Formal Verification Adoption Made Easy

JasperGold Apps and Design RTL-Bring Up

Alexandre Botelho – Application Engineer

September 2021

cadence®

Agenda

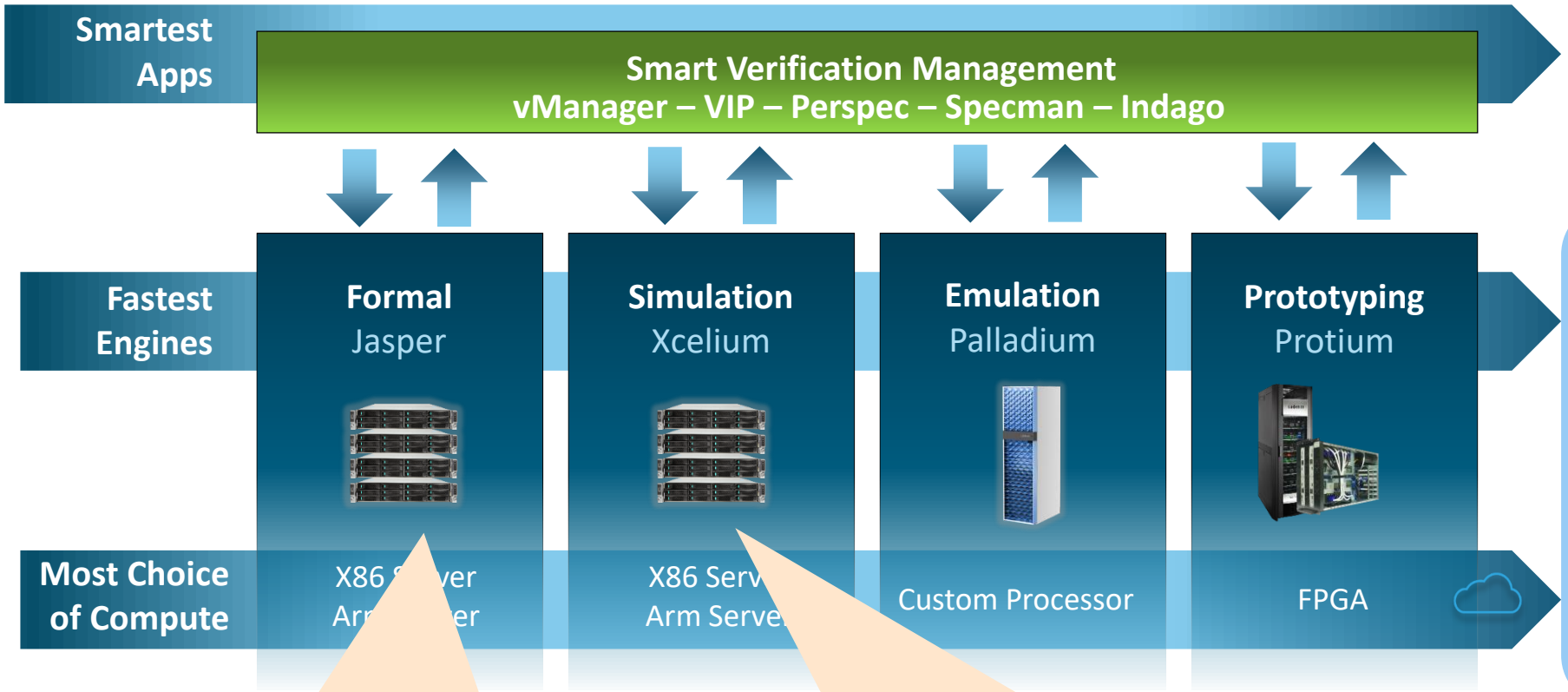
- Verification Throughput and JasperGold® Apps
- When is your design ready to handoff to verification?
- RTL Design Bring-Up Challenges
- An advanced design bring-up methodology using JasperGold® Platform
- Summary
- Q&A

Agenda

- Verification Throughput and JasperGold® Apps
- When is your design ready to handoff to verification?
- RTL Design Bring-Up Challenges
- An advanced design bring-up methodology using JasperGold® Platform
- Summary
- Q&A

Formal: A Key Component for Verification Throughput

Find and fix the most bugs per $ invested in bare metal compute



*Huge progress in model checking **scalability** and **performance** in last 10+ years*

*Formal verification now a **mainstream** tool in the **verification toolkit***

- Mathematically proves properties
- Automatically exercises all input combinations without a testbench
- Helps verification to “shift left”
- Historical limitation: scalability beyond block/IP level

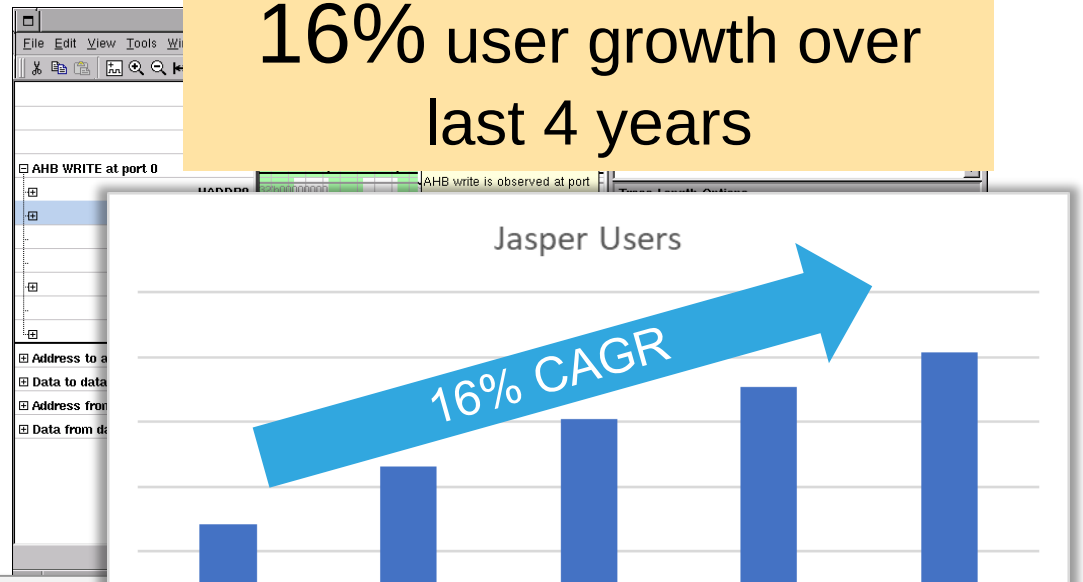
- Tests based on dynamic stimulus and checkers
- Testbench quality controls verification effectiveness
- Some directed tests early, full testbench later
- Scales well to block/IP and integration tests

