Verifying Microprocessor Debug-bus Connectivity Formally using VC Formal

Vinayak Kamath

Advanced Micro Devices, Inc.

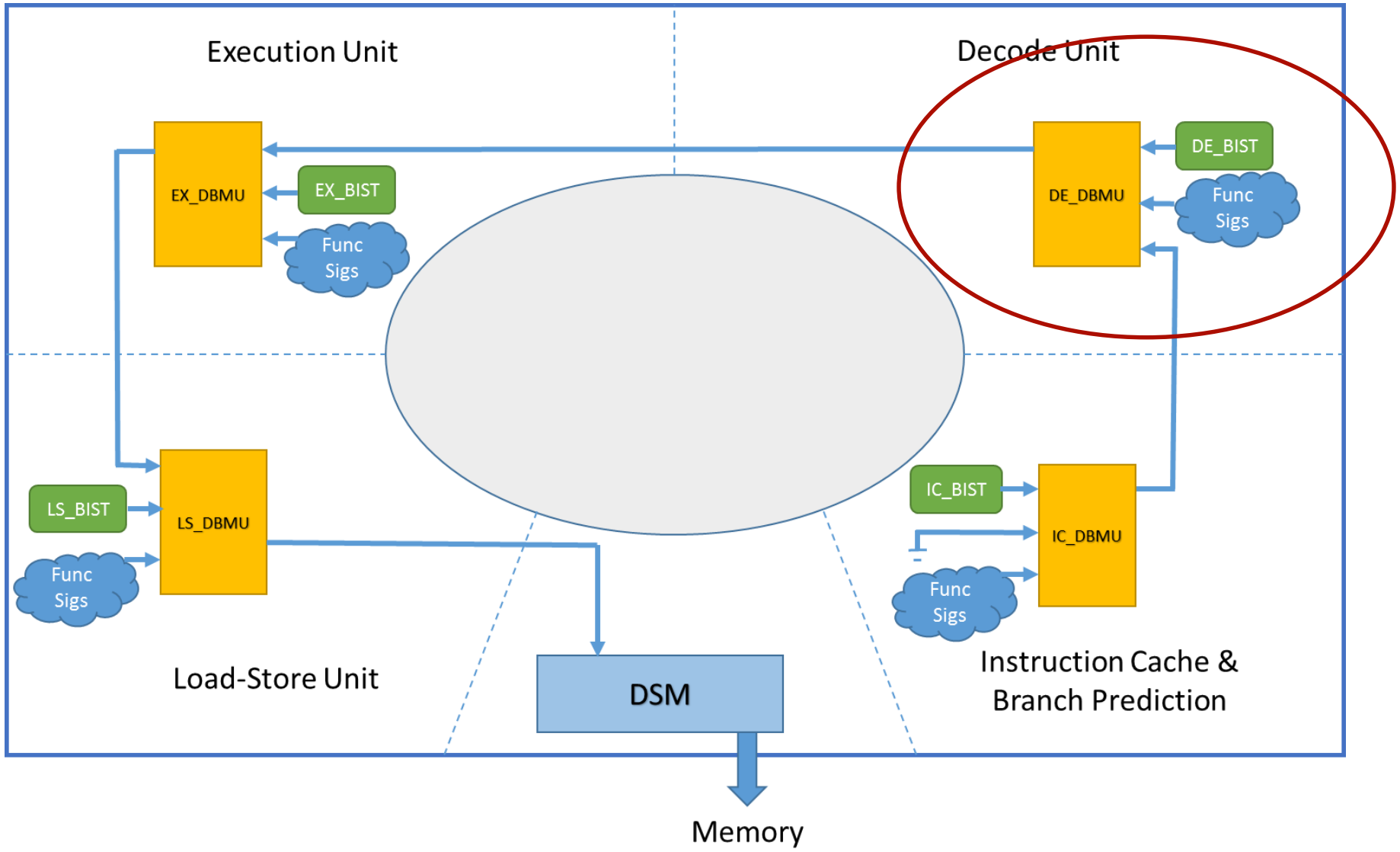
September 29, 2016

Austin, TX

