Formal Sign-off with Formal Coverage

ASHUTOSH PRASAD, VIGYAN SINGHAL (OSKI TECHNOLOGY)

VIKRAM KHOSA (ARM)

Unique Methodology. Highest Coverage. Fastest Time to Market.

Oski TECHNOLOGY

Agenda

- Sign-off
- Coverage
  - Simulation coverage
  - Formal coverage
- End-to-End Formal Verification
- Sign-off with Coverage
- Use on the ARM® Next-Generation CPU Design

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Sign-off

Unique Methodology. Highest Coverage. Fastest Time to Market.

Oski TECHNOLOGY

Sign-offs

- What does sign-off mean to program managers?
  - Ready for tape-out
- Sign-off requires
  - Commitment to finish – else task is optional, and may be killed
  - Metrics to measure progress
  - Nightly/weekly regression runs
- Common sign-off flows
  - Static timing
  - Simulation (spreadsheet and coverage)
  - Power
  - RTL-vs-gates LEC

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

What Do We Mean by Formal Sign-off

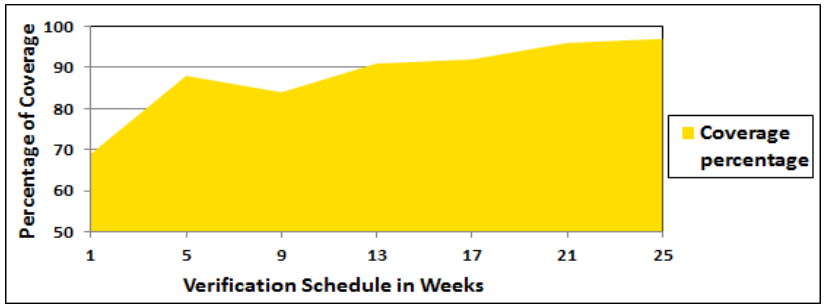
- When formal is done, no bug is left behind
- In addition to Formal Sign-off methodology, we also need:
  - Commitment & accountability
    - For chosen design blocks, formal is the only technology used for functional verification, replacing not supplementing simulation
    - Cannot tapeout until formal is done
  - Metrics to measure progress
    - Manager's dashboard – coverage, bug rate, run time
    - Progress tracker through weighed run-status transition graph
  - Weekly/nightly regression runs
    - Setup, once formal testbench matures, to easily catch bugs due to RTL changes

Formal Sign-off Can Offer More Confidence Than Simulation Sign-off!

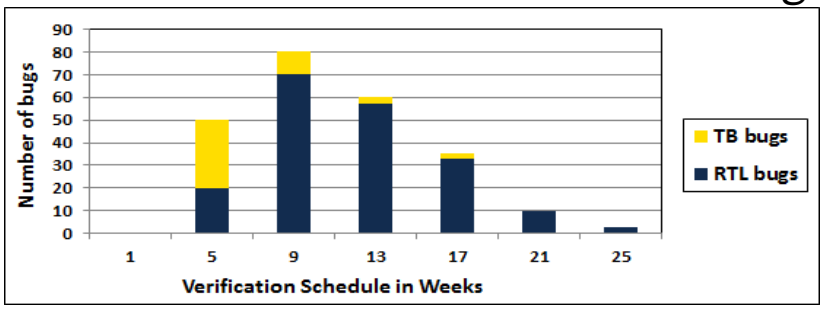
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

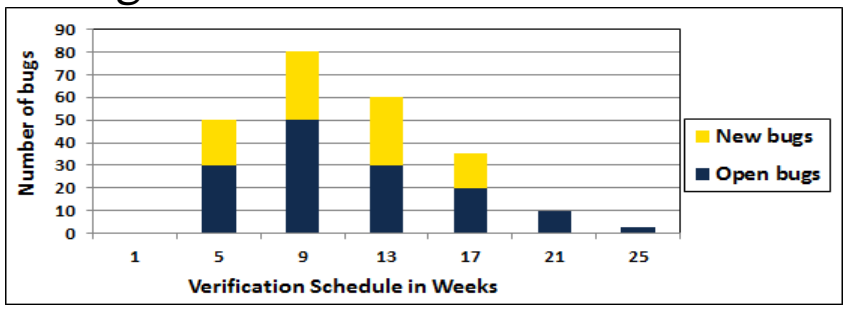
Tracking Progress Is Important for Verification Sign-off

Coverage tracking

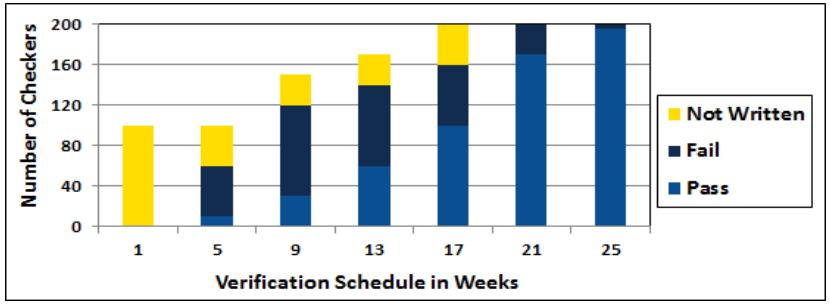


Bug tracking



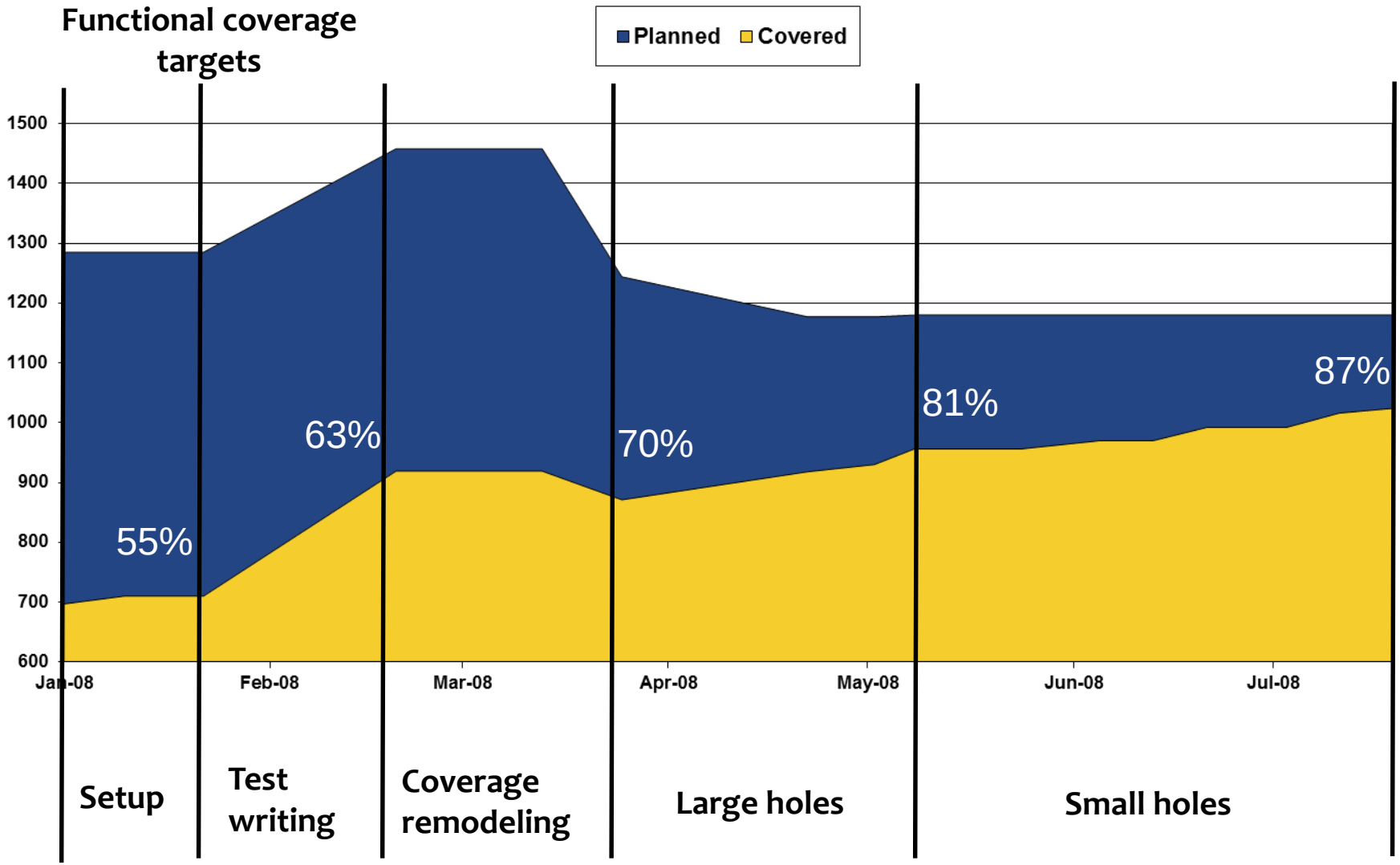


Runtime status



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Coverage Closure Is an Important Sign-off Item



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

But Coverage Is Not the Be-all and End-all

"The perfect is the enemy of good"

-Voltaire (1772)

- Coverage is not perfect
  - Bugs are missed even with 100% coverage
- But…
  - Helps measure progress
  - Helps identify blind-spots

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

How to Pick the Right Formal Metrics?

- To track progress, an ideal metric answers:
  - How much of the design and what is being verified?
    - Quantifies work done so far
  - How much of the design and what is not being verified?
    - Quantifies work left to do

- To indicate sign-off, an ideal metric answers:
  - Checkers: is my list of checkers complete?
  - Constraints: have I over-constrained the testbench to disallow legal stimuli?
  - Complexity: is my achieved proof depth sufficient?

We will use Formal Coverage to quantify each of the three questions

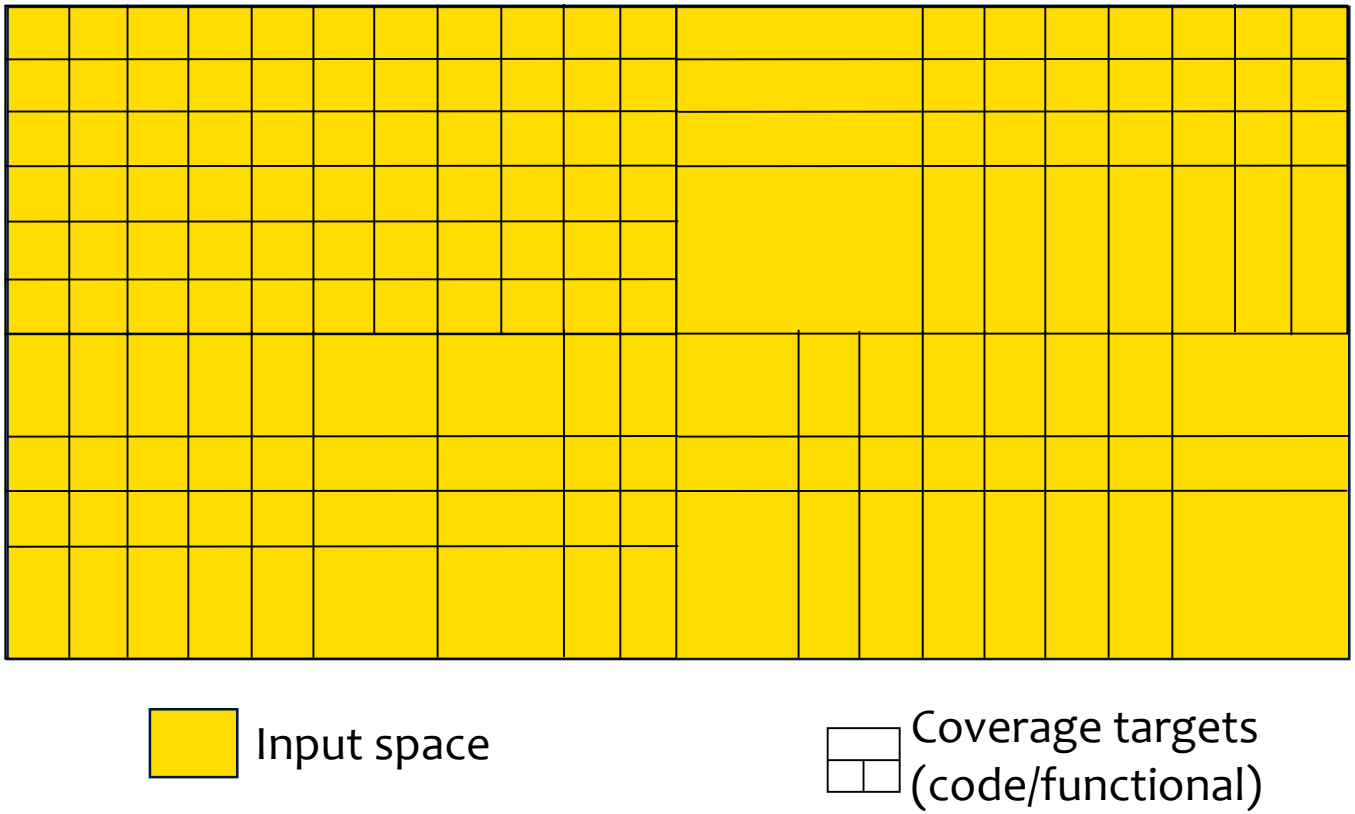
Oski TECHNOLOGY
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Coverage

Unique Methodology. Highest Coverage. Fastest Time to Market.

