Problem A: Multi-bit Large-scale Boolean Matching

Chung-Han Chou, Chih-Jen (Jacky) Hsu, Chi-An (Rocky) Wu, and Kuan-Hua Tu, Kei-Yong Khoo
Cadence Design Systems, Inc.

- 2023.02.13 First Version
- 2023.04.13 Update description, evaluation criteria, and primitive gates

1. Introduction

Boolean Matching [1] (Shown in Figure 1) is one of the most widely used techniques in electronic design automation. It determines whether two Boolean functions are functionally equivalent under the permutation and negation of inputs and outputs, and achieves better quality in synthesis [3][4] and technology mapping [2]. Large-scale Boolean Matching has been applied in ECO [5][6], equivalence checking [7] or even hardware Trojan detection [8].

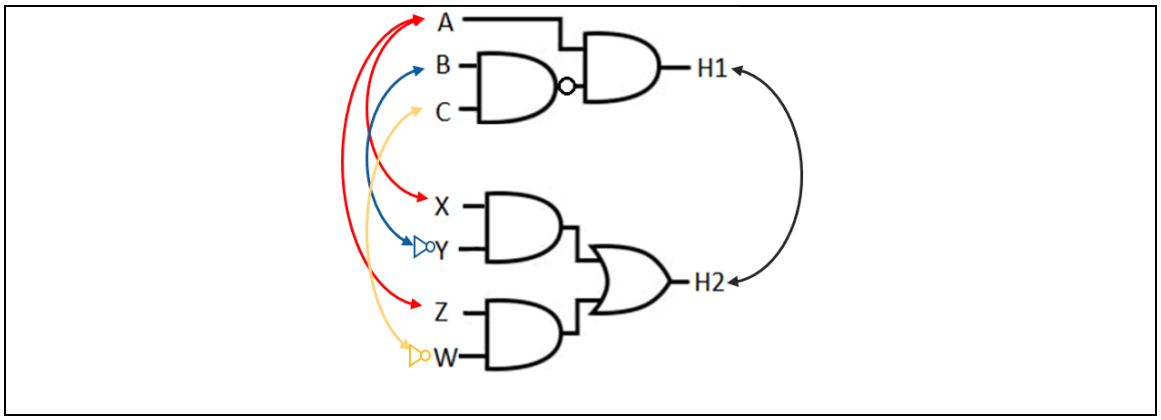

Figure 1. Example of Boolean Matching.

In the basic Boolean Matching problem, we negate and permute circuit input/output, so that the two Boolean functions can be equivalent. The time complexity for an exhaustive solution is $O(2^{n+1}n!)$ [20]. However, such brute-force method is extremely time-consuming when the numbers of input/output ports are large.

In modern digital IC design, the large number of input/output ports are usually caused by the buses or datapaths in the design. If it is known that some input/output ports come from the same bus, can we have an efficient algorithm to solve the Boolean matching

problem or even reduce the complexity of solving the Boolean matching problem? For example, Figure 2 shows two designs with five input ports and four output ports, respectively. If we solve the Boolean matching problems by exhaustive search, the number of permutations could be $5! \times 4! = 2880$. However, if we know that there are four buses in the first circuit (Circuit I), {a0, a1}, {b0, b1}, {h0, h1}, {m0, m1} , and four buses in the second circuit (Circuit II), {x0, x1}, {y0, y1}, {u0, u1}, {w0, w1}, the permutations can be significantly reduced to 64.

In this contest, contestants need to find out the matched ports between two given combinational circuits, Circuit I and Circuit II. The goal of this contest is to maximize the number of the matched ports between Circuit I and Circuit II. In addition, matching the input ports of Circuit II to constant one (1'b1) or constant zero (1'b0) is allowed, and matching multiple input/output ports of Circuit II to those of Circuit I is also allowed. Take Figure *2* as an example, if we have the following matched ports, (a0, x0), (a1, x1), (b1, y1), (b0, y0), (c, z), then there are two equivalent output buses, {h0, h1} = {u0, u1} and {m0, m1} = {w0, w1}. However, if the matched ports are (a0, y0), (a1, y1), (b1, x1), (b0, x0), (c, z), then we only get one equivalent output bus, {h0, h1} = {u0, u1}.

