

Using a Formal Property Checker for Simulation Coverage Closure



Tim.Blackmore@infineon.com
David.Halliwell@infineon.com
Phil.Barker@infineon.com



Kerstin.Eder@bristol.ac.uk
Naresh.Ramaram@infineon.com

Overview

- Motivation** - Difficulties with achieving **Coverage Closure** during simulation-based testing
- Describe how a **Formal Property Checker** can be used to aid Coverage Closure
 - Based on Temporal Induction
 - Give example properties

Coverage Holes

- Code Coverage** indicates how thoroughly the testbench has exercised the source code
- Analyzing Coverage holes** determines whether more tests are needed, or if that code is unreachable

```
343      if (A && B)
344  512      X = X + 2 ;
345      else if (C)
346  **0**    X = X + 3;
347      else
348  417      X = X + 4;
```

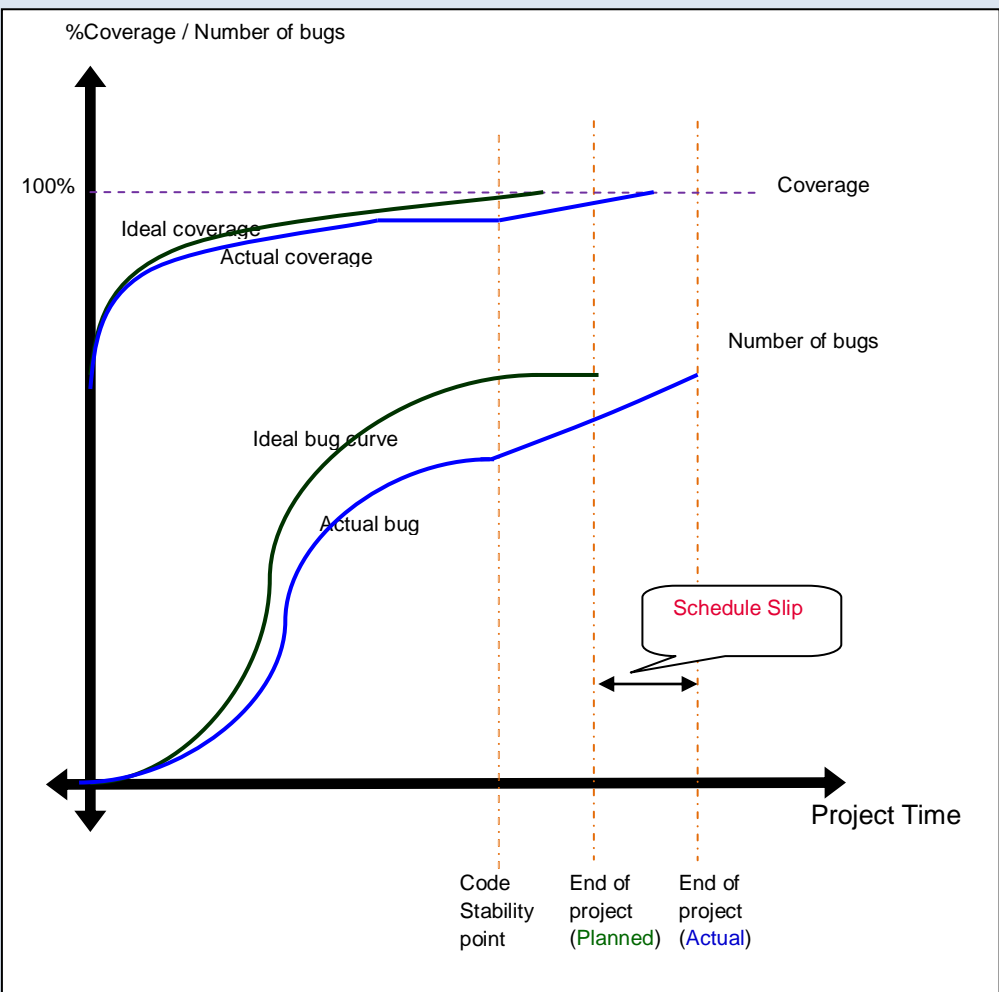
Figure1: RTL code with coverage hole. Holes are usually identified by line number.

```
343      if ( MODE == 2 )
344      begin
345      if (A && B)
346  512      X = X + 2 ;
347      else if (C)
348  **0**    X = X + 3;
349      else
350  417      X = X + 4;
351      end
```

Figure2: After bug fix, line numbers have changed. The hole needs to be re-analysed.

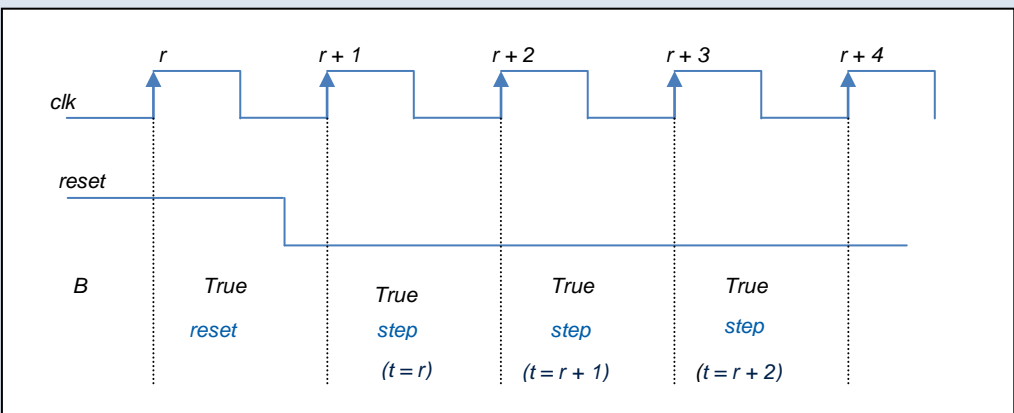
Coverage Closure

- Coverage analysis** is left until the code stability point
- Bugs** found during Coverage closure potentially result in schedule slips