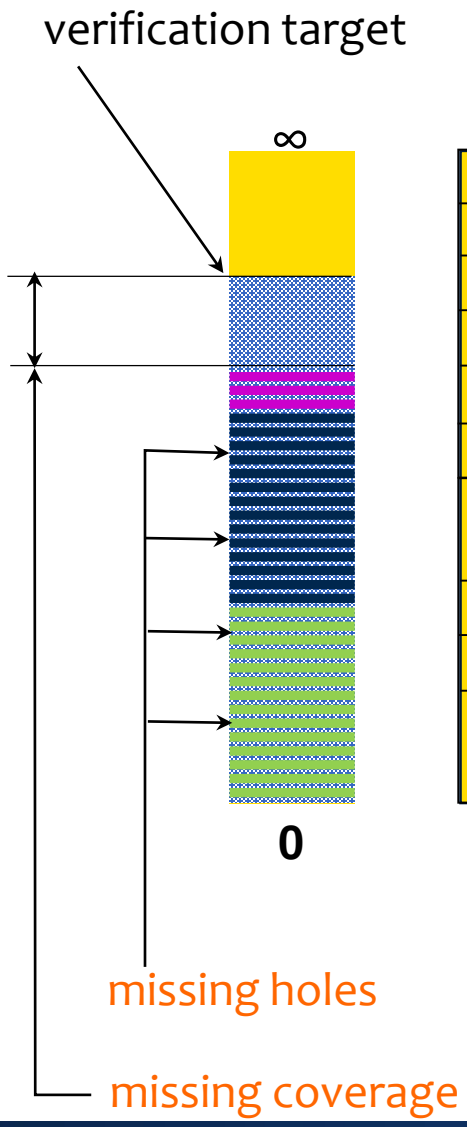
Oski TECHNOLOGY

Coverage Grid

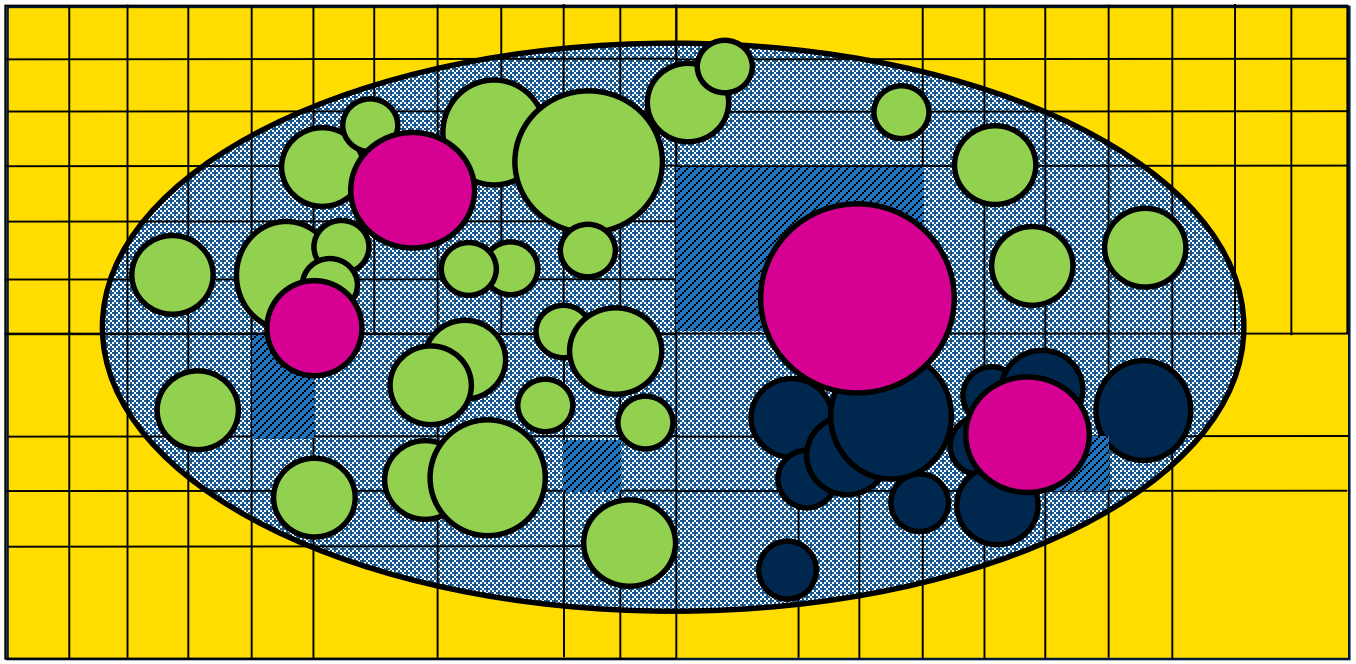


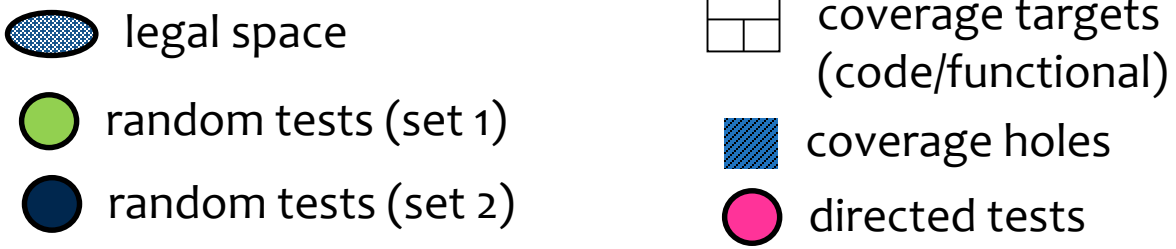
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Formal & Simulation Difference: Verification Closure (1/2)



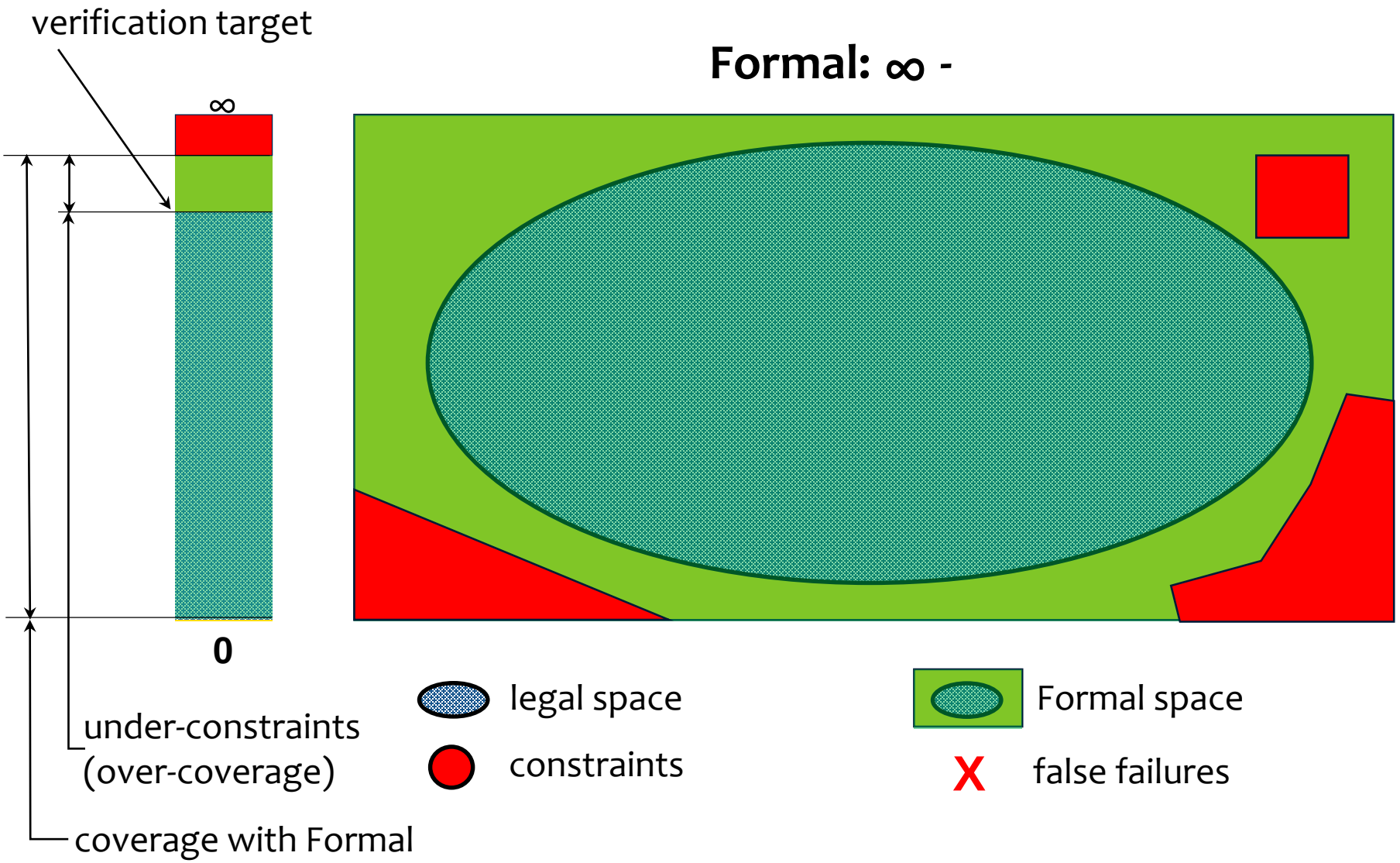
Simulation: 0+





COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Formal & Simulation Difference: Verification Closure (2/2)