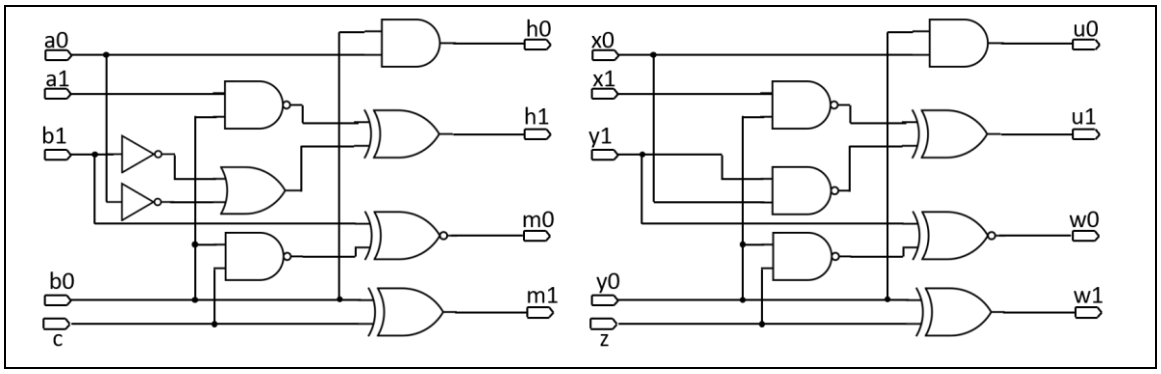

Figure 2. Boolean Matching on design with buses.

2. Background

The most matured approaches to solve the basic Boolean matching problem include canonical-form-based, signature-based, and SAT-based algorithms. Each approach has its own pros and cons. The canonical-form-based algorithms check whether two Boolean functions can be represented by the same canonical-form [2][7][9][10][11][12][13]. The signature-based algorithms use structural or functional properties as the signatures in order to search for inputs/outputs relation of two Boolean functions [13][14][15][16][17]. The SAT-based algorithms convert the Boolean matching problem into a SAT

problem[15][18][19]. Contestants can either apply the existing algorithm or explore new ideas to solve this problem.

3. Problem Formulation and Input/Output Format

Given two combinational circuits, Circuit I and Circuit II, and the list of buses, the contestants need to develop their programs to maximize the number of matched ports between Circuit I and Circuit II.

3.1 Program requirement

The program must be performed on the Linux platform provided by the contest organizer with a time limit, 3600 seconds, for each benchmark. Parallel computation with multiple threads or processes is NOT allowed. The program must be named as `bmatch` and accept two arguments, "`<input>`" and "`<match>`". The input file `<input>` includes the file path of both Circuit I and Circuit II and the bus information. Contestants need to output the file `<match>`, which describes the matching result. The command-line execution format will be "`./bmatch <input> <match>`". The formats of both `<input>` and `<match>` are given in the following subsections.

3.2 Input file format

Figure 3 shows an example of the input file <input>, which include the information of Circuit I and Circuit II. The first line gives the file path of Circuit I, followed by an integer N indicating the number of buses in Circuit I, and the details of each bus. For each bus, it starts with an integer M. After that, there will be M input/output port names which are considered as a group of signals in the bus. The rest of Figure 3 shows Circuit II, which has the same format as Circuit I does.

```
circuit_1.v
4
2 a0 a1
2 b0 b1
2 h0 h1
2 m0 m1
circuit_2.v
4
2 x0 x1
2 y0 y1
2 u0 u1
2 w0 w1
```

Figure 3. Example of input file.

Notes that:

- Both circuits are given in Verilog format without hierarchy. There will be only one module for both circuits named "top".
- The maximum length of input/output names is 256 chars.
- The netlist is composed of:
    1. primitive gates (and, or, nand, nor, not, xor, xnor, buf)
    2. wires
    3. constant values (1'b1, 1'b0)
- All the signals including primary inputs and primary outputs are scalars. That means, they are all one-bit signals.

3.3 Output file format and requirements

The output file <match> includes three types of matching groups, including the matched input ports, the matched output ports, and the matched constant values. The details for each group will be further explained in the following.

```
INGROUP
1 + <input_name1>
2 + <input_name2>
2 - <input_name3>
…
END
```

(a)

```
OUTGROUP
1 + <output_name1>
2 - <output_name2>
2 + <output_name3>
…
END
```

(b)

```
CONSTGROUP
+ <input_name1>
- <input_name2>
+ <input_name3>
…
END
```

(c)

Figure 4. Example of output file.

Input (Output) group: The input (output) group, or INGROUP (OUTGROUP), as shown in Figure 4(a) (Figure 4 (b)), indicates the matched input (output) ports between Circuit I and Circuit II. Each input (output) group starts with the keyword INGROUP (OUTGROUP). After the keyword, the contestants need to list the input (output) ports line by line. Each line

starts with an integer, either 1 or 2, denoting Circuit I or Circuit II, followed by the character "+" or "-", indicating positive or negative matching. After the character, each line ends with the name of an input (output) port. After all the matched ports are listed, the keyword **END** finishes the input (output) group.

Constant group: Only the input ports of Circuit II can be binded to the constant values. Figure 4(c) gives an example of a constant group. It starts with the keyword **CONSTGROUP**. Each line after the keyword denotes the constant binding for the inputs of Circuit II, where "+ <name1>" means binding the input port <name1> to 1'b0, and "- <name2>" means binding the input port <name2> to 1'b1.

