Automated Debugging of Missing Input Constraints in a Formal Verification Environment

Brian Keng and Andreas Veneris
University of Toronto

FMCAD 2012

Outline

- Motivation
- Background
- Debugging of Missing Input Constraints
- Experimental Results

Motivation

- Formal Property Checking
  - Can "prove" correctness about design blocks
  - Exhaustively check state-space of a design for violations of properties (assertions)
    - Can return hard to find corner case
    - Returns counter-example that excites failure
  - Properties written in formal specification language (e.g. SVA, PSL)

Input Constraints

- Limit input space explored by formal tool
- Need to find all missing constraints before "real" bugs can be found
- Causes of missing constraints:
  - Undocumented assumptions
  - Adjacent design blocks limit possible inputs to DUV
- Debugging missing input constraints is hard!
  - Failing counter-example could be due to design bug, bug in assertion, or missing input constraint
  - Time-consuming (guess & check)

Outline

- Motivation
- Background
- Debugging of Missing Input Constraints
- Experimental Results

Background

- Given a UNSAT Boolean formula Φ in CNF:
  - UNSAT Cores:
    - Subset of clauses in Φ that are UNSAT
  - Minimal Unsatisfiable Subset (MUS)
    - UNSAT core where every proper subset is SAT
  - Minimal Correct Set (MCS)
    - Minimal subset of clauses in Φ such that removing these clauses will make Φ SAT

Relationship between MUSs and MCSs

- Duality relationship between MUSs and MCSs
  - Given all MUSs (MCSs), it is possible to compute one from the other [1]
- Computing MCSs:
  - Add fresh *relaxation* variable to each clause
  - Cardinality constraints to limit active relaxation vars
  - Active relaxation variables correspond to MCS
  - Idea used in modern Max-SAT solvers e.g. [2]

[1] Liffiton, Sakallah, "On Finding All Minimally Unsatisfiable Subformulas," *SAT 2005*
[2] Marques-Silva, Planes, "Algorithms for maximum satisfiability using unsatisfiable cores," DATE 2008

Outline

- Motivation
- Background
- Debugging of Missing Input Constraints
- Experimental Results

Debugging of Missing Input Constraints

- Goals:
  - Give suggestions for the missing constraints
    - User must make final decision to add constraint
  - Simple, easy to understand properties
    - Cannot be complicated sythesized function
  - Quick feedback to user
    - Faster than guess & check method

Debugging Flow

- Input:
  - Design, property, counter-example
- Flow:
  - Extract “bad” input combinations from counter-example
  - Generate list of fixed cycle properties from counter-example
  - Filter generated properties
- Output:
  - List of fixed cycle properties that prevent “bad” input combinations from counter-example