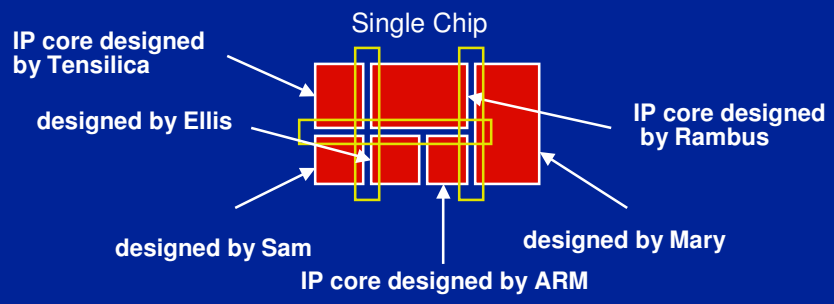
Deriving a Simulation Input Generator and a Coverage Metric From a Formal Specification

Kanna Shimizu and David L. Dill

Stanford University, Stanford, CA

Formal Specification?



1. Interfaces need to be clearly defined and communicated
2. Interface protocols need to be thoroughly debugged

Formal interface specifications would be optimal

Formal vs Informal

Informal Specification

Target subsequent latency is the number of clocks from the assertion of IRDY# and TRDY# for one data phase to the assertion of TRDY# or STOP# for the next data phase in a burst transfer. The target is required to complete a subsequent data phase within eight clocks from the completion of the previous data phase. This requires the target to complete the data phase either by transferring data (TRDY# asserted), by doing target Disconnect without data (STOP# asserted, TRDY# deasserted), or by doing Target-Abort (STOP# asserted, DEVSEL#

Official PCI Specification

Formal Specification

```
prev(!final_dphase_done & !idle)
        IMPLIES cbe_e;
prev(frame) IMPLIES (frame | irdy);
prev(stop & irdy) IMPLIES !stop;
prev(stop & irdy) IMPLIES !irdy;
prev(!frame) IMPLIES (frame | ad);
prev((counter = 3) & a) IMPLIES !a;
prev((m_initial = 7) & !irdy)
        IMPLIES irdy;
prev((m_subseq =7) & !irdy)
        IMPLIES irdy;
```

Example

Changes Needed

- **Cost Goal:** Decrease formal specification cost
  - writing & debugging
  - shorter development time
  - less expertise required

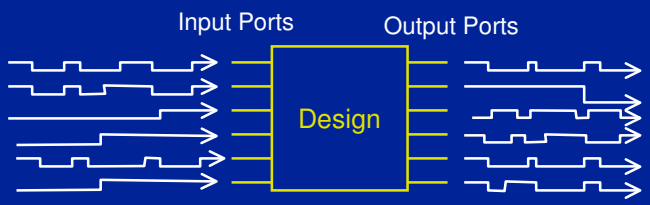
FMCAD '00, CHARME '01

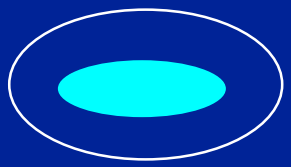
- **Value Goal:** Increase formal specification value
  - beyond debugging the protocol?
  - assist with simulation?

DAC '02

Verify Design

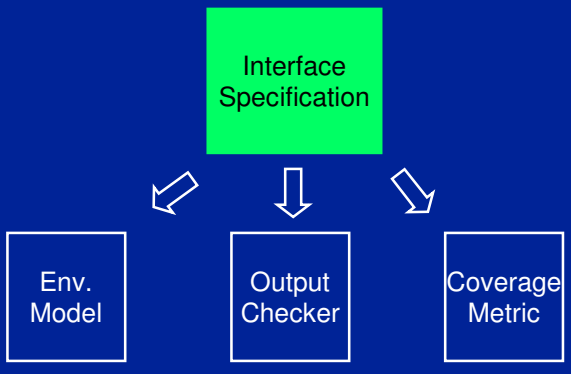
Many tools are needed for simulation and functional verification





Coverage metric?

