



Formal-based Coverage-Driven Verification

Sia Karthik Madabhushi | May 15, 2014

Preface



- In the future formal apps & methodologies will be the default for verification
- Consequently, it's critical for formal verification practitioners to have metrics to measure their progress independent of simulation-related data.
- Hence, this presentation will focus on formal-specific coverage

Coverage Closure Challenges

- *Is my property set complete or are there gaps?*
- *Are my constraints allowing ALL possible legal stimulus into the design?*
- *How do I measure the extent of design verification attributable to partially or completely proven properties?*
- *How do I show the contribution formal is making to the overall verification effort?*
- *Can I use formal to reduce or even eliminate simulations?*

Jasper's Coverage-Driven Verification Strategy

Provide solutions to accelerate the overall verification coverage closure process.

1. Use formal to help identify holes and unreachables in your simulation (via UCIS/UCDB or vendor-specific interfaces)
[Make formal relevant to your sim-centric colleagues]

Today's focus

2. Provide coverage metrics for formal verification to establish confidence in formal results, and eliminate redundant simulation tasks
[Enable you & management to see your progress & success]

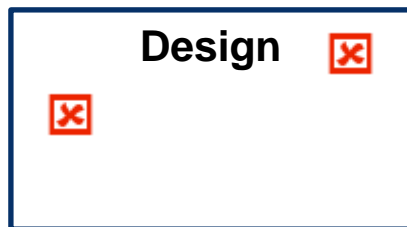
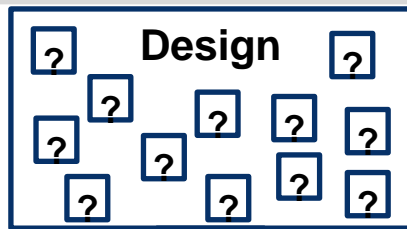
Formal-Specific Coverage Metrics

- Measuring completeness of a formal testbench
 - Stimuli coverage: completeness of stimuli applied to the design under the given set of constraints
 - Property completeness: Completeness of property set applied to the DUT
- Measuring verification coverage after formal analysis
 - Proof coverage: coverage for properties fully proven
 - Bounded proof coverage: coverage for properties with bounded proofs

Benefits

- **Protect against the potential over-constraint problem to eliminate false confidence in design correctness**
- **Provides an empirical measurement of the ROI of your formal verification**

Stimuli Coverage Illustrated



Dead Code

- 1) Designer Investigates
- 2) Fix, or waive from coverage

- Engines attempt to hit all cover items
- Formally proves items that are unreachable

App further categorizes unreachable items:

- Dead code: impossible for any stimulus to hit
- Environment overconstrains possible stimulus

