Next-Generation Constraint Solvers for Circuits and Their Applications to Test and Verification

K.T. Tim Cheng

Department of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106, USA

The Boolean Satisfiability (SAT) Problem

- Given a Boolean function \( f(X) \):
  - Find an assignment \( X^* \) such that \( f(X^*) = 1 \), or
  - Prove that such an assignment does not exist.

- In the “classical” SAT problem, \( f(X) \) is represented in Conjunctive Normal Form (CNF)
  - Example: \( \phi = (a + c) \, (b + c) \, (\neg a + \neg b + \neg c) \)

- Has many applications in EDA and other fields
  - Test generation for manufacturing defects
  - Equivalence checking
  - Bounded model checking
  - Circuit delay computation
  - FPGA routing, Noise analysis, etc…
Combining SAT and ATPG Solvers

- Tremendous progress in SAT solving in recent years
  - GRASP, SATO, ZCHAFF, BERKMIN, LIMMAT, ..., etc.
- ATPG is a 30+ years old field
  - Several commercial/industrial/academic tools are available
- Our goals:
  - Explore & combine techniques in SAT solvers & ATPG tools
  - Target circuit-oriented problem instances
  - Address both combinational & sequential SAT problems
  - Apply for property checking, equivalence checking, manufacturing testing, and functional vector generation

Sequential SAT for Circuits

- Find a sequence of vectors at inputs to:
  - Set “Out” to 1, or
  - Bring the circuit to state “\( FF_1=1 \) and \( FF_2=0 \)”
- Or, prove no such sequence exists
  - With a given initial state, or
  - Assuming the initial state is unknown
New Solvers

- CSAT: Fast Combinational Solver Using Circuit Structural Information for Learning
- Satori - Fast Sequential Solver
- Satori2 - Faster Sequential Solver With Improved Search Strategies
- Shiva – A Hybrid Constraint-Solver
  - Integrating Boolean Solver with Arithmetic Solver

Software and Publications

- Software released
  - CSAT: both source and executable are available on the web
  - Satori2: executable is available on the web; source is available for selected partners
  - Have been evaluated by, integrated into and used by: Intel, Calypto, Motorola, Cadence, Mentor Graphics
- Key Publications:
  - DAC'03: A Signal Correlation Guided ATPG solver and Its Applications for Solving Difficult Industrial Cases
  - ICCAD'03: SATORI – A Fast Sequential Justification Engine
  - HLDVT'03: A Comparison of BDDs, BMC, and Sequential SAT for Model Checking
  - ASPDAC'04: Efficient Reachability Checking using Sequential SAT
  - IEEE Design&Test: On Property Verification using Sequential SAT and Bounded Model Checking, March 2004
  - DAC'04: An Efficient Finite-Domain Constraint Solver for Circuits
SHIVA Hybrid Solver - Overview

- Objective: further improvement over pure Boolean solver
- Efficient constraint solver for Boolean and arithmetic domains
  - Boolean solver for control
  - Presburger Math for Data-path
  - Tightly integrated
  - Find good partition dynamically
  - Can handle bit-vector arithmetic
- Bound Hybrid Search space by
  - Constraint Propagation
  - Learning on interface points

Circuit Model for Hybrid Search
Overall Solver

Example

- Overall system
  - \( g = ( f == e) \land (e = ((\neg s . c \lor sel . d) \land (c = a_1 + a_2) \land (d = b_1 + b_2))) \)

- Value Assignments on sel and g by ATPG
  - Eg: \{sel, g\} = \{0,1\}
  - Solution is IS_Satisfiable \((f == a_1 + a_2)\)

- Advantage
  - Only decision points here are 1 bit g and 1 bit sel
  - If comparator was in Boolean solver, we have 64 decision points.
Example: Interface Constraints

- Loose conflict clause: \((g' + h + i' + s0 + s1')\)
- Tight conflict clause: \((g' + i')\)

Constraint Learning for Efficient Bounding

- Constraint: Set of signal line assignments that cannot be true simultaneously
- Boolean constraints:
  - Learnt during combinational solving
  - Contains Boolean literals
  - Stored as clauses
  - If \("a = 1, c = 1\) and \(f = 0\) cause conflict, insert \((\neg a + \neg c + f)\) into the problem instance
- Interface constraints:
  - Learnt during satisfiability checking of Boolean-arithmetic boundary assignments
  - Contains Boolean literals
  - Stored as clauses
- State constraints:
  - Contains boolean literals and arithmetic relations
  - Stored as a clause and partial STG
Summary - Shiva

- HDPLL is promising
  - Hybrid UIP learning for efficient conflict analysis
  - Use of structure information for decision ordering
  - Constraint propagation for efficient bounding

- Details available in DAC’04 paper

- Scheduled for release in Q1 of 2005