JasperGold: Easiest Formal Verification to Adopt

Solve **specific verification problems** with targeted JasperGold® Apps

- Formal Property Verification App
- SuperLint (AFL) App
- Design Coverage Verification App
- Sequential Equivalence Checking App
- X-Propagation Verification App
- Control/Status Register Verif. App
- Connectivity Verification App
- Coverage Unreachability App
- Clock Domain Crossing App
- Functional Safety Verification App
- Low Power Verification App
- Security Path Verification App

Highly interactive **formal debug** transforms to fit the App



16% user growth over last 4 years

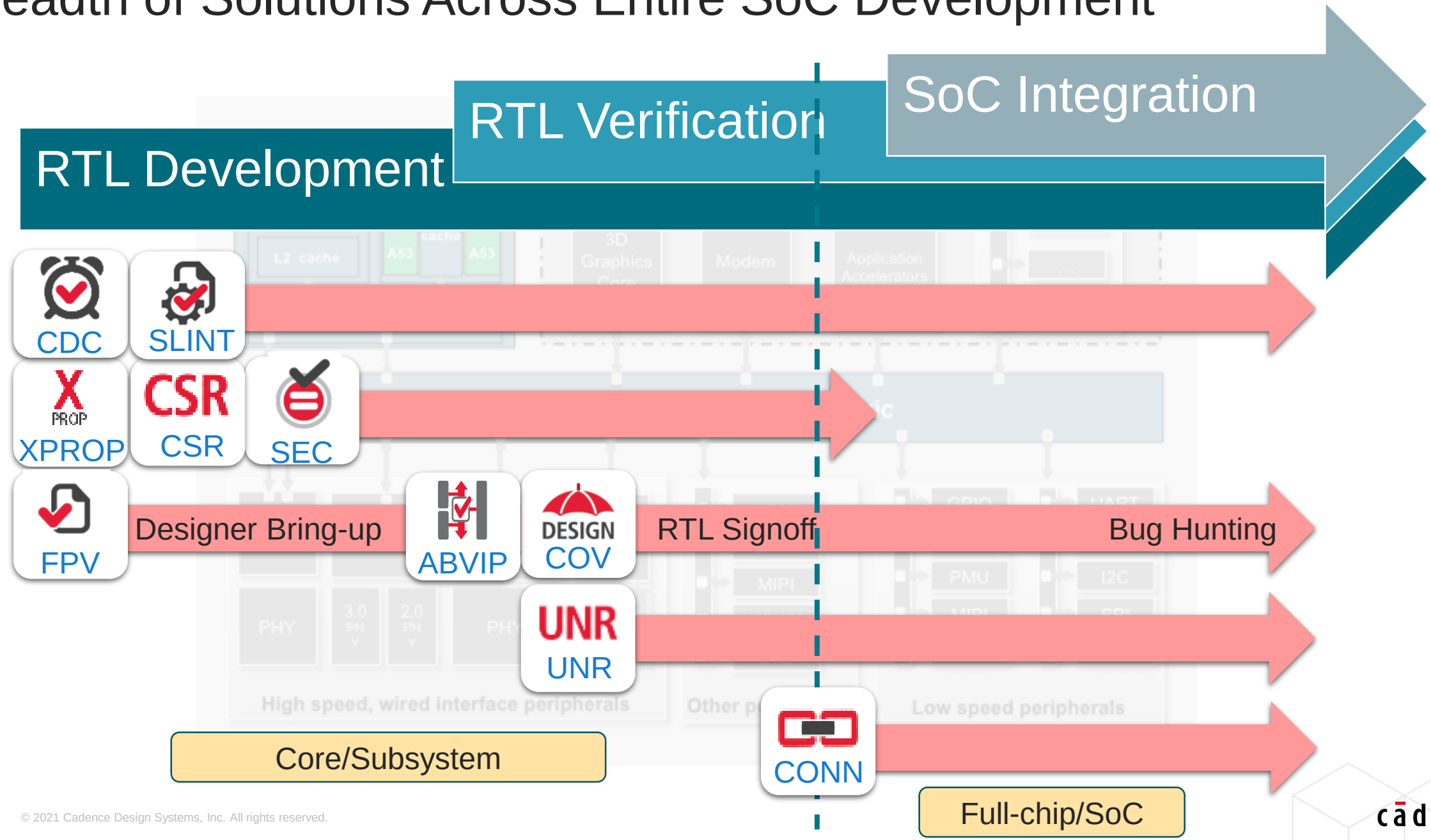
Broad **formal engine** and inf

Assertion Based Verification IPs for AMBA and o

Programmable Interface via TCL

ProofGrid™ Manager assigns best engine for task

Breadth of Solutions Across Entire SoC Development



Agenda

- Verification Throughput and JasperGold® Apps
- When is your design ready to handoff to verification?
- RTL Design Bring-Up Challenges
- An advanced design bring-up methodology using JasperGold® Platform
- Summary
- Q&A

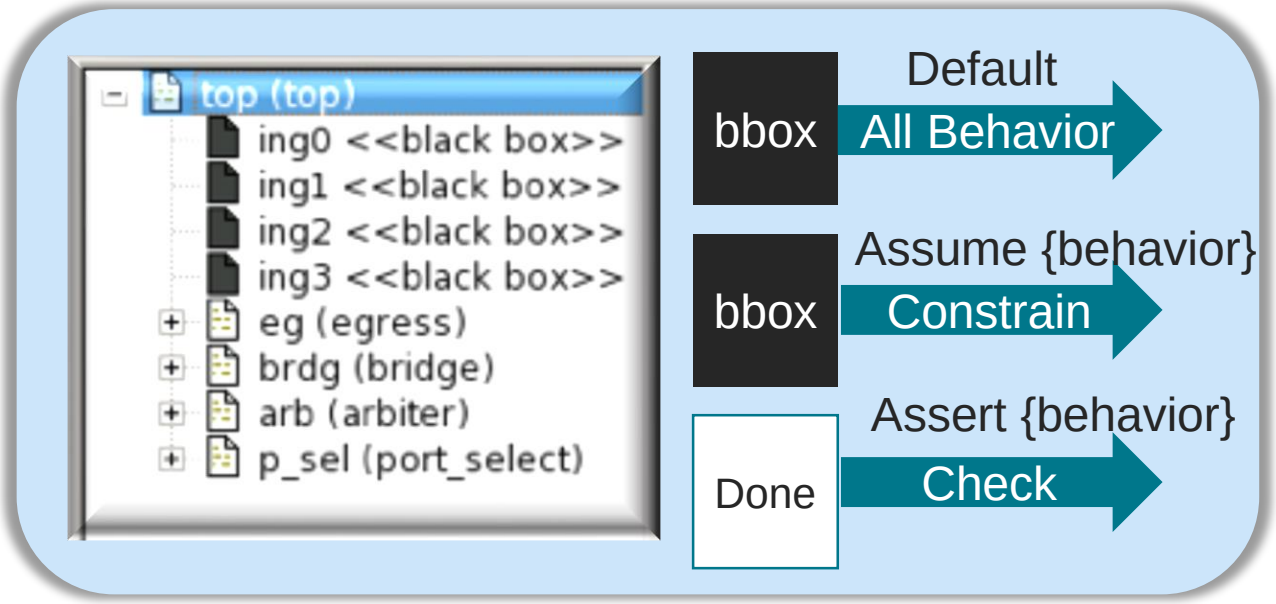
When is your design ready to hand-off to verification?

