Overview

• Decision Procedures for Hardware Design and Verification
  – constrained random simulation\(^1\)
  – semi-formal\(^2\) and bounded model checking\(^3\)
  – equivalence checking\(^4\) and model checking\(^5\)
  – inductive techniques\(^6\) and theorem proving\(^7\)
  – combinational\(^8\) and sequential\(^9\) synthesis

• New Results in SAT Preprocessing
  – BCE = blocked clause elimination
  – witnesses for mixing BCE with equivalence reasoning
Constrained Random Simulation

**Hardware**

- **stimulus**
- **design under test**
- **output checker**

**input generation**

**procedural**
- (non-synthesizable) Verilog
- e.g. compute checksum of input packet
- use random number generators

**declarative**
- constraints
- e.g. assert checksum field to be correct
- specify distribution of input values
hit functional coverage target, specify light house as intermediate step

focus on coverage not on formal property verification
check temporal properties, focus on falsification, use SAT solvers

used in many companies effectively
Equivalence Checking

Verification → high-level model → implementation → Synthesis

<table>
<thead>
<tr>
<th>high-level model</th>
<th>implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL: Verilog, VHDL</td>
<td>gate- or switch-level</td>
</tr>
<tr>
<td>C models</td>
<td>Verilog</td>
</tr>
</tbody>
</table>

RTL vs gate in widespread use
• check temporal properties of RTL models
  – *verification* instead of *falsification* as focus
  – needs quantifier elimination, BDDs, $k$-induction or interpolation
  – which in turn scales not as good $\Rightarrow$ capacity issues

• **commercially far less successful** than equivalence checking
  – specifications in temporal logic are hard to obtain
  – modeling the environment is even harder

• capacity is increasing, but not much research in academia
  – 3rd Hardware Model Checking Competition this year at FLOC

• *explicit model checking for protocols is working*
how to prove pipelined processor to implement the architecture

real pipelined, out-of-order processor

hypothetical non-pipelined processor

one pipeline cycle

flush

execution of one machine instruction

needs additional state invariants, completion-function as inductive invariant for flush
Inductive Techniques\textsuperscript{6}: Engineering Inductive Invariants

- reachable states as strongest inductive invariant
  - hard to compute; symbolic reachability is PSPACE complete
  - shows that inductive invariants can always be used (in theory)

- $k$-inductive invariants do not have to be inductive (= 1-inductive)

\[
G^k p \equiv \bigwedge_{i=0}^{k-1} X^i p, \quad p \text{ is } k\text{-inductive} \iff M \models G^k p \rightarrow XG^k p \quad \text{iff} \quad M \models G^k p \rightarrow X^k p
\]
  - strengthening of properties (with other invariants) still useful

- symbolic trajectory evaluation (STE) and also OneSpin’s approach
  - user specifies property automaton: assume / guarantees on transitions
  - over design variables, inductively show guarantees in parallel product
• (academic) examples of full blown processor verification
  – using ACL2, PVS, Isabelle for some processors
  – incorporation of SMT solvers is ongoing research

• special purpose applications
  – mostly for ALU but also more recently for instruction decoding
  – theorem prover connects lemmas which are proven with decision procedures
  – large theorems could involve reals (IEEE floating point standard conformance)

• specialized activity without many users
Combinational Synthesis\textsuperscript{8}

- generic problem: \( \exists p [\forall i [g(p, i) = s(i)]] \) inputs \( i \), parameter \( p \)

- for instance fitting / merging logic to a small part (rectification) …

- … or synthesizing clock gating control

- … or functional substitution (can actually be done with interpolants)

- … or fixing circuits automatically

Sequential Synthesis\textsuperscript{9}

- game theory: the spec \( s \) is a temporal property

- not only in theory much more complex, needs application specific restrictions
• decision procedures applied in HW vary on the amount of “formal”
  – constrained random simulation\textsuperscript{1}
  – semi-formal\textsuperscript{2} and bounded model checking\textsuperscript{3}
  – equivalence checking\textsuperscript{4} and model checking\textsuperscript{5}
  – inductive techniques\textsuperscript{6} and theorem proving\textsuperscript{7}

• combinational\textsuperscript{8} and sequential\textsuperscript{9} synthesis interesting topics

• if your problem is hard and does not scale \implies change the problem
Blocked Clauses

Definition

A literal $l$ in a clause $C$ of a CNF $F$ blocks $C$ w.r.t. $F$ if for every clause $C' \in F$ with $\overline{l} \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\overline{l}\})$ obtained from resolving $C$ and $C'$ on $l$ is a tautology.

