Reducing Interpolant Circuit Size by Ad-Hoc Logic Synthesis and SAT-Based Weakening

Gianpiero Cabodi
Paolo Camurati
Paolo Pasini
Marco Palena
Danilo Vendraminetto

Politecnico di Torino
Torino, Italy
Outline

● Motivations & background
  ● Craig interpolation in MC => Size bottleneck (>10^5 gates)
  ● Highly redundant circuits, missing ad hoc reduction

● Contribution
  ● Ad HOC (fast) logic synthesis (based on known techniques)
  ● SAT-based Weakening (and strengthening) with high compaction potential (strength/time for size)

● Experimental results & Conclusions
Motivations & related works

- Interpolation (ITP) still a major engine in UMC portfolio [McMillan CAV’03]
- Other ITP usages
  - Formula/constraint synthesis in predicate abstr.
- Scalability problem
  - ITPs are large and highly redundant
- Previous efforts [Marques-Silva CHARME’05, D’Silva et al. VMCAI’08, Cabodi et al. FMSD’15, Rollini et al LPAR’13 (PeRIPLO)]
  - Proof reduction
  - Interpolant compaction
Background: Craig Interpolant for IMG in Unbounded MC
Interpolant: set view

\[ A(x, y) \cap \Lambda \bigcup B(y, z) \]
Interpolant: set view

\( \neg \land I(y) \land A(x,y) \land B(y,z) \)
Craig Interpolant from refutation proof

A \quad B

CNF clauses

UNSAT problem
\((A \land B = 0)\)
Craig Interpolant from refutation proof

Resolution graph

Null clause
Craig Interpolant from refutation proof

Resolution graph

Null clause

Unsatisfiable core

resolution

\[(A \lor p) \land (\neg p \lor B) \Rightarrow (A \lor B)\]
Craig Interpolant from refutation proof

A
\[ I(Y) = \text{Interpolant} \ (A(X,Y),B(Y,Z)) \]

B

Resolution graph

AND-OR circuit

Null clause
Craig Interpolant from refutation proof

\[ I(Y) = \text{Interpolant}(A(X,Y),B(Y,Z)) \]

A gate for each resolution node

Null clause

Resolution graph

AND-OR circuit

Interpolant \((A(X,Y),B(Y,Z))\)
Redundancy / Compaction

$|\text{Interpolant}| = O(|\text{proof}|)$

highly redundant $\Rightarrow$ compaction

- Reduce proof
  - Recycle-pivots, local transformations, proof restructure
- Combinational logic synthesis
  - BDD/SAT-sweeping
  - Rewriting
  - Refactoring
- Ad hoc logic synthesis
  - Logic synthesis using proof graph
  - ITE-based decompose & compact
Contributions

- AD-hoc logic synthesis (rewrite/refactor)
  - Implemented/tuned for interpolants
  - Interpolant strength unchanged
  - Trade-off speed for optimality
- Compaction by strengthening/weakening
  - Gate Level Abstraction
  - Proof (core) based
  - Expensive (SAT needed)
  - Trade-off strength for size
Circuit graph decomposition
Circuit graph decomposition
Circuit graph decomposition

Macrogates
Circuit graph decomposition

Macrogates
Circuit graph decomposition
Circuit graph decomposition
Circuit graph decomposition

Dominators
Circuit graph decomposition

\[ d \text{ dominates } n \]
Rewrite: direct ODC removal

\[
\begin{align*}
v_1 & \quad v_N \\
\text{a} & \quad f
\end{align*}
\]
Rewrite: direct ODC removal

\[ F(Y,a) = a \land g(Y,a) = a \land g(Y,1) \]
Rewrite: direct ODC removal
Algorithm

DirectOdcSimplify()

forall clusters CL

find node v ∈ cut(CL) with fanout(v) > 1 in CL

take u,t ∈ fanout(v) ∩ CL

if (domG(t) dominates u)

u <= constant // u is redundant
Rewrite: transitive ODC
Rewrite: transitive ODC
Refactor: search sharable term
Refactor: search sharable term
Refactor: search sharable term
Refactor: search sharable term
Algorithm

MacrogateRefactor()

for all macrogates $G$

for all nodes $v \in \text{cut}(G)$

find $\text{AND}(u, v) \notin G$ such that $u \in \text{cut}(G)$

refactor $G$ using $\text{AND}(u, v)$
Experimental results

- Implementation on PdTrav tool
  - HWMCC ’07 to ‘15
- ITPs stored on files from MC runs
- Picked 87 instances
- experiments also on ITPs from PeRIPLO (Sharygina’s group, Lugano)
Logic synthesis