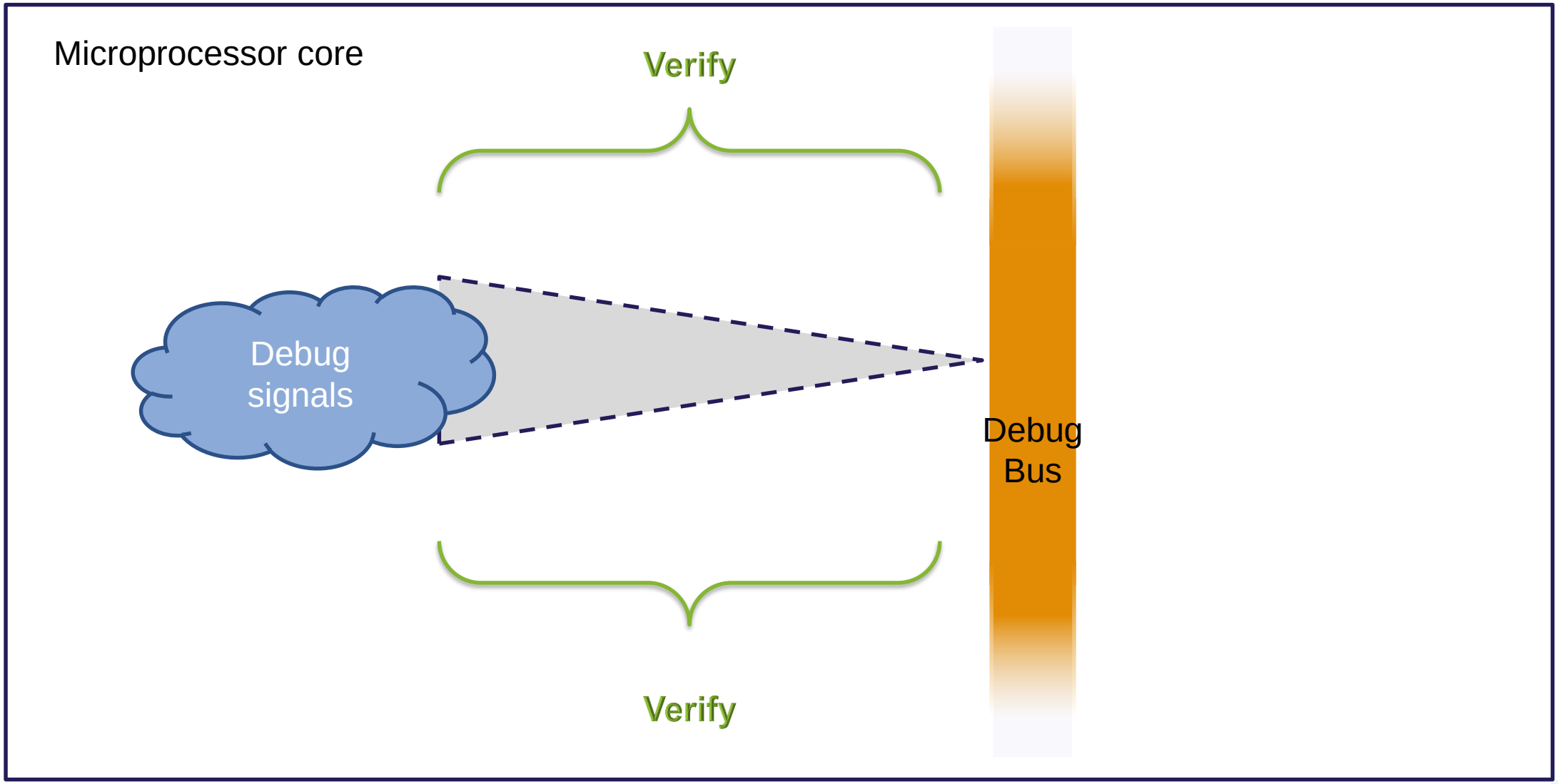
Agenda

- **What is debug-bus?**
- Traditional verification
- Why use formal?
- How to use formal
- Gotchas
- Issues identified
- Conclusion

