

Formal Verification Techniques

DV Club March 2007

Rekha Bangalore



What is Formal Verification?

- Formal verification is the process of proving or disproving properties using formal methods (i.e., mathematically precise, algorithmic methods). A formal proof of a property provides a guarantee that no simulation of the system considered will violate the property. This eliminates the need for writing additional test cases to check the property.

What are Formal Verification properties?

- Properties for formal verification are represented in a mathematically precise, machine-readable language for which there are formal semantics.
- Assertions are the preferred language for writing such properties.
- Adding assertions will help verify the design feature sets and also obtain coverage for assessing IP quality.

Formal Verification in Verification space

- Formal Verification can be applied at block level to eliminate functional bugs early in the design cycle and match block specifications.
- It is difficult to assess the need for additional patterns for exhaustive verification if coverage goals are not integrated as part of formal verification.

Formal Verification in Verification space

- Formal Verification Tools can be applied at SoC level for
 - Static design rule checks
 - Check for tied ports at SoC level
 - Bus contention
 - Verifying inter modular constraints (Case study)
 - > This is not an easy task as all assertions are built in for stand alone verification and to move to the next level requires:
 - Understanding of module interactions in the application.
 - Understanding of integration of the modules at SoC level
 - Porting of assertions at SoC level can slow down the speed of the tools to process the design and prove the properties.
 - Sequential depth will become a harder issue to resolve.

Features of Formal Verification

- Checks in Formal Verification tools include
 - Synthesis pragmas
 - Cross clock domains
 - X-state propagations in the design
 - Tri state and bus contention
 - Dead code in the design
 - Checks for Case and branch enable statements
 - Checks for Structural modeling
 - Checks for clock related errors
- Checks for user defined properties in Formal Verification tools include
 - Specifications in form of assertions
 - Specifications in form of constraints

Advantages of Formal Verification

- Reduce vector set for either dynamic or random simulations. Formal verification is vectorless and checks assumptions made by designers
- Smaller vector set for corner cases are easier to detect at block level and bugs can be fixed prior to SoC integration
- Automatic generation of testbench
- Bugs found at Post layout due to incorrect timing constraints are detected early. Example: Multi cycle paths and False Paths
- Validates constraints from designers
 - Synthesis constraints are input to Formal Verification
 - Helps in any invalid timing constraints for IP by the designers
 - Helps in reducing post layout debug time when IP is integrated in SoC

Challenges of Formal Verification tools

Challenges of current tools:

- Capacity issues
- Custom cells integrated at block level.
- Cannot generate functional coverage metrics from assertions
- Supporting all features of LRM
- Automatic generation of assertions/constraints based on scenarios
- Formal compatible VIP for standard Protocols
- Cannot generate Code coverage reports based on the proof

Challenges of Formal Verification methodology

- Challenges of using Formal techniques on large designs
 - Apply constraints for intermodular signals (Not tried it yet.) and see if violations show up.
 - Check for tied ports in the design and correlate with toggle coverage tools.
 - Complex SoC vector initialization is required.
 - Analog and Memory behavioral model checks are not fully available in EDA tools currently
 - Cannot use for multiple functional modes. Separate initialization sequences are required.

- Formal Verification is a must for quality verification
- Efficient strategy is required to plan the flow for both block and full chip.
- Tools must be updated to improve the engines internally to handle larger designs and have the capability of changing the user defined rules dynamically.