AMD

DESIGN AUTOMATION CONFERENCE 56

LEAKING UNVERIFIED CORNER CASES

WAYNE YUN

ADVANCED MICRO DEVICES, INC.

AGENDA

FORMAL SIGN-OFF USING A THEORETIC METRIC

I Have a Dream

Output Enforcement Analysis

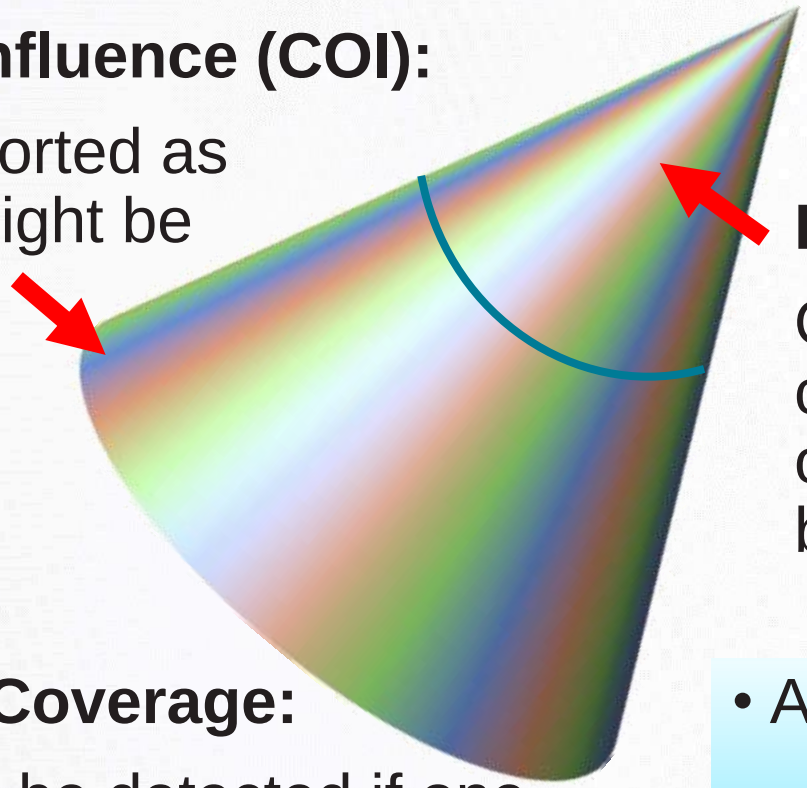
Sign-off Using OEA

Summary

FORMAL ASSERTION COVERAGE

Cone of Influence (COI):

Logics reported as covered might be unverified.



Formal Core:

One usage is covered, others could be unverified.

"In spite of the difficulties and frustrations at the moment, I still have a dream."

- Martin Luther King Jr.

Mutation Coverage:

A fault can be detected if one scenario triggers. Other scenarios might be unverified. Also the metric could consume much computation power.