- Linting passes a predetermined baseline
  - No basic structural problems
- Initialization looks correct
  - No unexpected X behavior coming out of reset
- Each line of RTL is reachable (or waived)
  - No surprises later with system-level coverage
- Interfaces are clean…no protocol violations
  - Our DUT communicates correctly with other blocks
- All specific configuration modes are covered
  - Our DUT can be configured into desired modes of operation
- Critical good behaviors can be covered
- No failing Generic Functional Behaviors
  - Checks for fundamental behaviors (Ex. overflow, deadlock) are not failing
- Waivers have been saved off
  - Our waivers allow us to track what we've already looked at
- User Interface Properties have been exported
  - Our interface properties will be double-checked at the system level

Agenda

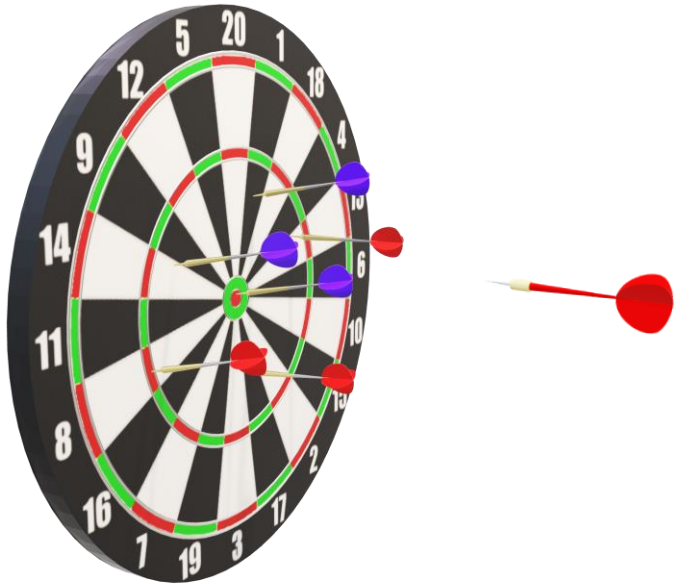
- Verification Throughput and JasperGold® Apps
- When is your design ready to handoff to verification?
- RTL Design Bring-Up Challenges
- An advanced design bring-up methodology using JasperGold® Platform
- Summary
- Q&A

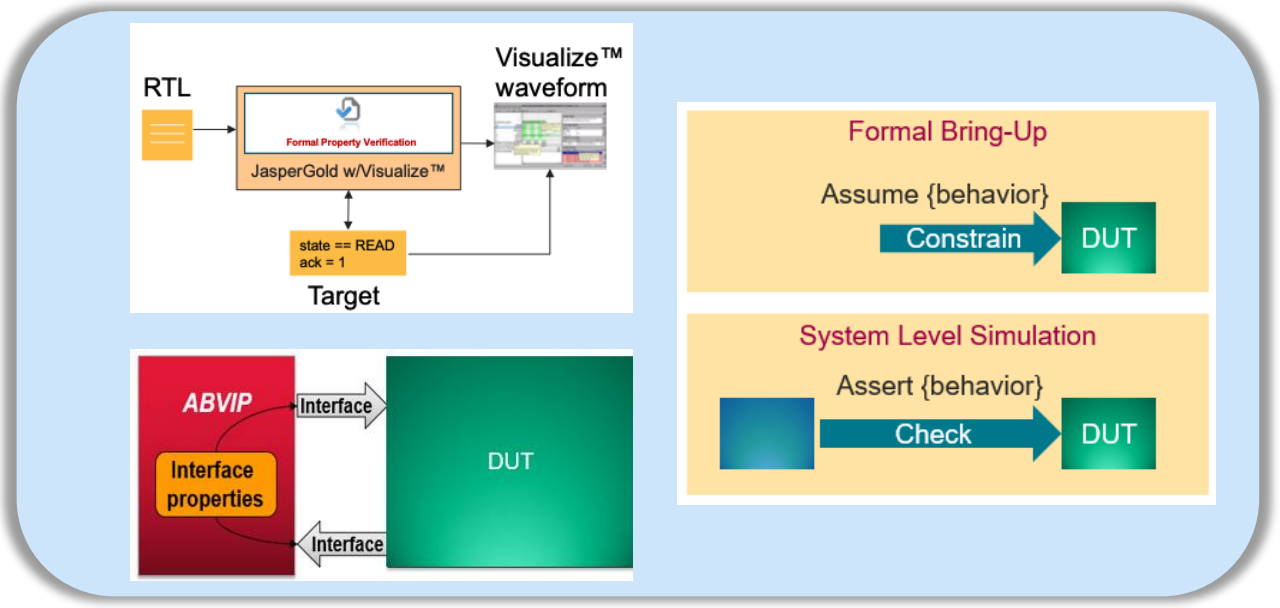
Bring-Up Challenge #1: Missing Pieces



- Traditional Bring-Up
  - You may not have all of the RTL completed, or it doesn't compile yet
  - Modeling these low level instances takes a lot of time and is prone to error
  - This modeling is often throw-away work and usually not verified in any way
- Formal Bring-Up
  - Missing modules can be black-boxed. By default, blackbox outputs allow all possible output behaviors
  - Optional assumptions can be added to blackbox outputs to constrain behavior
  - When the instance becomes available, the Assumptions can be changed to Assertions to check the behavior

