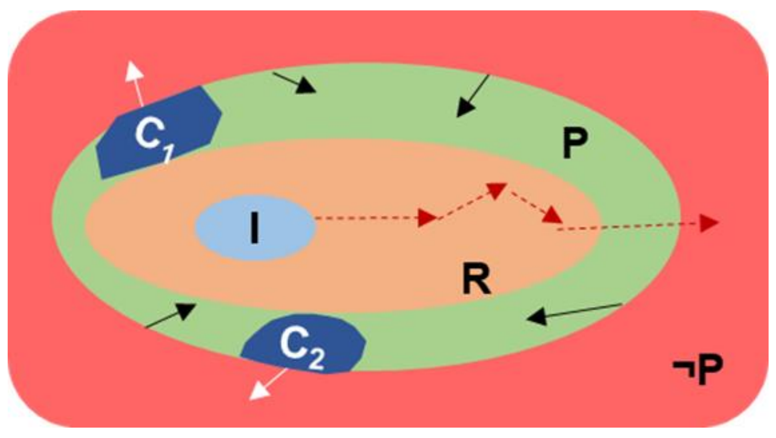
Empirical Evaluation of IC3-Based Model Checking Techniques on Verilog RTL Designs

Aman Goel and Karem Sakallah
University of Michigan, Ann Arbor, USA
{amangoel,karem}@umich.edu

Introduction

Problem: Given a transition system **TS** (defined by formulas for the transition relation **T** and the set of initial states **I**), check whether it meets a given *safety* property **P**, and, if not, produce a counterexample demonstrating how **TS** violates **P**.

Methodology: Use IC3-based techniques that perform property-directed approximate reachability analysis using incremental SAT solving. IC3 derives strengthening clauses from counterexamples-to-induction to incrementally make **P** inductive, or derives a counterexample trace disproving the property.



R : Reachable set of states
C_1, C_2 : Counterexamples to induction

[P ∧ T ∧ ¬P'] is SAT **Not Inductive**

[$\neg C_1 \wedge \neg C_2$ ∧ P ∧ T ∧ ¬P'] is UNSAT **Inductive**

Schematic diagram of model checking using IC3

For hardware, IC3 can be performed on the synthesized netlist (bit level) using SAT solvers, or directly at the register-transfer level (word level) using a variety of abstraction techniques and SMT solvers.

In this work, we rigorously evaluate different IC3-based techniques, including both bit-level and word-level model checkers, and identify their benefits and shortcomings, and opportunities for improving scalability.

Experimental Setup

We analyzed 535 safety checking problems from different sources:

+ ***opensource*** includes 141 problems from publicly available sources. Problems include cores from picoJava, USB 1.1, CRC generation, Huffman coding, mutual exclusion algorithms, simple microprocessor, etc.
● ***industry*** includes 370 proprietary problems from real-world industry designs, involving control-centric properties on large designs with wide data paths.
× ***crafted*** includes 24 synthetically created problems involving both control- and data-centric properties.

We evaluated six techniques from 3 different tools:

From ABC:

- **pdr** is one of the best implementations of bit-level IC3
- **dprove** employs pre-processing (bounded model checking, retiming, simulation, interpolation, etc.) followed by *pdr*
- **pdr-nct** is *pdr* configured with improved generalization and localization abstraction

From nuXmv:

- **nuxmv-ic3** performs bit-level IC3 with pre-processing (latch equivalency, temporal decomposition)
- **nuxmv-ic3ia** uses word-level IC3 with implicit predicate abstraction

From Averroes 2:

- **avr** performs word-level IC3 with syntax-guided data abstraction (avr-ic3sa-uf)