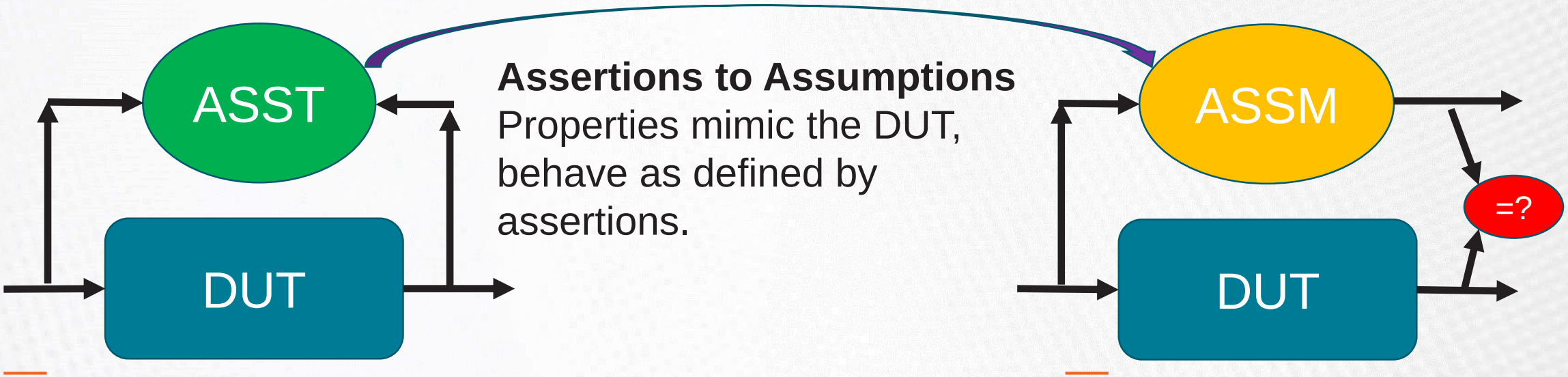
- An ideal verification metric would be one that:
  - Gives a timely accurate conclusion
  - Tells if there is any scenario not verified
  - Presents details of unverified scenarios
- Output Enforcement Analysis (OEA) can help

OUTPUT ENFORCEMENT ANALYSIS

FORMAL PROPERTY VERIFICATION

ASSUME-GUARANTEE METHOD

SEQUENTIAL EQUIVALENCE CHECK



Assertions to Assumptions
Properties mimic the DUT, behave as defined by assertions.

Create Assertions
In an end-to-end sign-off setup, assertions are intended to verify the behavior of the DUT. However, corner cases could be left unverified and result in behavior gaps.

Catch the Gap
The difference of behavior between the DUT and assumptions is the gap. An unverified corner case can be caught together with such a gap.

EXAMPLE OF A TWO-LEVEL PIPELINE

```
// oea_fpv.sv, for FPV run
module oea (
  input clk,   input rst, input q,
  input a,     input o
  );
  wire q; reg qq;
  always @(posedge clk) qq <= rst ? 1'b0 : q;
  assign o = qq;
  m_r: assert property (@(posedge clk)
                        $fell(rst) |-> !q);
  m_a: assert property (@(posedge clk)
                        ##1 q == $past(a));
endmodule
```

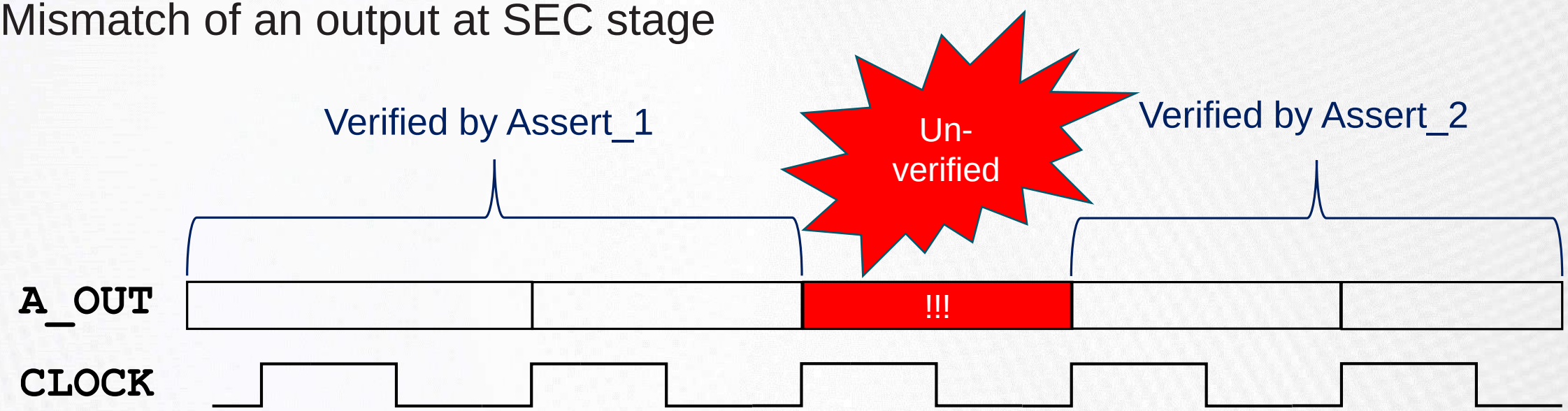
```
// dut.v DUT same for FPV and SEC
module dut (
  input clk,   input rst,
  input a,     output o
  );
  reg q, qq;
  always @(posedge clk) q <= rst ? 1'b0 : a;
  always @(posedge clk) qq<= rst ? 1'b0 : q;
  assign o = qq;
endmodule
```

```
// oea_seq.sv, for SEC run
module oea (
  input clk,   input rst,
  input a,     output o
  );
  wire q; reg qq;
  always @(posedge clk) qq <= rst ? 1'b0 : q;
  assign o = qq;
  m_r: assume property (@(posedge clk)
                        $fell(rst) |-> !q);
  m_a: assume property (@(posedge clk)
                        ##1 q == $past(a));
endmodule
```