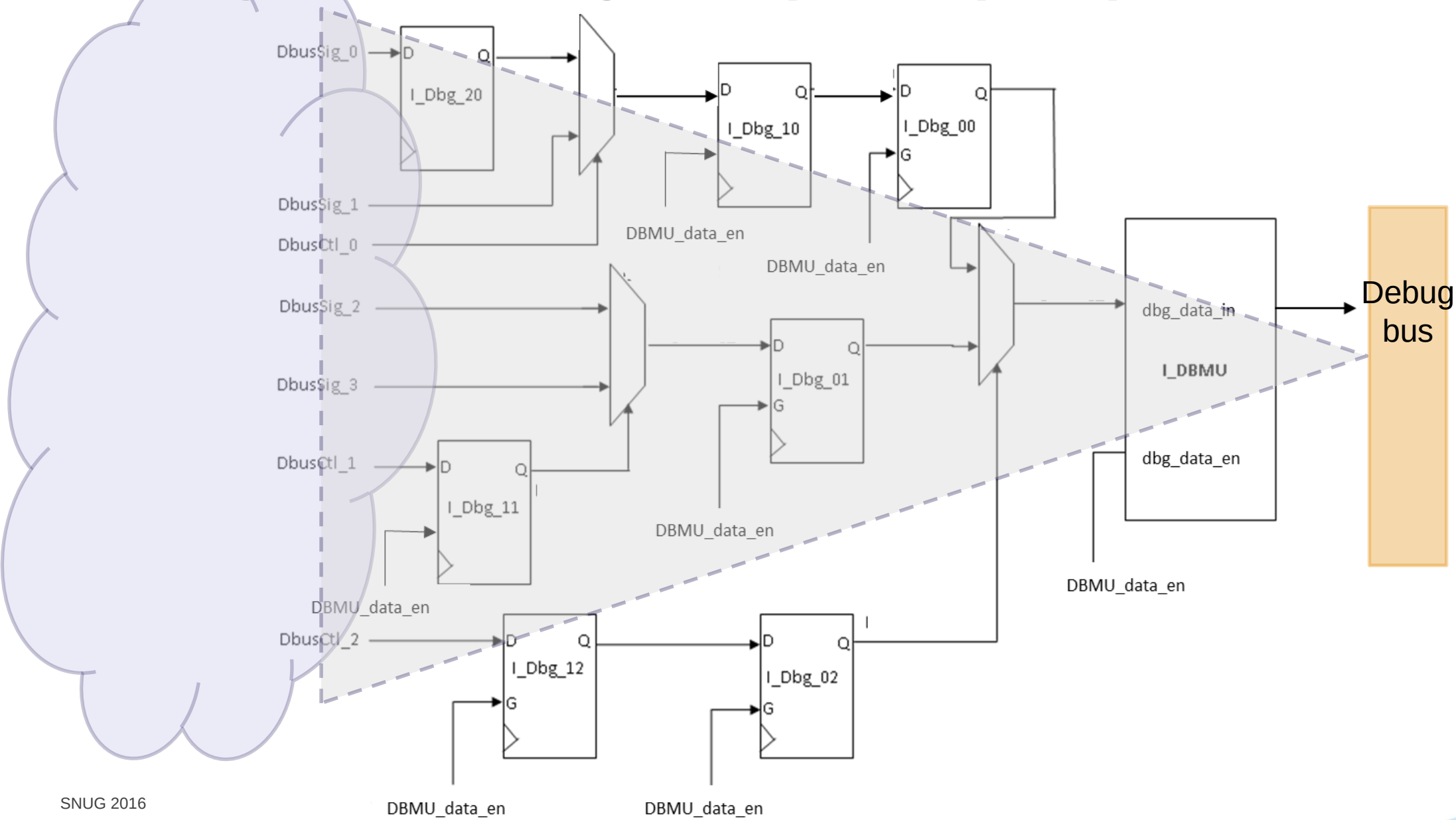
What is debug-bus?



