Monitor-Based Formal Specification of PCI

Kanna Shimizu, David L. Dill,
Alan J. Hu

Stanford University,
University of British Columbia

Background

How to write a specification for interfaces?

- Hardware Designs
- Pin Level Signaling
- Examples
  - Busses
  - IP core interface

Background

- Specification currently in use? Informal, “English” specifications are,
  - Ambiguous
  - Incorrect
  - Incomplete

Background

- Formal Specification is one possible answer to these problems.
  - Eliminate Ambiguity
  - Automatic Checks for Correctness
    - –Eliminate Contradictions
    - –Closer to Completion
  - Useful

Background

"Formal specifications are hard to write."

Contributions

- With our methodology,
  - Protocols are more easily specified
    - –PCI protocol specified
  - Good debugging strategies
    - –PCI Protocol Bugs Discovered

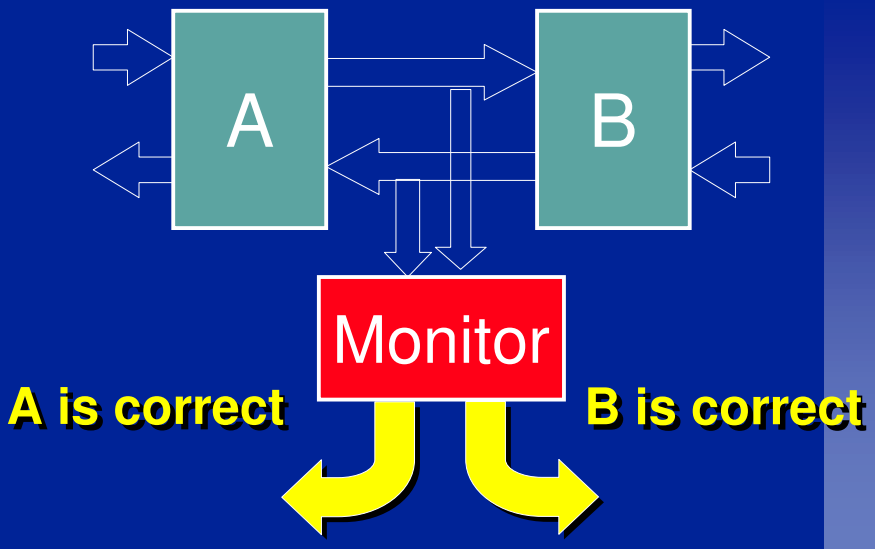
Methodology

1. Specification
2. Debugging

Specification Methodology

Characteristic 1

Specification Methodology — Characteristic 1

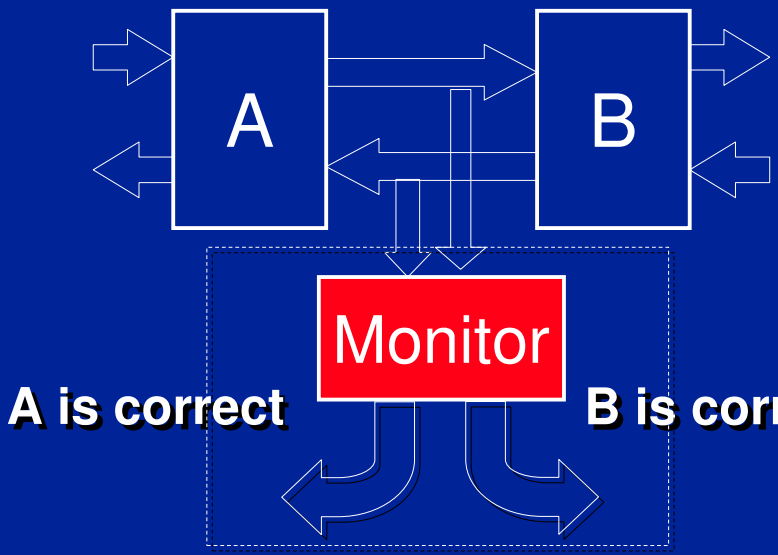


Specification Methodology — Characteristic 1

Usefulness

- Formally verify A without building B
  - Assume-Guarantee Reasoning
- Simulation Checker
  - After implementations of A and B are built