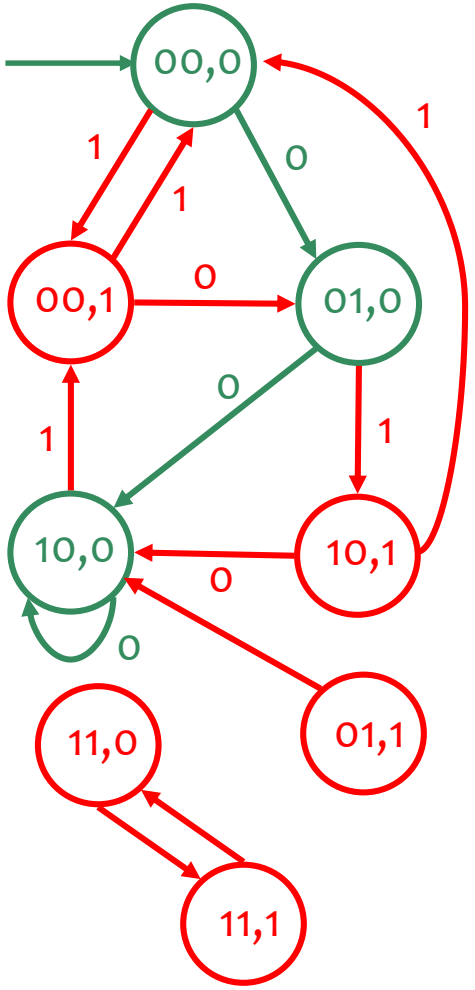
Different Types of Coverage

- What to Measure (targets)
  - Structural coverage
    - Statement (line) coverage
    - Branch coverage
    - Expression coverage
    - FSM coverage
    - Toggle coverage
  - Functional coverage
  - Will not cover in this talk – assume line coverage from now on
- When to Measure
  - Controllable (controlling inputs)
    - Reachability coverage (analogous of classical simulation coverage)
  - Observable (observing at outputs or checkers)
    - COI coverage
    - Proof Core coverage (analogous to mutation coverage in simulation tools)

Oski TECHNOLOGY
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Simulation Coverage (a = 0)

```
input a;
reg b;
reg [1:0] st;

always @(posedge clk or negedge rst)
  if (~rst) st <= 2'b00;
  else case( st )
    2'b00: if (~a) st <= 2'b01;
    2'b01: st <= 2'b10;
    2'b10: if (a) st <= 2'b00;
    endcase

always @(posedge clk or negedge rst)
  if (~rst) b <= 1'b0;
  else if (~a | b) b <= 1'b0;
  else b <= 1'b1;
```



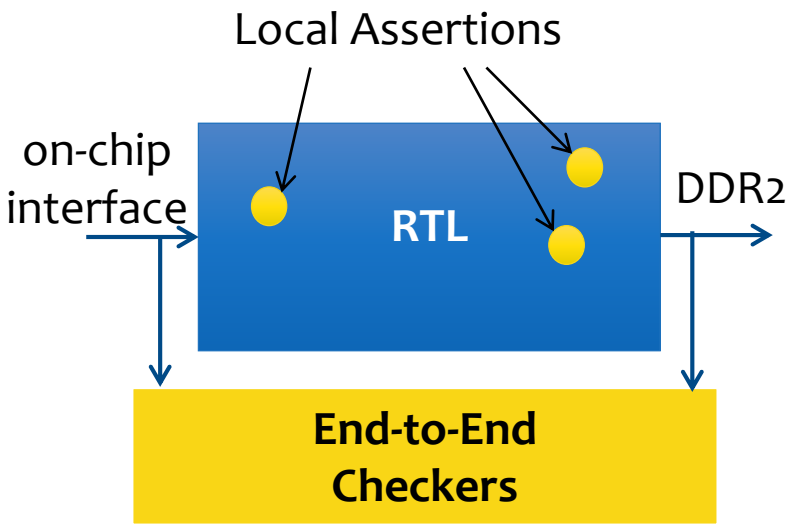
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

End-to-End Formal Verification

Unique Methodology. Highest Coverage. Fastest Time to Market.

What is End-to-End Formal?

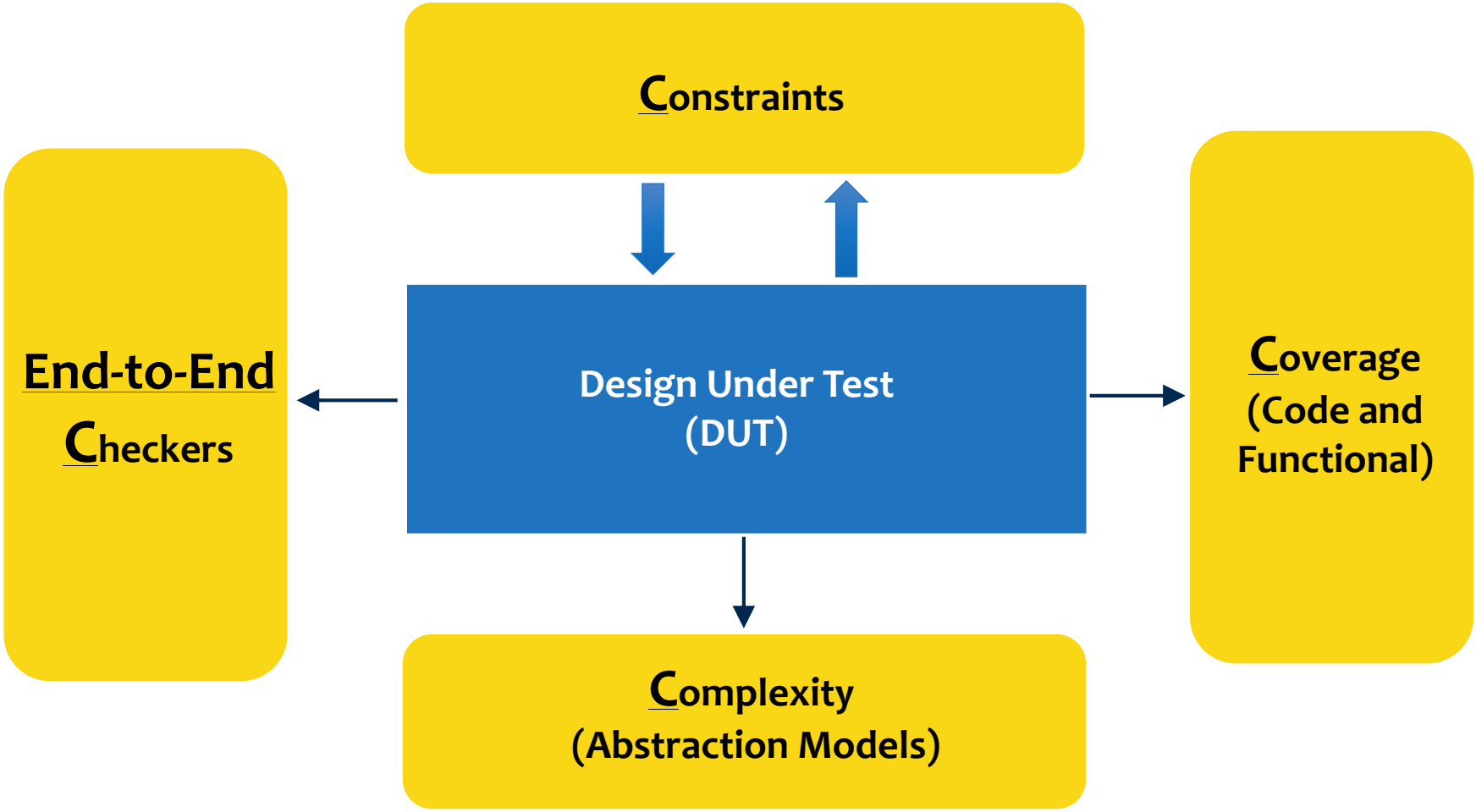
- Local Assertions – easier to verify
  - Internal RTL assertions, embedded in RTL
- End-to-End Checkers – hardest to verify
  - Model end-to-end functionality
  - **Can replace simulation**
  - Often requires Abstraction Models to manage complexity



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Testbench Architecture for End-to-End Formal

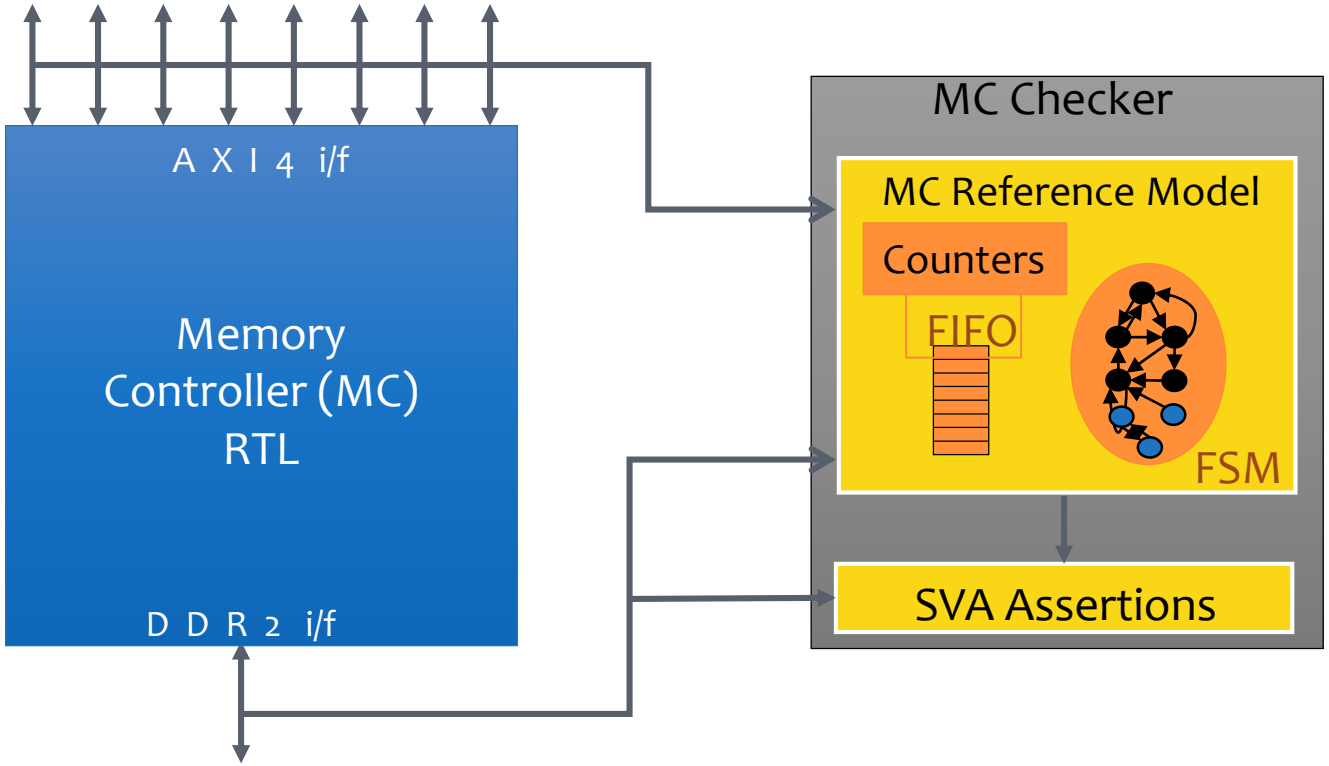
Need to Quantify Completeness of Other 3 C's



