A Two-Step Search Engine
For Large Scale Boolean Matching
Under NP3 Equivalence

Chak-Wa Pui, Peishan Tu, Haocheng Li, Gengjie Chen, Evangeline F. Y. Young
CSE Department, Chinese University of Hong Kong, Hong Kong

Speaker: Jordan, Chak-Wa Pui
Outline

• Introduction
• Algorithms
• Experimental Results
• Conclusion
Background

• What is Boolean matching
  • determine whether two Boolean functions are functionally equivalent under some constraints.
  • The relationship between the inputs(outputs) are called permutation, while the sign decision is called negation
  • Many variants, such as NPNP, PP, etc.

• Really difficult to solve even under P
• Widely used in security, library binding, ECO, etc.
Representation of Boolean Matching

• Any matching can be represented by these two matrices.

• By limiting the value assignment of $M_I, M_O$, we can formulate all kinds of Boolean matching problem into these framework.
  
  • For example, in P equivalence, $M_O$ is given. A matching result under P must satisfy:
    
    $\sum_{j=1}^{m_I} a_{i,j} = 1, \forall i = 1, ..., n_I$
    
    $\sum_{i=1}^{n} a_{i,j} = 1, \forall j = 1, ..., m_I$
    
    $b_{i,j} = 0, a_{i,m_I+1} = 0, \forall i, j$
Problem Formulation

• What is NP3
  • the permutation and negation of inputs/outputs like NPNP.

• Non-Exact and Projection (NP)
  • Allow unmatched outputs in ckt0
  • Allow constant binding in inputs of ckt1
  • Allow one-to-many binding in ckt0 inputs

• Aims at maximizing the number of mapped outputs of both ckt0 and ckt1.
Previous works

• Three types
  • Signature-based
    • prune the Boolean matching space by filtering impossible I/O correspondences
  • Canonical form-based
    • compare the canonical representations of two Boolean functions to find valid I/O matches
  • SAT-based
    • Good scalability and high efficiency as the SAT solver become stronger nowadays

• Limitation:
  • To our knowledge, there is no previous works on NP3 equivalence, which is a more general formulation of Boolean matching and have applications in security and ECO.
Algorithms

• Overall framework
• Output Solver
• Input Solver
Algorithms

• Overall framework
  • Output Solver
  • Input Solver
Overview of the framework

• Two-step
• SAT-based backtracking output solver
• SAT-based input solver
• Incremental

1. Add output constraints
2. Get the matched output pair

1. Add input constraints
2. Get the matched input pair
3. Construct mitter

Final Solution
Algorithms

- Overall framework
- Output Solver
  - Overview
  - Output functional constraints
  - Output solver heuristics
- Input Solver
Output Solver

- Feedbacks from input solver
  - If success, keep the current matched POs. O.W., forbid in later iterations
- Backtracking
  - No more pairs can be found if the current matched POs are kept
- Disable projection until no more output matching result can be found

\[
f(x) \quad g(x)
\]
Output Functional Constraints

• Definitions:
  • Function support
  • Structural support

• Constraints:
  • Forbid outputs $f_i$ and $g_j$ to be matched if $\text{FuncSupp}(f_i) > \text{FuncSupp}(g_j)$
  • Equal constraint enables faster output matching if some outputs share the same source
Output Solver Heuristics

• Output matching order heuristics
  • First match outputs with less functional/structural support and fanin

• Output grouping heuristic
  • Bad matched output pairs in early stage
    • Consider two functions $f$ and $g$, where $|f| = |g| = 4$, and the numbers of functional supports of $f_1, f_2, f_3, f_4$ are 1, 2, 3, 5, and the numbers of functional supports of $g_1, g_2, g_3, g_4$ are 2, 2, 4, 6. if $f_1$ is matched to $g_4$ at the beginning, either $f_3$ or $f_4$ cannot be matched to any $g_i$.

• How to avoid
  • For two circuits with the same number of outputs, do grouping
  • Avoid matching across groups
Algorithms

• Overall framework
• Output Solver

• Input Solver
  • Overview
  • Input Functional Constraints
  • Input Solver Heuristics
  • Input Symmetric Constraints
Input Solver

• Similar to [1][2]
  • Use counter example to prune solution space
    • Given $f_p$ and $g_q$ for Boolean matching under NP3 equivalence, if $f_p(\vec{x}) \neq g_q(\vec{y})$, then any PI matching is infeasible if it maps $\vec{x}$ to $\vec{y}$.

• Incremental
  • Counter examples from previous iterations of output and input solvers will be reused

Output Solver

1. Add previous counter examples
2. Add constraints
3. Get input pairs
   • If not success, end the loop
4. Construct miter
5. Solve miter
   • If not success, add counter example, goto 1
   • o.w., end the loop

Final Solution

Input Functional Constraints

• Remove redundant literal in a counter example
  • $\vec{x} = 1101, \vec{y} = 0101$, with $y_2 = 0, y_3 = 1$, $y_0, y_1$ is redundant, hence implying three more counter examples $\vec{y} = 0101, 0001, 1001, 1101$
  • Reduce time since most of the time is spent in SAT solving

• Two inputs are allowed to match if their supported outputs are matched

• Bind irrelevant inputs in circuit 1 to constant
Input Solver Heuristics

• Output grouping heuristic
  • Avoid bad matched output pairs in early stage
  • Group the outputs and avoid matching across groups