or

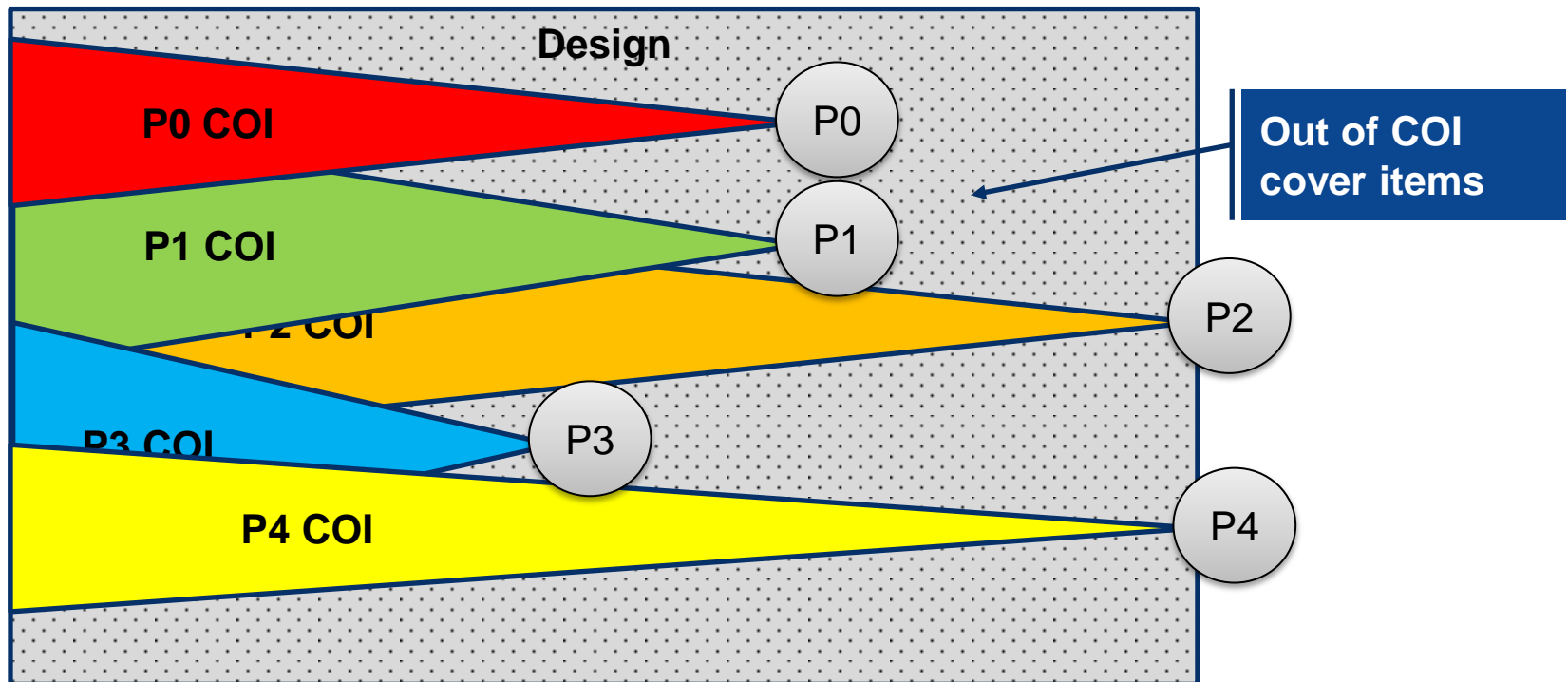


Overconstrained Stimulus

- 1) Overconstraint identified
- 2) Fix, or waive from coverage report

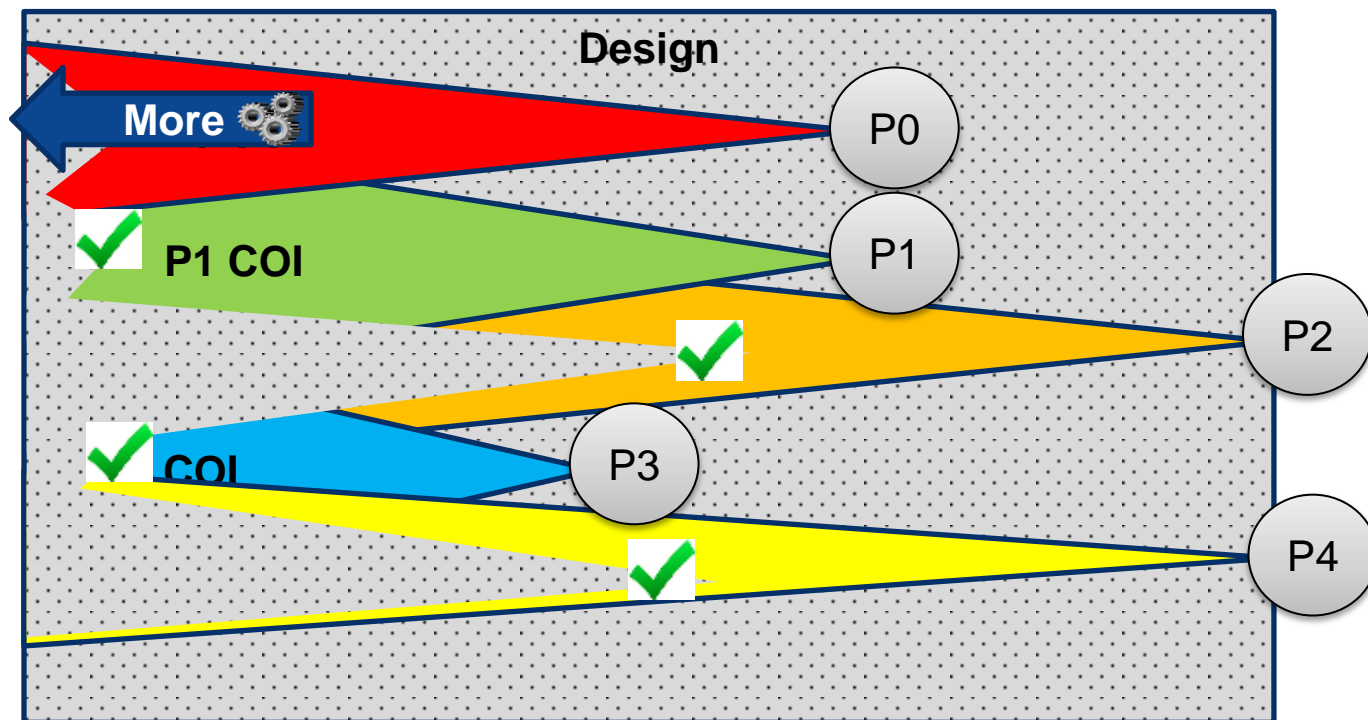
Property Completeness Analysis

- 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 assertion set at this hierarchical level



Property Completeness Progress Measured By Size of “Proof Core”

- Proof Core of an assertion: Subset of the logic contained in the COI of the property that is capable of establishing the correctness of the assertions
- **Key metric for showing formal verification progress**



Summary:

Full Proofs And Coverage Results

Key Task

Identify which cover points within the proof core of the COI have been covered

Desired Results

A full proof result implies that entire reachable state-space is traversed, and no violation of the assertion is encountered.

What can I do when my formal analysis is not converging?

Option 1



Option 2



Option 3

Use coverage from
“Bounded Proofs”!