Problem statement



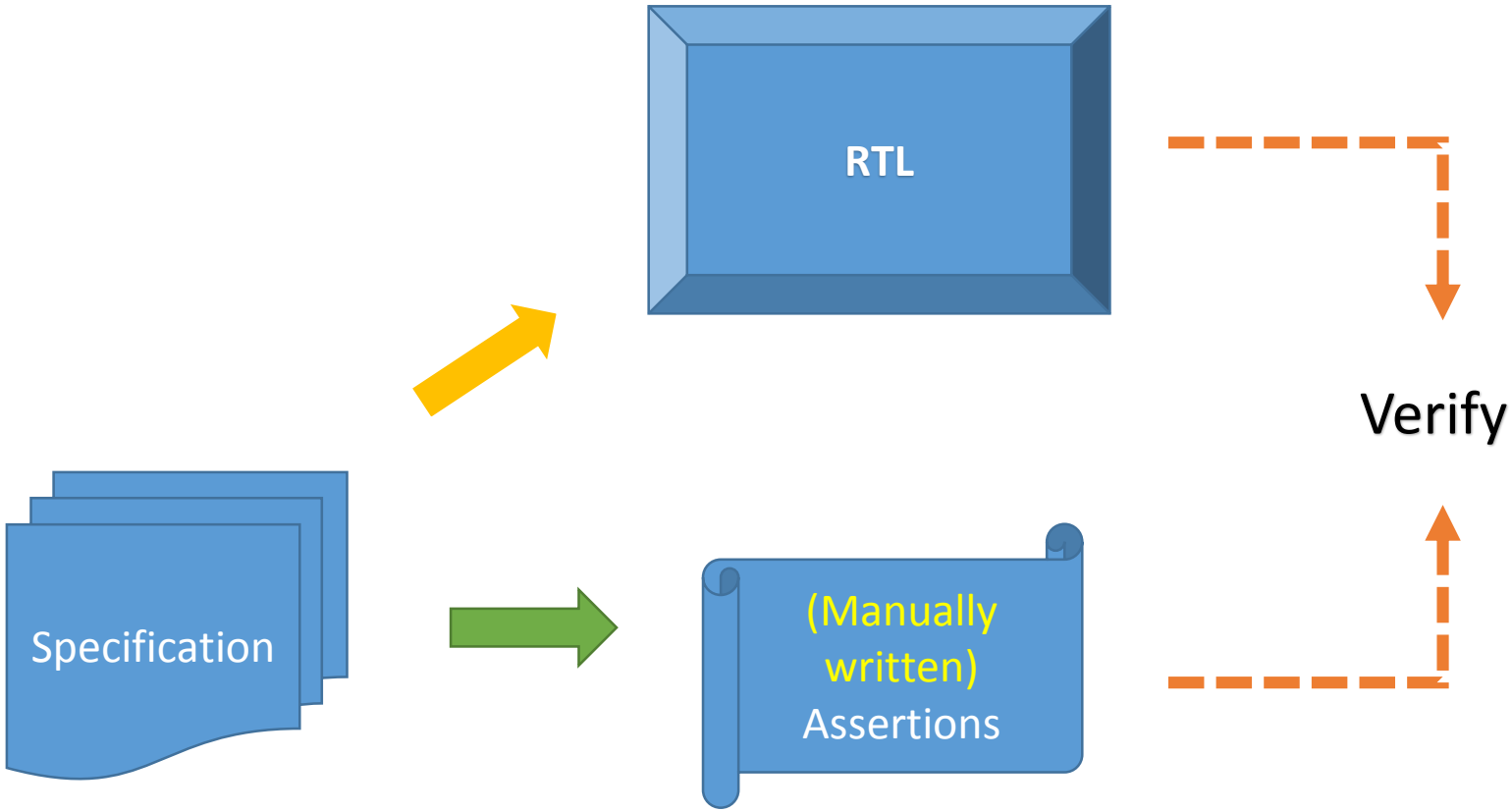
Debug-Bus Muxing Unit (DBMU) stop



Agenda

- What is debug-bus?
- **Traditional verification**
- Why use formal?
- How to use formal
- Gotchas
- Issues identified
- Conclusion

Simulation-based verification



Determining verification effort

- ▲ Critical piece of debug hardware
- ▲ Behaves like a hardware assertion

- ▼ No direct customer value
- ▼ Writing D-tests is monotonous and labor intensive
- ▼ Debug signals change constantly as design evolves

Drawbacks of traditional verification

Design

- Keeping assertions up-to-date is time-consuming
- Even simple RTL changes require assertion updates