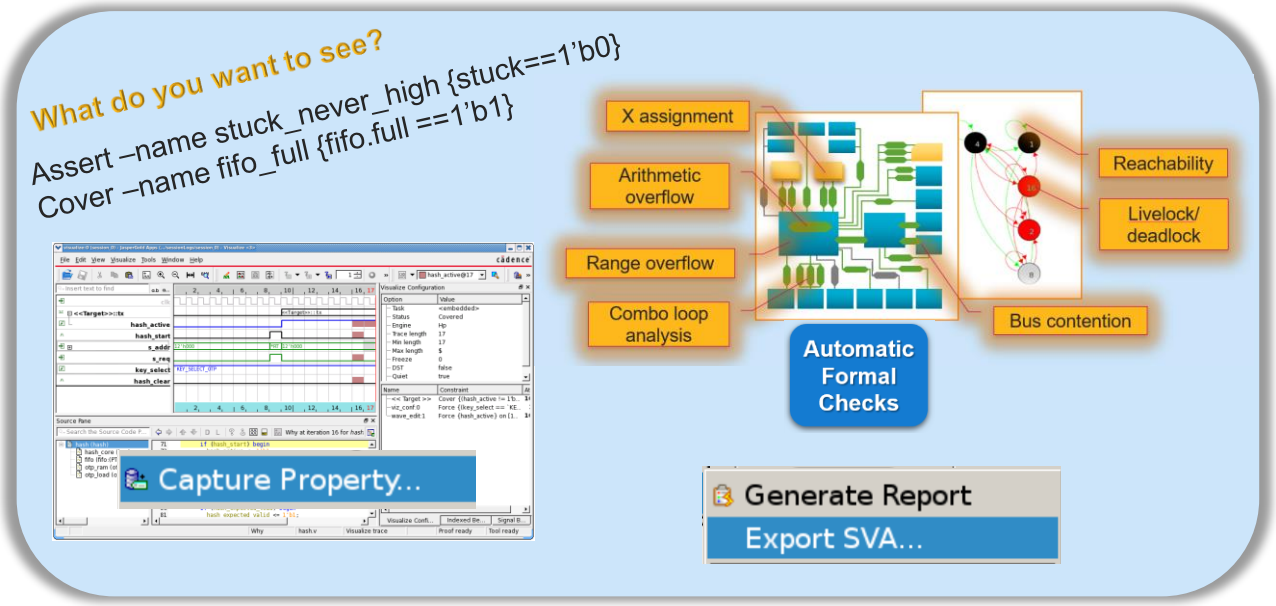
Bring-Up Challenge #2: Stimulus





- Traditional Bring-Up
  - You spend lots of time trying to inject the correct stimulus for each test
  - It is time consuming to generate stimulus that is protocol-compliant
  - Stimulus used at the block level is never “double-checked” at the system level
- Formal Bring-up
  - Choose the target behavior you want to see and formal creates the stimulus for you
  - Leverage Protocol-compliant constraints & checks that may be available for common interfaces (Ex. Cadence Assertion-Based Verification IP)
  - Any SVA you use to constrain your DUT can be verified at the system-level, and can also check the behavior of the upstream block!

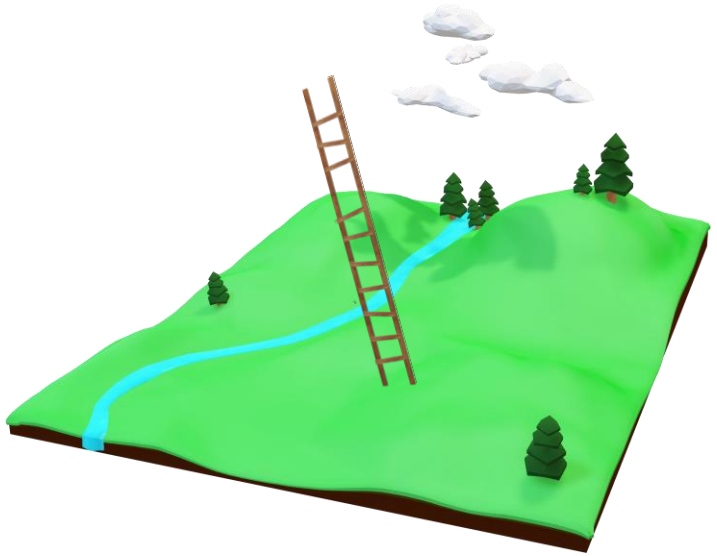
Bring-Up Challenge #3: Checkers

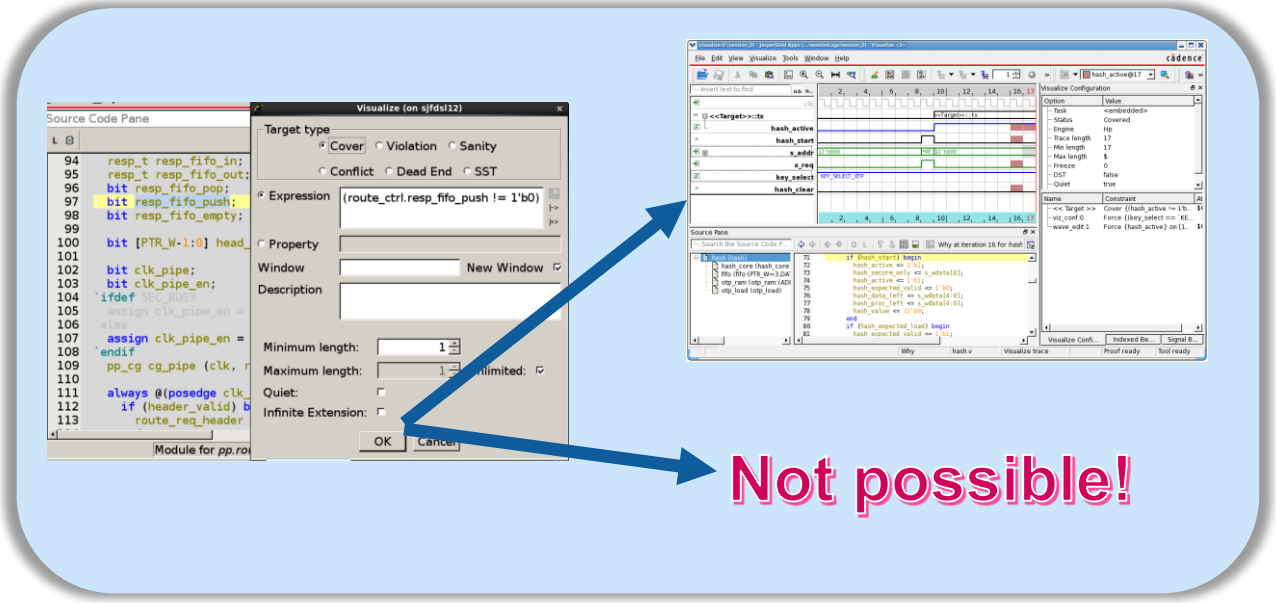


- Traditional Bring-Up
  - You spend lots of time developing checkers and scoreboards
  - Each check you perform relies on specific stimulus
  - Checkers you create usually provide little value to other RTL developers or the verification team

- Formal Bring-up
  - No need to worry about the stimulus required to exercise the check. Focus on the end behavior and formal shows you the stimulus required
  - You may be able to capture checker properties directly from interesting waveforms (Ex. JasperGold® Visualize™)
  - Leverage Automatic Formal checks for generic behavior such as arithmetic overflow, deadlocks and more (Ex. JasperGold® Superlint App)
  - All checks can be used by the verification team (simulation or formal) and by other designers