- Same property file could be used for FPV and SEC
- The first level is verified by properties
- The second level is checked by formal model
- Code tested with Synopsys VC Formal 2018.09

FIRST UNVERIFIED CASE CAUGHT BY OEA

- In a mature end-to-end sign-off setup supported by COI, Formal Core, and Mutation
  - Mismatch of an output at SEC stage

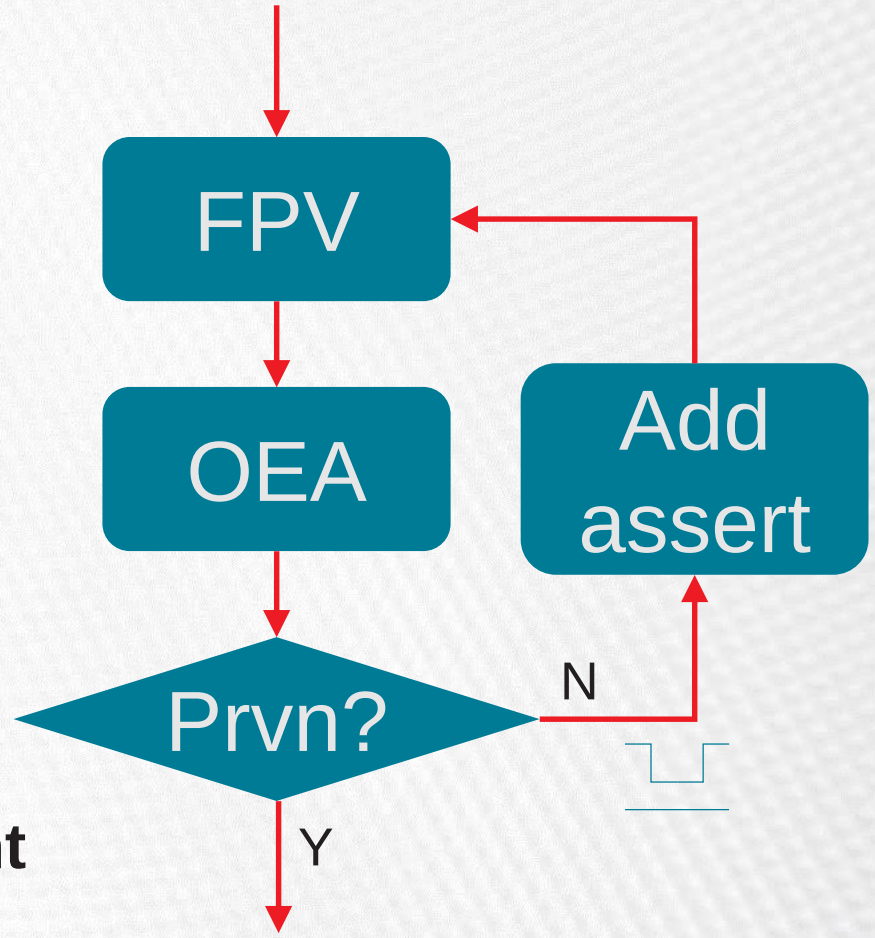


- More mature setups were tried
- ***OEA can be a strong sign-off metric***

Design Size	+	++	+++
Findings	1	~25	???