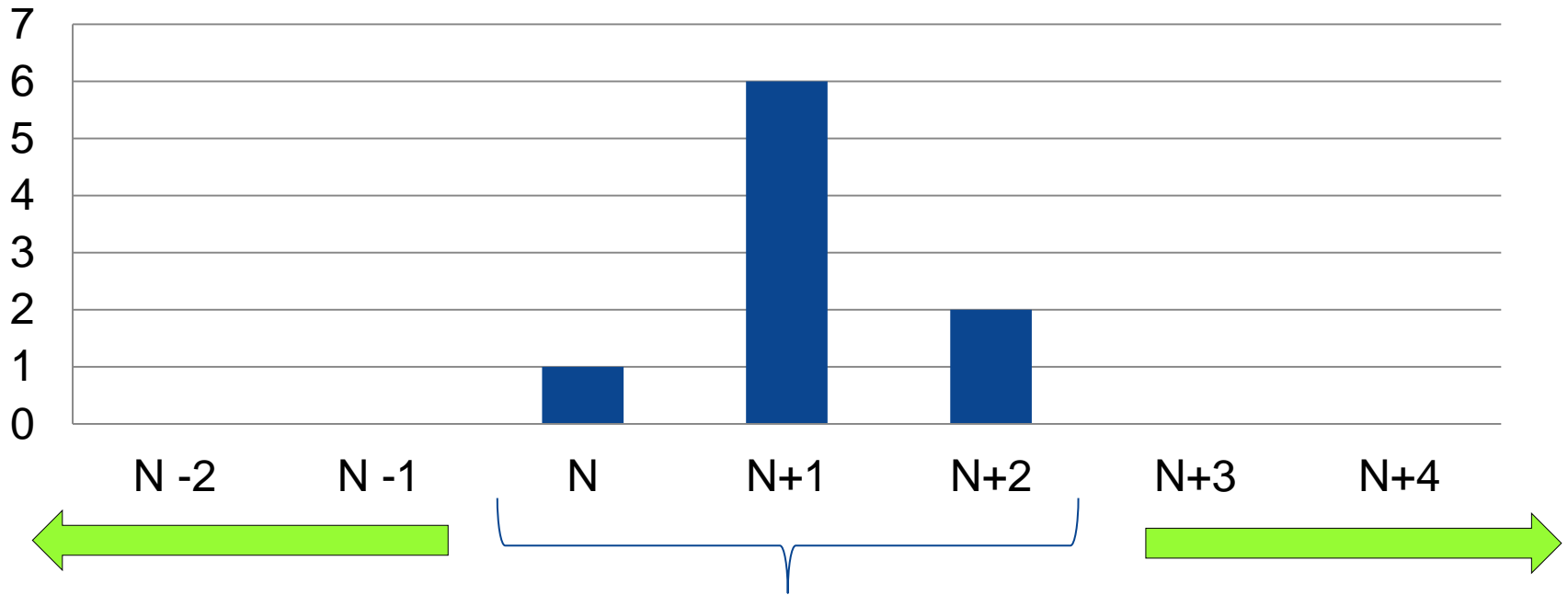
Bounded Proof Definition

- A bounded proof result implies that only a subset of the reachable state-space is traversed, and no violation of the assertion is encountered in that subset
- Bounded proof of “N” cycles == all states reachable within “N” cycles from the design’s reset state have been analyzed
- This implies that all events possible within “N” cycles from the reset state have been triggered