Bring-Up Challenge #4: Dead Code & Unreachable Behaviors

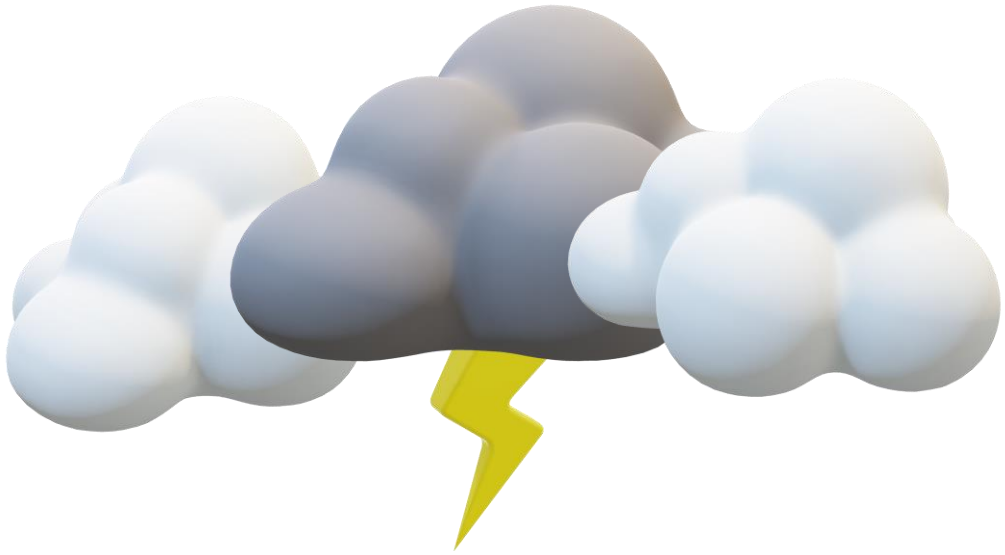


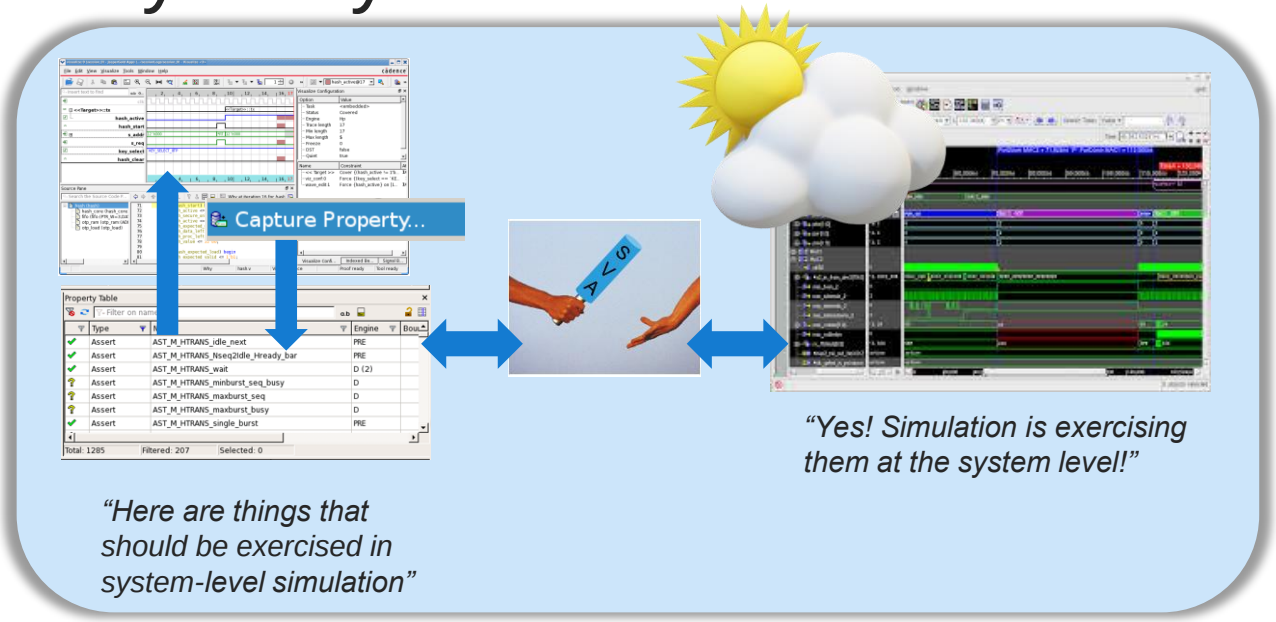


- Traditional Bring-Up
  - You may spend lots of time trying to exercise code and behaviors that are actually unreachable
  - Some unreachable behaviors are expected, but you have no way to confirm this with simulation

- Formal Bring-up
  - If you ask formal to demonstrate (cover) a behavior, it will either show you a waveform or say it is impossible
  - Good behaviors that are not possible could indicate an RTL bug
  - You may also be able to leverage automatic formal Deadcode checks (Ex. JasperGold® Superlint App)

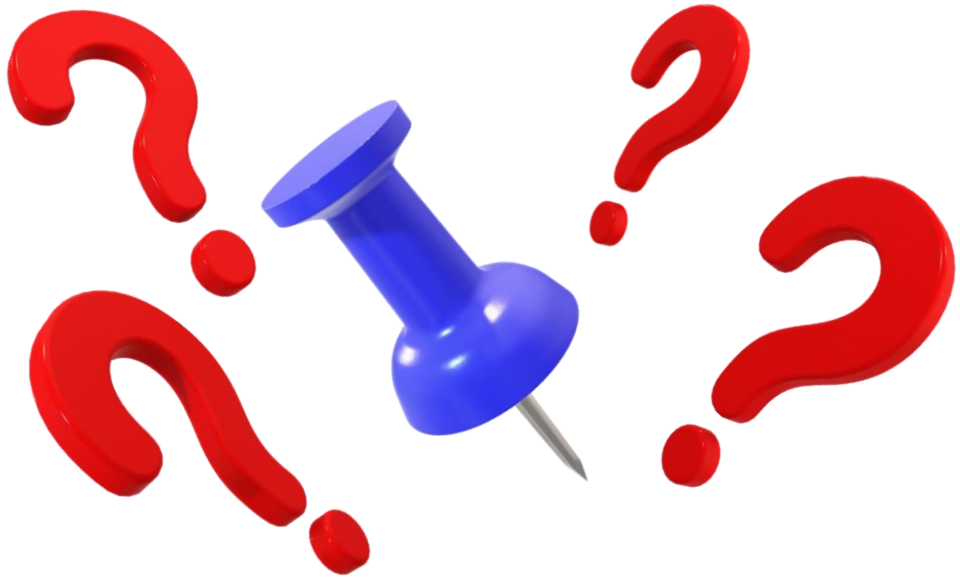
Bring-Up Challenge #5: Poor Visibility At System-Level

