T-79.298 Postgraduate Course in Digital Systems Science

Model Checking Using SAT Solvers

Satu Virtanen, satu@cs.hut.fi

3 December 2001

Bjesse, Chapter 7

- searching for bugs in a memory subsystem of Alpha, i.e. searching for counterexamples to desired properties of hardware
- model checking methods that use SAT solvers instead of BDDs
- two methods:
  (i) *bounded model checking*
  (ii) *symbolic trajectory evaluation*
- experiments comparing the runtime of different approaches

Research on using symbolic model checking to **find bugs** in a next-generation Alpha processor, focusing at *register transfer level* bugs.

Using Cadence SMV, runtimes required for checking simple properties were too high $\Longrightarrow$ instead of resorting to BDD-based model checkers such as SMV, experiments using SAT solvers.

Circuits on synchronous gate level viewed as *finite transition systems*

- states are assignments to the *state variables* (a vector s)
- transition system described by *propositional* formulas:
  (i) $Init(s)$ that evaluates to true for all initial states
  (ii) $Trans(s, s')$ that evaluates to true for proper state transitions
- three inputs: the above formulas together with a description of the property to check $Prop(s)$
- the aim is to construct a *counterexample* to the specified property instead of attempting to show that one does not exist

Bounded model checking (BMC)

- previously applied to industrial verification (Power PC)
- a method for applying SAT solvers in model checking
- not used for finding as "deep" bugs before
- sufficient capacity for realistic industrial application when combined with efficient SAT solvers
- tries to find bugs by attempting to construct a formula that is satisfiable exactly when a counterexample of a given length N or shorter exists: $\mathit{Init}(s_1) \wedge \mathit{Trans}(s_1, s_2) \wedge \ldots \wedge \mathit{Trans}(s_{N-1}, s_N) \wedge (\neg \mathit{Prop}(s_1) \vee \ldots \vee \neg \mathit{Prop}(s_N))$
- the resulting formula is evaluated by some external SAT solver
- obviously incapable of proving the absence of bugs

Symbolic trajectory evaluation (STE)

- mixes abstract interpretation and symbolic evaluation
- not previously combined with SAT solvers, always with BDDs
- now applied to verification at the *synchronous gate level*, which is a high level of abstraction for STE
- takes as input $\mathit{Trans}(s, s')$ and a two lists that form a **trajectory assertion** $\mathit{Ant} \Rightarrow \mathit{Cons}$, for which a boolean expression ok is computed that evaluates to true whenever the assignments for the states fulfill the assertion (ok then given to an external SAT solver)
- the equal-sized lists contain information of the system state on step i
- ok may contain values $\{\mathit{True}, \mathit{False}, X, \top\}$ as not everything is properly specifiable ($X \triangleq$ unknown, $\top \triangleq$ over-specified)
- *delayed* `or` as an example: $[s.a = x \wedge s.b = y,\ \langle\cdot\rangle] \Rightarrow [\langle\cdot\rangle,\ s.o = x \vee y]$

Merge buffer

- subbox of the MBox (for execution of memory reference instructions)
- receives requests to write into memory and merges stores to the same physical address
- must communicate with four other subboxes to achieve correct merging
- a large and complex buffer

1. starting from the original RTL description
2. reducing the size of the model by symmetry reductions
3. problems of the reduced model should remain problems in the original model
4. restricting the input by adding transactor state machines
5. abstracting the circuit:
   - optimization by an RTL compiler
   - removal of redundant or transparent latches
6. property specification and abstraction
7. verification and inspection of the counterexample

Symbolic model checking with BDDs

- evaluation of several BDD-based tools, SMV being the only promising one
- SMV ported to 64-bit Alpha with 8 GB main memory
- the merge buffer is too large to handle: some inputs were fixed
- verification takes several hours
- many bugs were found

Bounded Model Checking

- SAT-based model checking workbench FixIt
- `Prover` not available for Alpha $\Longrightarrow$ experiments on a 32-bit PC
- time required for the verification magnitudes less than with SMV
- Captain Prove searches for models using *strategies*
- a simple strategy: timed strategy consisting of *1-saturation* followed by backtracking

SAT-based symbolic trajectory evaluation

- a version SAT-based of STE implemented in FixIt
- difference to BMC: possible to give concrete values to some state variables and leave some as X $\Longrightarrow$ potentially deeper exploration of the state-space quickly
- writing specifications is slow: what values to assign, propagation of Xs $\Longrightarrow$ iteration required

Methodology proposal

1. begin analyzing a new subbox with BMC
2. use a small bound to make the inspection quick
3. eliminate false counterexamples by modifying the transactors
4. check longer and longer runs with the timed strategy
5. abstract the failure trace to check for other similar failures
6. after a bug has been found and fixed, ensure removal by STE
7. when BMC starts to take too long ($\geq \frac{1}{2}$ h), use STE in parallel
8. when nothing is found anymore, try SMV or move on