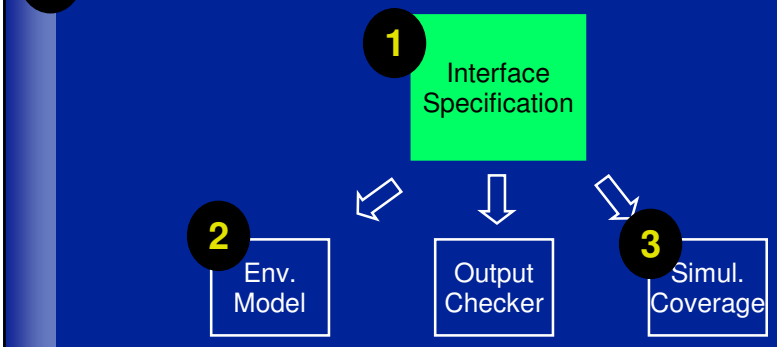
Unified Framework Approach



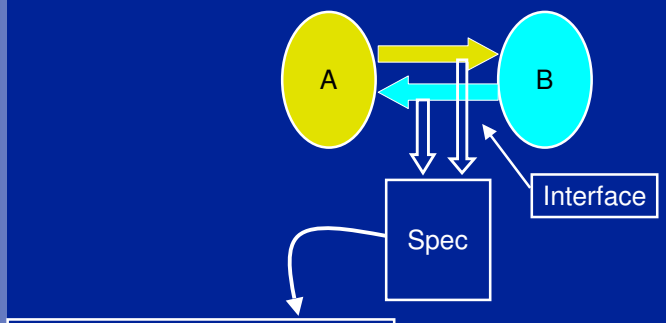
1. Black Box & Automatic → Saves Time & Effort
2. Fewer bugs in tools
3. Change in interface, not a problem!

Outline

1. Specification Style
2. Environment Model
3. Simulation Coverage
4. Experimental Results



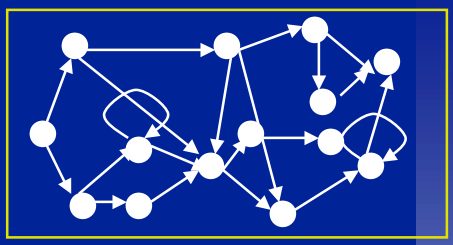
1. Properties-Based Specification Style




```
previous(a0 & b0) IMPLIES a0
previous((counter=8)) IMPLIES a1
previous(a1) IMPLIES b0 | b1 | b2
```

1. Properties-Based Specification Style

- previous(frame) IMPLIES (frame | irdy);
- previous (stop & irdy) IMPLIES !stop;
- previous (stop & irdy) IMPLIES !irdy;
- previous (!frame) IMPLIES (frame | ad);
- previous ((counter = 3) & a) IMPLIES !a;



many compact properties

one large complex state machine

O

X

2. "Previous Implies Current" Specification Style

All properties must be

previous expression IMPLIES current expression

- previous(frame) IMPLIES (frame | irdy);
- previous (stop & irdy) IMPLIES !stop;

