



{ assertive DESIGN }

DesignCon 2006



DesignCon 2006

A Comparative Study on the Effectiveness of Automated Assertions over Traditional Methods in a Project Design Flow

DesignCon 2006



What happens when a design has a complete set of functional assertions?

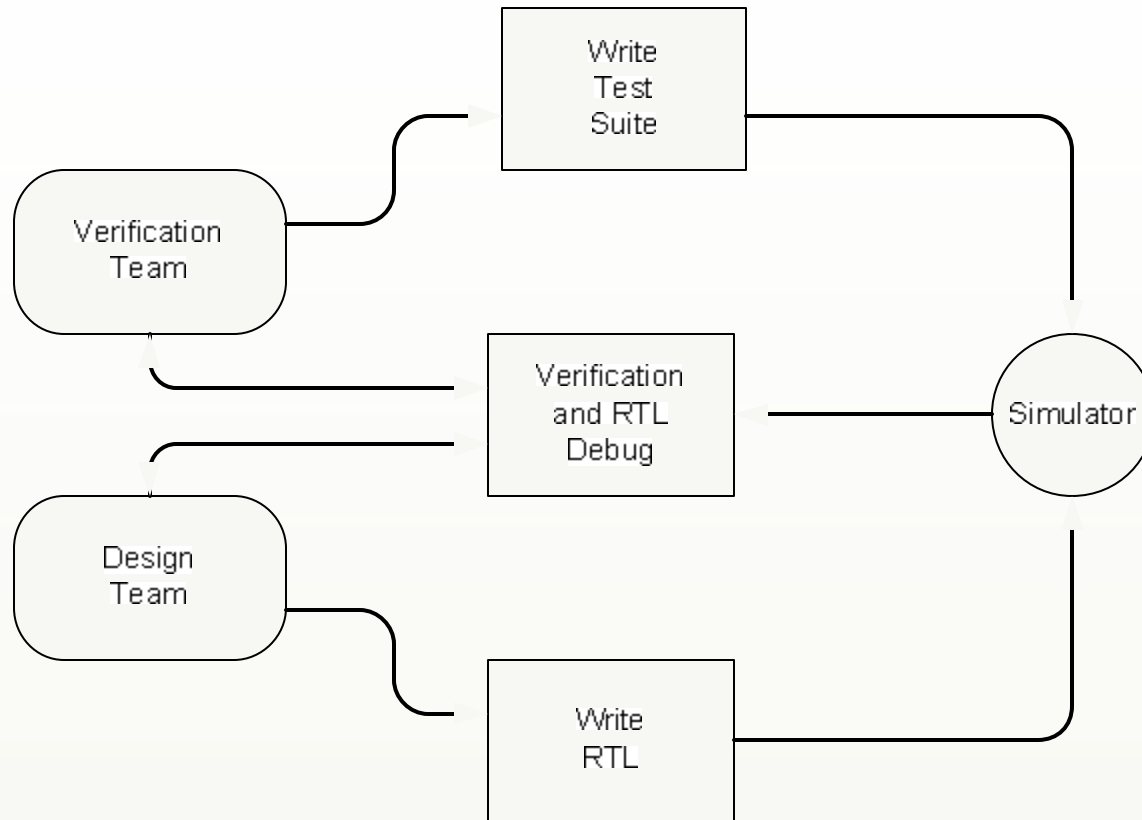


Introduction

- Examined 3 digital design projects
 - All could compare standard EDA tool flow to automated assertion flow
- All had comparable RTL models
 - Rewrite projects
 - Existing RTL design used as one model
 - New assertion-based design ported from existing RTL
 - Some added features
 - Redesign projects
 - Existing RTL design used as one model
 - New assertion-based design built from specification
- All projects had some level of existing verification environment



