Expanding the Reach of Formal

Ziyad Hanna PhD.  | January 13, 2014
Sr. VP of R&D
Jasper Israel GM
Jasper R&D Team

- Offices in US / Sweden / Brazil / Israel
- > 70 in R&D (out of >135 total employees)
- > Many PhD and experts in Formal Verification, EDA
- Highest number of formal R&D in EDA: est. >3x nearest competitor
Jasper Academic Relations

- Prestigious Jasper Technology Advisory Board (TAB)

- Japer sponsors major conferences in FV domain
  - FMCAD, SAT, CAV, HVC

- Japer serves in program committees …
  - DAC, SAT, HVC, FMCAD, CAV, …
Upgrade your Design & Verification with JASPER design automation

JasperGold® Apps

Common Database • Common Interface • Simplified Interaction Between Apps • Flexible Deployment

Formal Property Verification App
- Protocol certification
- End-to-end packet integrity
- Asynchronous clocking effects
- Assertion-based verification
- Proofs for critical functionalities
- Debug isolation and fix validation

Connectivity Verification App
- Properties automatically extracted from table input
- Sub-system and Chip-level connectivity
- Conditional connection with latency

X-Propagation Verification App
- Automatic property generation
- Unexpected X detection and debugging

Control/Status Register Verification App
- Comprehensive
- Standard and proprietary protocols

Behavioral Property Synthesis
- Synthesis of multi-cycle, hand-shake, implication, black-box, and white-box properties from simulation
- Automated and manual property ranking and classification
- Feedback properties into formal or simulation environments
- VCD, FSDB and PLI support

Architectural Modeling App
- Pre-RTL modeling
- Capture executable spec
- Absence of deadlock
- Cache coherency verification
- Liveness and latency

Low Power Verification
- Creating power-aware formal model
- Automatic extraction of power-related functionality and sequencing properties
- Formal verification of power optimized SoCs

Design Coverage Verification
- Coverage metrics generation from formal verification
- Coverage metrics to establish quality of formal testbench and for bounded/full proof result
- Interacting with coverage metrics from simulation via an external DB (e.g. UCDB)

Sequential Equivalency Checking
- Sequential, temporal and functional equivalence verification
- Multi-value logic
- Full chip capacity

Security Path Verification
- Identify unintentional/illegal read access to secure data (leakage)
- Verify absence of illegal secure data overwrite (sanctity)
- Fault-tolerant security verification

Post-Silicon Debugging
- Failure signature matching
- Root cause isolation
- Candidate cause elimination
- Validation of fixes before re-spin

Other SoC-Related Applications
- iPXACT
- Design-based verification w/o testbench
- Glitch detection
- System-level deadlock

Higher Capacity
Verify complex 100M+ gate designs

Interactive Debug
Modify/create properties on the fly to explore design behavior

Increased Throughput
Utilize multiple proof engines on parallel compute resources

Wider Deployment
Proliferate across engineering teams with unique adoption model
Jasper’s Coverage-Driven Verification Strategy

*Provide solutions to accelerate the overall verification coverage closure process.*

1. Provide **coverage metrics for formal verification** to establish confidence in formal results, and eliminate redundant simulation tasks

2. Use **formal** to help identify holes and unreachables in your simulation (via UCIS/UCDB)
JG-COV Usage Models in Formal Flow

Stimuli Coverage

Formal Setup

DUT

How restrictive the design behavior under the formal setup? What dead code does it generate?

Property Completeness Coverage

Formal Setup

DUT

Properties

How complete my property set? Do I capture all design behaviors?

Proof Coverage

Formal Setup

DUT

Proven Properties

What coverage is achieved by the proven properties?

Bounded Proof Coverage

Formal Setup

DUT

Bounded Proof Properties

What is coverage of bounded proof? Is the bound enough? How to do better?
JG-COV Usage Model: Integration with Simulation Database

- **Testbench**: DUT
- **Simulator**: Write API, Read API
- **Coverage DB**: Write API, Read API
- **JG-COV**: Read API, Write API

Identify unreachable coverage targets: reduce the denominator
Provide coverage information from formal analysis: increase the numerator
Formal Coverage Model – Basic Definitions

- A **Cover Item** $C$ is an implicit or explicit functional event of RTL design $D$.

- A **Coverage Model** $M$ for design $D$ is a finite set $\text{CI}(D) = \{C_0, C_1, \ldots, C_n\}$ of Cover Items $C_i$.

- **Coverage Analysis** is a process to compute if a cover item $C_i$ is *triggered* in simulation or is *reachable* in formal analysis.