OEA FLOW AND METRIC

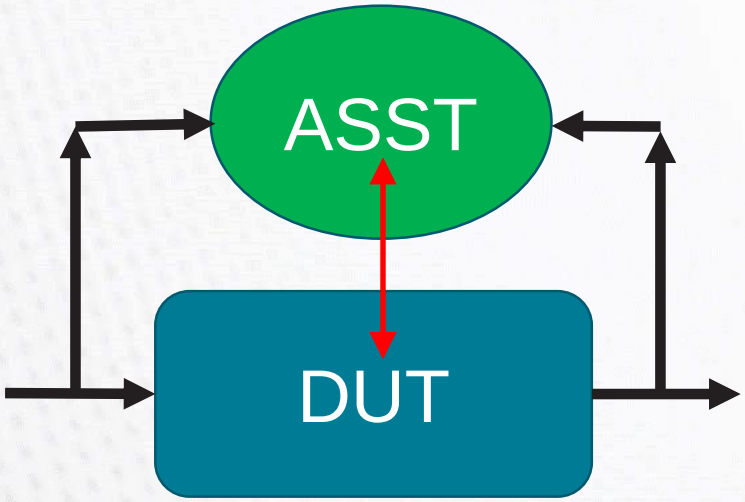
- OEA flow starts with assertions proven by FPV
- Sequential Equivalence Check (SEC/SEQ/SLEC) compares DUT to assumptions (were assertions)
- Properties model the DUT at any equivalent output, hence, this output is proven to be verified
- Otherwise, a formal tool can generate a waveform showing a scenario and indicating the difference
- Unverified case can result in a difference at outputs
- Improve assertions to check such cases and re-run
- Successful SEC proves the DUT is verified
  - Within constraints/assumptions from FPV
  - In theory, properties model DUT in every scenario - no case unverified
- **OEA Metric = the percentage of outputs equivalent**
  - 100% = each case verified within constraints



HANDLING COMPLEXITY

SEC Challenge

- To check properties vs RTL
- Using intermediate nodes of DUT
  - Equivalent between properties and DUT
  - Simplifying SEC problem



