# 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

香港中文大學

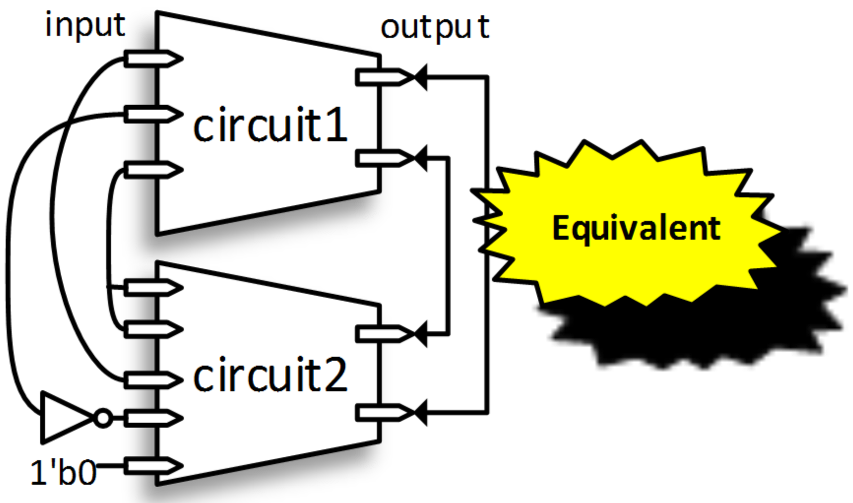
The Chinese University of Hong Kong

# 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:
    - $\sum_{j=1}^{m_I} a_{i,j} = 1, \forall i = 1, \dots, n_I$
    - $\sum_{i=1}^{n} a_{i,j} = 1, \forall j = 1, \dots, m_I$
    - $b_{i,j} = 0,\ a_{i,m_I+1} = 0,\ \forall i, j$

$$M_I = \begin{array}{cc} & \begin{array}{ccccccc} x_1 & \neg x_1 & \cdots & x_{m_I} & \neg x_{m_I} & 0 & 1 \end{array} \\ \begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_{n_I} \end{array} & \begin{bmatrix} a_{1,1} & b_{1,1} & \cdots & a_{1,m_I} & b_{1,m_I} & a_{1,m_I+1} & b_{1,m_I+1} \\ a_{2,1} & b_{2,1} & \cdots & a_{2,m_I} & b_{2,m_I} & a_{2,m_I+1} & b_{2,m_I+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ a_{n_I,1} & b_{n_I,1} & \cdots & a_{n_I,m_I} & b_{n_I,m_I} & a_{n_I,m_I+1} & b_{n_I,m_I+1} \end{bmatrix} \end{array},$$

$$M_O = \begin{array}{cc} & \begin{array}{ccccc} f_1 & \neg f_1 & \cdots & f_{m_O} & \neg f_{m_O} \end{array} \\ \begin{array}{c} g_1 \\ g_2 \\ \vdots \\ g_{n_O} \end{array} & \begin{bmatrix} c_{1,1} & d_{1,1} & \cdots & c_{1,m_O} & d_{1,m_O} \\ c_{2,1} & d_{2,1} & \cdots & c_{2,m_O} & d_{2,m_O} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{n_O,1} & d_{n_O,1} & \cdots & c_{n_O,m_O} & d_{n_O,m_O} \end{bmatrix} \end{array}.$$

# 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.

$$M_I = \begin{array}{c|ccccccc} & x_1 & \neg x_1 & \cdots & x_{m_I} & \neg x_{m_I} & 0 & 1 \\ \hline y_1 & a_{1,1} & b_{1,1} & \cdots & a_{1,m_I} & b_{1,m_I} & a_{1,m_I+1} & b_{1,m_I+1} \\ y_2 & a_{2,1} & b_{2,1} & \cdots & a_{2,m_I} & b_{2,m_I} & a_{2,m_I+1} & b_{2,m_I+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ y_{n_I} & a_{n_I,1} & b_{n_I,1} & \cdots & a_{n_I,m_I} & b_{n_I,m_I} & a_{n_I,m_I+1} & b_{n_I,m_I+1} \end{array},$$