Methodology

- Based on **Temporal Induction**
- To prove a behaviour (**B**) for a synchronous design
 - Prove **B** at reset (**reset** property)
 - Prove that if **B** is true at time **t**, then it is true at **t+1** (**step** property)It follows that the behaviour is always true
- Simple properties over very small time windows
- Usually run in seconds on a (bounded) property checker



Property Generation

```
343      if (A && B)
344  512      X = X + 2 ;
345      else if (C)
346  **0**    X = X + 3;
347      else
348  417      X = X + 4;
```

```
branch_unreachable_346_reset;
branch_unreachable_346_step_t;
```

```
macro branch_346
  !( A && B ) && C;
endmacro;
```

```
property
branch_unreachable_346_step_t =
  !branch_path_346 =>
    next(!branch_path_346) ;
endproperty
```

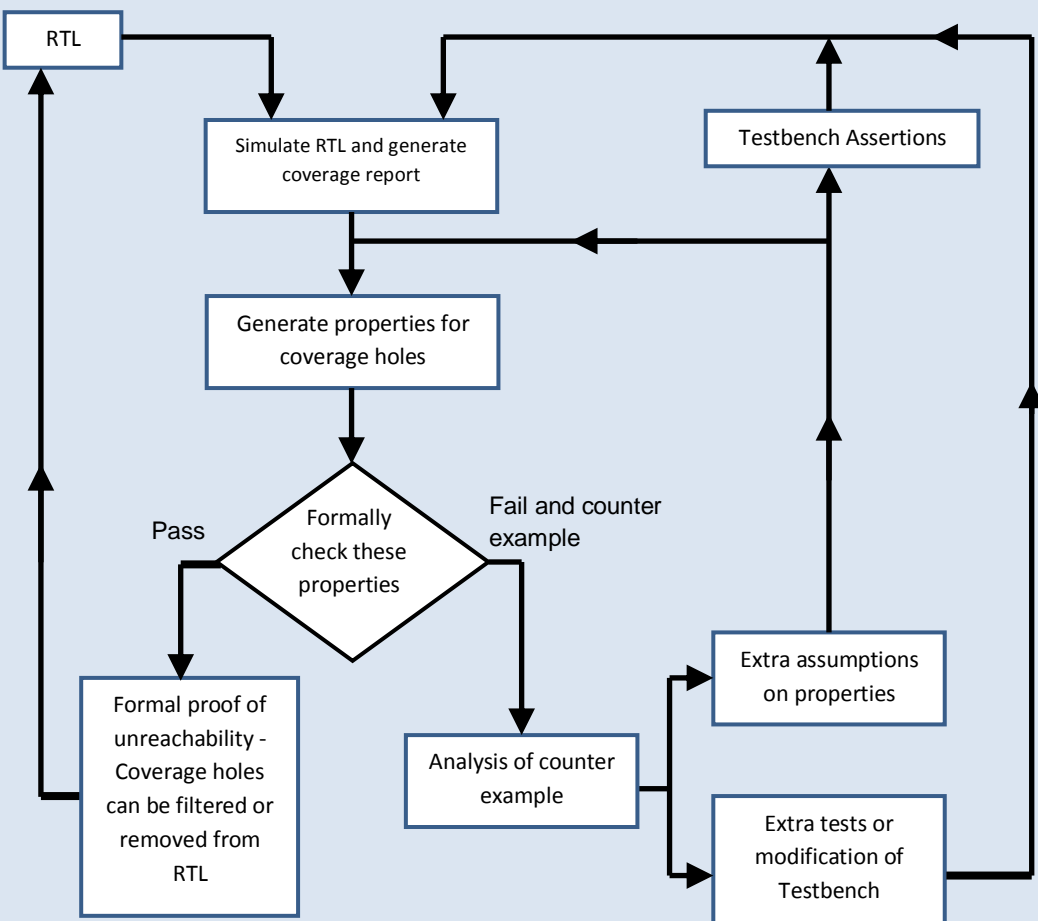
- Properties generated by script for each coverage hole
- No need to wait for code stability to start running scripts and analyse remaining holes

```
343      if ( MODE == 2 )
344      begin
345      if (A && B)
346  512      X = X + 2 ;
347      else if (C)
348  **0**    X = X + 3;
349      else
350  417      X = X + 4;
351      end
```

```
branch_unreachable_348_reset;
branch_unreachable_348_step_t;
```

```
macro branch_348
  MODE ==2 && !(A && B ) && C;
endmacro;
```

Flow



Results

Coverage Metric	Total	Coverage holes during simulation	Coverage holes excluded by Methodology
Statement	41074	331	309
Branch	12341	353	334
FEC	27230	1581	1080

Advantages

Versus Traditional Method

- Robust against RTL changes – can start earlier
- Use machine time instead of engineering time
- Less prone to human error as exclusions formally proven
- Counter-examples help fill coverage holes efficiently
- Extend use of formal methods to non-experts

Versus 'Off-the-shelf' tools

- Runtime is hours rather than days
 - Only run properties for coverage holes rather than all code
- White box approach, so can make properties more powerful
 - Include branch nesting and prioritisation
 - Add assumptions to all properties reflecting testbench constraints