- The **Coverage** of design $D$ is a percentage of the triggered or reachable $C_i$ in $M$.
Stimuli Coverage

- **CI_R(E,D)** = \{C_i | C_i is **reachable** or covered in design D under formal environment E\}
  - Used to measure the coverage of design D

- **CI_U(E,D)** = \{C_i | C_i in CI where C_i is **unreachable** or covered in design D under formal environment E\}
  - Used for dead code analysis

- **CI_B(E,D)** = \{C_i | C_i in CI where C_i is **unresolved** or got a **bounded** proof in design D under formal environment E\}
  - Used for bounded proof debug and analysis

- **CI = CI_R(E,D) union CI_U(E,D) union CI_B(E,D)**
Property Completeness Coverage

- \( \text{COI}(P,E,D) = \{ C_i \mid C_i \text{ in CI that intersect with the cone of influence of property } P \text{ in design } D \text{ under the environment } E \} \).

- For a set of properties \( S = \{ P_0, P_1, \ldots, P_k \} \), we denote the \( \text{COI}(S) = \text{union of all } \text{COI}(P_i) \text{ Where } P_i \text{ is in } S \).
Property Completeness Coverage Illustrated

- Determine the cover items in the COI of each assertion
- Find the union of the assertion COIs
- The remaining out-of-COI cover items indicate holes in the property set at this hierarchical level

![Diagram showing coverage illustration]
Proof Core for Property:
\[ \text{state}[2] \implies \text{state}[3] \]
Verification Coverage Metric for Fully Proven Properties

- **PC(Pj):** set of $C_i$ in $CI(E,D)$ that intersects with the proof core of $P_j$
- **PC(Pj)** is a subset of $COI(Pj)$
- **PC(S):** union of all PC($P_j$)
- **PC_U(S), PC_R(S), PC_B(S):** Defined for unreachable, reachable, and bounded cover items that form the coverage set PC(S)
Measuring Property Completeness

RTL + Assertions + Constraints

Property Coverage Reporting
- Branch coverage
- Statement coverage
- Fault coverage

JasperGold w/Visualize™

Textual Report | GUI-based Report | Jasper DB

Bottom-line
Users can empirically determine the “sufficiency” of their “formal testbench”
Example: Coverage Measurement from Bounded Proofs

RTL

+ 

Assertions

+ 

Constraints

Bounded Proof Coverage Reporting

- Branch coverage
- Statement coverage
- Expression coverage
- Functional coverage

Design Coverage Verification

JasperGold w/Visualize™

Textual Report
GUI-based Report
Jasper DB

Bottom-line

Valuable verification progress info is available even if a property is not yet proven.
Case Study: D&V Group In UK

- **Background**
  - Customer D&V engineering group focused on ARM-based CPU peripherals
  - IPs include sensor controllers, clock & reset management blocks
  - Engineers were moderately familiar with ABV / formal

- **Process**
  - Used Coverage App to eliminate all dead code / work toward 100% code coverage goal
  - Used App to confirm there was no over-constraining
  - Leverage data from bounded proofs to increase coverage

- **Benefits**
  - Were able to quickly meet the project’s for coverage closure specification
  - Very high satisfaction with the results given the Coverage app / underlying formal analysis exhaustively exercised all functionality
In the Future…

- Jasper's formal technology will continue to expand to support larger designs, more complex problems, and more types of problems.
- Jasper will continue to push formal into spaces that were previously not conceived as applicable for formal.
- Users will continue to see more and more ROI from Jasper's formal solutions, creating a positive feedback loop of more usage, more users, and even higher ROI.
What This Will Mean to **You** Someday

- Formal will be your default choice for virtually every verification task
- The tools will have the capability to selectively apply the right heuristics for each situation, under-the-hood
- The tools will guide the user’s learning process, and will produce the needed metrics for management
- Engineering productivity will sky-rocket
Conclusions

- Formal technology has become an essential component for modern SoC design and verification
- Four types of coverage measurement in formal verification discussed
  - Measurement of stimuli coverage under constraints
  - Measurement of completeness of a property set
  - Measurement of verification coverage for bounded proofs
  - Measurement of verification coverage for full proofs
- Using Coverage DB API, formal tools can help coverage closure by:
  - Identifying unreachable coverage targets in a coverage DB
  - Providing coverage information from formal proofs in the coverage DB
“FORMAL WILL DOMINATE VERIFICATION”

Kathryn Kranen
October 22, 2013