Using Bounded Proofs

i.e. Getting value from incomplete proofs

CEX's per Cycle



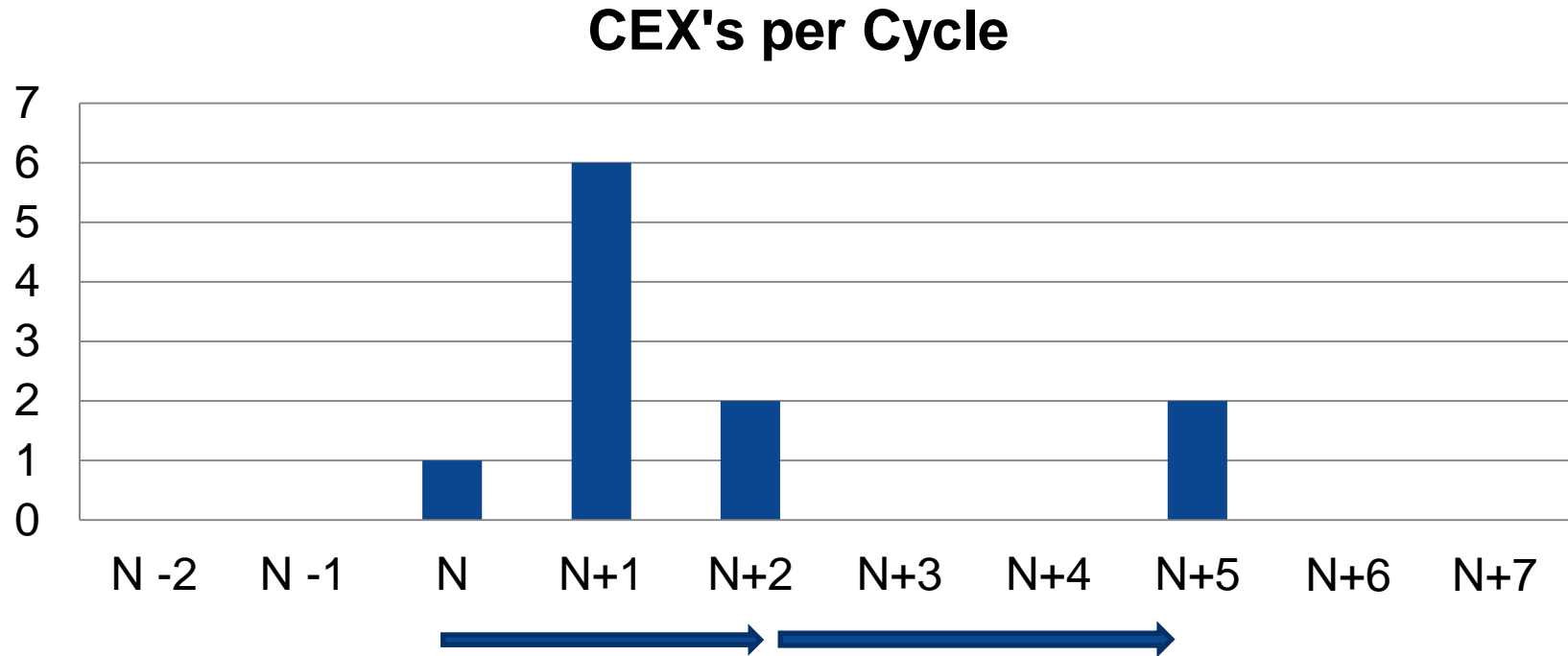
**Good news:
No failures
before "N"**

Guesstimating "N"

- **Min. cycles from input to output**
- **Counter size**
- **State machines**
- **FIFO depths, etc.**

