EDA View of Formal Verification

Robert Kurshan
FMCAD07
November 11, 2007
Formal Functional H/W Verification IN USE in Industry Today

Equivalence checking
Theorem Proving on data paths -- ALUs (AMD, INTEL, ...)
Model Checking of protocol models -- cache coherence (INTEL, HP, ..)
  - MurPhi

Model Checking of block-level and interface properties ("static ABV")
  - arbitration
  - resource allocation (request/grant)
  - flow control
  - message delivery (block-level)
  - serialization (block-level)

Many companies are doing MC today, supported by EDA vendor tools:
  Cadence IFV; Synopsys Magellan; Mentor 0-In; Jasper; OneSpin; RealIntent Verix; Averant Solidify; Axiom (was @HDL)
Assertion-Based Verification

- Assume-Guarantee reasoning
  - Use some assertions as assumptions to help prove others
  - Must avoid circular reasoning
  - Working on automation

- Assertions as constraints
  - Assertions on inputs can be cast as constraints (assumptions)
  - Guided-Random simulation (with constraint solver)

- Automatic test bench generation
  - Use constraint solver to automatically generate simulation test vectors that satisfy given constraints
  - Can generate vectors that satisfy a given density distribution
  - Can handle both combinational and sequential constraints
Engines

- **BDD**
  - Forward search, backward search or both
  - Counterexample-guided refinement

- **SAT**
  - Bounded model checking (Clarke et al)
  - Abstraction-refinement (McMillan, Amla)
  - Interpolation (McMillan)

- **ATPG**

- **Model checking/Simulation hybrid**

- **Simulation** (guided-random using constraint-solving)
Benchmark results
The BIG Verification Problem

Verification (intrinsically) DOESN’T SCALE

– Component interactions grow exponentially with the number of system components, while conventional system test at best can increase coverage as a linear function of allotted test time.

– Likewise, capacity limitations are commonly cited as the essential gating factor that restricts the application of automatic formal verification (model checking) to at most a few design blocks.
Abstraction has long been used successfully in pilot projects to apply model checking to entire systems. Abstraction in conjunction with guided-random simulation can be used in the same way to increase coverage for conventional test.
Abstraction as Hierarchical Design

Utilize design hierarchy for verification

But: **NO REPEATED VERIFICATION AT SUCCESSIVE LEVELS** as is the case with current hierarchical methods

- Implement **CONTROL BEFORE DATAPATH**
  - More logical: CONTROL = high-level behavior
  - Use formal **STUBS** for datapath

- Design properties before design coding
  - Properties part of test plan
  - Design and verification done together

- Supports **earlier debug**
  - Thus accelerates time to market
  - Leads to higher quality/more robust design
The Technology Transfer Problem

• Catch22
  – For support, need demand
  – For demand, need support

• Acceptability is inversely proportional to change in user interface

• A methodology change is a killer for tech transfer
  – Takes much time to generate confidence in a new technology
  – Takes a compelling need

• Anything new is suspect (and for good reason)
  – Competition breeds confidence
Framework for Technology Transfer

• Small Steps
  – Each step involves very small change for user
  – Each step produces some positive benefit

• Road map
  – From where we are to where we want to get to
  – Small steps
  – Major challenge: getting from here to there
  – Be prepared for many false starts