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

End-to-End Checkers

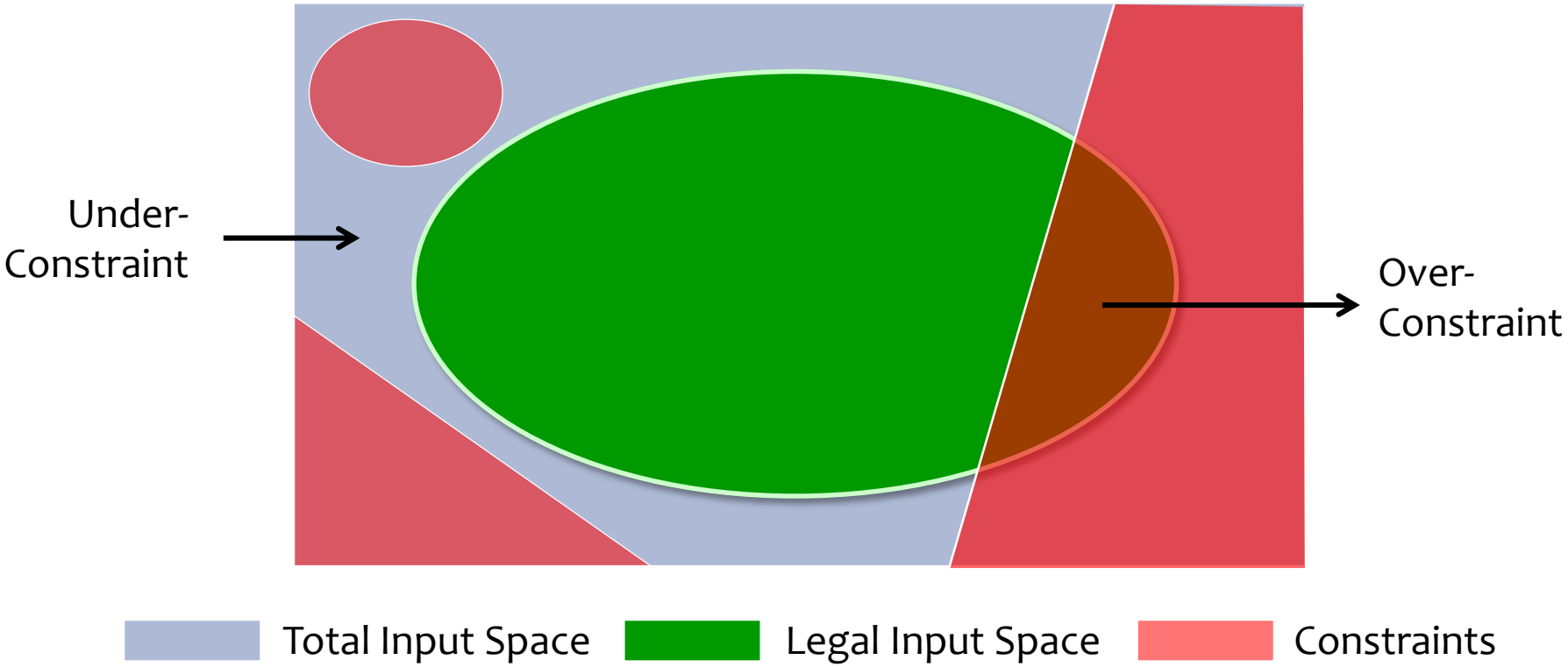
- 95% of End-to-End Checker is in SV or Verilog; rest is SVA
  - Developing synthesizable reference model is as big an effort as writing RTL



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Constraints

- Used to restrict input space to contain only legal input sequences
- Need to watch out for under-constraints & over-constraints



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

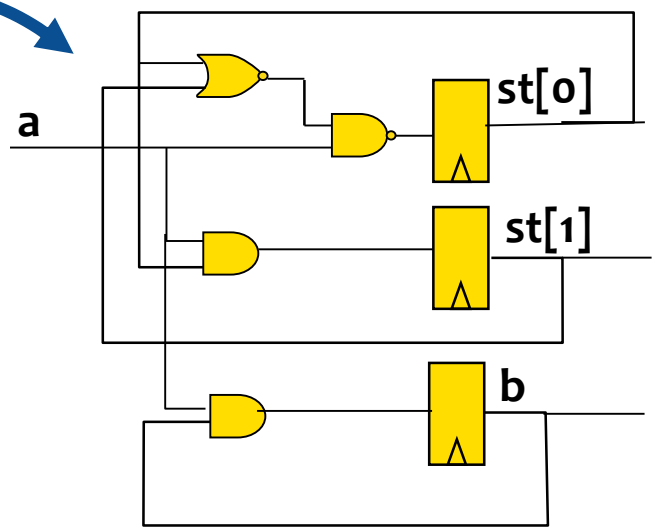
Where Complexity Comes From

Checker: (st == 2'b01) => ~b

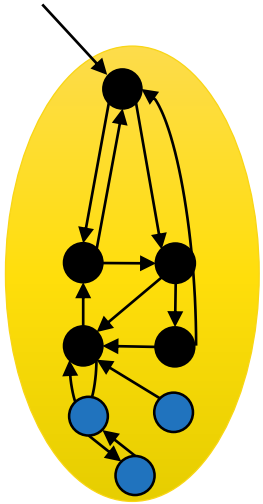
RTL

```
input a;
reg b;
reg [1:0] st;

always @(posedge clk or negedge rst)
 if (~rst) st <= 2'b00;
 else case( st )
       2'b00:  if (~a) st <= 2'b01;
       2'b01:  st <= 2'b10;
       2'b10:  if (a) st <= 2'b00;
       endcase

always @(posedge clk or negedge rst)
 if (~rst) b <= 1'b0;
 else if (~a | b) b <= 1'b0;
 else b <= 1'b1;
```

Internal Netlist



Internal STG



$2^3 = 8$

$2^{10} = 1{,}024$

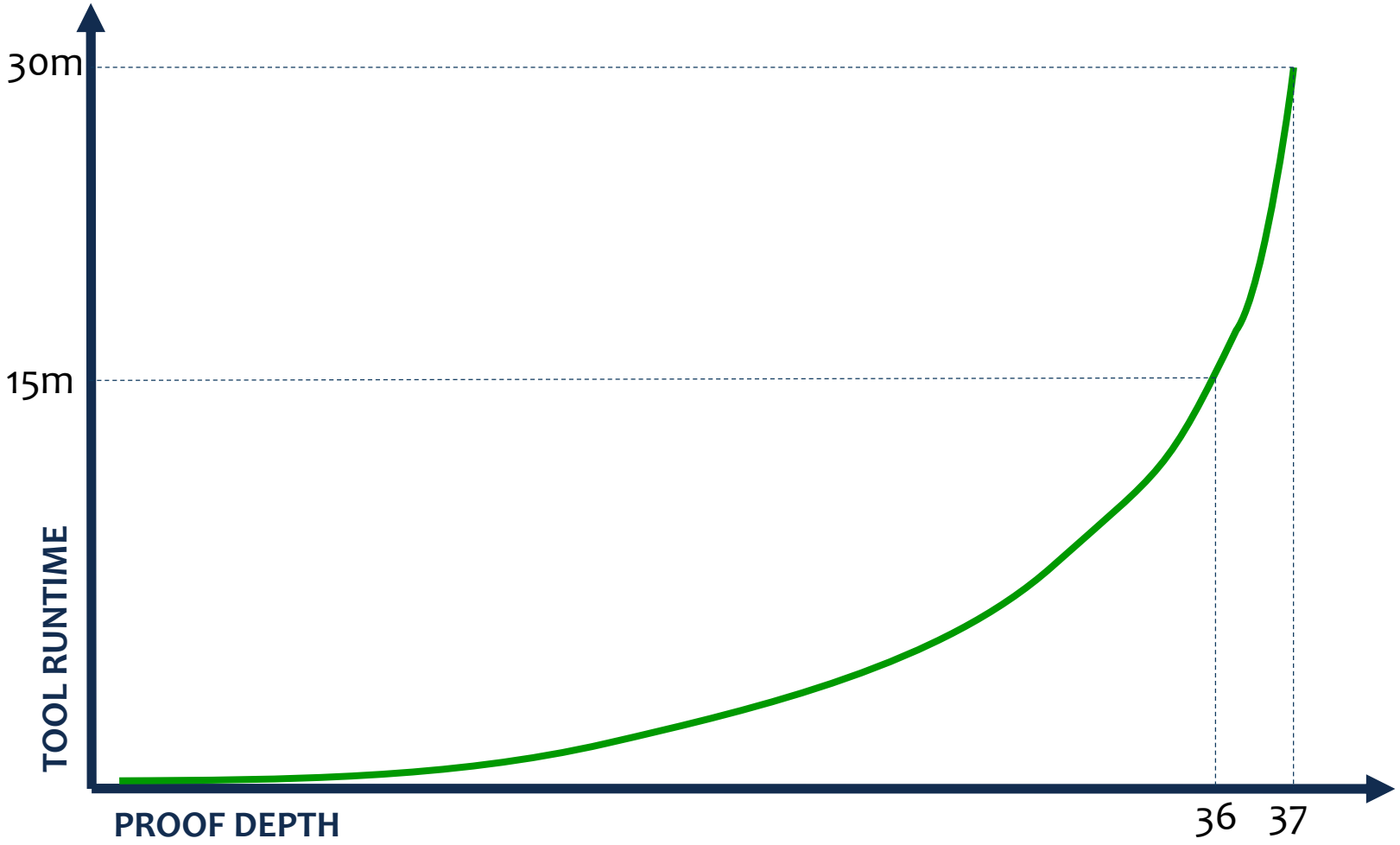
$2^{20} = 1{,}048{,}576$

$2^{30} = 1{,}073{,}741{,}824$

Formal Complexity Comes from Search Space Explosion and can lead to inconclusive (bounded) results!

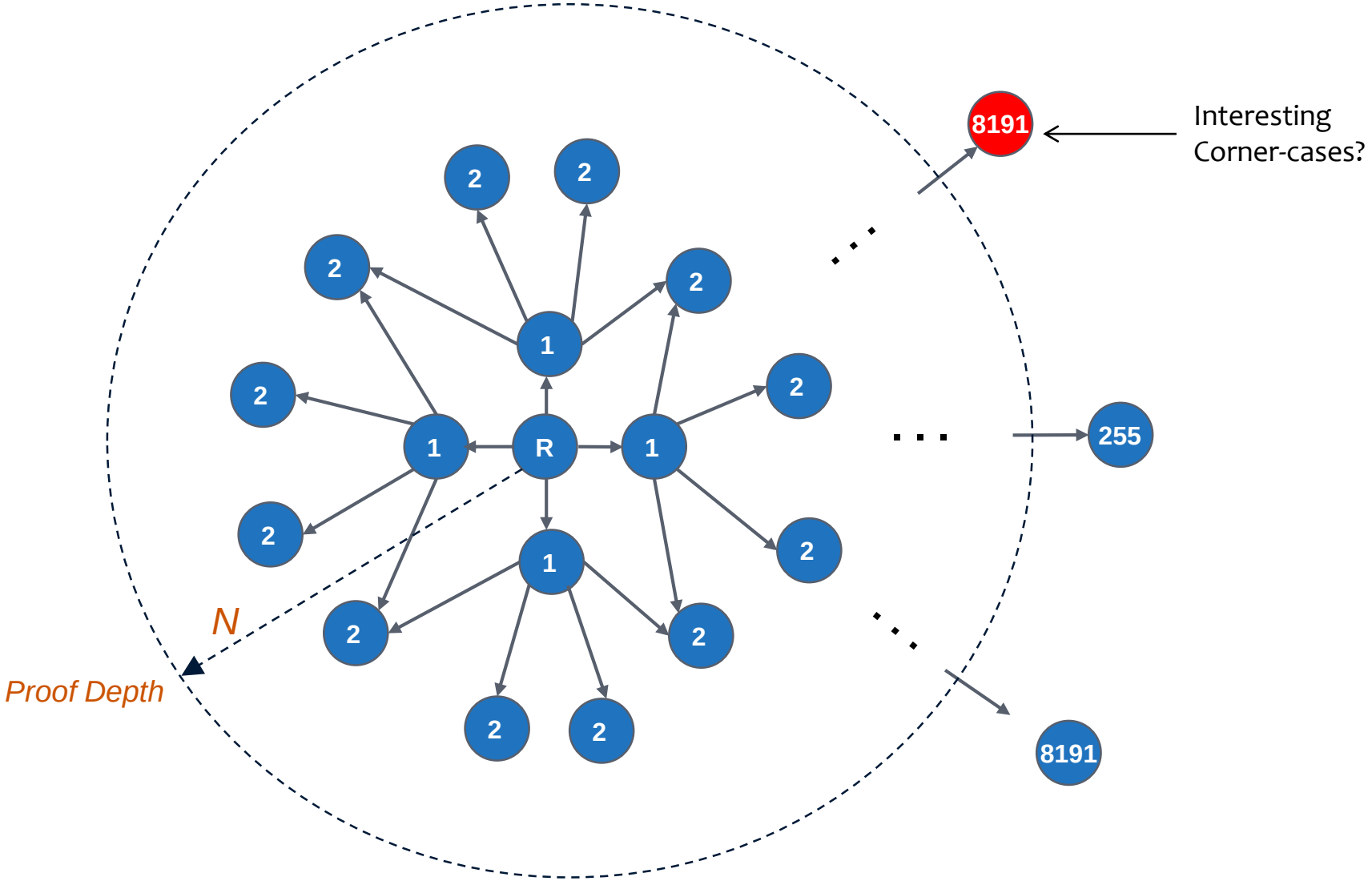
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Can't Fight the Exponential Complexity!