Activating Logic

Constraining Logic

2. “Previous Implies Current” Specification Style

previous(a0) IMPLIES a0 | a1

!previous(a0) | a0 | a1

previous(a0) & !a0 IMPLIES a1

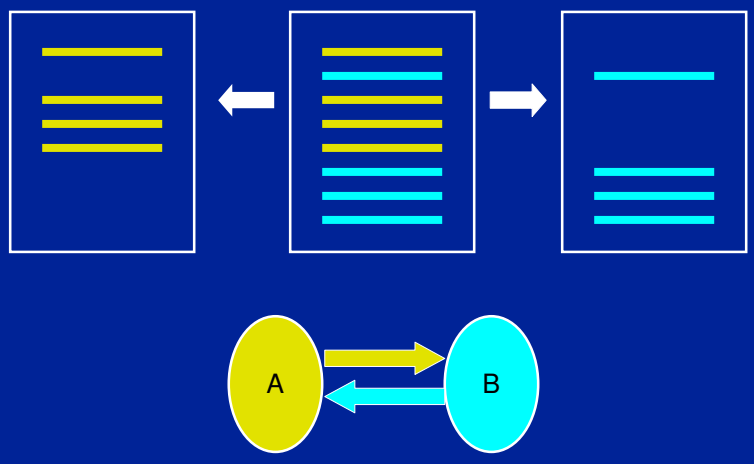
“Previous Implies Current” Form

Free Form

O

X

3. Output Separability Specification Style



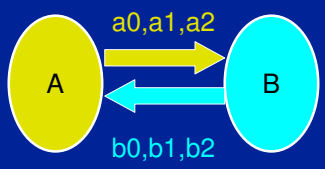
3. Output Separability

previous(a0 | b0) IMPLIES (a0 | a1)
previous(counter=8) IMPLIES (a1 & a2)
previous(b0) IMPLIES (b0 | !b2)

O

previous(a0) IMPLIES (a0 | b1)
previous(counter=8) IMPLIES (!a0 | !b0)
previous(b0) IMPLIES (a1 & b1 & b2)