Traditional Tool Flow Diagram

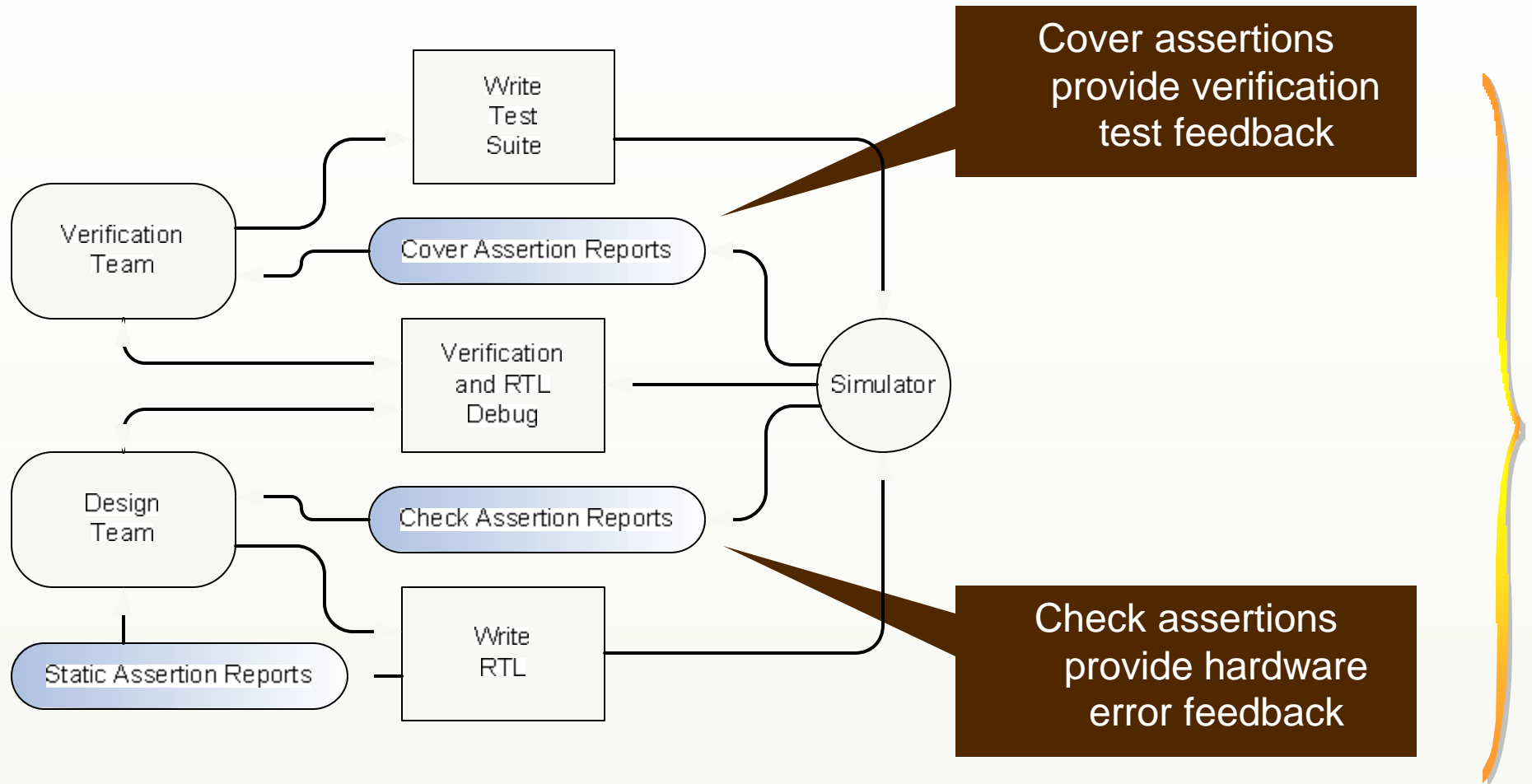


Assertion Flow Introduction

- DesignPSL
 - RTL hardware design tool
 - Automatically generates functional assertions
 - Check assertions
 - Cover assertions
 - Fits into existing design flows
 - Mix and match with existing RTL



Assertion Flow Diagram



Comparison of Projects

- What happens to a project when assertions are inserted into the RTL.
 - Check assertions: about 2 per line of RTL
 - Cover assertions: about 1 per line of RTL
- That information is available to design and verification teams dynamically
- Look at differences between traditional and assertion based methods.



Static Assertions

- Detected at compile time
- Look a lot like syntax errors – only fire on issues that will result in simulation errors
- Currently around 10% of assertions



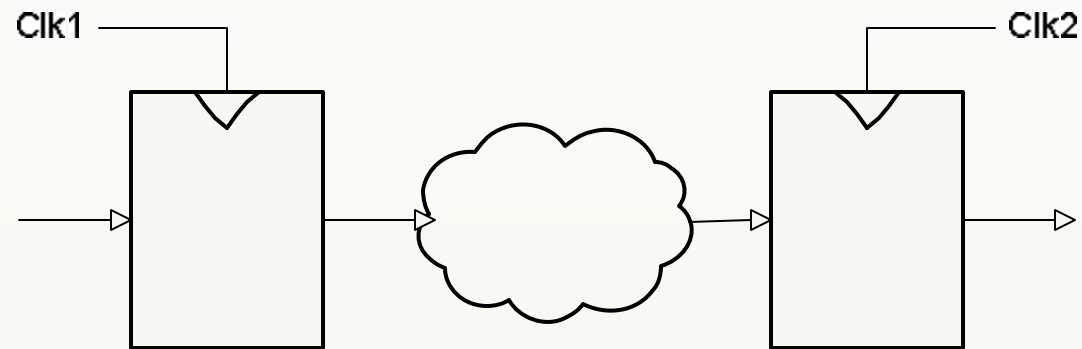
Static Assertion Distribution

Assertion Type	Percentage
Sizing Assertions	55%
Clock Crossing	30%
Illegal Conditions	10%
Out of Scope	2%



