Formal Verification of an IBM CoreConnect Arbiter

- **Faculty:** Randy Bryant, Randy.Bryant@cs.cmu.edu
- **Researcher:**
  - Amit Goel, agoel@ece.cmu.edu, <status: hired>
- **Description:**
  - We wish to better understand the issues associated with formally verifying on-chip bus architectures. To do this, we performed model checking on an arbiter built for the IBM CoreConnect architecture as a test-case. Our work found bugs in the arbiter implementation and discovered inconsistencies and ambiguities in the specification.
Formal Verification of an IBM CoreConnect Arbiter

• **Expected Outcome:**
  • Dec 99:
    • *Verification of the arbiter*
    • *Insight into problems associated with specification and verification of bus architectures*
  • Dec 00:
    • *Method and tools to verify consistency of specifications*
PLB Architecture

- High speed peripherals in CoreConnect system connect to Processor Local Bus
- Decoupled address, read data and write data buses
- Multiple masters and slaves
- Each PLB Master has an independent interface to the arbiter
- Slaves are connected to the arbiter via a shared but decoupled interface
Verification by Model Checking

Model
- Master
- Arbiter
- Slave

Specification
- Properties

RuleBase

Verification Result
Correct or Counter-example
Formal Verification Methodology

- Verification based on PLB Architecture Specifications
- Restriction on master/slave signals built into master/slave models
- Restrictions/requirements for PLB signals specified as temporal logic properties
- Models built try to capture all possible behavior based on broad interpretation of specifications so as not to be optimistic in our verification
Formal Verification Methodology: Models

Independent Master Interfaces
⇒ Multiple master modules

Shared Slave Interface
⇒ Single slave module instantiated

Master and Slave non-deterministic modules written in RuleBase’s EDL modeling Language

RTL used to verify the implementation
Results: Bugs Found by Formal Verification

The core had been verified by simulation, but…

…bugs escaped simulation … and found by FV.

The bugs found could be classified as:
- soft: lead to performance degradation
- hard: “bad” behavior
- Problems in Specifications

Model checking runs were fast for verification with two masters

… ran out of resources for a setup with eight masters (maximum for the core)

Example: Arbiter exhibits undefined behavior when slave issues rearbitrate for secondary request
Problems in Specifications

Inconsistency: Descriptions don’t match

Redundancy: Unnecessary information (e.g. design guide) included

Ambiguity: different interpretations possible

Incompleteness: Behavior not always defined