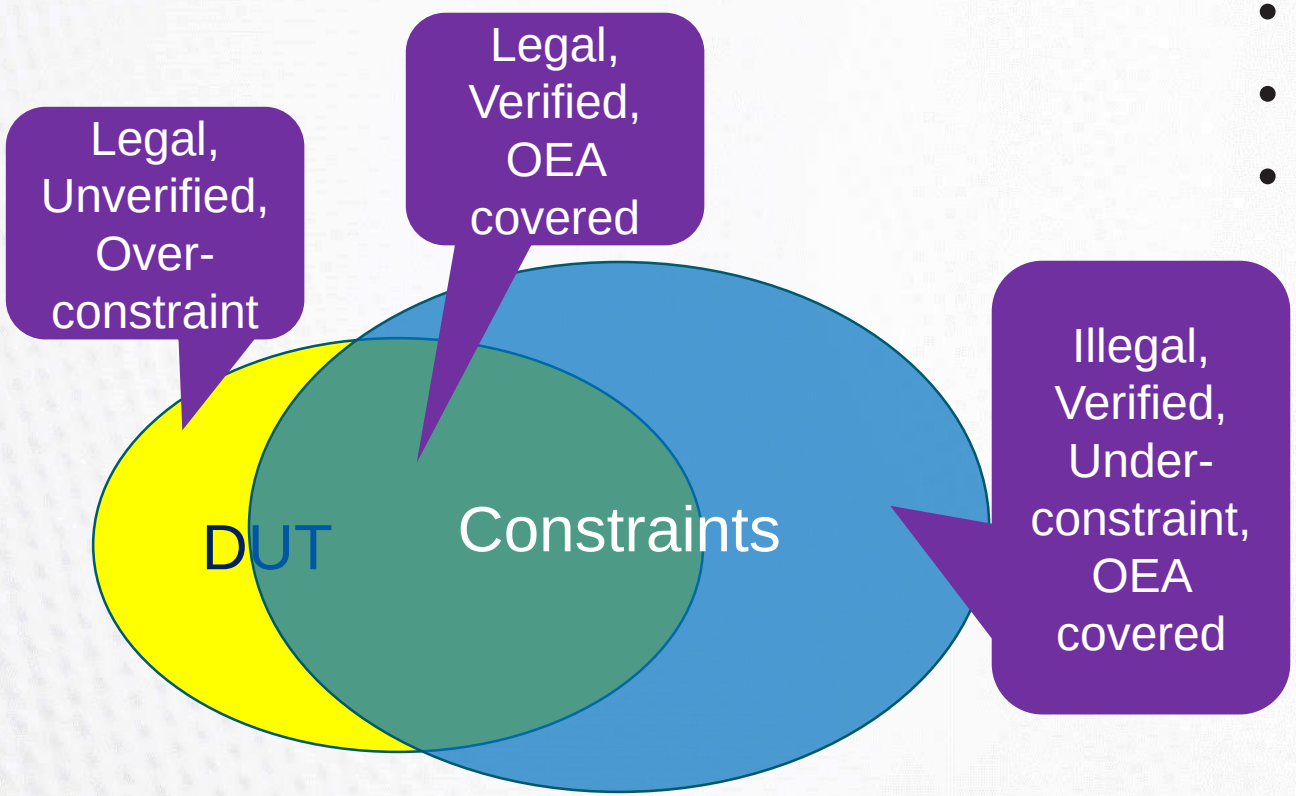
Bounded proofs

- Are pushed to a depth deemed proven in FPV
- SEC can take them as proven

SEC	Action
Proven	Deem as proven
Bounded, deep enough	Deem as proven
Bounded, not deep enough	Improve setup
Failed	Case unverified or assertion falsified

SIGN-OFF WITH OEA

- OEA Metric is complete within constraints at theory level
- Over-constraint becomes major concern



Strategies for over-constraint

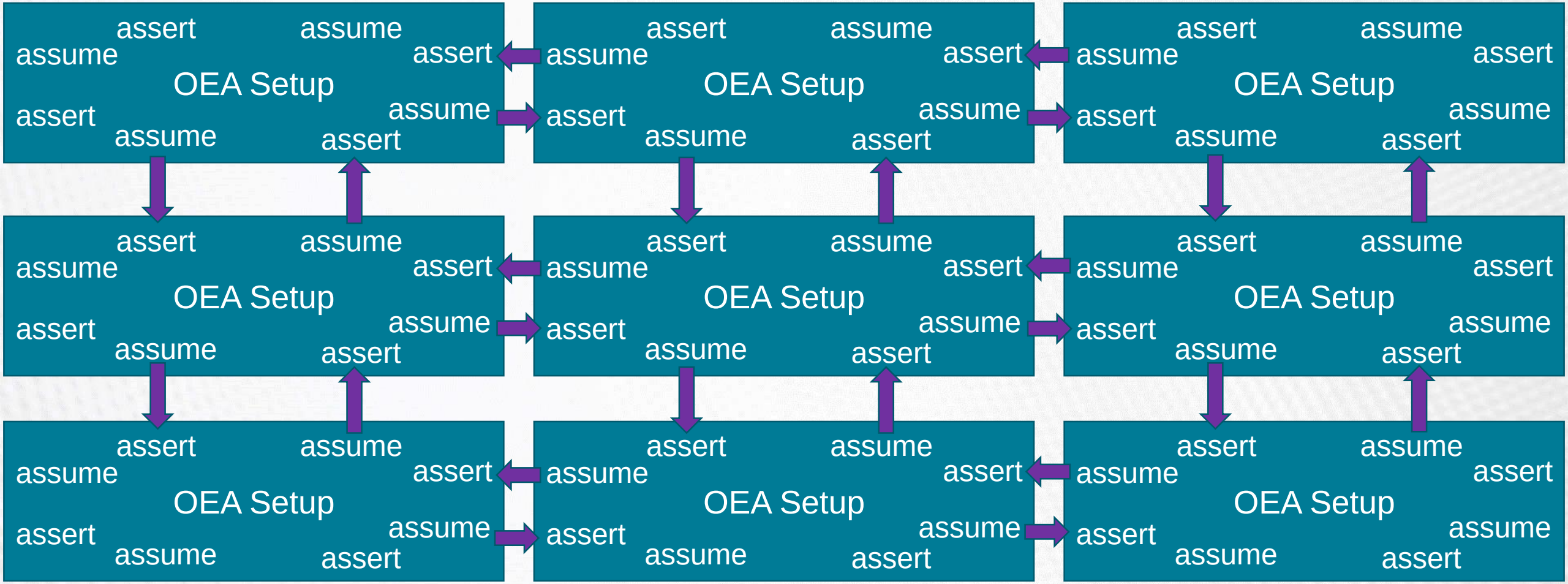
- Over-constraint analysis
- Prove constraints
- Simulate constraints
- Mutation coverage
- Cover properties/groups

