On Bounded Invariant Checking of Blackbox-Designs

Marc Herbstritt

(joint work with Bernd Becker)

Institute of Computer Science
Albert-Ludwigs-University
Freiburg im Breisgau, Germany

Presentation at IEEE MTV 2005, Nov 03 2005
Overview

1. Introduction
2. Bounded Invariant Checking
3. 01X
4. Experiments
5. Conclusions
Applications of Blackboxing

Abstraction: Hide components that are not necessary.
Applications of Blackboxing

\[
\text{op}(A,B,+,\text{enc}(\text{bin})) = \text{enc}(\text{bin},A)+\text{enc}(\text{bin},B) \quad ?
\]

Implementation of Shifter and BCD–SUB unit not finished

Partial Verification: E.g. in early design stage.
Applications of Blackboxing

Check whether error lies within the blackbox region

Diagnosis: Localisation of error.
Validity vs. Realizability

Validity
For all blackbox implementations, does the property hold?  $[\forall]$  

Realizability
Does there exist at least one blackbox implementation, such that the property holds?  $[\exists]$
### Verification vs. Falsification

<table>
<thead>
<tr>
<th>Verification</th>
<th>Falsification</th>
</tr>
</thead>
<tbody>
<tr>
<td>To prove that the property holds. ( \top )</td>
<td>To find a counterexample that violates the property. ( \bot )</td>
</tr>
</tbody>
</table>
Validity vs. Realizability
Verification vs. Falsification

Relationship 1

Verifying validity of property $\varphi$ $\iff$
Falsifying realizability of property $(\neg \varphi)$

$[\forall, \top, \varphi] = [\exists, \bot, (\neg \varphi)]$
Validity vs. Realizability

Verification vs. Falsification

Relationship 2

Verifying realizability of property $\varphi$

$\Leftrightarrow$

Falsifying validity of property $(\neg \varphi)$

$[\exists, \top, \varphi] = [\forall, \bot, (\neg \varphi)]$
Bounded Invariant Checking for Blackbox-Designs

- **Given:**
  - Incomplete circuit implementation, i.e. circuit containing blackboxes
  - Invariant, i.e. a property $\varphi$ that should always hold

- **Wish:** To know that the circuit is already correct wrt. $\varphi$ (i.e., verifying validity)

- **Implementation:** If we can falsify realizability of $\varphi$, we know definitely that the system is incorrect wrt. $\varphi$.

**Relationship 3**

Verifying validity $\Rightarrow$ Verifying realizability.
Falsifying realizability $\Rightarrow$ Falsifying validity.
Methodology

Algorithms and Data structures

**BMC:** Integrate blackbox modelling into BMC formulation

**AIG:** Use AND/INV-graphs for problem representation

**SAT:** Apply dedicated SAT-engine for AIGs

**Logic:** Extend decision procedure to 01X-logic
Finite unfolding for BMC

$n$ blackboxes and unfolding depth $k$
$\Rightarrow (n \times k)$ blackboxes in unfolded transition relation!
Modelling blackboxes with 01X-logic

Modelling unknown blackbox functionality

Value ‘X’ on blackbox output models the meaning ‘Value is either 0 or 1, but unknown’.
Modelling blackboxes with 01X-logic

Modelling unknown blackbox functionality
Value ’X’ on blackbox output models the meaning ’Value is either 0 or 1, but unknown’.

Satisfiability = Counterexample
Satisfiable solution of BMC that is independent of X-value is a concrete counterexample that violates property $\varphi$. 
How to handle 01X

1. Adapt encoding (**ENC-SAT**)
   - For each 01X-variable $x$, introduce new propositional variables $x_0$ and $x_1$
   - Adapt operators: $\text{AND}_{01X}(x, y) = [x_0 + y_0, x_1 \cdot y_1]$
   - Relies on propositional decision procedure
   - Jain et al. (VTS 2000)

2. Adapt SAT-engine (**01X-SAT**)
   - Extend propagation rules: $\text{AND}_{01X}(1, X) = X$
   - Keep problem representation
Comparison
Assume to check value 0 at AND-gate output with one input constrained to X.
Comparison: X-Propagation

ENC-SAT can propagate set of values.
Comparison: Justification selection

1 justification in 01X-SAT ⇔ 2 justifications in ENC-SAT
Comparison: Justification selection

Propositional SAT-engine can be misguided in ENC-SAT.
Conflict Learning

Conflict due to $(u \equiv 1)$, $(v \equiv 1)$, and $(w \equiv 1)$

$\Rightarrow$ add vertices that force $(u \cdot v \cdot w) \equiv 0$. 
X-values disable conflict detection in the usual way and new implications within propositional fragment: \( w \) could be 0 or \( X \)!
### Comparison: Summary

<table>
<thead>
<tr>
<th></th>
<th>ENC-SAT</th>
<th>01X-SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation</td>
<td>$2 \cdot n$</td>
<td>$n$</td>
</tr>
<tr>
<td>Propagation</td>
<td>set of values</td>
<td>single value</td>
</tr>
<tr>
<td>Justification</td>
<td>ignoring</td>
<td>structure</td>
</tr>
<tr>
<td></td>
<td>structure</td>
<td>aware</td>
</tr>
<tr>
<td>Conflict Learning</td>
<td>full</td>
<td>limited to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>prop. fragment</td>
</tr>
</tbody>
</table>
Experiments: Setup