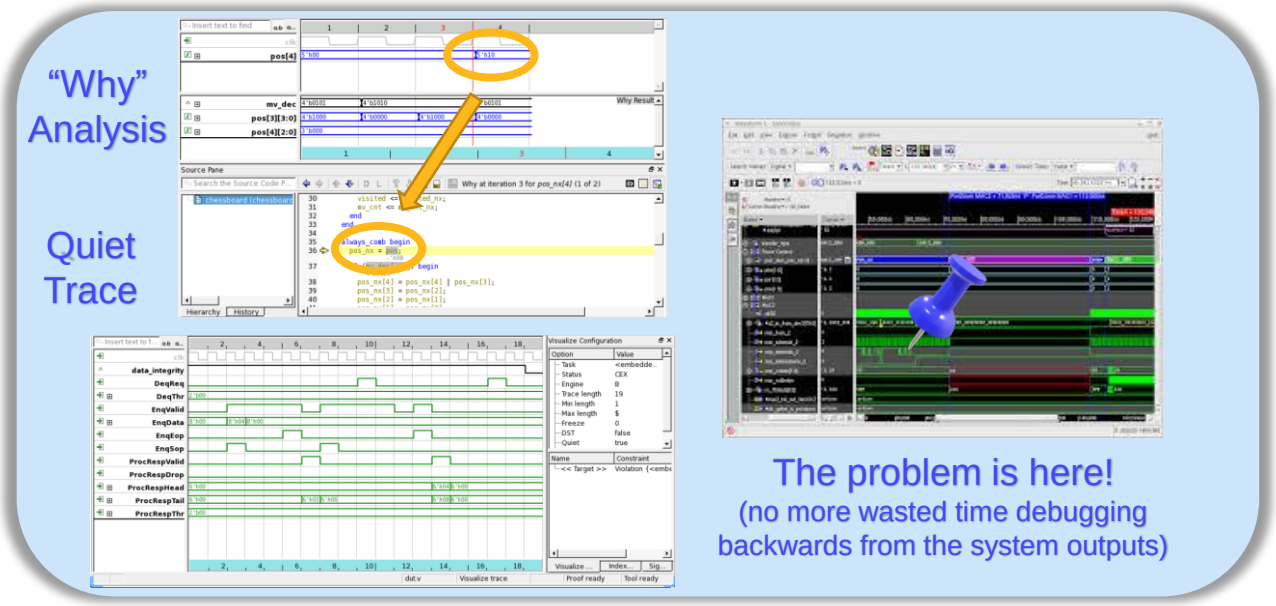




- Traditional Bring-Up
  - Once you send your RTL "over the wall", is the verification team exercising things you are concerned about?
  - Bug escapes often occur where the system- level tests are not testing what you need them to
- Formal Bring-up
  - Behaviors that you visualize during design bring-up are written and captured in the form of SVA properties
  - SVA Properties can be handed off to the verification team to give you confidence that these behaviors are also being exercised at the system-level

Bring-Up Challenge #6: Pinpointing The Root Cause





- Traditional Bring-Up
  - A unit-level testbench produces failing traces that are much longer than formal traces
  - When a failure happens at the system-level, root-cause analysis takes much longer
    - Which block failed?
    - Where is the failure inside the block?

- Formal Bring-up
  - During bring-up, formal finds the shortest waveform that shows the behavior, minimizing debug time
  - Take advantage of formal tool capabilities you may have to quickly root-cause issues (Ex. JasperGold® Visualize "Why" and "QuietTrace")
  - Exported SVA properties that you run in system-level simulations pinpoint the exact time and location of the behavior

Agenda

- Verification Throughput and JasperGold® Apps
- When is your design ready to handoff to verification?
- RTL Design Bring-Up Challenges
- An advanced design bring-up methodology using JasperGold® Platform
- Summary
- Q&A

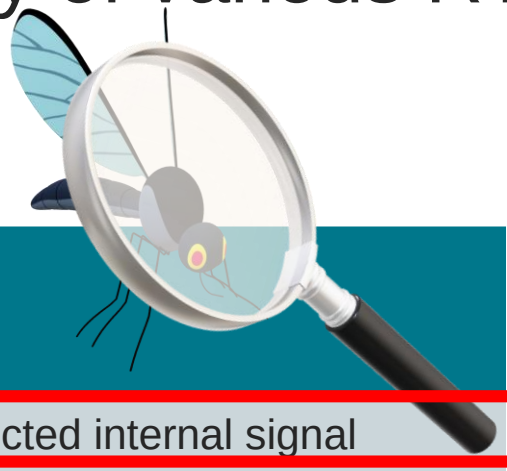
Anatomy of various RTL bugs

Bug Type	Detectable by structural checks?	Visible while exercising expected behavior?	Would violate generic behavioral checks?	Would violate protocol at external ports?	Would violate function-specific Assertions?

Likelihood of detecting the bug using this method

LOW | MEDIUM | HIGH

Anatomy of various RTL bugs



	Detectable by structural checks?	Visible while exercising expected behavior?	Would violate generic behavioral checks?	Would violate protocol at external ports?	Would violate function-specific Assertions?
Unconnected internal signal	HIGH	MEDIUM	HIGH	MEDIUM	HIGH
Latched (stuck) signal	LOW	MEDIUM	HIGH	MEDIUM	HIGH
FIFO push when full	LOW	MEDIUM	MEDIUM	HIGH	MEDIUM
Performance bug	LOW	HIGH	LOW	LOW	MEDIUM
Register missing a reset assignment	HIGH	MEDIUM	MEDIUM	MEDIUM	HIGH
Out-of-bound indexing	LOW	LOW	HIGH	LOW	MEDIUM
Corner-case protocol violation	LOW	LOW	LOW	HIGH	LOW
Unreachable case	MEDIUM	MEDIUM	HIGH	LOW	MEDIUM
FSM deadlock	LOW	LOW	HIGH	LOW	MEDIUM
Function-specific bug	LOW	MEDIUM	LOW	LOW	HIGH