Recommendation:

- **OEA and Proving Constraints**
- **Mutation as safety net**

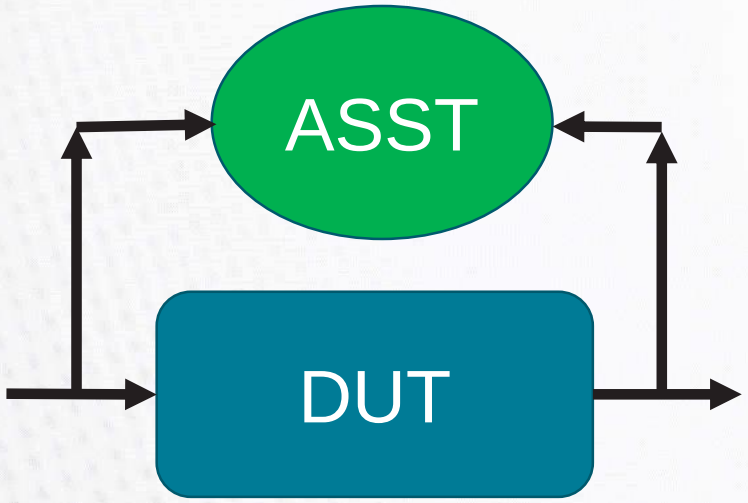
FROM BRICKS TO WALL

- Glue OEA setups with assume-guarantee method to cover large design



PROPERTY STYLES

- A property defines a relationship among a group of signals
- Based on if each signal is input or output of DUT, properties can be of in-in, in-out, or out-out style
- Some styles can work better in OEA scheme than others



Property	In-In	In-Out	Out-Out
Assertion	☹	☺	☺
Assumption (on signals of same interface)	☺	☺☹	☹

SUMMARY

- Popular metrics of formal assertion coverage used for sign-off may NOT be complete
  - Corner cases could be left unverified between assertions
  - A barrier for wide adoption of formal sign-off
- Output Enforcement Analysis (OEA) Metric is *complete* at theory level
  - Sequential Equivalence Checking (SEC) can contribute to the completeness
  - OEA Metric is defined as percentage of DUT outputs found equivalent
  - 100% indicates each corner case verified within constraints
  - The remaining challenge is mainly over-constraint, recommended to prove them
  - Listed strategies for complexity, bounded proof, and large design
- This desired innovation can propel exhaustive verification, confident sign-off, and early/quick fix of issues

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.

ATTRIBUTION