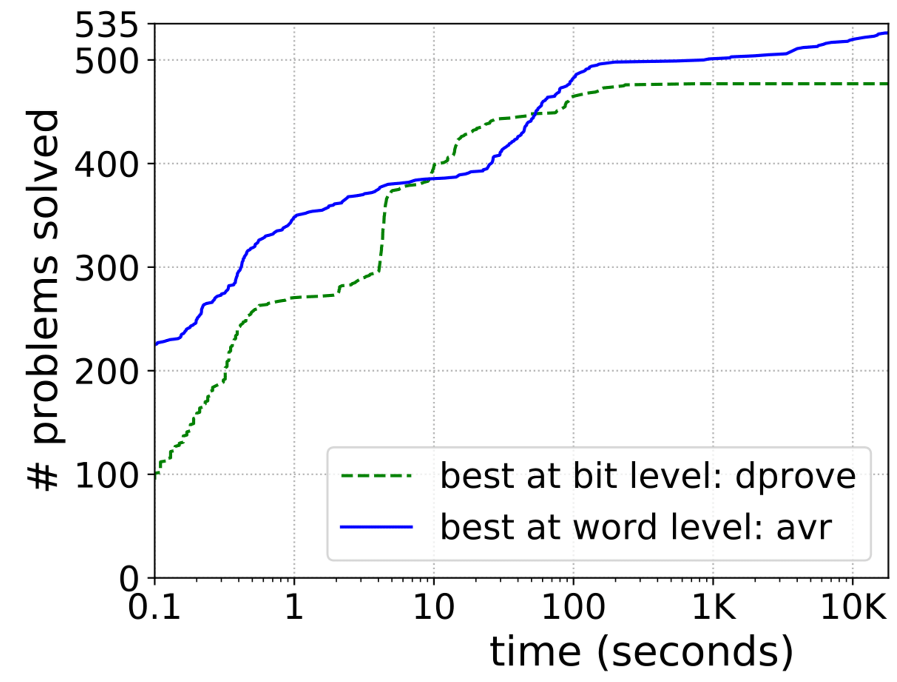
Results and Discussion

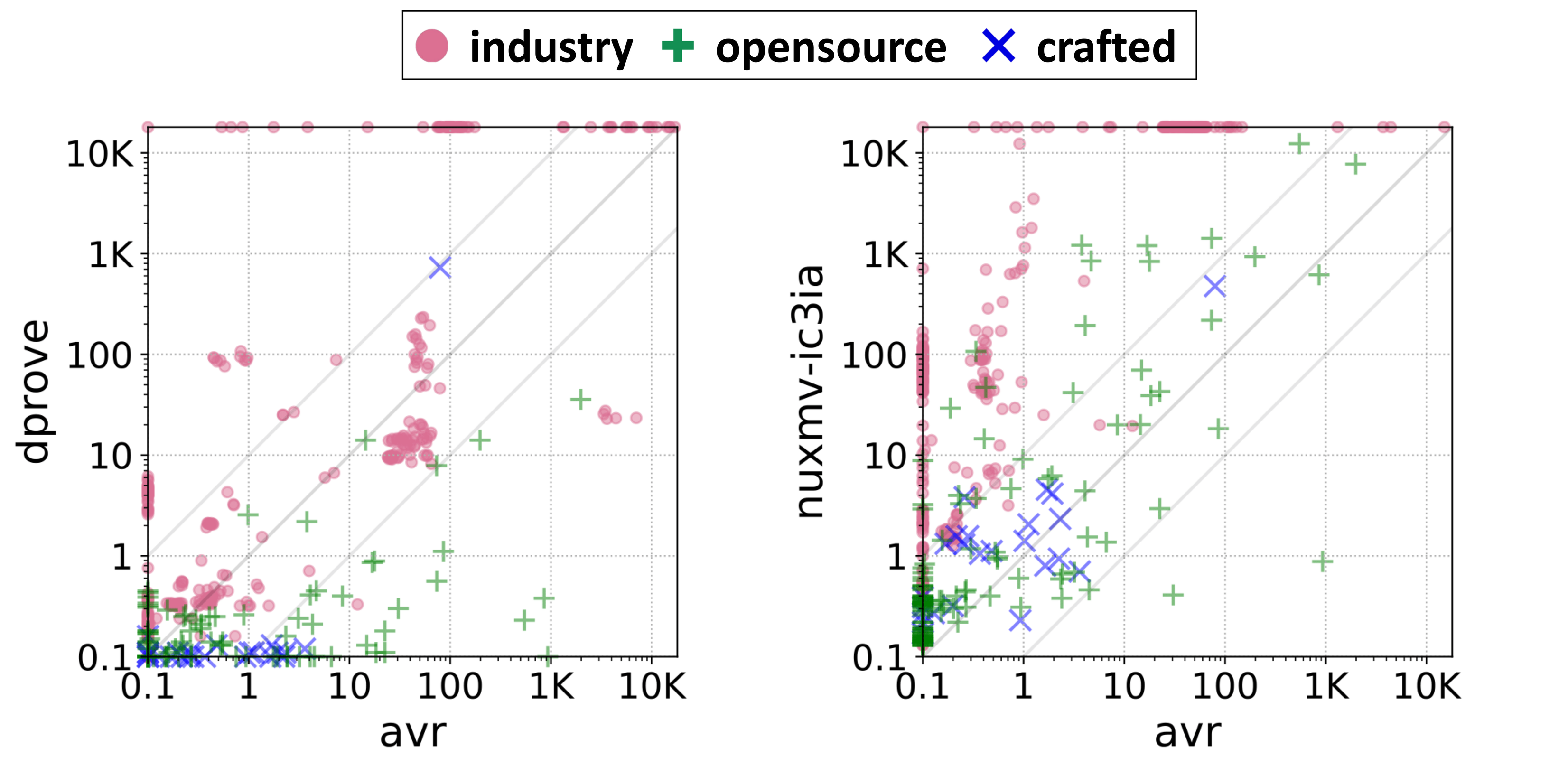
- Data abstraction using word-level information helped *avr* solve more problems than any other solver, especially dominating in the industry category.
- Word-level techniques require orders-of-magnitude fewer SMT solver calls compared to the number of SAT solver calls made by bit-level techniques.
- Fewer solver calls is justified by strong word-level clause learning performed by word-level techniques.
- Using word-level techniques has the additional advantage of producing concise and informative word-level inductive invariants which can be easily related to the Verilog RTL design.
- Pre-processing used by *dprove* is helpful, suggesting that pre-processing techniques similar to ones at the bit level may further help scale word-level techniques.