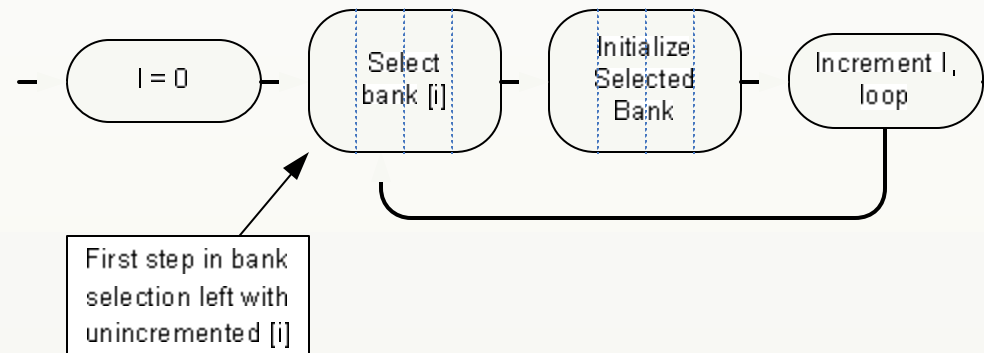
Static Assertion Example - 1

- Clock crossing
 - signals that “leak” across a clock boundary
 - using the wrong reset on a register



Static Assertion Example - 2

- Out of scope signals
 - signals used outside of an expected range
 - tough to find in simulation (may actually pass all tests)



Dynamic Assertions

- Require simulations to trigger
- Find discrepancies even when simulation is examining other conditions
 - dynamic assertions fired on passing simulations
- Assertions are classified by type




Dynamic Assertion Distribution

Assertion Type	Percentage
Transition Failure	15%
Default Max Repeat	14%
Optional Boolean	11%
Sere-OR Multiple Path	9%
Sere-OR No Path	9%
Unknown Condition	7%
Register Collision	3%



Types of errors detected

- Early errors:
 - Forgot to set or clear signals (transition)
 - State machine hangs (max repeat)
 - Missing initialization (unknown)
 - Mis-wired ports (unknown, deviation)
 - Later errors:
 - Incorrect equation (deviation)
 - State machine interactions (collision)
 - Protocol mismatches (deviation, SERE, transition)
 - Architectural issues (optional boolean)
- 

Assertion vs test failure reports

- During initial debug
 - all assertions, all the time – the majority of errors
- After initial debug
 - assertions fire, test hangs.
- During later debug
 - assertion firing and test failure
 - assertion fires immediately, test reports failure later
 - test failed without assertion firing
 - on outside boundaries of design
 - assertions fired without test failure
 - functionality that wasn't tested



Functional Coverage Overview

- Measures activity in all areas of design
- Categorized and prioritized so that users aren't flooded with information
- Dynamically available to randomized tests



Coverage Categorization

- State Coverage
 - Really really need to hit this
- Transition Coverage
 - Really need to hit this
- Boundary Condition Coverage
 - Important
 - Back-to-back operations
 - Min/max values hit
- Path Coverage
 - Case by case




Functional Coverage Purpose


- See what is happening in simulation runs
 - What has a test done
 - Is a random test making forward progress
 - Is it time to stop
 - What additional tests are needed
 - Is it time to change constraints
- Grade test quality

Traditional RTL Functional Coverage	DesignPSL state & transition
98%	82%
97%	23%


Using Dynamic Coverage

- Easy:
 - Determine when to finish a test – no more forward progress on functional coverage
 - Hard:
 - Determine what hasn't been covered and fix/add tests
 - Hardest:
 - Watch for specific coverage information, and dynamically change test constraints to steer test
- 

Functional Coverage Data

- Massive amount of data is generated (billions of assertions firing in most tests)
 - Categorization of data to manage importance and relevance
 - Contextual display of data
 - millions of lines of text are not useful
- 

Unexpected Coverage Usage

- Designer visibility into hardware failures
 - on failures, designers often started by bringing up coverage plot to see how they got there
 - became a hardware debug tool – how did it go wrong?
 - Dead code finder
 - Verification view into hardware design
 - Verification environment debugger
 - visibility into coverage after each test – did the test do what it was supposed to do?
- 

Project Results

- Assertions result in significantly steeper bug-open rates and bug-close rates.
 - none of the projects are ideal for comparisons
- Assertions find bugs that get past traditional simulations
- Assertions are useful for both hardware and verification environment debug
 - 15% of assertions fired due to test issues
- Coverage has more uses than we expected





assertive DESIGN

DesignCon 2006