X

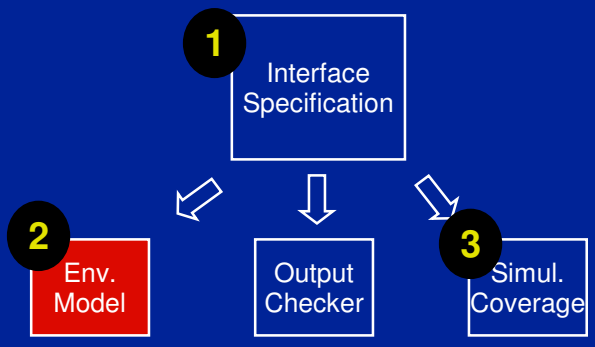


Summary

1. Properties-Based
2. “Previous Implies Current”
3. Output Separability

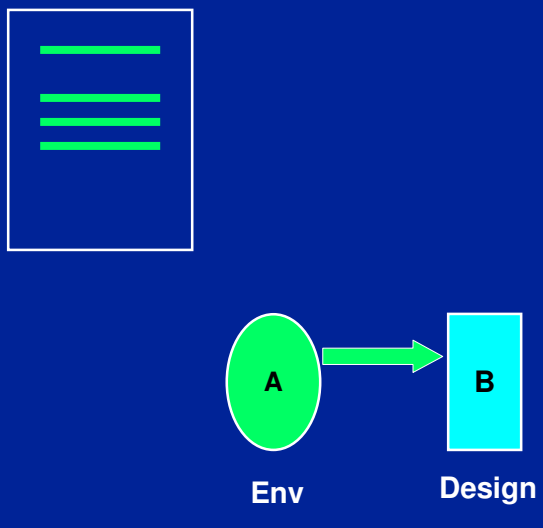
Outline

1. Specification Style
2. Environment Model
3. Simulation Coverage
4. Experimental Results



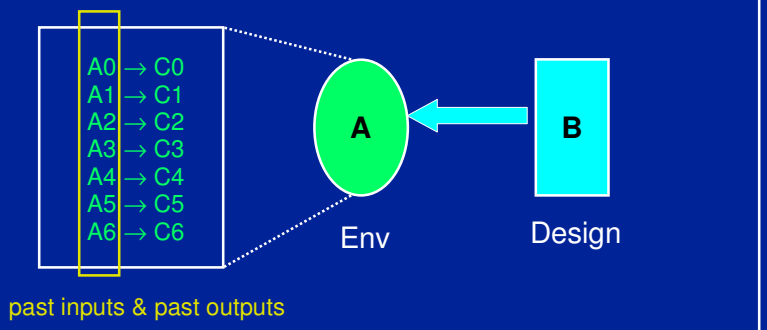
Preparation Step

Environmental Model



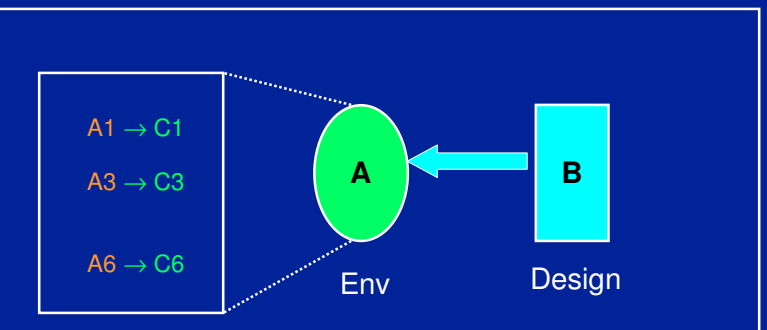
Dynamic, Repeated Step

Environmental Model



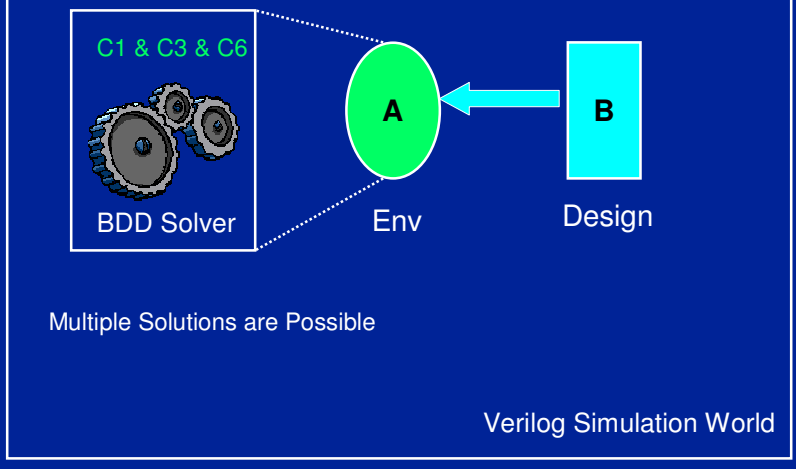
Dynamic, Repeated Step

Environmental Model



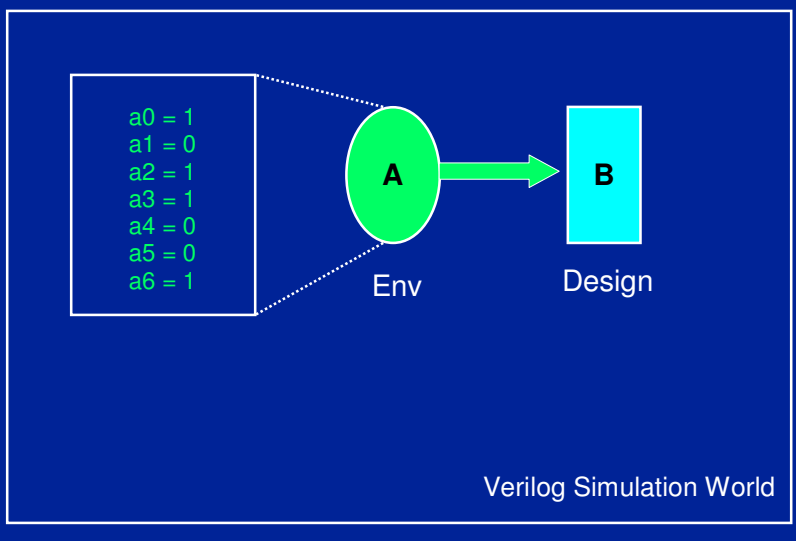
Dynamic, Repeated Step

Environmental Model