Tool		Solved	Unique	industry	opensource	crafted
pdr	(B)	466	1	308	137	21
dprove	(B)	477	3	315	**138**	**24**
pdr-nct	(B)	466	1	308	137	21
nuxmv-ic3	(B)	385	0	228	134	23
nuxmv-ic3ia	(W)	389	0	232	133	**24**
avr	(W)	**526**	**55**	**368**	134	**24**

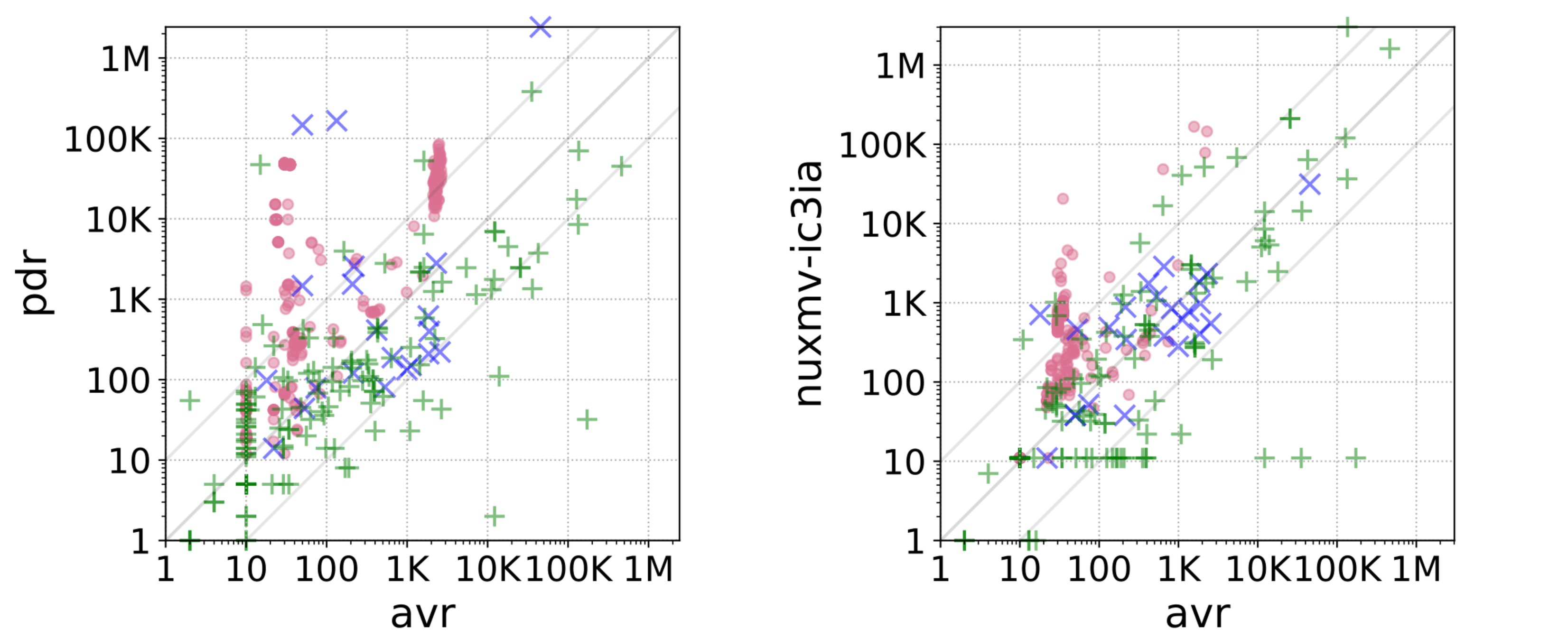
Word-level Model Checking

- ✓ More Industry Problems Solved
- ✓ Fast
- ✓ SMT Solving
- ✓ Fewer Solver Calls
- ✓ Word-level Clause Learning
- ✓ Rich Inductive Invariant
- ✓ Data Abstraction
- ✗ Limited Word-level Pre-processing Techniques