$$M_O = \begin{array}{c|ccccc} & f_1 & \neg f_1 & \cdots & f_{m_O} & \neg f_{m_O} \\ \hline g_1 & c_{1,1} & d_{1,1} & \cdots & c_{1,m_O} & d_{1,m_O} \\ g_2 & c_{2,1} & d_{2,1} & \cdots & c_{2,m_O} & d_{2,m_O} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ g_{n_O} & c_{n_O,1} & d_{n_O,1} & \cdots & c_{n_O,m_O} & d_{n_O,m_O} \end{array}.$$

$$\sum_{i=1}^{n_O}\sum_{j=1}^{m_O}(c_{i,j}+d_{i,j}) > 0, \tag{1}$$

$$\sum_{j=1}^{m_O}(c_{i,j}+d_{i,j}) \le 1, \quad \forall i = 1,\cdots,n_O, \tag{2}$$

$$\sum_{j=1}^{m_I+1}(a_{i,j}+b_{i,j}) = 1, \quad \forall i = 1,\cdots,n_I, \tag{3}$$

$$\Big(\bigwedge_{c_{j,i}=1} f_i \equiv g_j\Big) \wedge \Big(\bigwedge_{d_{j,i}=1} f_i \equiv \neg g_j\Big), \tag{4}$$

$$\Big(\bigwedge_{a_{j,i}=1} x_i \equiv y_j\Big) \wedge \Big(\bigwedge_{b_{j,i}=1} x_i \equiv \neg y_j\Big). \tag{5}$$

# 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

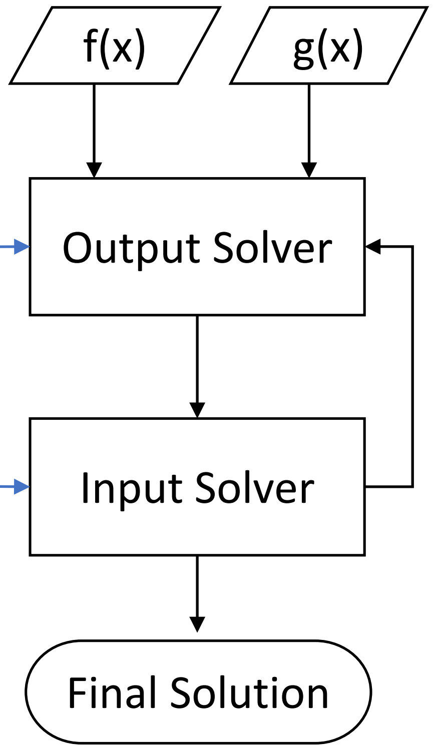
- Overall framework
- Output Solver
- Input Solver