• Output group signature heuristics
  • Given no projection and constant binding, two inputs must support the same corresponding groups in order to be matched
    • Can be matched if \( W_{x_i} = W_{y_j} \)
    • If \( W_{x_i} \subset W_{y_j} \), we relax the constraints, let \( w \in W_{y_j} \) and \( w \notin W_{x_i} \), for any \( g_p \in W \)
      • The number of PO \( y_j \) support in \( w \) is small
      • In the matching of \( g_p \) there must be constant binding or projection
Input Symmetric Constraints

• What is symmetric
  • A pair of input \((x_i, x_j)\) is
    • positive symmetric on \(f_p\) if \(f_p(\vec{x}|_{x_i=0, x_j=1}) = f_p(\vec{x}|_{x_i=1, x_j=0})\) for any \(\vec{x}\).
    • negative symmetric on \(f_p\) if \(f_p(\vec{x}|_{x_i=0, x_j=0}) = f_p(\vec{x}|_{x_i=1, x_j=1})\) for any \(\vec{x}\).

• Symmetric inputs can only be bound to symmetric inputs
  • True in NP problems
  • Not in NP3
    • For example, given \(g = (y_1 \oplus y_2) \land y_3\), if we bind \(y_1\) and \(y_2\) to the same constant or same input, \(y_3\) will become redundant to \(g\)
Input Symmetric Constraints

• In NP
  • SymmSign for each input, it’s a sequence of number.
    • SymmSign(2i) (SymmSign(2i + 1)) means the number of inputs it’s positive (negative) symmetric with on output i.
    • For any pair of matched outputs \((f_p, g_q)\) whose functional support sizes are the same, two inputs can be matched if and only if \(\text{SymmSign}(2p) = \text{SymmSign}(2q)\) and \(\text{SymmSign}(2p + 1) = \text{SymmSign}(2q + 1)\)
  • Fast matching on inputs symmetric to all outputs

• In NP3
Input Symmetric Constraints

- In NP
- In NP3
  - Find symmetric groups that cannot be broken in circuit 1
  - Build symmetric constraints on these groups

<table>
<thead>
<tr>
<th>Case#</th>
<th>With symm</th>
<th>Without symm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Score</td>
<td>Time(s)</td>
</tr>
<tr>
<td>12</td>
<td>60</td>
<td>29</td>
</tr>
<tr>
<td>14</td>
<td>84</td>
<td>324</td>
</tr>
<tr>
<td>15</td>
<td>120</td>
<td>38</td>
</tr>
<tr>
<td>17</td>
<td>120</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>120</td>
<td>82</td>
</tr>
<tr>
<td>20</td>
<td>108</td>
<td>479</td>
</tr>
<tr>
<td>22</td>
<td>60</td>
<td>25</td>
</tr>
</tbody>
</table>
## Experimental Result

<table>
<thead>
<tr>
<th>Case#</th>
<th>Ours</th>
<th>1st Place</th>
<th>2nd Place</th>
<th>3rd Place</th>
<th>[19]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Score</td>
<td>Time(s)</td>
<td>Score</td>
<td>Score</td>
<td>Score</td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>1</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>192</td>
<td>18</td>
<td>192</td>
<td>192</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>192</td>
<td>13</td>
<td>192</td>
<td>192</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>180</td>
<td>57</td>
<td>180</td>
<td>136</td>
<td>132</td>
</tr>
<tr>
<td>4</td>
<td>192</td>
<td>3</td>
<td>192</td>
<td>192</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
<td>1</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>60</td>
<td>29</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>13</td>
<td>120</td>
<td>718</td>
<td>120</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case#</th>
<th>Ours</th>
<th>1st Place</th>
<th>2nd Place</th>
<th>3rd Place</th>
<th>[19]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Score</td>
<td>Time(s)</td>
<td>Score</td>
<td>Score</td>
<td>Score</td>
</tr>
<tr>
<td>14</td>
<td>84</td>
<td>1</td>
<td>84</td>
<td>84</td>
<td>84</td>
</tr>
<tr>
<td>15</td>
<td>120</td>
<td>1</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>16</td>
<td>96</td>
<td>22</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>17</td>
<td>120</td>
<td>3</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>120</td>
<td>82</td>
<td>-</td>
<td>24</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>108</td>
<td>479</td>
<td>24</td>
<td>48</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>60</td>
<td>24</td>
<td>60</td>
<td>48</td>
<td>60</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>192</td>
<td>98</td>
<td>192</td>
<td>108</td>
<td>73</td>
</tr>
<tr>
<td>26</td>
<td>120</td>
<td>1</td>
<td>120</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>2005</td>
<td>-</td>
<td>1801</td>
<td>1469</td>
<td>1418</td>
</tr>
</tbody>
</table>
Conclusion

• A two-step search engine to solve large scale Boolean matching under NP3 equivalence is proposed.

• Several heuristics are used to accelerate the searching process, which include modifying the matching order of output pairs, output grouping, and output group signature.

• New constraints are proposed to solve the Boolean matching problem under NP3 equivalence, which include support group size dependency constraints and symmetry related constraints.
Thanks
Appendix

\[
\text{score} = \sum_{i=0}^{m} q(f_i),
\]  

(6)

where \( f_i \) denote the \( i \)th primary output of \( \text{ckt} \theta \) and \( q(f_i) \) is calculated as Equation (7).

\[
q(f_i) = \begin{cases} 
K + \sum_{j=1}^{n} (c_{j,i} + d_{j,i}), & \text{if } \sum_{j=1}^{n} (c_{j,i} + d_{j,i}) \geq 1, \\
0, & \text{otherwise}.
\end{cases}
\]  

(7)