- Benchmark set (87 ITPs)
  - Size range: $4 \cdot 10^5$ – $8.5 \cdot 10^6$ gates
  - Average size: $2 \cdot 10^6$ gates

- Experimental set-up
  - Time limit: 900 seconds
  - Memory limit: 8 GB

<table>
<thead>
<tr>
<th>Completed Benchmarks</th>
<th>Average compaction rate</th>
<th>Average execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance</td>
<td>87</td>
<td>61.31 %</td>
</tr>
<tr>
<td>ITP Simplify</td>
<td>68</td>
<td>83.06 %</td>
</tr>
<tr>
<td>No Direct ODC</td>
<td>68</td>
<td>80.83 %</td>
</tr>
<tr>
<td>No Refactoring</td>
<td>86</td>
<td>69.28 %</td>
</tr>
<tr>
<td>No Transitive ODC</td>
<td>72</td>
<td>80.93 %</td>
</tr>
<tr>
<td>ABC</td>
<td>31</td>
<td>94.96 %</td>
</tr>
</tbody>
</table>
Cumulative size

![Graph showing cumulative size for different benchmarks]

- Balance
- ITPsimplify
- NoDirectODC simplify
- NoMacrogateRefactor
- NoTransitiveODC simplify
- ABC simplify
Cumulative time
ITP weakening by SAT GLA

- GLA: Gate Level Abstraction
  - used here to abstract combinational circuit

- SAT-based abstraction (PBA)
  - SAT($I \land B$) => UNSAT CORE => abstract

- Given cut var, monotonicity => existential quantification by constant substitution

- NNF encoding
  - monotone w.r.t. all circuit nodes except Pis.
Interpolant weakening

\[ I(y) \land B(y, z) \]
Interpolant weakening

\[ I(y) \]

\[ I_W(y) \]

\[ B(y,z) \]
Interpolant weakening

$B(y, z) \land I(y)$
Interpolant weakening

\[ B(y, z) \land \text{UNSAT} \implies \text{CORE} \]
Interpolant weakening

\[ I(y) \]
Interpolant weakening

\[ I(y) \wedge B(y, z) \]
Interpolant weakening

\[ B(y, z) \land I(y, N) \]

EXTRA VARIABLE
Existential quantification

NNF encoding

NNF circuit monotone w.r.t. internal nodes

Quantification by constant substitution

\[ \exists N I(Y, N) = I(Y, 1) \]

1. \( I(Y) \Rightarrow I_{\text{NNF}}(Y) \)
2. \( \text{SAT}(I_{\text{NNF}}(Y) \land B(Y, Z)) \)
3. \( I_{\text{NNF}} \Rightarrow (\text{uns. core, abstract, inject 1}) \Rightarrow I_{\text{NNF, weak}} \)
4. \( I_{\text{NNF, weak}} \Rightarrow I_{\text{weak}} \)
Interpolant strengthening

\[ A(x,y) \land \neg \overline{l}(y) \]

Strengthening by weakening complement of interpolant w.r.t. A
SAT-based weakening

• Benchmark set (87 circuits)
  • Size range: $4 \cdot 10^5 – 8.5 \cdot 10^6$ gates
  • Average size: $2 \cdot 10^6$ gates

• Experimental set-up
  • Time limit: 3600 seconds
  • Memory limit: 8 GB

<table>
<thead>
<tr>
<th></th>
<th>Completed Benchmarks</th>
<th>Average compaction rate</th>
<th>Average execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>72</td>
<td>82.24 %</td>
<td>902.33 s</td>
</tr>
<tr>
<td>AB</td>
<td>63</td>
<td>97.99 %</td>
<td>1495.78 s</td>
</tr>
<tr>
<td>ABAB</td>
<td>57</td>
<td>99.59 %</td>
<td>1649.79 s</td>
</tr>
<tr>
<td>B</td>
<td>68</td>
<td>97.53 %</td>
<td>1324.51 s</td>
</tr>
<tr>
<td>BA</td>
<td>67</td>
<td>98.03 %</td>
<td>1371.30 s</td>
</tr>
<tr>
<td>BABA</td>
<td>60</td>
<td>99.56 %</td>
<td>1470.84 s</td>
</tr>
</tbody>
</table>
Cumulative size
Cumulative time

![Graph showing cumulative time against benchmark]
Conclusions

- Contributions:
  - Adapting logic synthesis for fast/scalable ITP compaction
  - More expensive SAT-based compaction (weakening/strengthening)

- Evaluation
  - Fast synthesis always used
  - Expensive weakening/strengthening
    - Avoid memory explosion (or time not critical)
    - Strength may impact on quality of result
Thank you!