# 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**

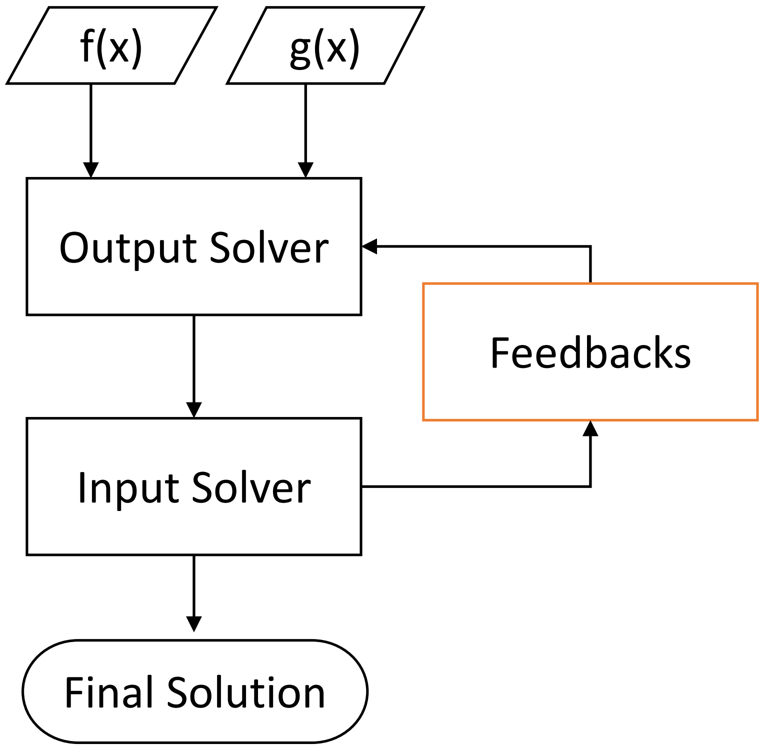


# Algorithms

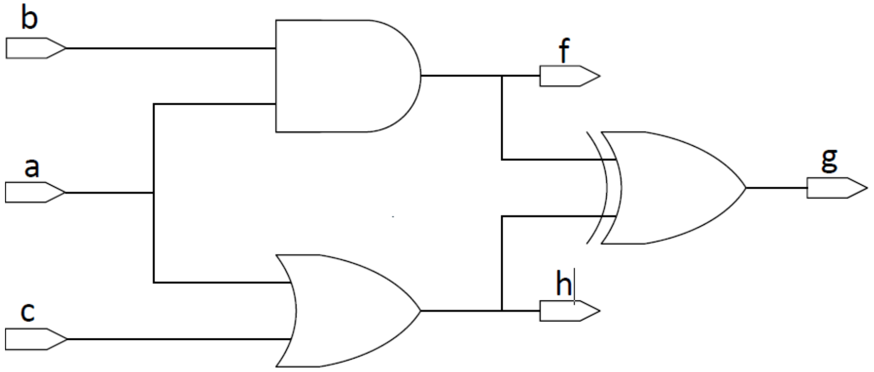
- Overall framework
- Output Solver
  - Overview
  - Output functional constraints
  - Output solver heuristics
- Input Solver

# 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



# Output Functional Constraints




- Definitions:
  - Function support
  - Structural support
- Constraints:
  - Forbid outputs $f_i$ and $g_j$ to be matched if $FuncSupp(f_i) > FuncSupp(g_j)$
  - Equal constraint enables faster output matching if some outputs share the same source

# Output Solver Heuristics

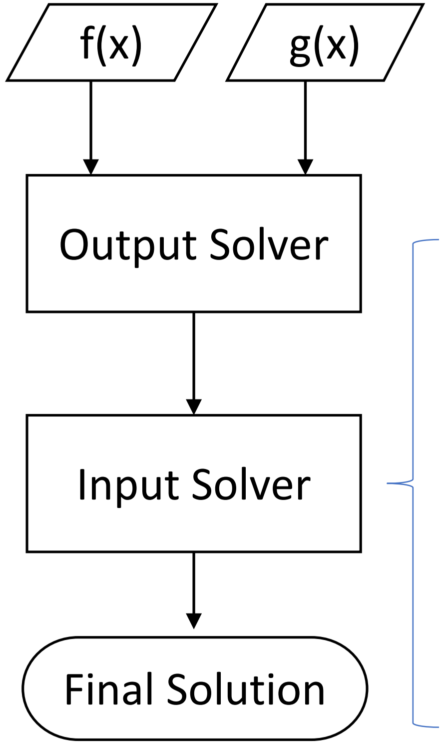
- Output matching order heuristics
  - First match outputs with less functional/structural support and fanin
- Output grouping heuristic
  - Bad matched output pairs in early stage
    - Consider two functions $f$ and $g$, where $|f| = |g| = 4$, and the numbers of functional supports of $f_1, f_2, f_3, f_4$ 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 $f1$ 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 maching is infeasible if it maps $\vec{x}$ to $\vec{y}$.
- Incremental
  - Counter examples from previous iterations of output and input solvers will be reused




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

[1] C.-F. Lai, J.-H. R. Jiang, and K.-H.Wang. Boolean matching of function vectors with strengthened learning. ICCAD2010.
[2] C.-F. Lai, J.-H. R. Jiang, and K.-H. Wang. Boom: a decision procedure for boolean matching with abstraction and dynamic learning. DAC2010.