Notes that the resulting output file must strictly follow the constraints below:

- Each input/output port can only appear in either one of the groups.
- For each input/output group, there must be exactly one input/output port coming from Circuit I.

4. Evaluation Criteria

The runtime limit for each case is 3600 seconds. The more ports are matched, the higher score the contestant will receive. The detailed grading criteria are given below.

- There will be no point for the benchmark if the matching result violates the constraints. For example, any of the input/output ports appears in more than one group, or more than one port from Circuit I appears in the same group.
- For each output group, if all the group matched ports are verified equivalence, then the team gets the ***points*** equal to the number of matched ports.
- As a result, the maximum points for each benchmark is equal to the total number of output ports in both circuits. We define

max_pts for each benchmark = ***#output_port(Circuit I) + #output_port(Circuit II)***

- The ***score*** of each benchmark = $\frac{\boldsymbol{points}}{\boldsymbol{max\_pts}} \times 100\%$
- ***final_score*** is the summation of the $\boldsymbol{scores}$ of all benchmarks.
- The team having the highest ***final_score*** wins the contest.

5. References

[1] Hadi Katebi and Igor L. Markov, Large-scale Boolean Matching, DATE, 2010

[2] Luca Benini and Giovanni De Micheli, A Survey of Boolean Matching Techniques for Library Binding,Design Automation Electronic System, ACM, 1997.

[3] J. Cong and Y.-Y. Hwang, Boolean Matching for LUT-Based Logic Blocks with Applications to Architecture Evaluation and Technology Mapping, CAD Integrated Circuit System, IEEE, 2006.

[4] C. Yu, L. Wang, C. Zhang, Y. Hu, and L. He, Fast Filter-Based Boolean Matchers, Embedded Systems Letters, IEEE, 2013.

[5] S. Krishnaswamy, H. Ren, N. Modi, and R. Puri, DeltaSyn: An efficient logic-difference optimizer for ECO synthesis, ICCAD, 2009.

[6] S.-L. Huang, W.-H. Lin, C.-Y. (Ric) Huang, Match and replace - A functional ECO engine for multi-error circuit rectification, ICCAD, 2011.

[7] J. Mohnke, P. Molitor, and S. Malik, Application of BDDs in Boolean matching techniques for formal logic combinational verification, Software tools for Technology Transfer, 2001.

[8] P. Swierczynski, M. Fyrbiak, C. Paar, and C. Huriaux, Protecting against Cryptographic Trojans in FPGAs, FCCM, 2015.

[9] S. Chatterjee, A. Mishchenko, R. Brayton, Reducing Structural Bias in Technology Map, ICCAD, 2005

[10] G. Agosta, F. Bruschi, G. Pelosi, and D. Sciuto, A Transform-Parametric Approach to Boolean Matching, CAD Integrated Circuit System, IEEE, 2009.

[11] Z. Huang, L. Wang, Y. Nasikovskiy, and A. Mishchenko, Fast Boolean matching based on NPN classification, ICFPT, 2013.

[12] A. Petkovska, M. Soeken, G. De Micheli, P. Ienne and A. Mishchenko, Fast hierarchical NPN classification, FPL, 2016.

[13] J. Zhang, G. Yang, W. N. N. Hung, and J. Wu, A Canonical-Based NPN Boolean Matching Algorithm Utilizing Boolean Difference and Cofactor Signature, IEEE Acess, 2017.

[14] A. Abdollahi, Signature based Boolean matching in the presence of don't care, DAC, 2008.

[15] K.-H. Wang, C.-M. Chan, and J.-C. Liu, Simulation and SAT-Based Boolean Matching for Large Boolean Networks, DAC, 2009.

[16] F. Wang, J. Zhang, L. Wu, W. Zhang and G. Luo, Search space reduction for the non-exact projective NPNP Boolean matching problem, ISCAS, 2017.

[17] J. Zhang, W. Guo, G. Yang, Y. Zhu, and X. Lv, A Heuristic Boolean NPN Equivalent Matching Verification Method Based on Shannon Decomposition, IEEE Acess, 2022.

[18] C.F. Lai, J.-H.R. Jiang, and K.-H. Wang, BooM: A Decision Procedure for Boolean Matching with Abstraction and Dynamic Learning, DAC, 2010

[19] C.F. Lai, J.-H. R. Jiang, and K.-H. Wang, Boolean Matching of Function Vectors with Strengthened Learning, ICCAD, 2010

[20] J. Zhang, L. Ni, S. Zheng, H. Liu, X. Zou, F. Wang, and G. Luo, ''Enhanced fast Boolean matching based on sensitivity signatures pruning,'' ICCAD, 2021.