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Formal Covers All State Transitions Within Proof Depth



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

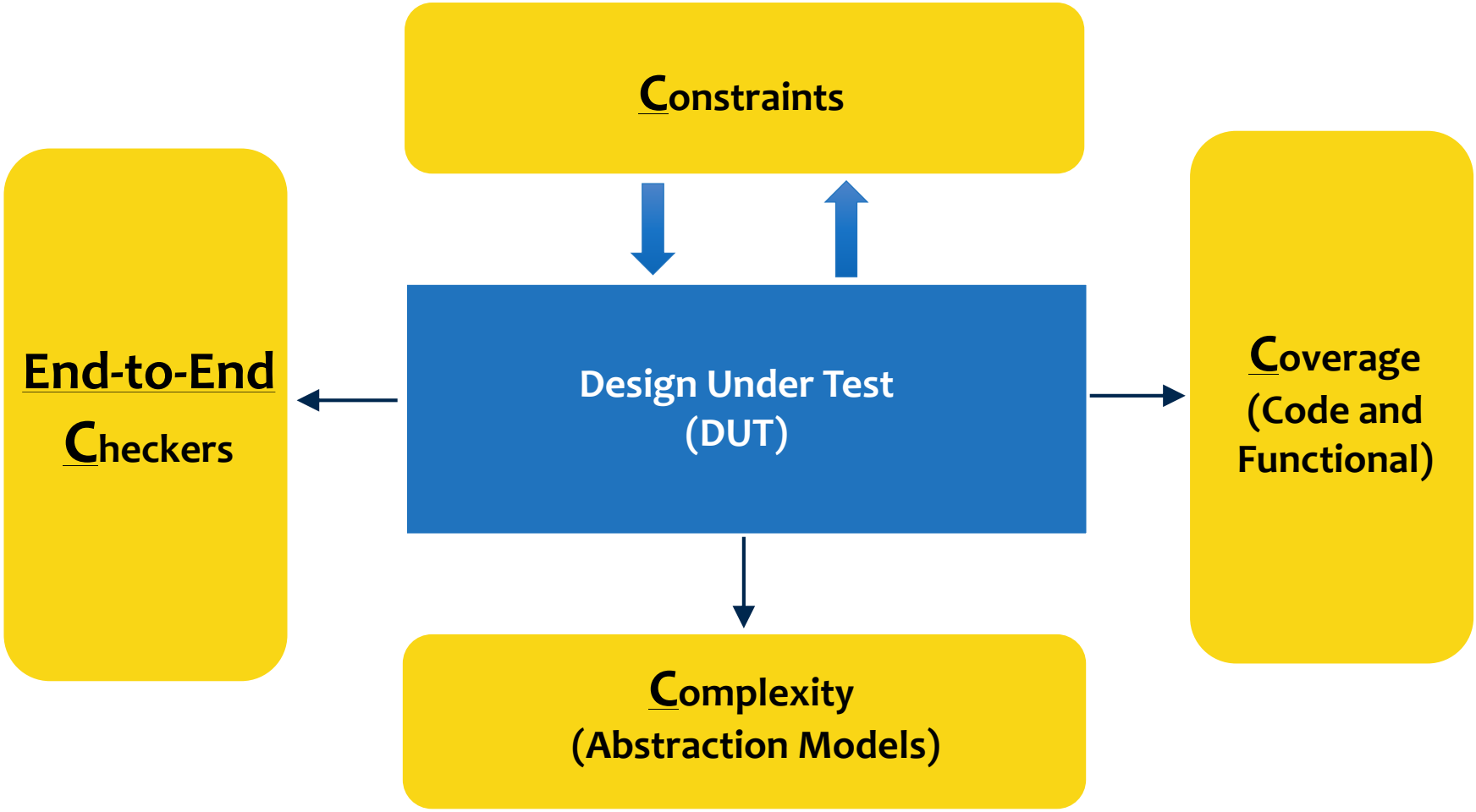
Formal Coverage

Unique Methodology. Highest Coverage. Fastest Time to Market.

Oski TECHNOLOGY

Testbench Architecture for End-to-End Formal

Quality of Formal Depends on all 4 Cs!



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

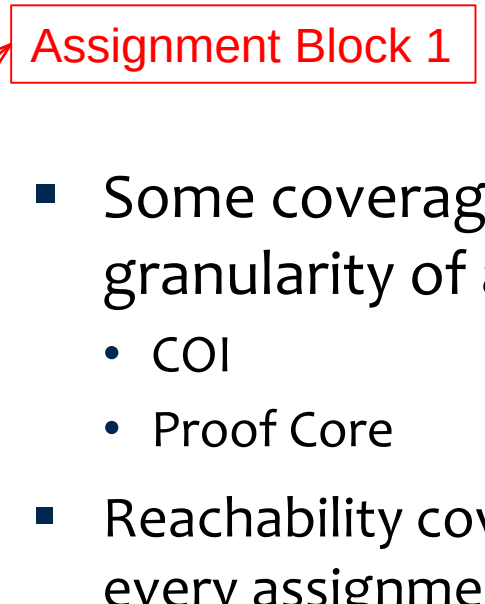
Formal Coverage Measure Goodness of Three C's

- Constraints: Environment may be over-constrained
  - Intentional: avoided some hard to model or verify input combinations
  - Unintentional: bugs in constraints; forgot to remove intentional over-constraints
- Complexity: Sign-off on bounded proof
  - A checker is verified up to proof depth N
  - If a target is not covered in N cycles, proof depth is not sufficient
- Checkers: Measures completeness of the checkers
  - Which portion of logic is used to prove the checkers

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Assignment Blocks

```
reg p, q;
always @(*) begin
  if (a) begin
    p = d1;
    q = e1;
  end else begin
    p = d2;
    if (b) begin
      q = e2;
    end else begin
      p = d3;
      q = e3;
    end
  end
```

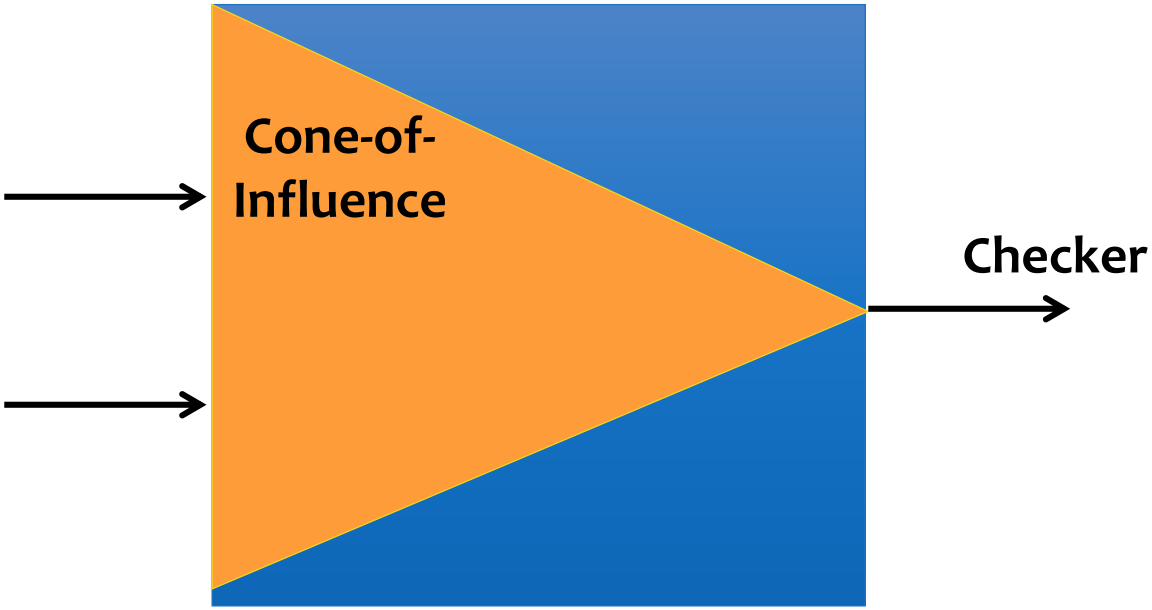


- Some coverages are at the granularity of assignment blocks
  - COI
  - Proof Core
- Reachability covers some portion of every assignment block!
  - Unless testbench is totally vacuous

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Cone-of-Influence (COI) Coverage

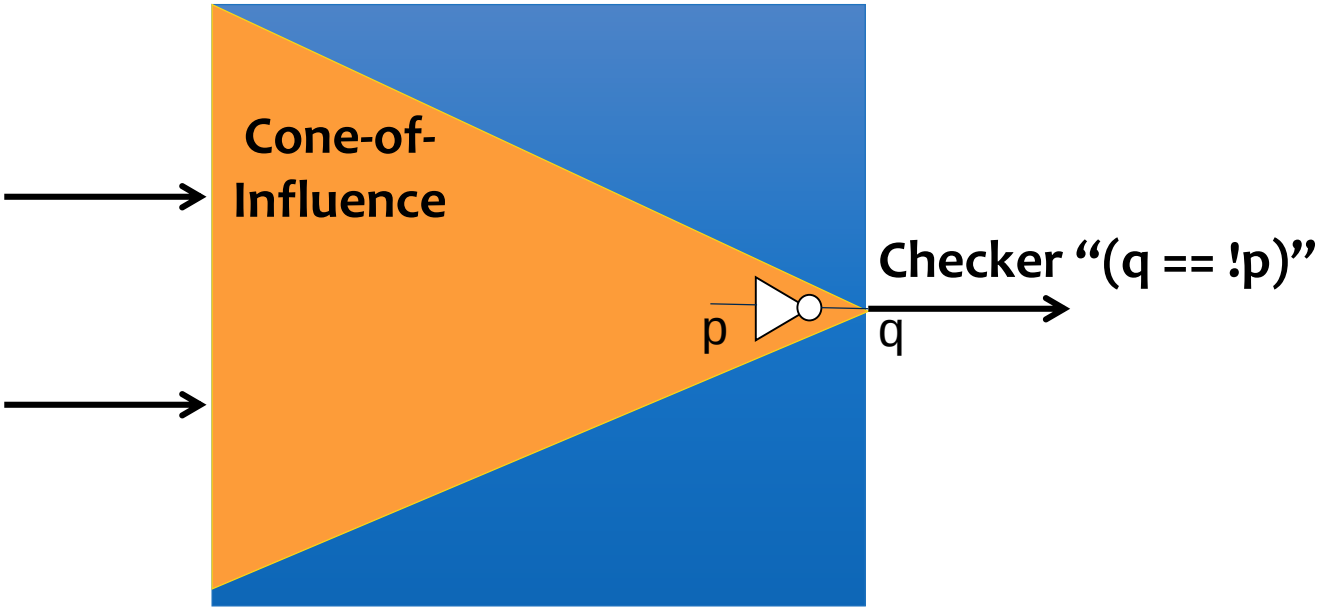
- COI is cheap to compute, but is conservative



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Cone-of-Influence (COI) Coverage