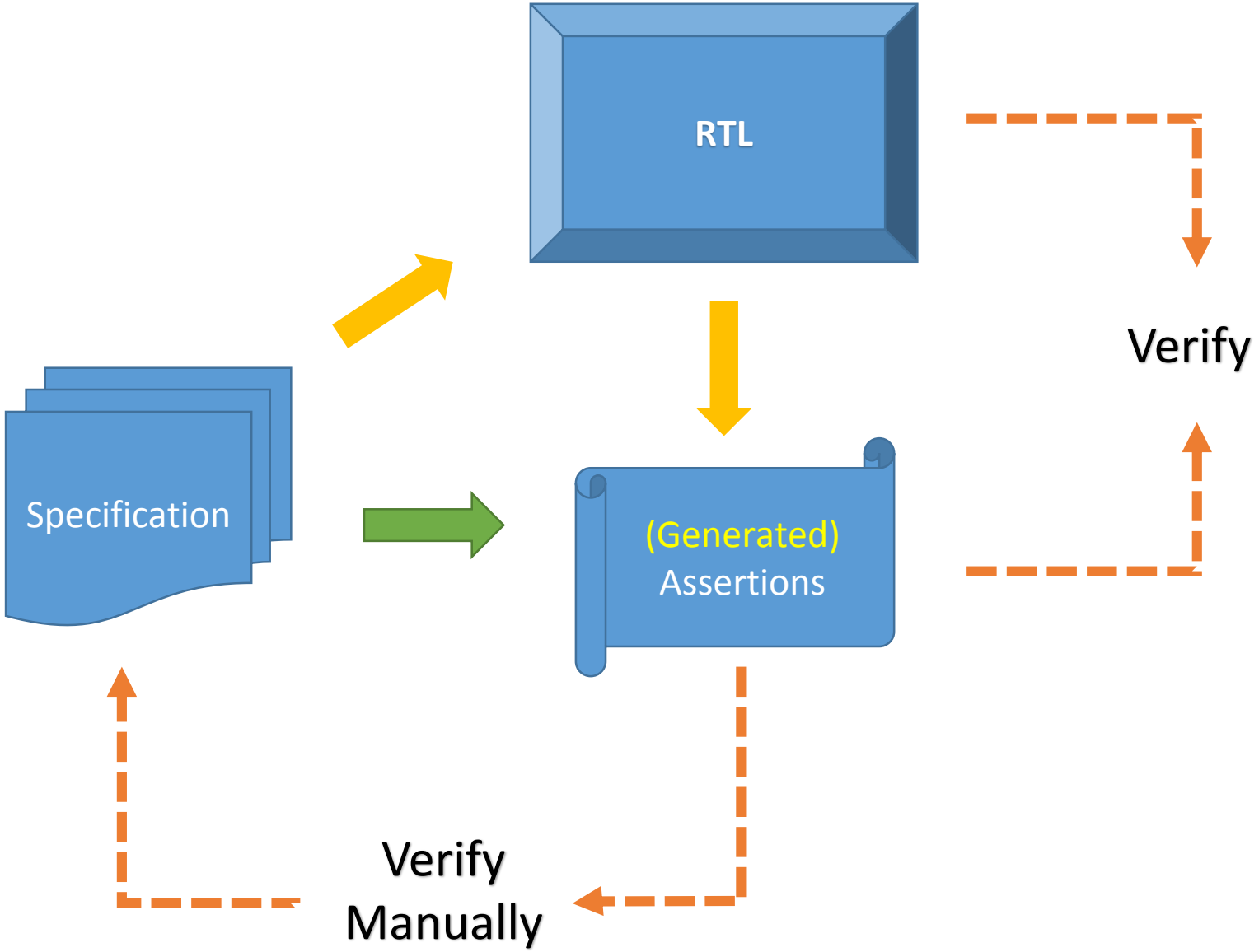
Verification

- Lots of false positive fails
- Closing assertion coverage is cumbersome

Agenda

- What is debug-bus?
- Traditional verification
- **Why use formal?**
- How to use formal
- Gotchas
- Issues identified
- Conclusion