Design Exploration + Bring-up

Compile Design & Structural Inspection

Linting Clean with Waivers

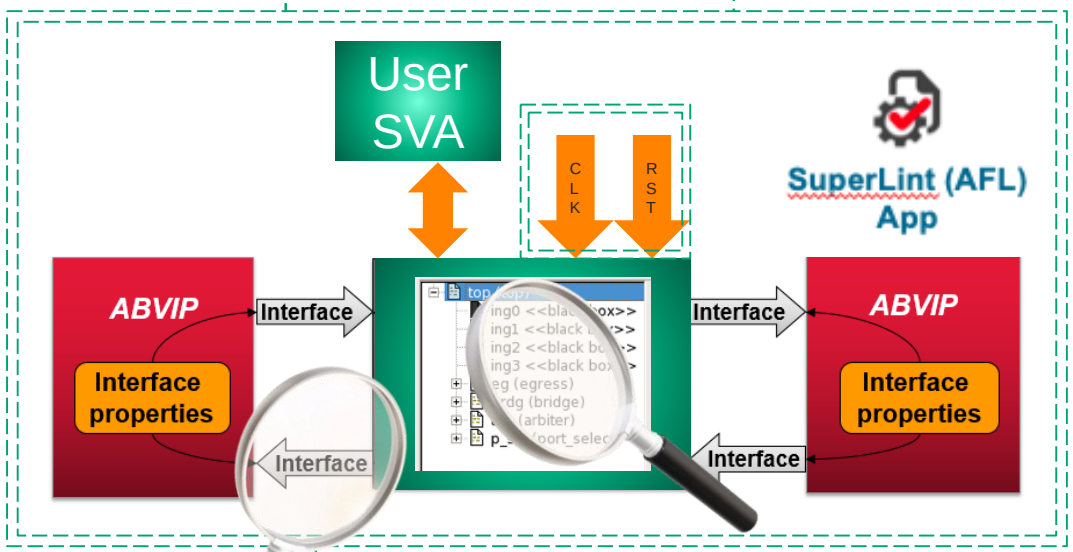
Superlint DeadCode Clean with Waivers

Write properties for interesting behaviors

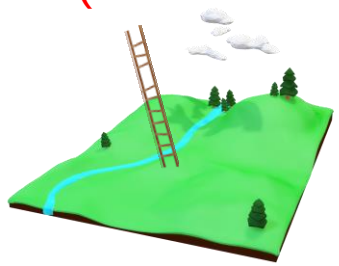
Let Jasper generate stimulus for interesting behaviors

More behaviors to explore?

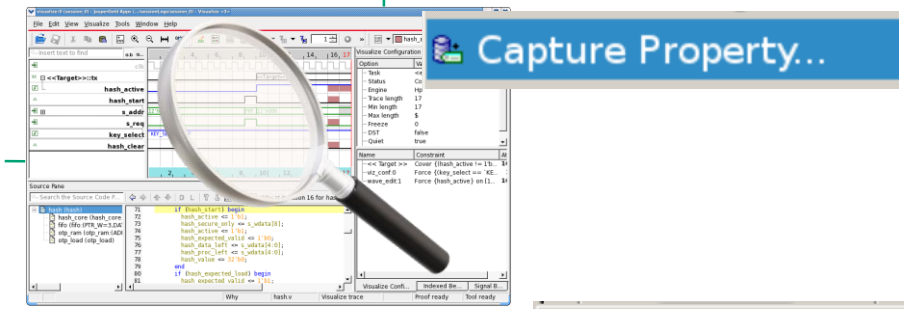
Yes!



Behavior is unreachable (Possible bug!)



- ✓ Plot additional cycles
- ✓ Make changes to trace as you go
- ✓ View all signals



Debug Interface failures

Debug Generic Behaviors w/ Superlint

Debug Design Specific Behaviors

Save off Waivers

Export properties for Simulation

Design Exploration + Bring-up

Compile Design & Structural Inspection

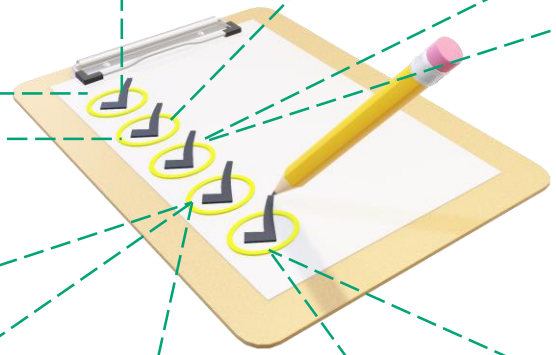
Linting Clean with Waivers

Superlint DeadCode Clean with Waivers

Write properties for interesting behaviors

Let Jasper generate stimulus for interesting behaviors

More behaviors to explore?



Debug Interface failures

Debug Generic Behaviors w/ Superlint

Debug Design Specific Behaviors

Save off Waivers

Export properties for Simulation

Handoff!

Agenda

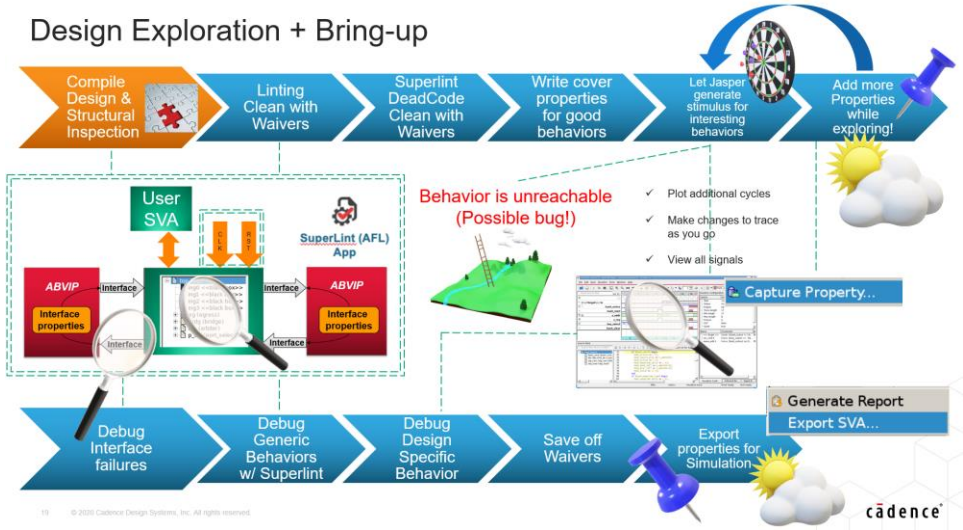
- Verification Throughput and JasperGold® Apps
- When is your design ready to handoff to verification?
- RTL Design Bring-Up Challenges
- An advanced design bring-up methodology using JasperGold® Platform
- Summary
- Q&A