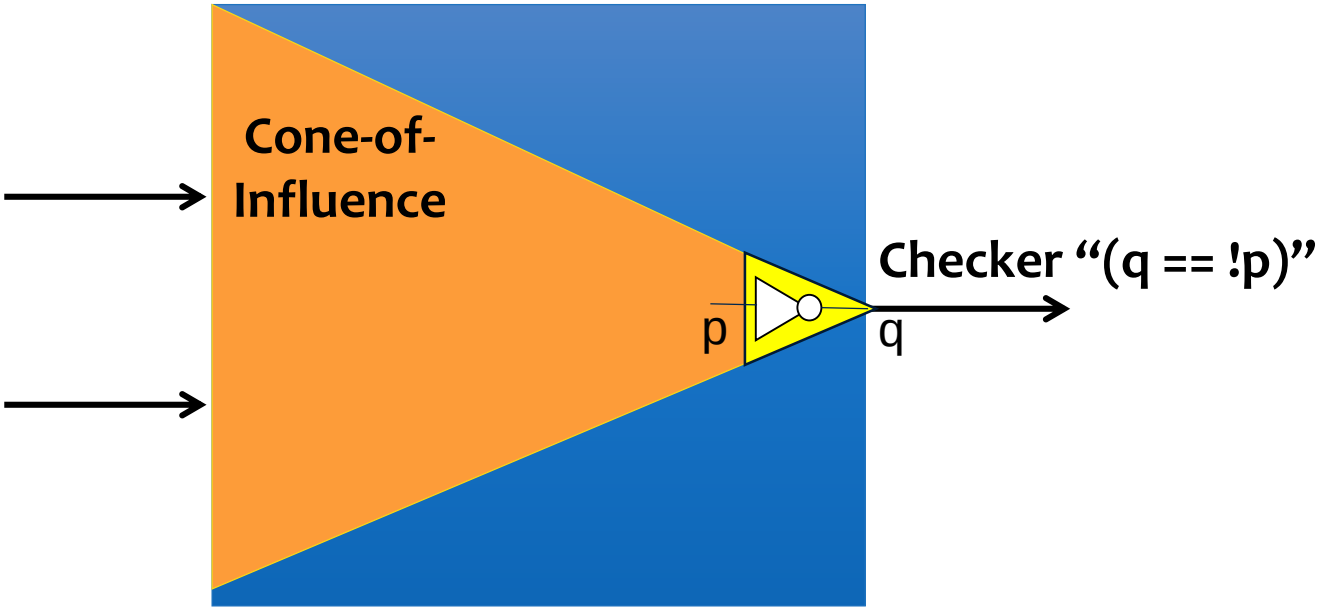
- COI is cheap to compute, but is conservative



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Cone-of-Influence (COI) Coverage

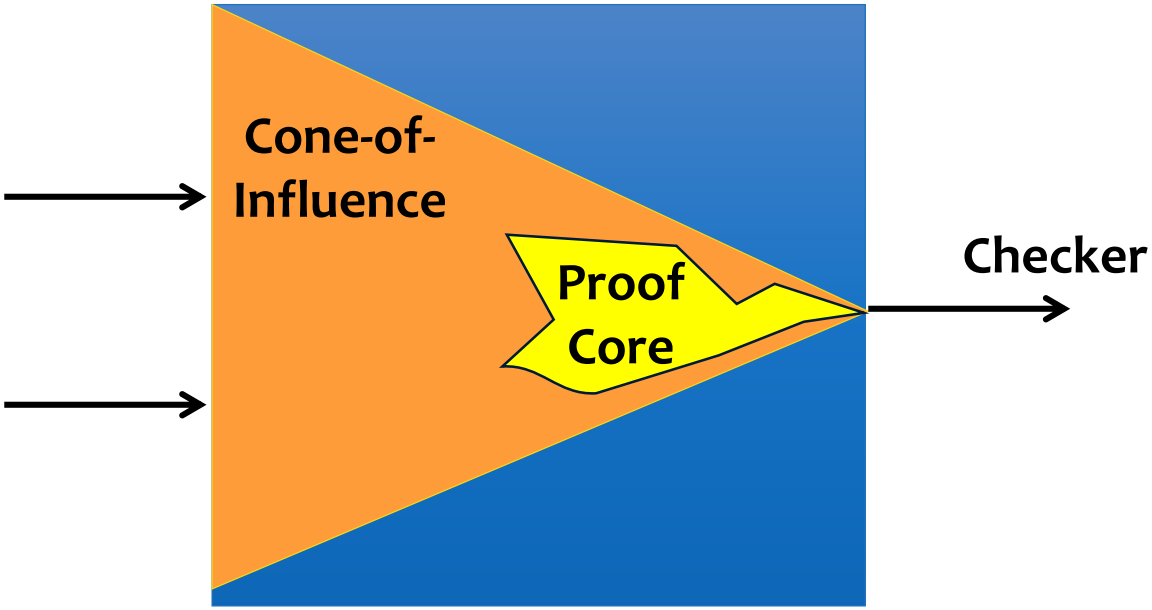
- Would like to cover only the logic needed for the proof



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

COI vs Proof Core (for Checkers)

- Proof Core is more expensive to compute, but is almost "exact"
  - Granularity is assignment block (not fine-grained), like COI, not Reachability
  - May be slightly conservative sometimes (not so much in practice)
    - Not much conservative in practice
    - 100% coverage does not mean no bugs, so "almost exact" is "good enough"



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

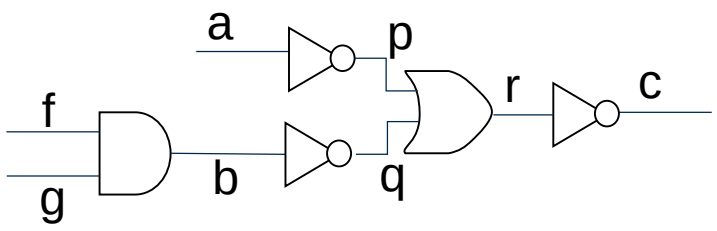
COI vs Proof Core Trade-offs

	COI	Proof Core
Cost to compute	Small (netlist traversal)	Larger (must finish proofs)
Value	Some	A lot

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Example – Proof Core

```
reg b, p, q, r, c;
always @(*) begin
  b = f && g;

  if (a) p = 1'b0;
  else p = 1'b1;

  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```

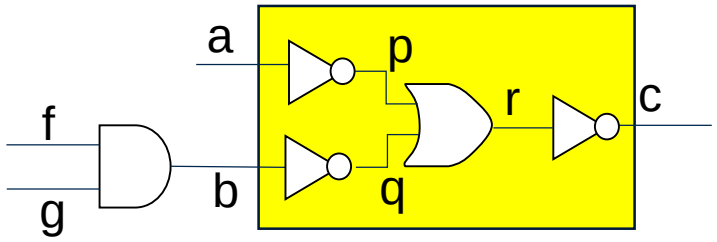



```
prop_a_and_b:
assert property(@(posedge clk)
disable iff (rst)
   (c = (a && b)));
```


COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Example – Proof Core

```
reg b, p, q, r, c;
always @(*) begin
  b = f && g;

  if (a) p = 1'b0;
  else p = 1'b1;

  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```

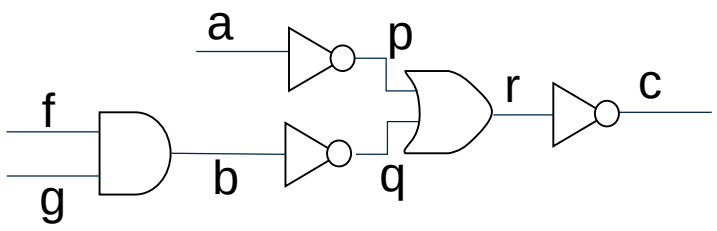



```
prop_a_and_b:
assert property(@(posedge clk)
disable iff (rst)
   (c = (a && b)));
```


COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Example – Proof Core

```
reg b, p, q, r, c;
always @(*) begin
  b = f && g;

  if (a) p = 1'b0;
  else p = 1'b1;

  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```



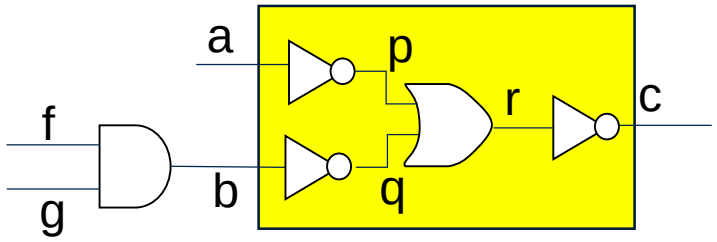

```
prop_a_and_b:
assert property(@(posedge clk)
disable iff (rst)
   (c = (a && b)));
```

```
prop_a:
assume property(@(posedge clk)
disable iff (rst)
   a);
```

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Example – Proof Core

```
reg b, p, q, r, c;
always @(*) begin
  b = f && g;

  if (a) p = 1'b0;
  else p = 1'b1;

  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```




```
prop_a_and_b:
assert property(@(posedge clk)
disable iff (rst)
   (c = (a && b)));
```

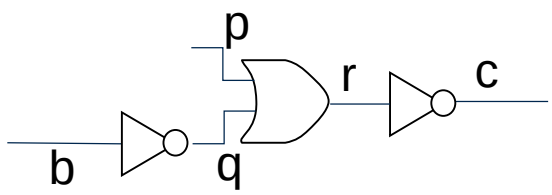
```
prop_a:
assume property(@(posedge clk)
disable iff (rst)
   a);
```

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Reachability Coverage

- In N cycles (depth of the shallowest proof):
  - Which coverage targets are reached?
  - Which coverage targets cannot be reached (including provably unreachable)?
  - Which are undetermined? (likely none within the Proof Core)
- Unless total vacuity (constraints conflict), every assignment block will have something covered!

```
reg q, r, c;
always @(*) begin
  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```

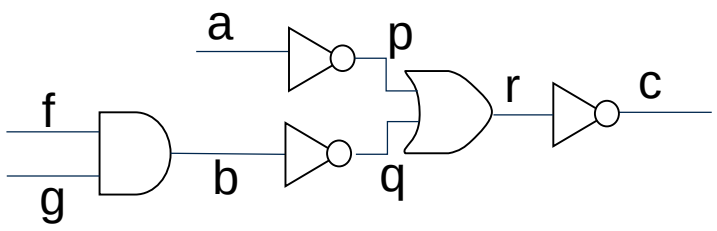



```
prop_a:
assume property(@(posedge clk)
disable iff (rst)
   b);
```


COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Example – Reachability

```
reg b, p, q, r;
always @(*) begin
  b = f && g;

  if (a) p = 1'b0;
  else p = 1'b1;

  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```



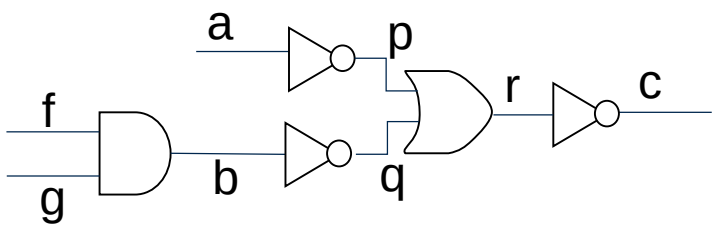

```
prop_a_and_b:
assert property(@(posedge clk)
disable iff (rst)
   (c = (a && b)));
```

```
prop_a:
assume property(@(posedge clk)
disable iff (rst)
   a);
```

