IC3-Guided Abstraction

Jason Baumgartner
Alexander Ivrii
Arie Matsliah
Hari Mony

IBM

Outline

- Overview of localization abstraction
  - How to use priorities of variables to improve its quality
- Overview of IC3
  - How to produce priorities based on an incomplete run
- Experiments
  - Compare localizations/runtimes with and without priorities
- Conclusions / Future Work

Abstraction by Localization

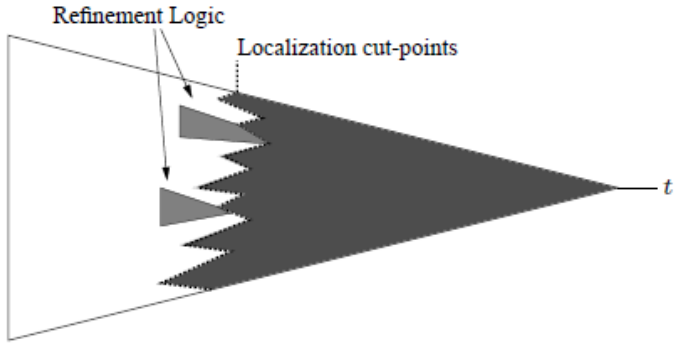
- Replace registers / gates by cutpoints

- Over-approximation:
  - Proofs on the localized netlist are valid
  - Counterexamples might be spurious

- We want to create a "perfect" abstract design
  - Small
  - No spurious counterexamples
  - Ultimately passed to a proof engine

Localization Strategies

- Counter-example based abstraction (CBA/CEGAR):
  - Start with an empty (or a very small) approximation
  - Run bounded model checking to see if target can be hit in N time-steps
  - Refine by ruling out spurious counter-examples



- Proof-based abstraction (PBA):
  - Run bounded model checking on the entire design for N time-steps
  - Look at the proof of unsatisfiability to decide what logic is necessary
- Hybrid method:
  - Interleave CBA and PBA

Localization with Guidance

- Priorities = rate relative importance of various state variables
  - From 0 (highest) to ∞ (lowest)

- Goal: use priorities to guide hybrid localization
  - Refinement based on many heuristics
  - Cumulative choices

Guiding CBA

Strategy:

- Initial abstraction = empty
- Refinement: add only state variables with highest priority

Improvement:

- Initial abstraction = all state variables with highest priority

IC3

- Incrementally refines and extends a sequence of clause sets
- On iteration=k:
  - $F_1, \ldots, F_k$ - bounded invariants
  - Property holds for k time-steps

- Standard optimization:
  - F_∞ - absolute invariants

- Example:
  - k=1: $F_1 = \{C_1, C_2\}$
  - k=2: $F_1 = \{C_1, C_2, C_3\}$ $F_2 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$
  - k=3: $F_1 = \{C_1, C_2, C_3, C_4\}$ $F_2 = \{C_1, C_3, C_4\}$ $F_3 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$

Producing Priorities with IC3

- Assign priorities to clauses
- Priority of a variable = minimum priority of clauses it's contained in

- Example:
  - k=1: $F_1 = \{C_1, C_2\}$
  - k=2: $F_1 = \{C_1, C_2, C_3\}$ $F_2 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$
  - k=3: $F_1 = \{C_1, C_2, C_3, C_4\}$ $F_2 = \{C_1, C_3, C_4\}$ $F_3 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$

Producing Priorities - 1

Method 1:

- Priority of each clause is 0
  - $Prio(C_1) = Prio(C_2) = Prio(C_3) = Prio(C_4) = 0$

- All clauses are equally important
- Abstraction with priorities 0 satisfies the property for k time-steps

- Example:
  - k=1: $F_1 = \{C_1, C_2\}$
  - k=2: $F_1 = \{C_1, C_2, C_3\}$ $F_2 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$
  - k=3: $F_1 = \{C_1, C_2, C_3, C_4\}$ $F_2 = \{C_1, C_3, C_4\}$ $F_3 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$

Producing Priorities - 2

Method 2:

- Priority of each clause = first k requiring it
  - $Prio(C_1) = Prio(C_2) = 1$, $Prio(C_3) = 2$, $Prio(C_4) = 3$

- Clauses for proofs of smaller bounds are more important
- Abstraction with priorities $\leq t$ satisfies the property for t time-steps

- Example:
  - k=1: $F_1 = \{C_1, C_2\}$
  - k=2: $F_1 = \{C_1, C_2, C_3\}$ $F_2 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$
  - k=3: $F_1 = \{C_1, C_2, C_3, C_4\}$ $F_2 = \{C_1, C_3, C_4\}$ $F_3 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$

Producing Priorities - 3

Method 3:

- Priority of each clause = how close it is to k
  - Prio(C_1) = Prio(C_3) = 0, Prio(C_4) = 1, Prio(C_2) = 2

- Clauses for larger bounds are more important
- Absolute invariants have priority 0

- Example:
  - k=1: $F_1 = \{C_1, C_2\}$
  - k=2: $F_1 = \{C_1, C_2, C_3\}$ $F_2 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$
  - k=3: $F1 = \{C_1, C_2, C_3, C_4\}$ $F_2 = \{C_1, C_3, C_4\}$ $F_3 = \{C_1, C_3\}$ $F_\infty = \{C_3\}$

Experimental results

- Implemented in the IBM verification tool Rulebase-SixthSense
- Used 465 single-target benchmarks from HWMCC 2011
- Comparing localization with hints (Method 2) and without hints
- Effect on Abstraction Size:
  - Run IC3 for 120 seconds, localization for 300 seconds
  - 294 instances: solved by IC3/localization alone
  - 171 remaining instances: 14.5% cumulative reduction (6.8% median)
- Effect on IC3 Resources:
  - Run IC3 with a 900 second time limit on 171 localized designs
  - Localization without hints: solved 15
  - Localization with hints: solved 24 (a strict superset)

Concluding remarks

- Higher-quality abstractions based on an incomplete IC3 run
  - Smaller and easier to verify
- A powerful verification tool will likely run IC3 for a small time-bound early in its strategy
  - Extracting localization hints poses virtually no overhead

- Future Work:
  - Improve heuristics on prioritizing state variables
  - Explore the effects on heavier-weight verification flows
  - Explore methods to prune irrelevant IC3 invariants
  - Explore the use of IC3 hints on proof-based abstraction

Thank You!