Specification Methodology — Characteristic 1



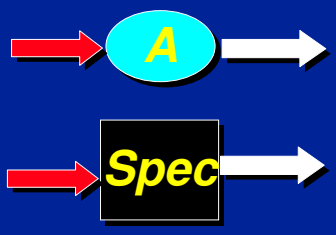
Specification Methodology — Characteristic 1

Traditionally:

Specification's

input: agent input

output: agent output

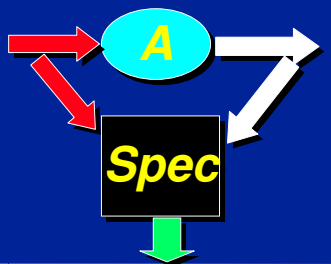


Monitor-Based:

Specification's

input: agent input and agent output

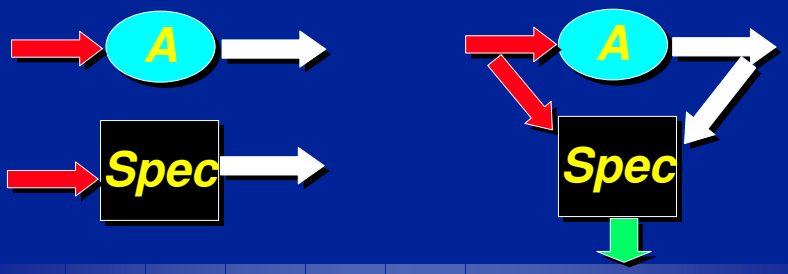
output: correct_i



Specification Methodology — Characteristic 1

Traditionally: Non-Deterministic

Monitor-Based: Deterministic



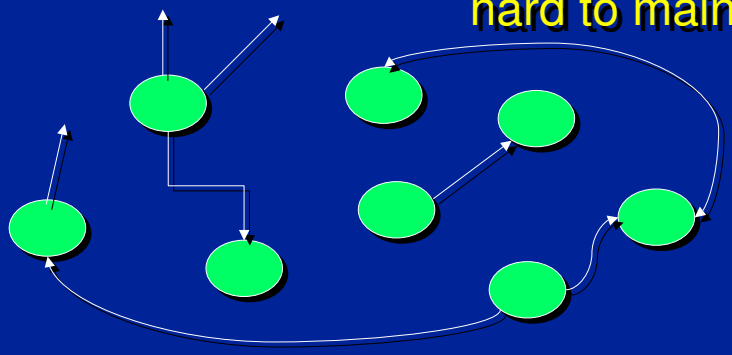
Specification Methodology

Characteristic 2

Specification Methodology — Characteristic 2

Traditionally,

confusing
hard to maintain



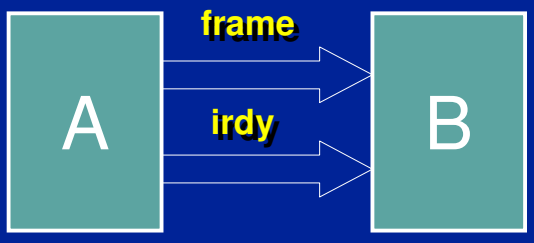
Specifications have used huge state machines

Specification Methodology — Characteristic 2

Specify using small, simple constraints

"Only when IRDY# is asserted can FRAME# be deasserted"

prev(frame) → frame | irdy

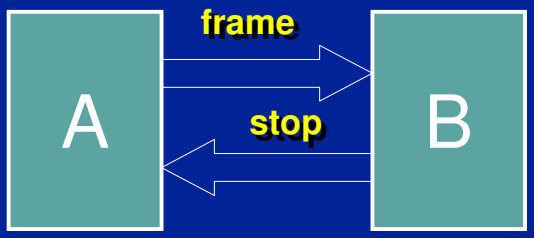


Specification Methodology — Characteristic 2

Specify using small, simple constraints

"Once FRAME# is deasserted, STOP# must be deasserted"