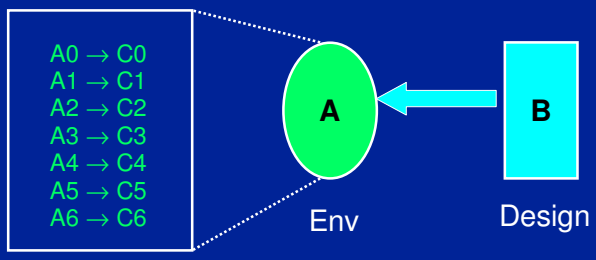


Dynamic, Repeated Step

Environmental Model

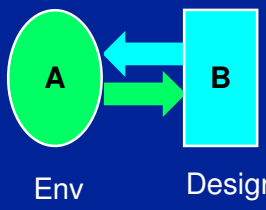


Dynamic, Repeated Step



Verilog Simulation World

Summary

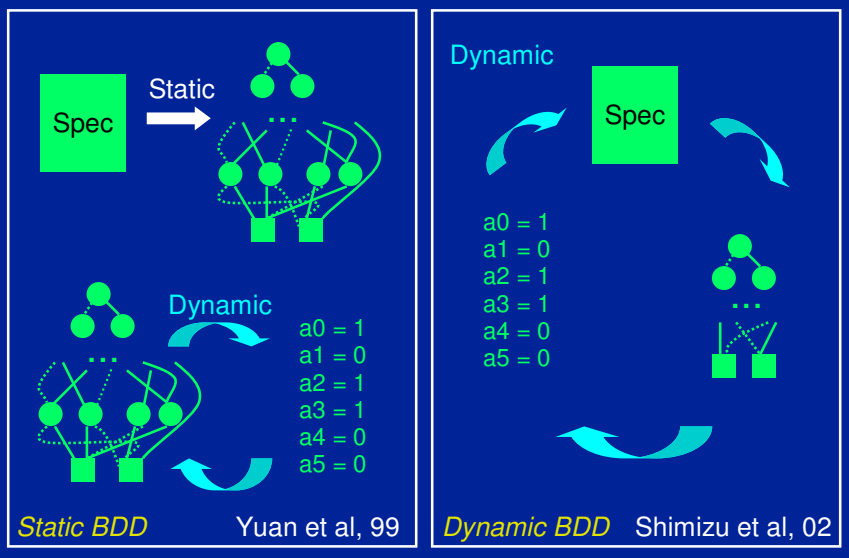


1. Reactive
2. Complies with protocol

Verilog Simulation World

Previous Work

Environmental Model



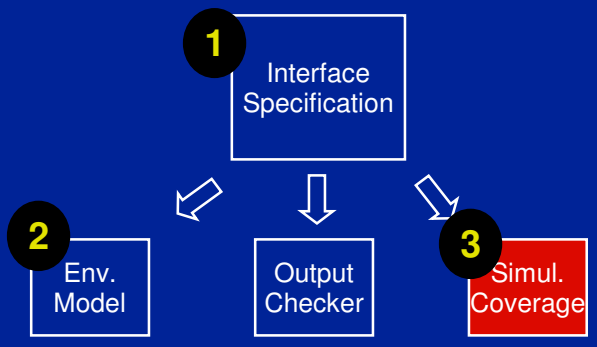
Advantages

Environmental Model

1. Much smaller BDD
   1. Only activated properties
   2. Only consequents
      - BDDs are purely of output variables
      - Example : BDD variables from 161 to 15
2. Technique useful for large interfaces
   - Avoid BDD blowup