**No more CEX's →
Suggests all clear**

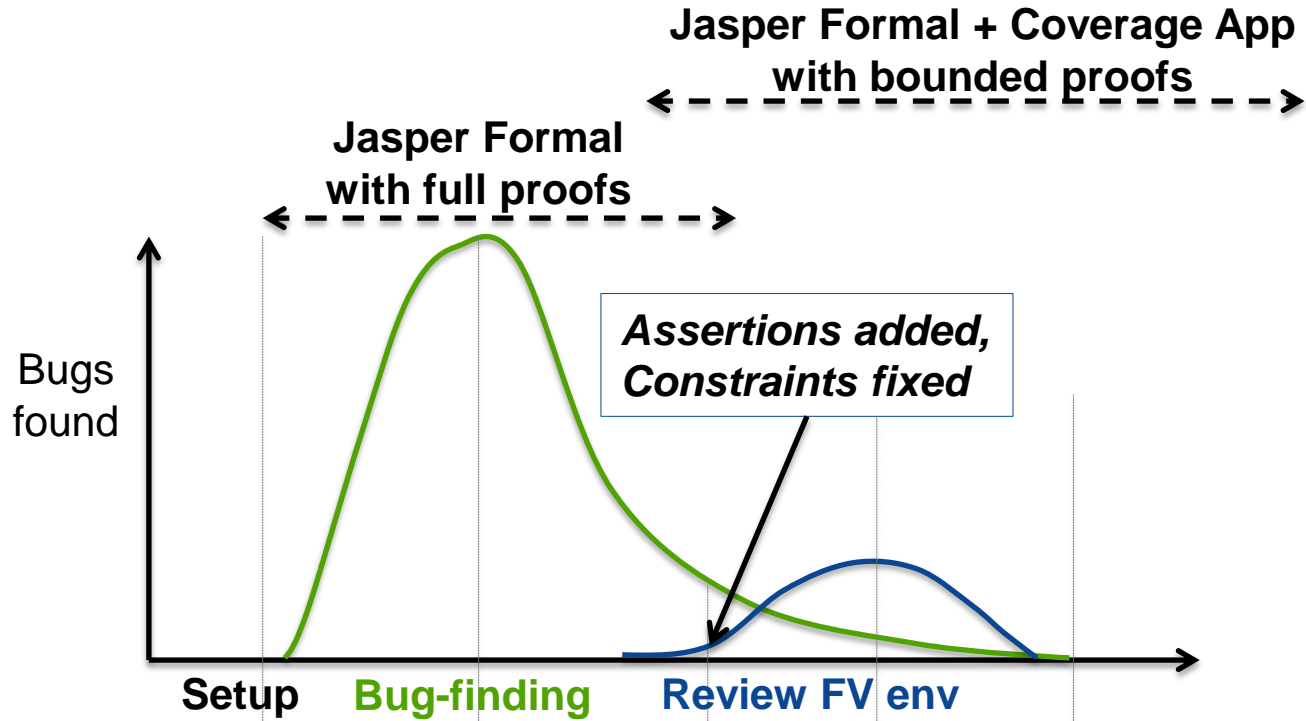
Potential Gotcha: Modal Behavior



Run a little longer to confirm there are no:

- Jumps in FSM states
- Latencies across data packet boundaries
- Different modes of operation

Summary: Overall Proof+COV Strategy



Proven:	✓ 0	✓ 10	✓ 20	✓ 50	✓ 77
Failures:	✗ 0	✗ 45	✗ 40	✗ 25	✗ 0
Undetermined:	✗ 0	✗ 9	✗ 7	✗ 5	✗ 3

Customer Example: APM at JUG 2013, “A Conceptual Toolkit for Formal Verification”

**Productivity –
Jasper’s Coverage app
unlocks value from
bounded proofs**

**Quality –
dramatically
higher coverage**

How the COV app helps

- COV app quantifies the significance of bounded proofs, telling you exactly what logic is covered
- Some preliminary results: on our MMU design, coverage increased from 19% to 99% when bounded proofs were included
- If these results are valid, bounded proofs are providing a 80% improvement in formal coverage!
- Bounded proofs on a block can be more important than the full proofs

Summary: Formal Coverage-Driven Verification

- Key capabilities include:
 - Protection against over-constraints
 - Measurement of assertion set completeness
 - Support for bounded and full proofs
 - Easy, persistent exclusion/waiver of illegal areas
 - Interaction with coverage data from simulation via UCDB or other APIs
- Benefits
 - Increased quality & confidence from exhaustive formal analysis
 - Rapidly identify code and functional coverage holes
 - Reach (formal and simulation) coverage closure faster
 - Reduce the number of simulations

Copyright Notice and Proprietary Information

Published: 16 May 2014

Copyright ©2013 Jasper Design Automation, Inc. All rights reserved. This software and document (“Materials”) are owned by Jasper Design Automation, Inc., or Jasper has the appropriate licenses to provide them, and may be used only as authorized in the license agreement controlling such use. No part of these Materials may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Jasper Design Automation, or as expressly provided by the license agreement.

These Materials are for information and instruction purposes. Jasper Design Automation reserves the right to make changes in specifications and other information contained in these materials without prior notice, and the reader should, in all cases, consult Jasper Design Automation to determine whether any changes have been made.

Disclaimer

JASPER DESIGN AUTOMATION, INC. DISCLAIMS AND MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, WITH REGARD TO THESE MATERIALS, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

IN NO EVENT SHALL JASPER DESIGN AUTOMATION, INC. BE LIABLE FOR ANY DIRECT, INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THESE MATERIALS OR THE INFORMATION CONTAINED IN THEM, HOWEVER CAUSED AND WHETHER BASED IN CONTRACT, TORT (INCLUDING NEGLIGENCE) OR ANY OTHER THEORY OF LIABILITY, EVEN IF JASPER DESIGN AUTOMATION, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Jasper Design Automation, the Jasper Design Automation logo, JasperGold, and ProofGrid are trademarks of Jasper Design Automation, Inc. All other trademarks or registered trademarks are the property of their respective owners.

707 California Street
Mountain View, CA 94041
Tel: (650) 966-0200
Fax: (650) 625-9840
<http://www.jasper-da.com>