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Example – Reachability

```
reg b, p, q, r;
always @(*) begin
  b = f && g;

  if (a) p = 1'b0;
  else p = 1'b1;

  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```



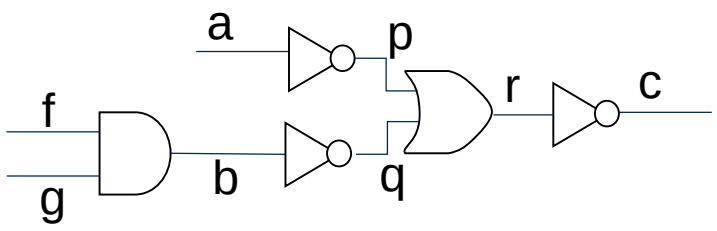

```
prop_a_and_b:
assert property(@(posedge clk)
disable iff (rst)
   (c = (a && b)));
```

```
prop_a:
assume property(@(posedge clk)
disable iff (rst)
   a);
```

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Example – Reachability

```
reg b, p, q, r;
always @(*) begin
  b = f && g;

  p = !a;

  if (b) q = 1'b0;
  else q = 1'b1;

  r = p || q;

  if (r) c = 1'b0;
  else c = 1'b1;
end
```

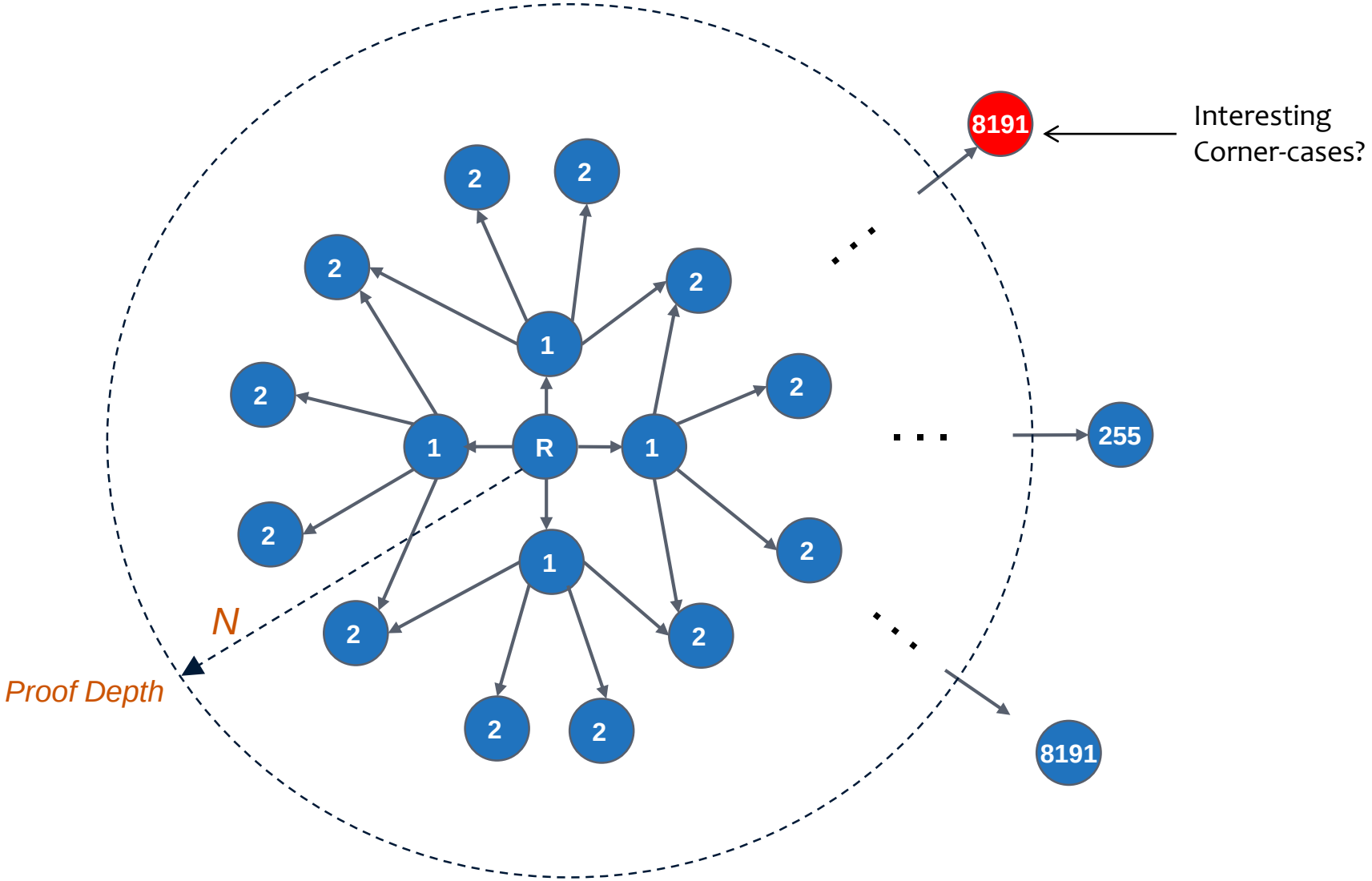



```
prop_a_and_b:
assert property(@(posedge clk)
disable iff (rst)
   (c = (a && b)));
```

```
prop_a:
assume property(@(posedge clk)
disable iff (rst)
   a);
```


COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

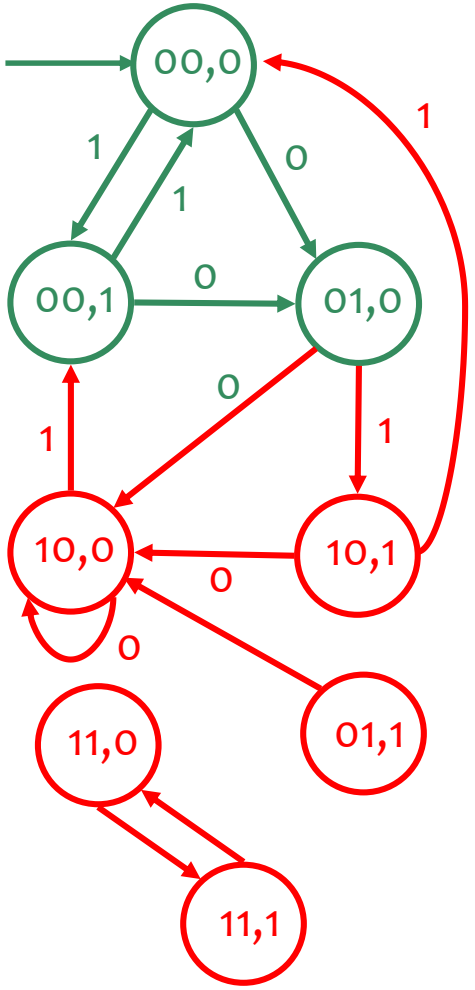
Formal Covers All State Transitions Within Proof Depth



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Formal Coverage (depth = 1)

```
input a;
reg b;
reg [1:0] st;

always @(posedge clk or negedge rst)
  if (~rst) st <= 2'b00;
  else case( st )
    2'b00: if (~a) st <= 2'b01;
    2'b01: st <= 2'b10;
    2'b10: if (a) st <= 2'b00;
    endcase