- Selected benchmarks from VIS benchmark suite: s1269 and PicoJava/biu
- Introduced single error into the circuits ⇒ violating at least one invariant property!
- Added blackboxes „around“ the error
- 2730 BMC problems containing blackboxes
- each BMC instance is checked with increasing depth up to a pre-determined threshold
Impact of problem representation

1. No propagation of X-values
2. Enabled propositional conflict learning

<table>
<thead>
<tr>
<th></th>
<th># SAT</th>
<th># UNSAT</th>
<th># UNRES</th>
<th># L¹</th>
<th># I²</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>01X</td>
<td>569</td>
<td>18754</td>
<td>366</td>
<td>48K</td>
<td>202M</td>
<td>13292</td>
</tr>
<tr>
<td>Enc</td>
<td>573</td>
<td>18529</td>
<td>346</td>
<td>18K</td>
<td>414M</td>
<td>20542</td>
</tr>
</tbody>
</table>

¹L = learned conflicts
²I = implications due to learning
Impact of problem representation

Comparison of 01X vs. two-valued encoding (Jain)
2005-10-21-experiments-XProp0.log

- no method fails
- 01X fails, Jain succeeds
- 01X succeeds, Jain fails
- both methods fail

ENC−SAT worse

01X−SAT worse
Impact of X-propagation

1. Enabled propagation of X-values
2. Disabled propositional conflict learning

<table>
<thead>
<tr>
<th></th>
<th># SAT</th>
<th># UNSAT</th>
<th># UNRES</th>
<th># D¹</th>
<th># A²</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>01X</td>
<td>573</td>
<td>18600</td>
<td>388</td>
<td>975M</td>
<td>33G</td>
<td>13049</td>
</tr>
<tr>
<td>Enc</td>
<td>581</td>
<td>19675</td>
<td>84</td>
<td>245M</td>
<td>18G</td>
<td>6241</td>
</tr>
</tbody>
</table>

¹D = decisions, i.e., selection of justifications
²A = assignments (incl. propagation)
Impact of X-propagation

Comparison of 01X vs. two-valued encoding (Jain)
2005-10-21-experiments-XProp1-wo-CL.log

- no method fails
- 01X fails, Jain succeeds
- 01X succeeds, Jain fails
- both methods fail
Combining Conflict Learning and X-propagation

1. Enabled propagation of X-values
2. Enabled propositional conflict learning

<table>
<thead>
<tr>
<th></th>
<th># SAT</th>
<th># UNSAT</th>
<th># UNRES</th>
<th># D</th>
<th># A</th>
<th># L</th>
<th># I</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>01X</td>
<td>572</td>
<td>18800</td>
<td>339</td>
<td>853M</td>
<td>31G</td>
<td>43K</td>
<td>137M</td>
<td>12493</td>
</tr>
<tr>
<td>Enc</td>
<td>589</td>
<td>19766</td>
<td>49</td>
<td>148M</td>
<td>12G</td>
<td>33K</td>
<td>129M</td>
<td>4336</td>
</tr>
</tbody>
</table>
Combining Conflict Learning and X-propagation

Comparison of 01X vs. two-valued encoding (Jain)
2005-10-21-experiments-XProp1.log

- no method fails
- 01X fails, Jain succeeds
- 01X succeeds, Jain fails
- both methods fail
How powerful is 01X-modelling

<table>
<thead>
<tr>
<th>circuit</th>
<th># BB</th>
<th>%area</th>
<th>errors found</th>
<th>detection rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1269</td>
<td>1</td>
<td>5</td>
<td>162/370</td>
<td>43.78</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>10</td>
<td>111/340</td>
<td>32.64</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>20</td>
<td>75/430</td>
<td>17.41</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>76/380</td>
<td>20.00</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>20</td>
<td>18/460</td>
<td>03.91</td>
</tr>
<tr>
<td>PicoJava/</td>
<td>1</td>
<td>5</td>
<td>65/150</td>
<td>43.33</td>
</tr>
<tr>
<td>biu</td>
<td>1</td>
<td>10</td>
<td>56/150</td>
<td>37.33</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>20</td>
<td>9/150</td>
<td>06.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>15/150</td>
<td>10.00</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>20</td>
<td>2/150</td>
<td>01.33</td>
</tr>
</tbody>
</table>

1 # BB = number of blackboxes
2 %area = size of blackbox(es)
Conclusions

- Bounded Invariant Checking for Blackbox Designs is feasible
- ENC-SAT profits from propositional core
- In its purity, 01X-SAT can benefit from structural knowledge
Future work

- Make propagation of a set of values applicable to 01X-SAT
- Adapt conflict learning to 01X-SAT
- Increase accuracy of blackbox modelling
- Develop structural QBF-solver
- Mixing 01X-logic and quantification
Thanks for your attention!