Definition [Blocked Clause] A clause is blocked if has a literal that blocks it.

Definition [Blocked Literal] A literal is blocked if it blocks a clause.

Example

$$(a \lor b) \land (a \lor \overline{b} \lor \overline{c}) \land (\overline{a} \lor c)$$

Only first clause is not blocked.

Second clause contains two blocked literals: $a$ and $\overline{c}$.

Literal $c$ in the last clause is blocked.

After removing either $(a \lor \overline{b} \lor \overline{c})$ or $(\overline{a} \lor c)$, the clause $(a \lor b)$ becomes blocked.

All clauses can be removed.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COI</td>
<td>Cone-of-Influence reduction</td>
</tr>
<tr>
<td>MIR</td>
<td>Monontone-Input-Reduction</td>
</tr>
<tr>
<td>NSI</td>
<td>Non-Shared Inputs reduction</td>
</tr>
<tr>
<td>PG</td>
<td>Plaisted-Greenbaum polarity based encoding</td>
</tr>
<tr>
<td>TST</td>
<td>standard Tseitin encoding</td>
</tr>
<tr>
<td>VE</td>
<td>Variable-Elimination as in DP / Quantor / SATeLite</td>
</tr>
<tr>
<td>BCE</td>
<td>Blocked-Clause-Elimination</td>
</tr>
<tr>
<td>encoding</td>
<td>b</td>
</tr>
<tr>
<td>-----------</td>
<td>---</td>
</tr>
<tr>
<td>T</td>
<td>V</td>
</tr>
<tr>
<td>SU</td>
<td>0</td>
</tr>
<tr>
<td>AT</td>
<td>12</td>
</tr>
<tr>
<td>AP</td>
<td>10</td>
</tr>
<tr>
<td>AM</td>
<td>190</td>
</tr>
<tr>
<td>AN</td>
<td>9</td>
</tr>
<tr>
<td>HT</td>
<td>147</td>
</tr>
<tr>
<td>HP</td>
<td>130</td>
</tr>
<tr>
<td>HM</td>
<td>6961</td>
</tr>
<tr>
<td>HN</td>
<td>134</td>
</tr>
<tr>
<td>BT</td>
<td>577</td>
</tr>
<tr>
<td>BP</td>
<td>542</td>
</tr>
<tr>
<td>BM</td>
<td>10024</td>
</tr>
<tr>
<td>BN</td>
<td>13148</td>
</tr>
</tbody>
</table>

S = Sat competition  
A = AIG competition  
H = HW model checking competition  
B = bit-vector SMT competition  
T = plain Tseitin encoding  
P = Plaisted Greenbaum  
M = MiniCirc encoding  
N = NiceDAGs
• equivalence reasoning

  – if \( l = k \) is derived, actually clauses \((\bar{l} \lor k)(l \lor \bar{k})\), then replace \( l \) by \( k \)

  – in practice use Tarjan’s union-find algorithm, where

  – each literal has a pointer to the representative of its equivalence class

  – for substituted variables get their value from the representative

• blocked clause elimination

  – save the removed blocked clauses on a stack

  – traverse stack in reverse order, values of blocked literals may need to be flipped

• how to obtain a witness if both techniques are used?
No Need to Flip Values of Substituted Literals

Theorem

Let $C = (l \lor l_1 \lor \ldots \lor l_n)$ be a clause blocked on $l$ wrt. $F$ and further assume $F \setminus C \models l = k$ then for any assignment $\sigma$ with $\sigma(F \setminus C) = \top$ we also have $\sigma(C) = \sigma(F) = \top$

Proof Sketch

$\sigma(l) = \bot \quad \sigma(k) = \top$

given resolution derivation of $\bar{l} \lor k$ from clauses $F \setminus C$, w.l.o.g. tree like

definition: $l$ forcing for a clause $l \lor k_1 \lor \ldots \lor k_m$ iff $\sigma(k_1) = \ldots = \sigma(k_m) = \bot$ (and $\sigma(l) = \top$)

show per induction: there is a path in the tree on which all clauses are $l$ forcing leaf on path is an input clause $B \in F \setminus C$, with $l \in B$ but also $\bar{k}_j \in B$ for one $j$ since $C$ is blocked since $B$ is $l$ forcing $\sigma(\bar{k}_j) = \bot$ and thus $\sigma(k_j) = \sigma(C) = \top$ q.e.d.