always @(posedge clk or negedge rst)
  if (~rst) b <= 1'b0;
  else if (~a | b) b <= 1'b0;
  else b <= 1'b1;
```



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

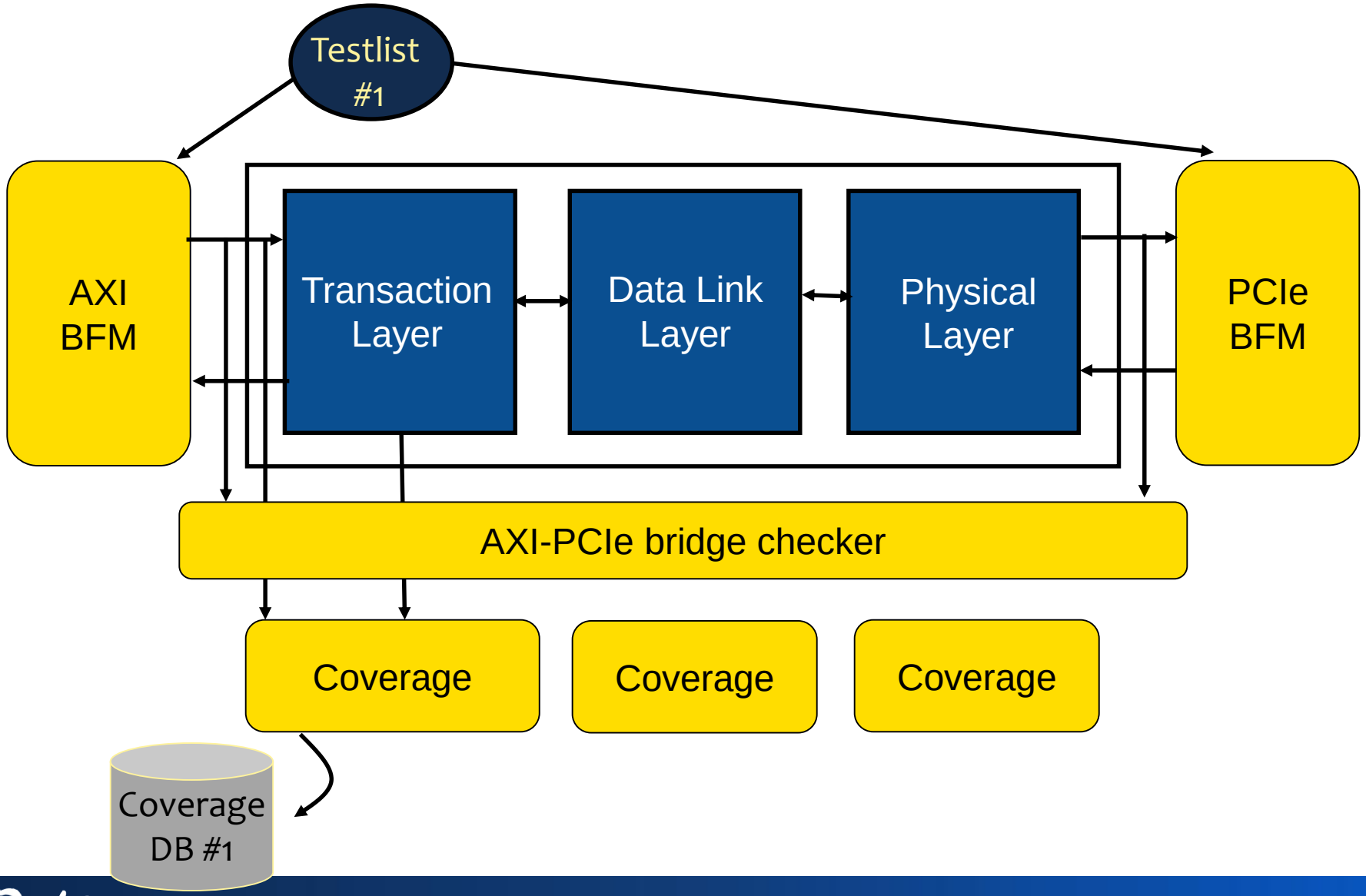
COI vs Proof Core Trade-offs

	COI	Proof Core	Reachability
Checkers	✔	✔ ✔	
Constraints		✔	✔ ✔
Complexity (Proof Depth)		✔	✔ ✔

- Best coverage option
  - Intersection of Proof Core and Reachability (up to N clocks)
    - Outside either means not used in verification, or not exercisable
  - If constraints are not mature (illegal inputs)
    - Proof Core is smaller than final Proof Core (a subset of logic is enough for CEX)
    - Reachability is larger than final Reachability (illegal transitions are possible)
    - Can start with COI

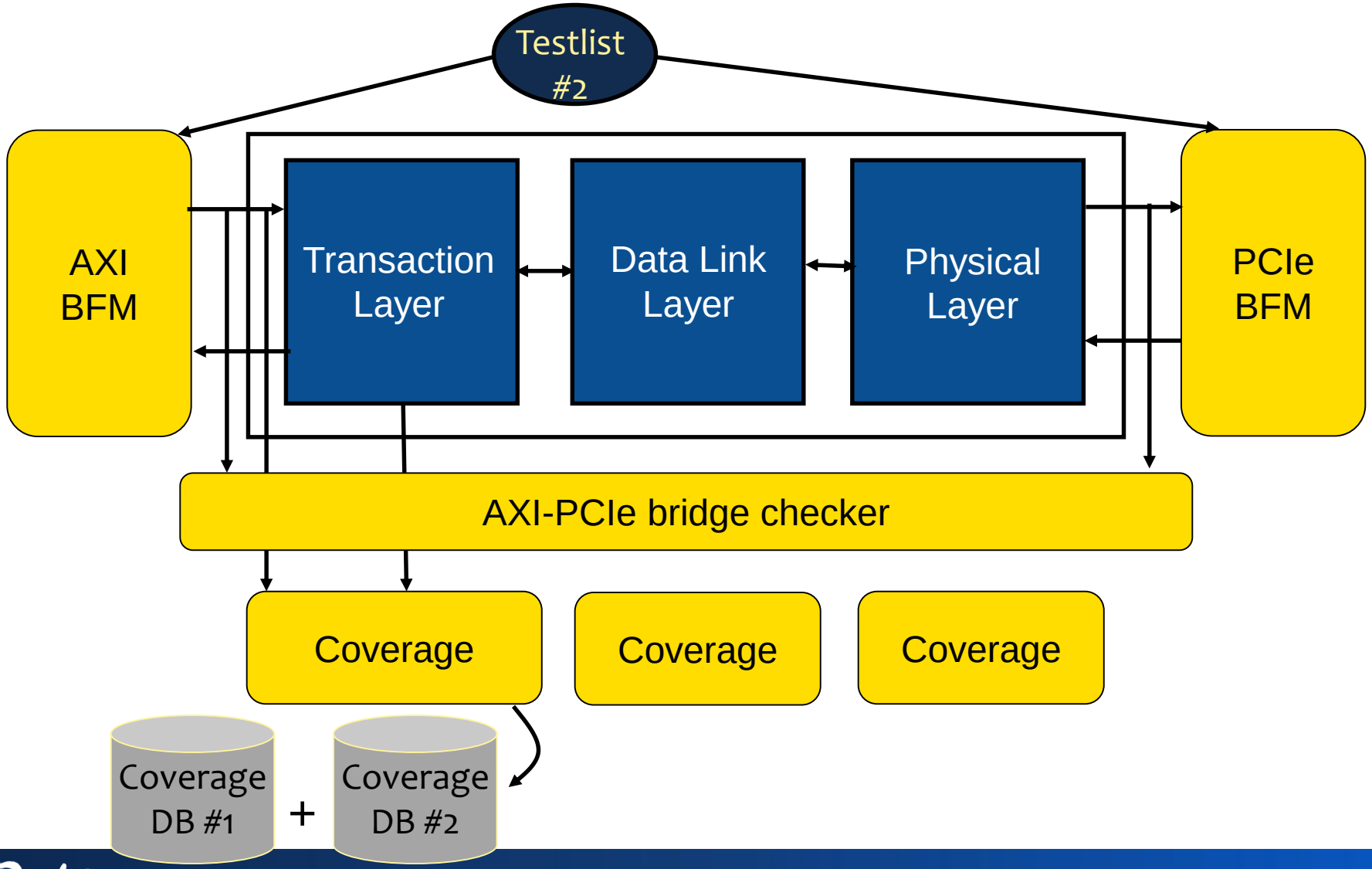
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Simulation Coverage Collection



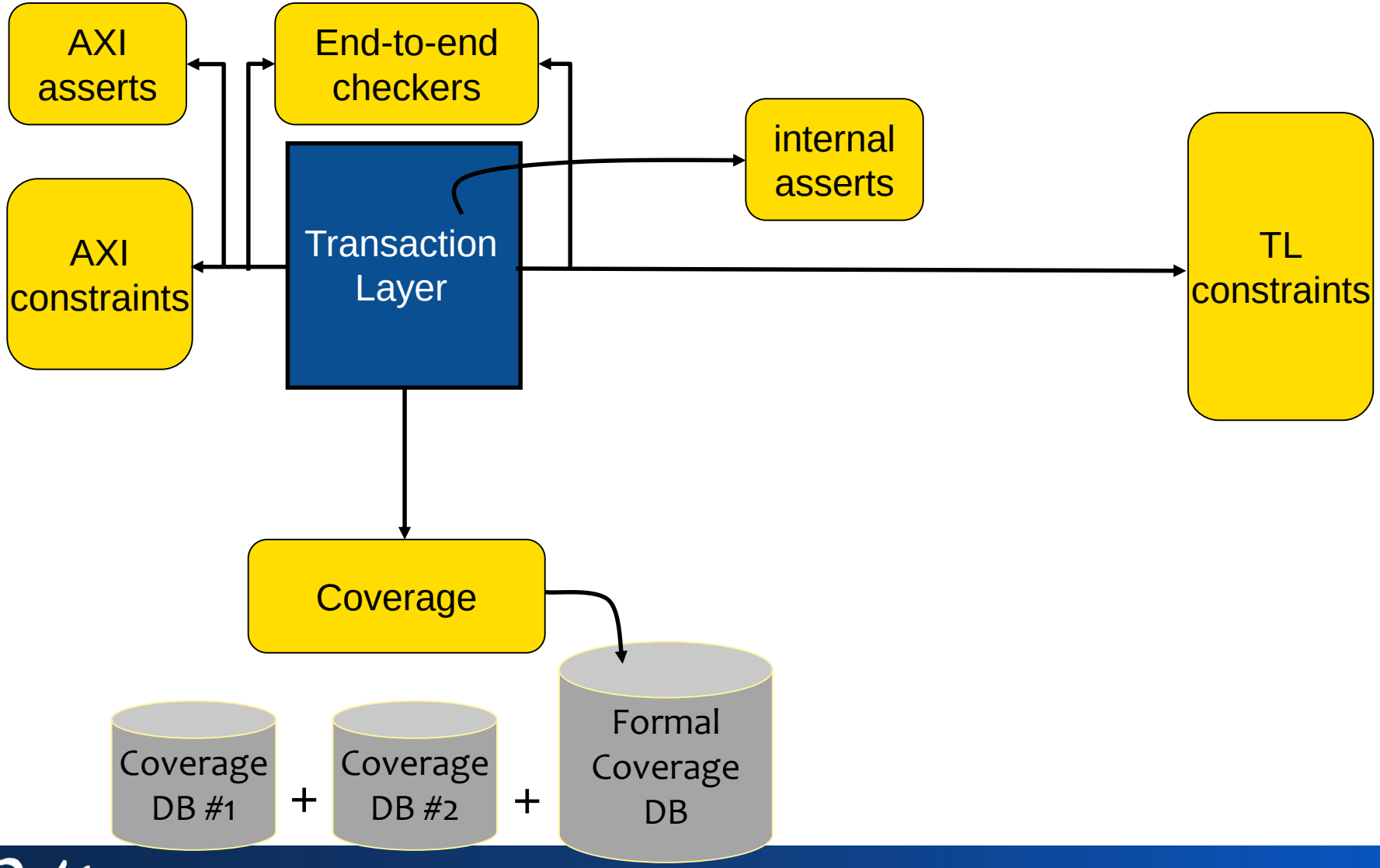
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Simulation Coverage Collection



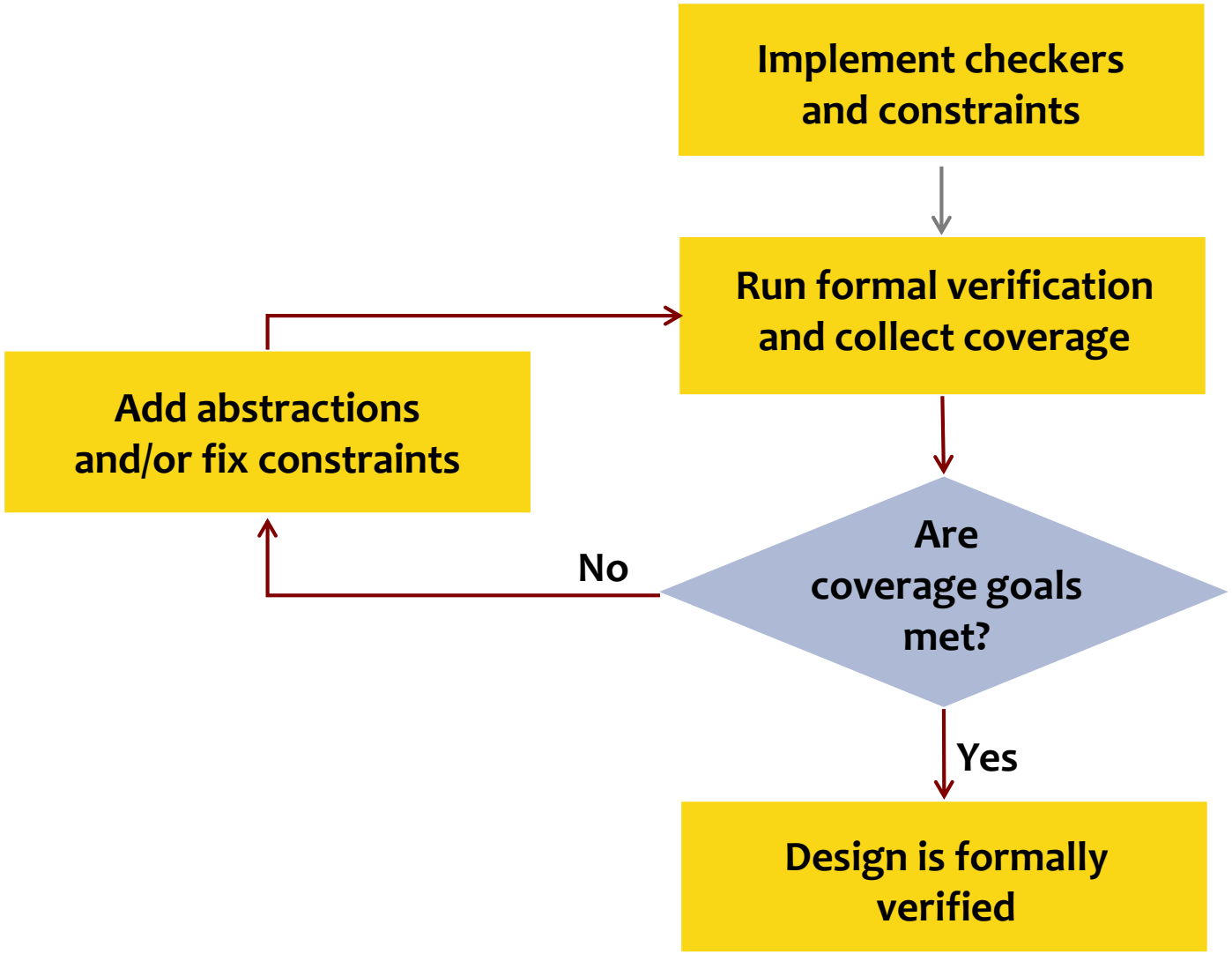
COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Merged with Formal Coverage – Some Day…



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Formal Coverage Methodology



COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Where Formal Coverage Is Important for Formal Sign-off

- Constraints – check for over-constrained environment
  - Intentional: avoided some hard to model, or verify, input combinations
  - Unintentional: bugs in constraints; forgot to remove intentional over-constraints
- Complexity – sign-off on bounded proof
  - All checkers are verified up to proof depth N
  - Any target that is not reachable in N clocks is not covered
- Checkers – does not verify completeness of Checkers
  - No different than simulation!

COPYRIGHT © 2011-15, OSKI TECHNOLOGY, INC.

Thank You!

Unique Methodology. Highest Coverage. Fastest Time to Market.

Oski TECHNOLOGY