

# 25 Years of Computer-Aided Verification

**Robert Kurshan**

**CAV'06 – 25MC**

**Wednesday, August 16, 2006**

# 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
  - International **standards** breed competition and facilitate tech transfer

# 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

# A Short History of Formal Verification - Prehistory

1900 Russell-Whitehead: Principia Mathematica

1936 Turing Machine

1960s Automated Theorem Proving (US gov funding)

Woody Bledsoe

1970s Gypsy (Ron Goode, JC Browne) [UT branch]

1977 Pnueli: LTL

1982 Boyer-Moore -> PVS (N. Shankar)

1980s R. Milner: ML -> HOL (orig for h/w) [Edinburgh branch]

Isabelle -> LAMBDA tool

Coq, Nuprl, ....

1980 Rudin-West (IBM) - reachability analysis for fsm  
Clarke-Emerson (CMU) - logic-based model checking

# A Short History of Formal Verification – last 25 years

- 1980 Rudin-West (IBM) - reachability analysis  
Clarke-Emerson (CMU) - logic-based model checking
- 1983 EMC (64K states)
- 1990 Symbolic Verification ( $10^{20}$  states, now  $10^{50}$  states)
- 1993 Chrysalis – equivalence checker
- 1994 FormalCheck commercial model checker
- 1995 LAMBDA commercial theorem prover (based on HOL)
- 1998 SAT-based bounded model checking
- 2003 Complete SAT-based model checking (McMillan Interpolation)
- 2006 Many commercial EDN model checking tools (Cadence, Synopsys, Mentor, Jasper ...)

# Standard Assertion Languages (2003)

- Open Verification Library (OVL)
- Accellera Property Specification Language (PSL)
- SystemVerilog Assertions (SVA)
  
- They differ in expressiveness and ease of use, but the basic idea is the same
- OVL is simplest, easiest to use, but least expressive
  
- All are international standards
  - Used first **for simulation test** (as monitors) – “small step”
  - Common interface breeds competition
  - Facilitates tech transfer: **KEY CATALYST**

# Use of Assertions

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

# Formal Functional H/W Verification IN USE in Industry – *The Compelling Needs*

Equivalence checking

Theorem Proving on data paths -- ALUs (AMD, INTEL, ...)

Model Checking of protocol models -- cache coherence (INTEL, HP, ..)

- MurPhi

**Model Checking** of local properties ("static ABV")

- arbitration

- resource allocation (request/grant)

- flow control

- message delivery (local)

- serialization (local)

Many companies are doing **MC** today, supported by EDA vendor tools:

Cadence FormalCheck, IFV, BlackTie, Conformal; Synopsys Magellan;

Mentor 0-In; Jasper; Calypto; RealIntent Verix; Averant Solidify; @HDL

# False Starts

- Power vs Automation
  - Theorem proving as a mainstream commercial tool
  - Even general model checker (FormalCheck vs StaticCheck)
  - NB: first accepted was .. **Equivalence Checker**
- Verification vs Falsification
  - **Small step**: *I believe an error track, not 'verified'*
  - Under-approximation (**restriction**) is easier
- Soundness vs Coverage
  - Allow fails in abstractions, passes in restrictions
- Interoperability (with simulation test)
  - Hybrid techniques
  - “Formal Verification” vs “Static Analysis”

# 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)

# Gating Issues for Property Checking

- Speed
- Capacity
- Improving speed:
  - do less work for same result
    - eg, avoid equivalent runs
      - remove irrelevant parts of the model: **localization reduction**
    - optimize engines
- Improving capacity: divide and conquer via decomposition