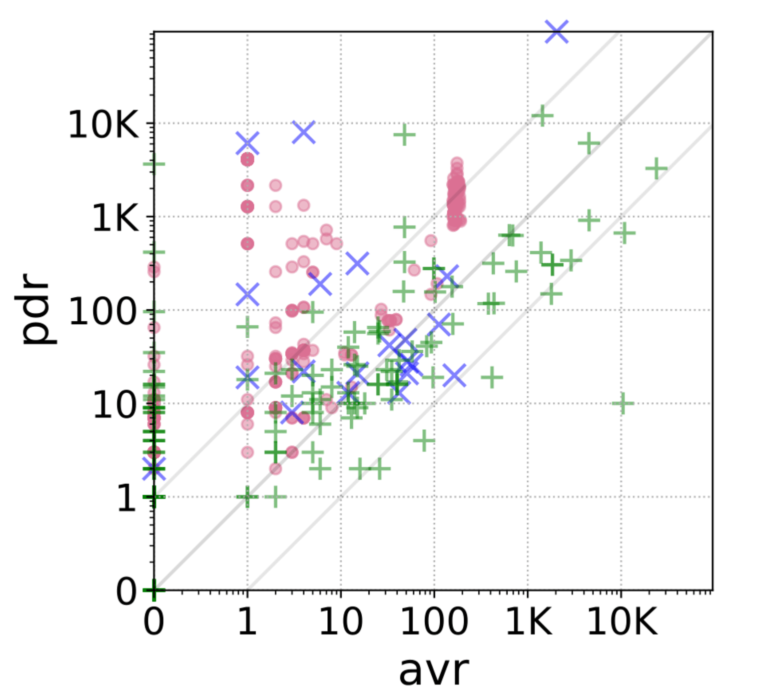




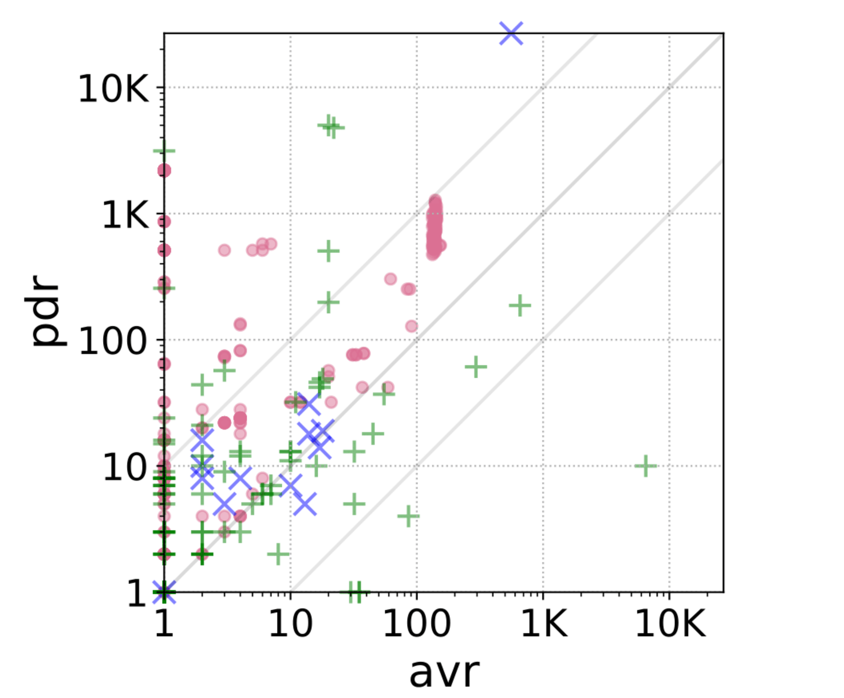
Time comparison (in seconds). *avr*'s times are better (resp. worse) above (resp. below) the diagonal



Solver calls comparison (SAT solver calls for *pdr,* SMT solver calls for *avr* and *nuxmv-ic3ia*)



Number of reachability clauses



Inductive invariant size

Take-away Points

- Word-level model checking has good potential to offer better scalability by taking advantage of high-level information.
- Word-level techniques learn strong clauses by performing many fewer solver checks.
- Pre-processing techniques and optimizations at the bit level can be adapted for word-level techniques for better scalability.

https://aman-goel.github.io/date19.html