# 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 $g = (y_1 \oplus y_2) \wedge y_3$, 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 $SymmSign_i(2p) = SymmSign_j(2q)$ and $SymmSign_i(2p+1) = SymmSign_j(2q+1)$
  - 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

| Case# | With symm | | Without symm | |
|---|---|---|---|---|
| | Score | Time(s) | Score | Time(s) |
| 12 | 60 | 29 | - | - |
| 14 | 84 | 324 | 36 | 1391 |
| 15 | 120 | 38 | 120 | 99 |
| 17 | 120 | 3 | 120 | 73 |
| 19 | 120 | 82 | 36 | 75 |
| 20 | 108 | 479 | 96 | 188 |
| 22 | 60 | 25 | - | - |

# Experimental Result

| Case# | Ours | | 1st Place | 2nd Place | 3rd Place | [19] | |
|---|---|---|---|---|---|---|---|
| | Score | Time(s) | Score | Score | Score | Score | Time(s) |
| 0 | **25** | **1** | 25 | 25 | 25 | 25 | 1 |
| 1 | **192** | **18** | 192 | 192 | 192 | 48 | 17 |
| 2 | **192** | **13** | 192 | 192 | 192 | 36 | 5 |
| 3 | **180** | **57** | 180 | 136 | 180 | 132 | 9 |
| 4 | **192** | **3** | 192 | 192 | 192 | 36 | 10 |
| 5 | **-** | **-** | - | - | - | - | - |
| 6 | **-** | **-** | - | - | - | - | - |
| 7 | **-** | **-** | - | - | - | - | - |
| 8 | **-** | **-** | - | - | - | - | - |
| 9 | **-** | **-** | - | - | - | - | - |
| 10 | **24** | **1** | 24 | 24 | 24 | 24 | 1 |
| 11 | **-** | **-** | - | - | - | - | - |
| 12 | **60** | **29** | 60 | 60 | 60 | - | - |
| 13 | **120** | **718** | 120 | - | - | - | - |

| Case# | Ours | | 1st Place | 2nd Place | 3rd Place | [19] | |
|---|---|---|---|---|---|---|---|
| | Score | Time(s) | Score | Score | Score | Score | Time(s) |
| 14 | **84** | **1** | 84 | 84 | 84 | - | - |
| 15 | **120** | **1** | 120 | 120 | 120 | 60 | 2 |
| 16 | **96** | **22** | 96 | 96 | 96 | 96 | 6 |
| 17 | **120** | **3** | 120 | 120 | 120 | - | - |
| 18 | **-** | **-** | - | - | - | - | - |
| 19 | **120** | **82** | - | 24 | - | - | - |
| 20 | **108** | **479** | 24 | 48 | - | 12 | 20 |
| 21 | **-** | **-** | - | - | - | - | - |
| 22 | **60** | **24** | 60 | 48 | 60 | - | - |
| 23 | **-** | **-** | - | - | - | - | - |
| 24 | **-** | **-** | - | - | - | - | - |
| 25 | **192** | **98** | 192 | 108 | 73 | - | - |
| 26 | **120** | **1** | 120 | 0 | 0 | - | - |
| Total | **2005** | **-** | 1801 | 1469 | 1418 | 469 | - |

# 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

$$\text{score} = \sum_{i=0}^{m_o} q(f_i), \tag{6}$$

where $f_i$ denote the $i$th primary output of *ckt0* and $q(fi)$ is calculated as Equation (7).

$$q(f_i) = \begin{cases} K + \sum_{j=1}^{n_O}(c_{j,i} + d_{j,i}), & \text{if } \sum_{j=1}^{n_O}(c_{j,i} + d_{j,i}) \geq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$