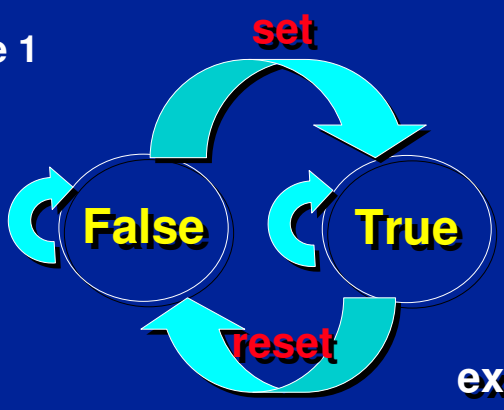
prev(prev(frame) & !frame) → !stop



Specification Methodology — Characteristic 2

Very simple state machines.

Type 1

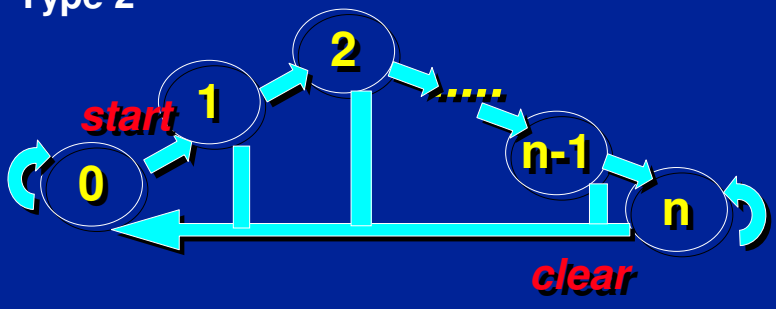


ex:*write_trans*

Specification Methodology — Characteristic 2

Very simple state machines.

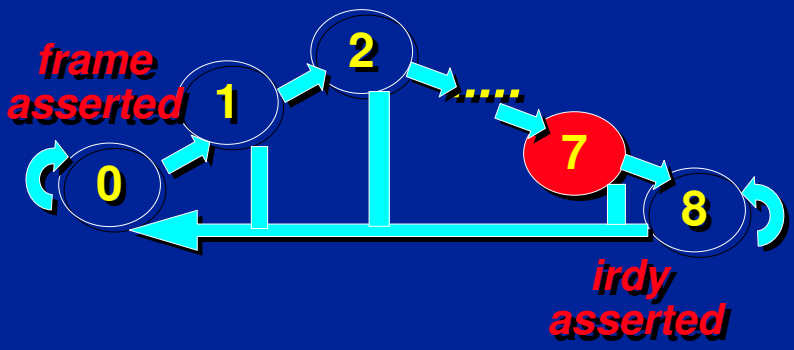
Type 2



Specification Methodology — Characteristic 2

Example

"master must raise irdy within 8 cycles of assertion of frame."



Specification Methodology — Characteristic 2

Example

"master must raise irdy within 8 cycles of assertion of frame."

$$prev((frame8 = 7) \;\&\; !irdy) \Rightarrow irdy$$

Specification Methodology — Characteristic 2

- Specify using small, simple constraints
  - No CTL
  - No LTL
  - No Large State Machines
  - Propositional Formulas + Very Simple State Machines

Specification Methodology

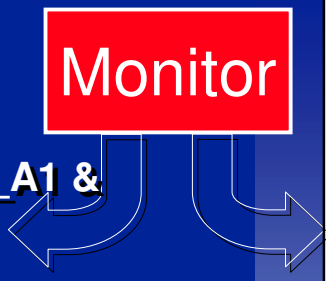
- Write the specification as a interface monitor which observes and judges

AND

- Write the specification as a collection of small, simple constraints

= ?