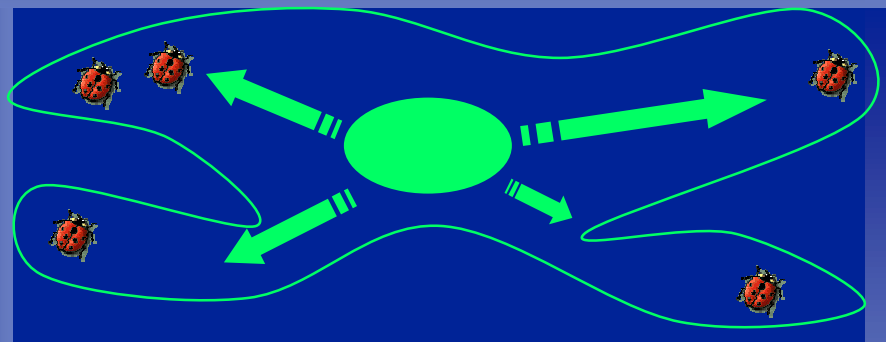
Outline

1. Specification Style
2. Environment Model
3. Simulation Coverage
4. Experimental Results



"Interesting" Scenarios

Coverage & Bias



1. Determining interesting scenarios
2. Push the simulation to that scenario

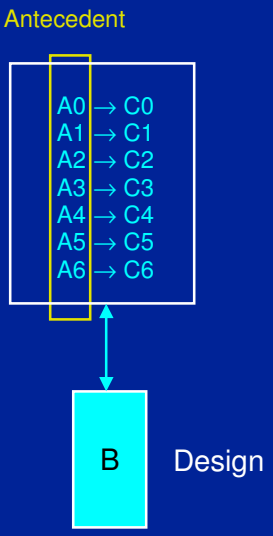
Proposed Coverage Metric

1. Determining interesting scenarios

previous(counter=8) IMPLIES (a1 & a2)

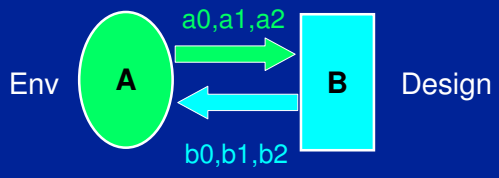
antecedent true== interesting scenario

Goal: Maximize the # of antecedents that have become true



Determine Input Biases

2. Push the simulation to that scenario



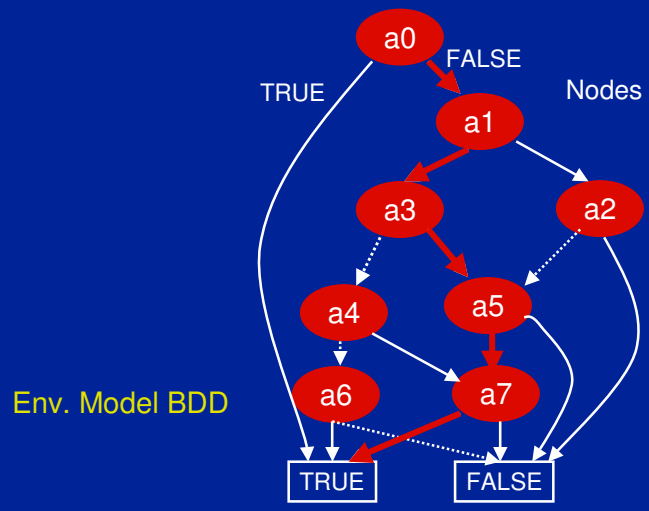
previous(!a0 & a1 & !b0) IMPLIES (b1)