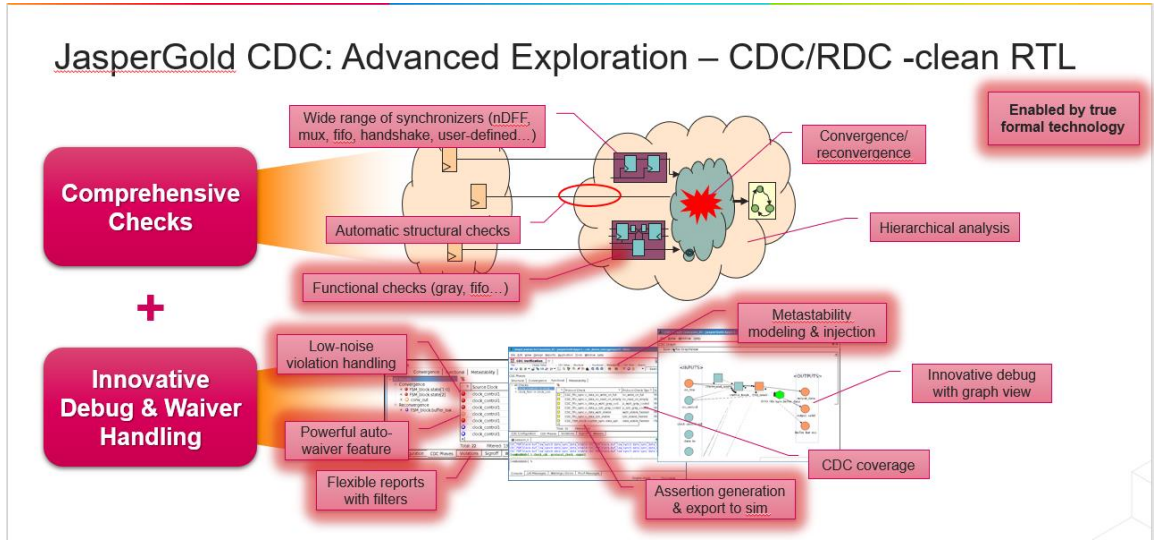
Summary

- Designers are faced with many challenges when trying to bring-up their designs
- Using a solid methodology combined with cutting-edge formal capabilities, designers can perform design bring-up and address each of these challenges
- By going through each step of the flow, designers can be confident they have addressed each criteria required for handing their RTL over to the verification team
- This design bring-up methodology can replace unit-level simulation while also providing team-friendly capabilities through the use of portable properties

What's next?

The most comprehensive flow for Designers!





cadence

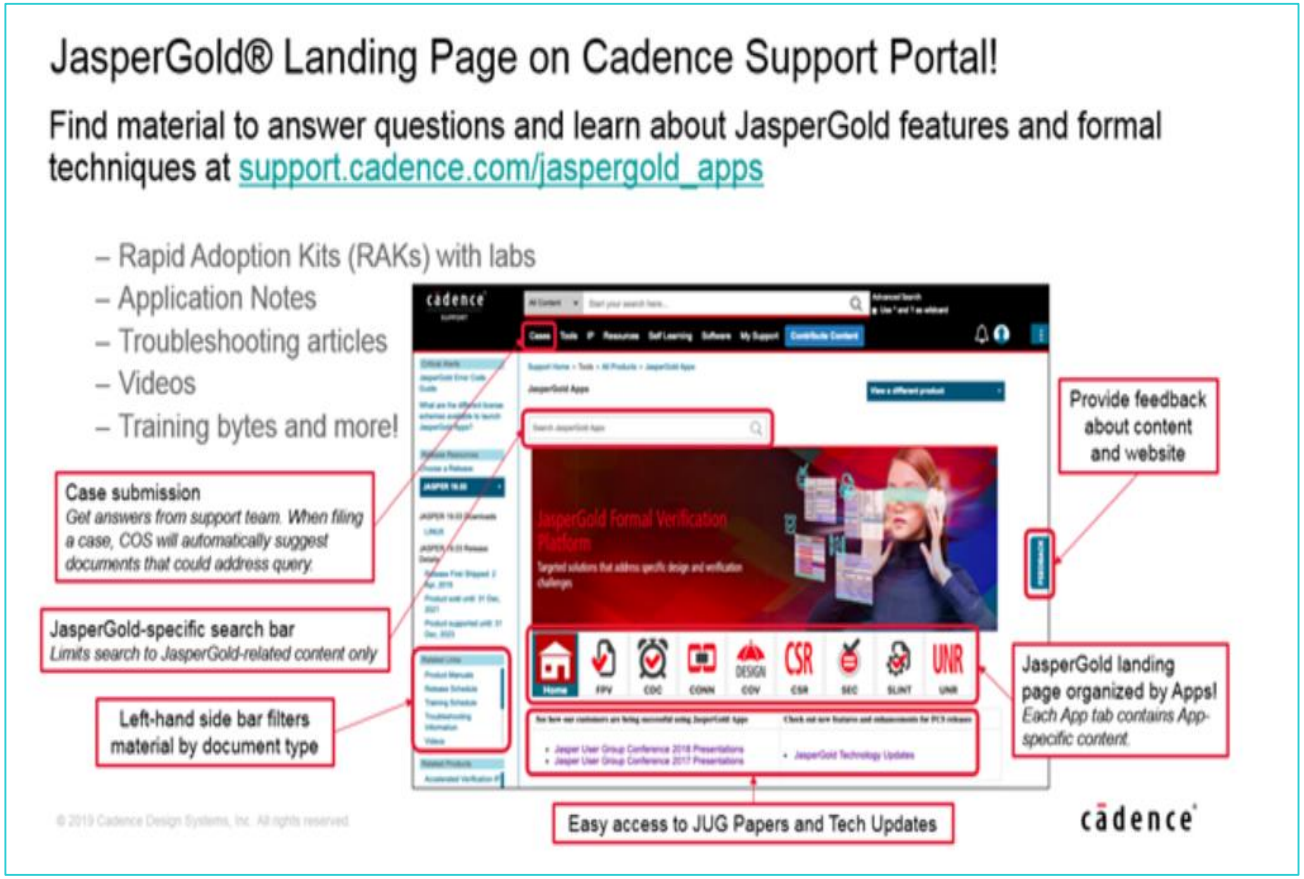
Agenda

- Verification Throughput and JasperGold® Apps
- When is your design ready to handoff to verification?
- RTL Design Bring-Up Challenges
- An advanced design bring-up methodology using JasperGold® Platform
- Summary
- Q&A

Questions?

- **My Contact Info**
  - Alexandre Botelho, Application Engineer
  - alexandre@cadence.com

- **Cadence Online Support**
  - More information about JasperGold Formal Property Verification, JasperGold Superlint and much more!
  - Rapid Adoption Kits (RAK's)
    - JasperGold RTL Design Bring-Up
    - JasperGold Superlint App



New! The Blended/Virtual Training Solution

Combining the best of all our training methods!

- **Live lectures** conducted by Cadence's expert instructors
- **Flexibility**—attend the training wherever you are
- **Self-study** lab work with instructor support
- **Interactive** live Q&A sessions and much more

Visit the **Cadence Training** website:
www.cadence.com/training

Cadence **Learning and Support** System

The one stop shop for all your learning and support needs!

- **Speed your** technical learning!
- **All online training** is **FREE** for Cadence customers with a Support account
- **Access** to hundreds of online courses available through https://support.cadence.com

You might be interested in this online course:

SVA, Formal & JasperGold® Fundamentals for Designers

NEW

Questions?
Reach out to us at eur_training@cadence.com