Formal verification



Pros and Cons of formal verification

- ▲ Can be setup as an automated flow
- ▲ Not having to write/update assertions
- ▲ Testing is exhaustive
- ▲ Turnaround time of 24 hours

- ▼ Learning formal verification
- ▼ Have to build test infrastructure from scratch
- ▼ Design has to be amenable

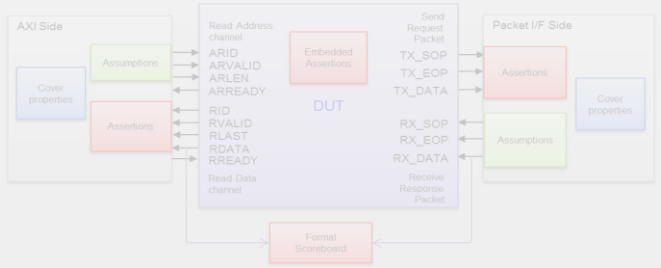
Agenda

- What is debug-bus?
- Traditional verification
- Why use formal?
- **How to use formal**
- Gotchas
- Issues identified
- Conclusion

VC Formal

Property Verification

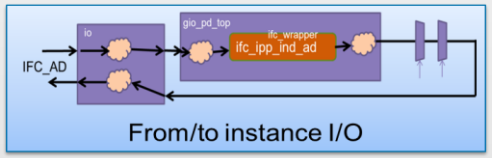
Verify property convergence



Connectivity Checking

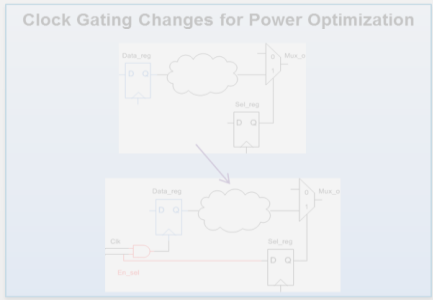
Verify connectivity correctness

to specification



Sequential Equivalence

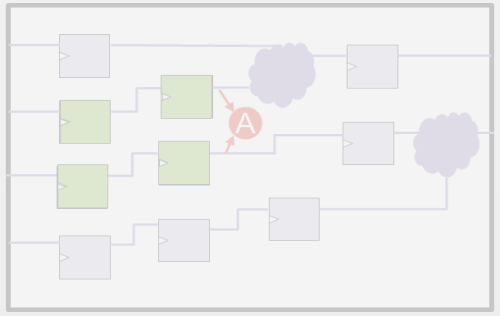
Catches bugs missed by other tools



Auto Checks

Formal aware structural design analysis

Easy setup and comprehensive checks



Formal Coverage Analyzer

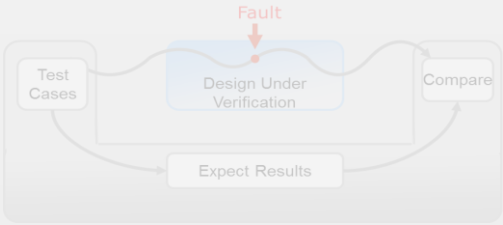
Native integration in VCS

Common coverage debug with Verdi

Formal Testbench Analyzer

Formal testbench completeness

High performance fault injection & analysis



Preparing for formal verification

1. Analyze design size and depth of logic

2. Follow a consistent signal naming convention

   Source: DbusSig*
   Sink: DBMU_LOCAL_DATA
   Control: DbusCtl*

3. Replace AND-OR logic with muxes using `defines

How it's done

Step 1: Assertion extraction

Initialize tool

Black-boxes unwanted modules

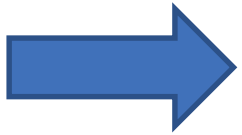
Setup clock and reset

Enable clock-gaters

Identify sources and destination

Identify control signals

Generate assertions



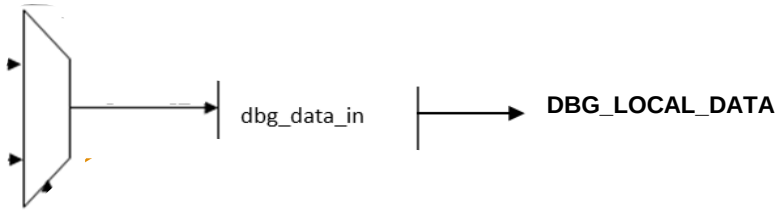
Step 2: Assertion proof

Define initial state

Prove assertions

(Debug and repeat)