a0: True 2%

a1: True 98%

Without Biasing

Coverage & Bias



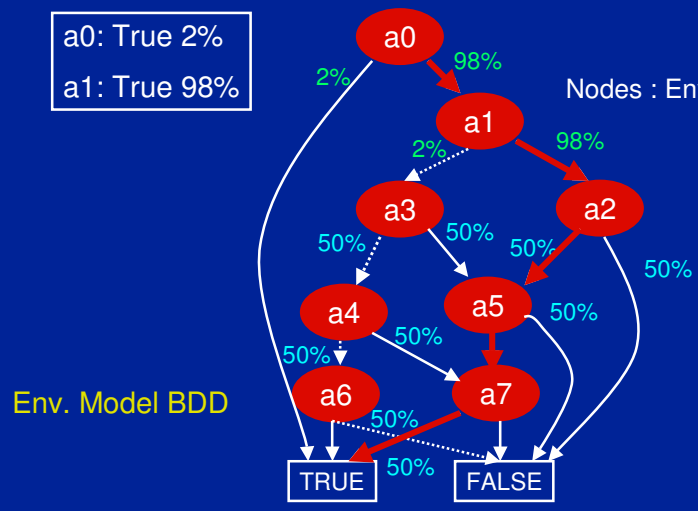
Env. Model BDD

With Biasing

Coverage & Bias

a0: True 2%

a1: True 98%

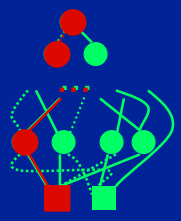


Env. Model BDD

Summary

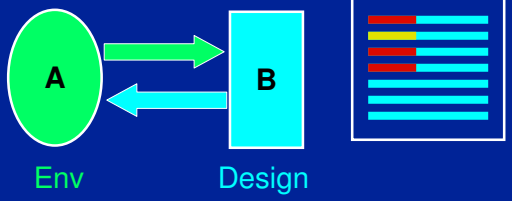
2. Bias Env's outputs

a0 : 98%
a1 : 2%
a5 : 98%



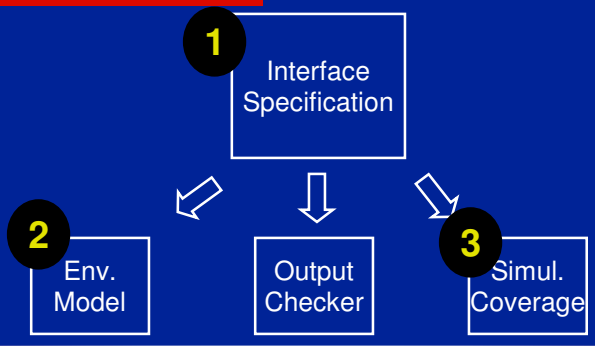
Biased Traversal

1. Target a missed antecedent



Outline

1. Specification Style
2. Environment Model
3. Simulation Coverage
4. Experimental Results

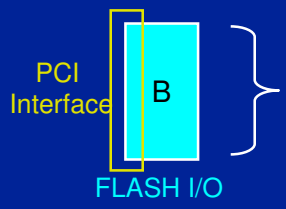


The Design

Experimental Results

Stanford FLASH I/O Design

- fabricated and functioning chip -

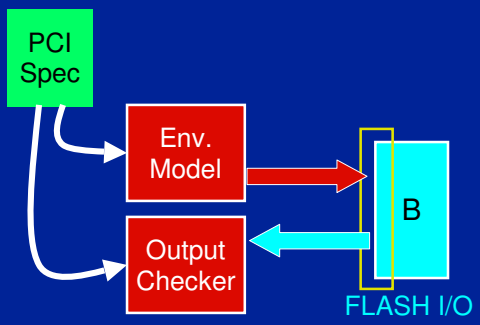


- 8000 lines of Verilog
- 283 (>1 bit) variables

Verified by Govindaraju et al, '00

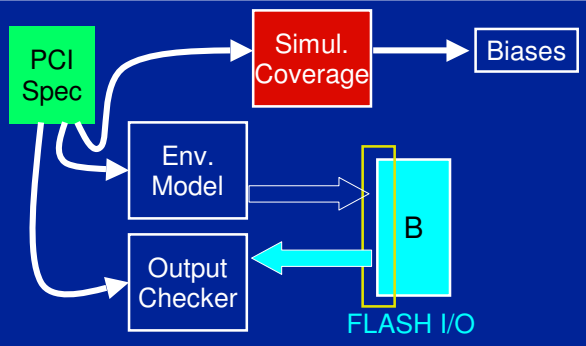
Simulation & Verification

Experimental Results



- 9 previously unreported bugs found
- Finding bugs: automated.

Coverage & Bias



Most bugs found during biased simulations

Performance Results

Memory usage small

- only 15 variables in BDDs (161 without technique)
- peak memory usage 4MB

Simulation time not a problem

- 12,000 time steps / run
- For all runs < 2 sec

Summary

- Methodology: Specification as an...
  1. Environment
  2. Output checker
  3. Coverage metric
  4. Bias determinator
- New Generation Algorithm: Smaller BDDs
- Application to a Large Design & Discovery of Bugs

Future Work

Further Uses for Specifications

- Better Coverage Metric
- Augmenting incomplete interface designs
- Better synthesis?