Specification Methodology



A_is_correct

= constraint_A0 & constraint_A1 & ...& constraint_AN

constraint_Ai = prev(frame) ⟶ frame | irdy

Specification Methodology

Monitor Specification can be written in Verilog!

constraint_Ai = prev(frame) ⟶ frame | irdy

Verilog

```
input frame, irdy;
wire constraint_Ai;
...
constraint_Ai = !p_frame |  (frame | irdy);
```


Specification Methodology

- **Summary**
  - Many, small propositional constraints
  - Group the constraints according to the agent (module) they specify
  - The conjunctions of these constraints are the outputs of the monitor

Specification Methodology

- PCI Result
  - Covers the core protocol excluding the “bus bridge” features. (sec. 3.1 to 3.6)
  - 83 constraints
  - Specification is 280 lines long.
  - Just 2 kinds of state machines

Methodology

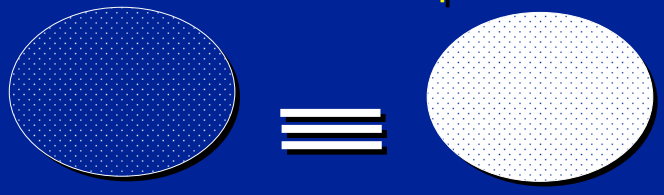
1. Specification
2. Debugging

Debugging

Background

What does it mean to "debug a specification"?

Sets of Event Sequences



Correct by Intention

Correct by Specification

Debugging

Background

Traditionally

Build Implementations and Simulate

Observe and Judge

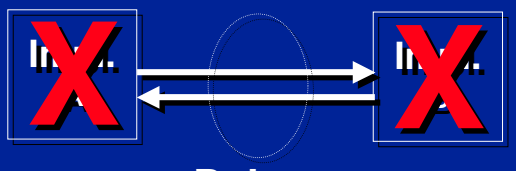
Time-Consuming!
Labor-Intensive!

Debugging

Background

With monitor-based specification

Specification checked *without* implementations



Debug Protocol

Specification debugged early in the design phase!

Debugging

Background

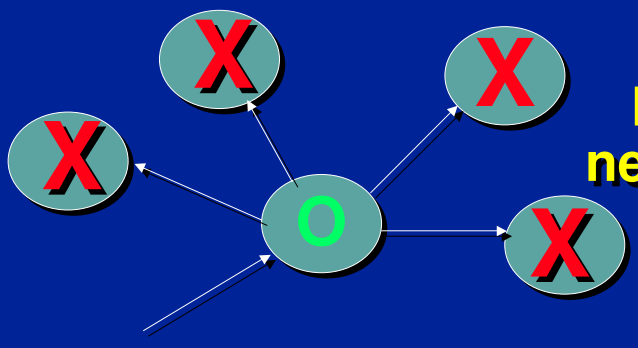
Debugging Monitor-Based Specifications

- 2 types of checks
- Using conventional model checkers
  - Cadence SMV

Dead State Check

Finds contradictions in the specification.

- Dead State Check
  - Assurance that the specification doesn't allow the following:



No legal next action!

Debugging Monitor-Based Specification — Check 1

- Dead State Check
  - Model Check
    for every possible state allowed by the specification,
    there is at least one legal next state.
  - Counterexamples
    ⇒ Conflicting Constraints

Debugging Monitor-Based Specification — Check 2

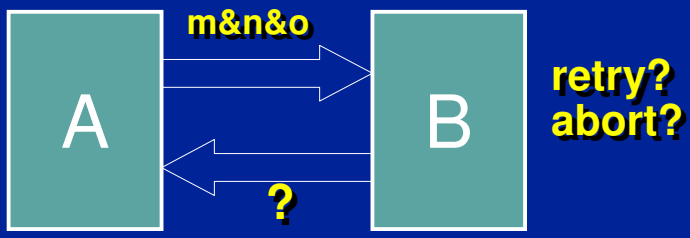
Characteristic Check

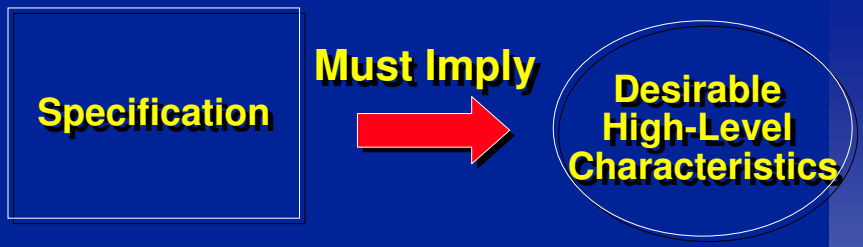
- Finds unexpected scenarios
  – finds "missing" rules or "excessive" rules

