On the Impact of Structural Circuit Partitioning on SAT-based Combinational Circuit Verification

(Presentation at IEEE MTV 2004)

Marc Herbstritt

(joint work with Thomas Kmieciak and Bernd Becker)

http://ira.informatik.uni-freiburg.de
herbstri@informatik.uni-freiburg.de

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

- Introduction
- Traditional techniques: SOG and AOG
- Discussion: SOG vs. AOG
- Idea for our approach based on output partitioning
- Two partitioning heuristics
- Partitioning-based verification using SAT
- Experimental results
- Conclusions
Single-Output Grouping (SOG): Verify *each* output *separately*

All-Outputs Grouping (AOG): Verify *all* outputs *at once*
Discussion: SOG vs. AOG

Partial Verification:

Report the equivalence status of those outputs for which the underlying verification method is able to check equivalence.
Discussion: SOG vs. AOG

Shared Components:

*Reuse computations made on components that are used by several outputs.*

Learn: \( (f=0) \implies (b=0) \)
Discussion: SOG vs. AOG

SOG
- **pros**: partial verification
- **cons**: no use of shared components

AOG
- **pros**: use of shared components
- **cons**: no partial verification
Our approach

We need partitionings of the primary outputs s.t.

- partial verification is possible
- computations on shared components are exploited

In this work, we ...

- ... use two already known output partitioning heuristics: WOG and BOG
- ... formally analyze WOG and BOG
- ... propose a SAT-based verification algorithm based on partitionings
WOG and BOG both compute ordered groupings

\[
\left( (p_{1,1}, p_{1,2}, \ldots, p_{1,k_1}), \ldots, (p_{n,1}, p_{n,2}, \ldots, p_{n,k_n}) \right)
\]

where

- \(p_{i,j}\) is primary output
- all \(p_{i,j}\) are pairwise disjoint
- \(n\) is the size of the grouping
- \(p_{i,1}\) is called the leader of group element \(i\)
WOG and BOG

WOG and BOG both compute ordered groupings

\[ ((p_{1,1}, p_{1,2}, \ldots, p_{1,k_1}), \ldots, (p_{n,1}, p_{n,2}, \ldots, p_{n,k_n})) \]

WOG and BOG differ in the support relation of the outputs in a group element \((p_{i,1}, p_{i,2}, \ldots, p_{i,k_i})\)

WOG: \( \forall j, 1 \leq j \leq k_i : \text{supp}(p_{i,j}) \subseteq \text{supp}(p_{i,1}) \)

(word-oriented, support relation is in terms of the leader)

BOG: \( \forall j, 1 \leq j \leq k_i : \text{supp}(p_{i,j}) \subseteq \text{supp}(p_{i,j-1}) \)

(bit-oriented, support relation is in terms of all previously included outputs)
How WOG works

ordering of outputs: $f_1 < f_4 < f_2 < f_3$
WOG computes: $((f_1, f_2, f_3), (f_4))$
How BOG works

BOG computes: \(((f_1, f_2), (f_4, f_3))\)
Notes on WOG and BOG

- WOG: \( (f_1, f_2, f_3), (f_4) \)
- BOG: \( (f_1, f_2), (f_4, f_3) \)

Computation of output relations by structural circuit analysis
\( \rightarrow \) Output-Correspondence Matrix

- BOG is not a refinement of WOG

- E.g. \( \not\exists \) subset of BOG for which the union results in \( (f_4) \)
**Lemma**: Let $\#W$ ($\#B$) be the number of group elements of the grouping generated by WOG (BOG). Then it holds

$$\#W \geq \#B.$$ 

**Proof**: Since BOG is not a refinement of WOG, the proof is more technical than expected.

**Hint**: Check which outputs are already handled at stage $i$ of the algorithms.
Verification using output partitioning

1. Input: $C_S$: specification circuit, $C_I$: implementation circuit
2. Output: Equivalence status
3. $(g_1, \ldots, g_t) \leftarrow \text{WOG-OR-BOG-GROUPING}(C_S)$
4. solved-outputs $\leftarrow \emptyset$; unresolved-outputs $\leftarrow \emptyset$
5. forall $m = 1$ to $t$ do
6. $C_M \leftarrow \text{CONSTRUCT-GROUP-ELEMENT-MITER}(C_S, C_I, g_m)$
7. res $\leftarrow \text{CHECK-SATISFIABILITY-OF-MITER}(C_M)$
8. if res = SATISFIABLE
9. return $C_S$ and $C_I$ differ on some primary output from $g_m$
10. endif
11. if res = UNRESOLVED
12. unresolved-outputs $\leftarrow$ unresolved-outputs $\cup$ $g_m$
13. else
14. solved-outputs $\leftarrow$ solved-outputs $\cup$ $g_m$
15. endif
16. endfor
17. return solved-outputs are equivalent,
   unresolved-outputs could not be proved
Comparison

- SINGLE-OUTPUT GROUPING (SOG)
- BIT-ORIENTED GROUPING (BOG)
- WORD-ORIENTED GROUPING (WOG)
- ALL-OUTPUTS GROUPING (AOG)
Partitioning results

Granularity := \frac{\text{Number of group elements}}{\text{Number of primary outputs}}
Verification results (script-rugged)
Conclusions

- Partitioning the outputs can have large impact
- Heuristics BOG and WOG are building a robust compromise between traditional techniques like AOG or SOG
- WOG compared to AOG: speedup of 276% on the average
- BOG compared to SOG: speedup of 75% on the average

Future work:
- Analyze in detail why SAT benefits from WOG/BOG
- Use of semi-canonical data-structures
- Partitioning heuristics incorporating more structure
- Development of SAT-oriented heuristics