Introduction

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# deasserted).

Official PCI Specification

Example

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prev(!final_dphase_done &amp; !idle) IMPLIES cbe_c;</td>
<td>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.</td>
</tr>
<tr>
<td>prev(frame) IMPLIES (frame</td>
<td>irdy);</td>
</tr>
<tr>
<td>prev(stop &amp; irdy) IMPLIES !stop;</td>
<td></td>
</tr>
<tr>
<td>prev(stop &amp; irdy) IMPLIES !irdy;</td>
<td></td>
</tr>
<tr>
<td>prev(!frame) IMPLIES (frame</td>
<td>irdy);</td>
</tr>
<tr>
<td>prev((counter = 3) &amp; a) IMPLIES !a;</td>
<td></td>
</tr>
<tr>
<td>prev((m_initial = 7) &amp; !irdy) IMPLIES irdy;</td>
<td></td>
</tr>
<tr>
<td>prev((m_subseq =7) &amp; !irdy) IMPLIES irdy;</td>
<td></td>
</tr>
</tbody>
</table>

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

Introduction

Many tools are needed for simulation and functional verification.

Input Ports  Output Ports

Correct?

Coverage metric?

Unified Framework Approach

Introduction

1. Black Box & Automatic → Saves Time & Effort
2. Fewer bugs in tools
3. Change in interface, not a problem!
1. Specification Style
2. Environment Model
3. Simulation Coverage
4. Experimental Results

1. Properties-Based Specification Style

A

B

Spec

Interface

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

many compact properties

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

<table>
<thead>
<tr>
<th>previous(a0) IMPLIES a0</th>
<th>a1</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous(a0) &amp; !a0 IMPLIES a1</td>
<td></td>
</tr>
</tbody>
</table>

“Previous Implies Current” Form

Free Form

[Diagram with symbols O and X]

3. Output Separability Specification Style

[Diagram with symbols A and B and arrows showing interaction]

Copyright by Kanna Shimizu 2002
3. Output Separability

### Specification Style

<table>
<thead>
<tr>
<th>Previous Conditions</th>
<th>New Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>previous(a0</td>
<td>b0) IMPLIES (a0</td>
</tr>
<tr>
<td>previous(counter=8) IMPLIES (a1 &amp; a2)</td>
<td>previous(counter=8) IMPLIES (la0</td>
</tr>
<tr>
<td>previous(b0) IMPLIES (b0</td>
<td>!b2)</td>
</tr>
</tbody>
</table>

### 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

Env  A  B  Design
Dynamic, Repeated Step

Environmental Model

Verilog Simulation World

A0 → C0
A1 → C1
A2 → C2
A3 → C3
A4 → C4
A5 → C5
A6 → C6

past inputs & past outputs

A

B

Env

Design

Verilog Simulation World

A1 → C1
A3 → C3
A6 → C6

A

B

Env

Design
Dynamic, Repeated Step

Multiple Solutions are Possible

Verilog Simulation World

Copyright by Kanna Shimizu 2002
Dynamic, Repeated Step

Environmental Model

Verilog Simulation World

A0 → C0
A1 → C1
A2 → C2
A3 → C3
A4 → C4
A5 → C5
A6 → C6

Summary

Environmental Model

1. Reactive
2. Complies with protocol

Verilog Simulation World
### Previous Work

**Environmental Model**

- **Static BDD**
  - *Spec*
  - Yuan et al., 99

- **Dynamic BDD**
  - *Spec*
  - Shimizu et al., 02

### 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

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

Env. Model BDD

_nodes: Env's Outputs_

- **a0**: True 2%
- **a1**: True 98%

Nodes: Env's Outputs

TRUE

FALSE

50%

50%

50%

50%

50%

50%
Summary

1. Target a missed antecedent
   - Bias Env’s outputs
     - a0: 98%
     - a1: 2%
     - a5: 98%

   Biased Traversal

Coverage & Bias

Outline

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

Interface
   - Specification
   - Env. Model
   - Output Checker
   - Simul. Coverage
The Design

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

- 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?