Characteristic Check Example

- "Termination signaling must be disjoint"

retry = m&n
abort = n&o

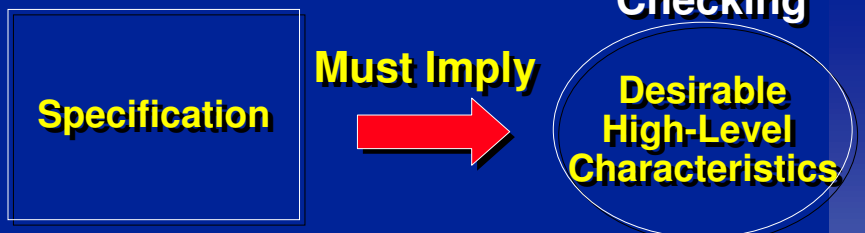




Debugging Monitor-Based Specification

Check 2

Checking



Simple Propositional Formulas

CTL, LTL

Debugging Monitor-Based Specification

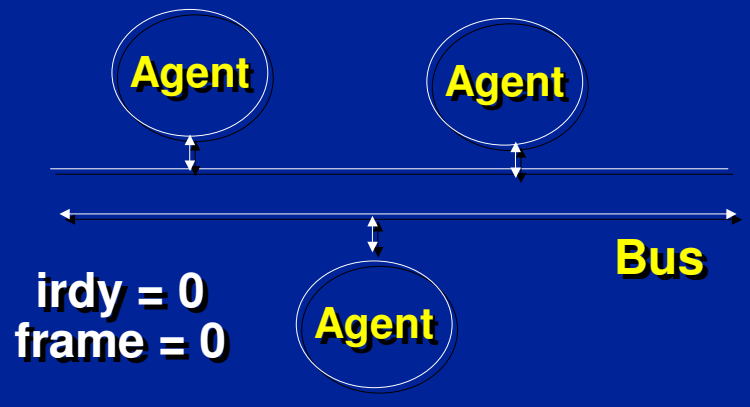
Summary

- Dead State Check
  - finds contradictions
  - easy for non-FV engineers
- Characteristic Check
  - finds missing and excessive rules
  - powerful but requires FV skills

PCI Result

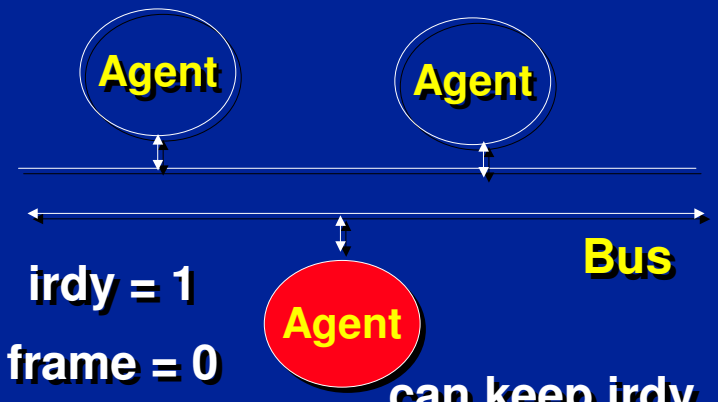
- Dead State Check
  - fine-tuned the specification
- Characteristic Check
  - found 3 bugs in the official protocol

In order to start a transaction...



Debugging Monitor-Based Specification — PCI Bug

Because of an omission in the specification...



Debugging Monitor-Based Specification — PCI Bug

1. One agent can lock the bus
2. Signaling definitions are not disjoint
3. An agent can unexpectedly change modes during a transaction.

Conclusion

- Easy yet powerful specification and debugging framework for interface specification presented
  - effective with PCI
  - PCI monitor specification available at http://radish.stanford.edu/pci

Contributions (Part II)

- Style Rules for the Constraints
- Guarantee of an Implementation
- Receptiveness