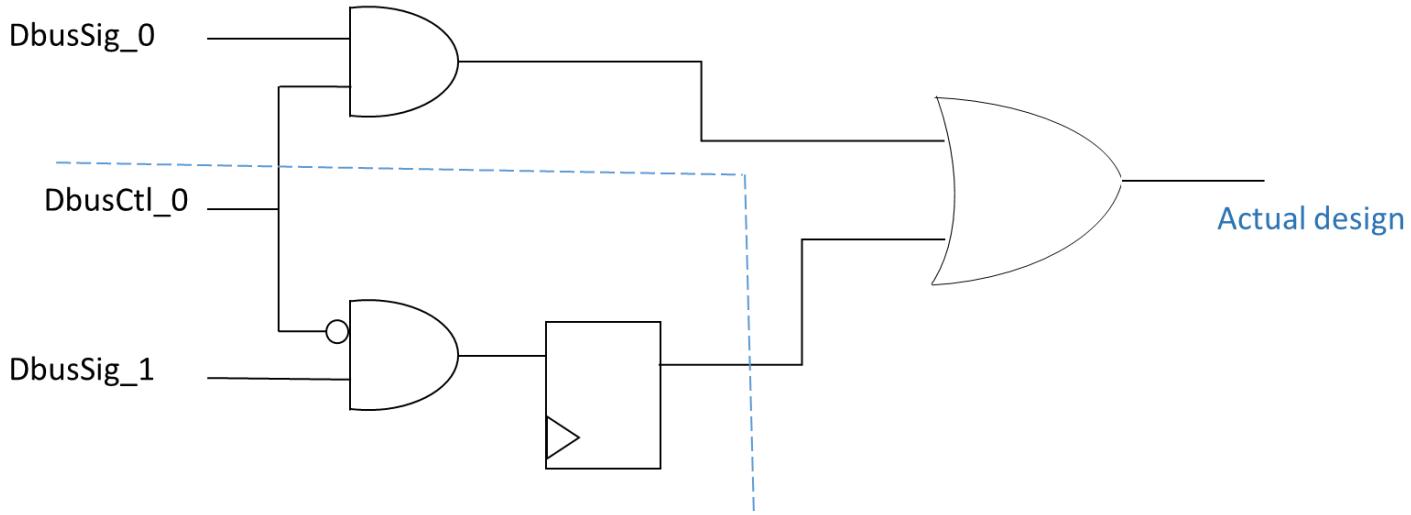
Example of SV assertions



Agenda

- What is debug-bus?
- Traditional verification
- Why use formal?
- How to use formal
- **Gotchas**
- Issues identified
- Conclusion

Combinational logic in datapath



Agenda

- What is debug-bus?
- Traditional verification
- Why use formal?
- How to use formal
- Gotchas
- **Bugs identified**
- Conclusion

Debug-bus design issues identified

- No path from debug signal to DBMU output

- Naming convention violations

Agenda

- What is debug-bus?
- Traditional verification
- Why use formal?
- How to use formal
- Gotchas
- Issue identified
- **Conclusion**

Conclusion

- Used VC Formal to perform processor core debug-bus connectivity check
- Checks are exhaustive; no separate coverage closure effort required
- Flow is automated. 24 hours turnaround time.
- Required design modifications to work around tool limitations
- No debug-bus issues in silicon

AMD

snug
Synopsys Users Group
Austin

Thank You

- When can the generated assertions fail?
  - When design used for assertion generation != design used for assertion proof (AND-OR -> muxes)
  - Use assertions as golden to check functional ECOs. Will see fails is debug-bus connectivity is disrupted.

Disclaimer & Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTIONS

Please double check the AMD Author’s Guidelines and Legal’s Trademark webpage to make sure you have included all of the appropriate AMD and 3rd party attributions:

http://amdcentral.amd.com/AMDTeams/Corporate/Legal/Pages/Trademarks.aspx