-defined metrics for tracking whether all required scenarios have been tested. The more randomization included in testing, the more functional coverage data is required. See also: Coverage-Driven Verification, Data-Oriented Functional Coverage, Control-Oriented Functional Coverage, and Structural Coverage.

Hot Spot
Design logic that is hard to verify.

Model Checking (or Property Checking)
Model checking is a method for formally verifying finite state concurrent systems. The system specifications are expressed using properties.

Monitors (or Protocol Monitors)
Modules that combine multiple checkers to capture complete interface protocols, check for violations in simulation, provide metrics for design exercise or coverage, and provide targets and constraints to formal verification.

Open Verification Libraries (OVL)
An industry-standard collection of assertions.

OSCI TLM-1.0
A standard set of interfaces and a transport mechanism implemented in SystemC along with their associated semantics for building TLMs.

Proof
The result of static formal verification when it can be determined that an assertion is never violated from the given initial state.

Property Checking
See: Formal Model Checking.

Property
A concise statement about a specific intended behavior of a design. Properties provide concise, mathematically precise descriptions of behavior about the design, or that constrain the operating environment of the block. In addition, properties can also describe coverage points or scenarios that must be exercised by the verification process. Properties can specify functionality, timing, or any other aspect of the design. The term assertion is often used interchangeably with property.

Property Specification Language (PSL)
An industry standard language for specifying assertions.

Simulation with Assertions (or Assertions in Simulation)
Checking assertions during design simulation and reporting violations. A violation can result from an incorrect assertion or a design implementation error that causes the simulated design to violate the assertion.

Static Formal Verification
Formal verification that analyzes all legal design behavior from a single design state.

Structural Coverage
The practice of recording component-specific functional coverage for common RTL design structures, such as FIFOs, arbiters and such. The coverage information recorded is targeted to report explicitly whether common corner cases known to be critical to the proper operation of the specific structure have been exercised. Initially developed by 0-In® and included in (but not limited to) the 0-In® CheckerWare® components.

SystemC
A standard language based on C++ which is ideally suited for building TLMs. Since TLMs require less detail than RTL models using them as part of your flow can significantly boost productivity.

System Verilog Assertions (SVA)
Assertions written in the SystemVerilog language.

Testbench
A verification environment providing the infrastructure necessary to drive stimulus into a design, record and analyze results. Provides a vehicle for which tests can be written to target and verify specific features of the design.

Testbench Automation (TBA)
Tools and methodologies used to assist in the assembly of testbenches and the generation of stimulus. Also ensures that the testbenches can be used (and re-used) effectively and reduces the number of tests that must be hand coded.

Transaction
A transaction is a single transfer of data or control between design elements. Transactions are often represented as chunks of data moving between communicating processes.

Transaction Level Modeling (TLM)
To handle increasingly complex designs, Transaction Level Modeling can be used for high level design and verification at both system and RTL levels in which the focus is placed on what behaviors are executing (reads, writes, etc.) as opposed to focusing on how the behaviors are implemented via signal-level transitions. The OSCI TLM standard is an API which provides a general purpose transport mechanism for doing Transaction Level Modeling. This API is available from OSCI and is integrated into Questa.

Transaction Level Model
A model of a system or component at an abstraction level higher than RTL in which the interface is defined in terms of transactions (often implemented as function/task calls), rather than signals. By abstracting out the details transaction level models run faster and can be written more easily.

Unbounded Proof
An assertion that can never be violated given the supplied constraints.

Verification Automation
The application of advanced methodologies such as Assertion-Based Verification (ABV), Coverage-Driven Verification (CDV), Testbench Automation (TBA), Transaction Level Modeling (TLM), and technologies such as static and dynamic formal verification model checking for the purpose of significantly improving verification productivity.

White-Box Verification
Functional verification that considers the implementation of a design. White box verification can be implemented using assertions, monitors, or any means for peering into the internals of a design.

Mentor Graphics®