Invited Paper: 2023 ICCAD CAD Contest Problem A: Multi-bit Large-scale Boolean Matching

Chung-Han Chou
Cadence Taiwan, Inc.
Hsinchu, Taiwan
chchou@cadence.com

Chih-Jen (Jacky) Hsu
Cadence Design System, Inc.
San Jose, CA
jackyhsu@cadence.com

Chi-An (Rocky) Wu
Cadence Taiwan, Inc.
Hsinchu, Taiwan
rockywu@cadence.com

Kuan-Hua Tu
Cadence Taiwan, Inc.
Hsinchu, Taiwan
sopare@cadence.com

Kei-Yong Khoo
Cadence Design System, Inc.
San Jose, CA
khoo@cadence.com

2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD) | 979-8-3503-2225-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCAD57390.2023.10323797

***Abstract*—Boolean Matching problem determines whether two Boolean functions are functionally equivalent under the permutation and negation of inputs and outputs. Prior contest[1] has extended the problem to allow inputs binding with constant value and the ports matching can be one-to-many projection. In this contest, we further extended the problem to consider the ports grouping information, which is caused by the buses or datapaths in the modern digital IC design.**

Keywords—Boolean Matching, Equivalence Checking, Engineering Change Order, Word-Level Datapath

I. Introduction

Boolean Matching [2] (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 [4][5] and technology mapping [3]. Large-scale Boolean Matching has been applied in ECO [6][7], equivalence checking [8] or even hardware Trojan detection [9].

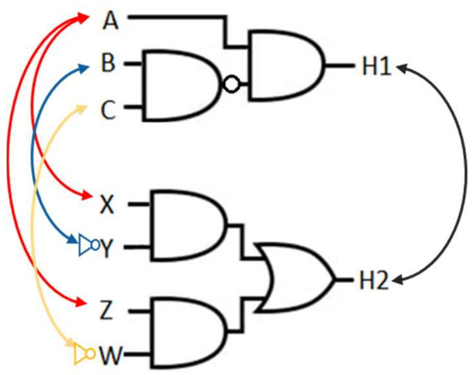

Figure 1. Example of Boolean Matching.

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(2n+1n!) [21]. 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! × 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),

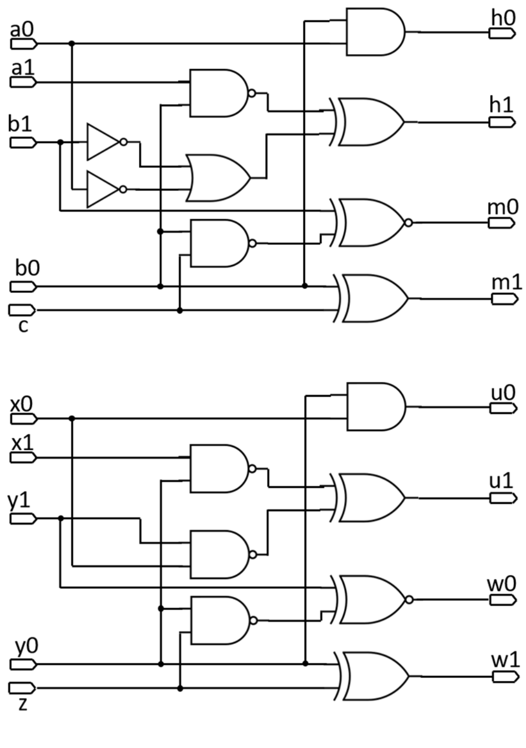

Figure 2. Boolean Matching on design with buses.

(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}.

Rest of the paper is organized as follows: The background is provided in section II, the problem formulation and detail input/output format are given in Section III. The evaluation criteria are shown in Section IV. The benchmark suite is present in Section V. Finally, we conclude this paper in Section VI.

II. 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 [3][8][10][11][12][13][14]. The signature-based algorithms use structural or functional properties as the signatures in order to search for inputs/outputs relation of two Boolean functions [14][15][16][17][18]. The SAT-based algorithms convert the Boolean matching problem into a SAT problem[16][19][20]. Contestants can either apply the existing algorithm or explore new ideas to solve this problem.

III. 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 num

ber of matched ports between Circuit I and Circuit II.

A. *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.

```
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.

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.

B. *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.

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.

C. *Output file format and requirements*

The output file <match> includes three types of matching

```
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.

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.

- 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 bound 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.

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

V. Benchmark Suite

The benchmark suite is generated from ISCAS'85 benchmark circuits and subset of industrial design. The overall testcase generating flow is as following:

1. Modify the netlist as following:
   - Select a subset of input/output, inverse the polarity.
   - Select a subset of input and merged as a pseudo input.
   - Select a subset of inputs and bind the values to constant 1 or constant 0.
2. Generate circuit I and circuit II by perform synthesis with different configuration/optimization stages using Cadence Genus Synthesis Solution for the original circuit/modified circuit.
3. Shuffle/rename the primary inputs, primary outputs, and nets.

Total 10 public cases and 10 hidden cases are generated in this contest benchmark suite. TABLE I. shows the testcase information for each public case. The first column is the name of each testcase. Column two is the number of gates of each case. The Third column is the number of primary inputs and the number of primary outputs. The number of buses of each case is given in column four. The number of inverted PI/PO, merged PI, and PIs bound to constant values are shown in column five.

TABLE I. Table Type Styles

#	#Gate	#pi / #po	#bus	#inv / #mer / #const
case01	9	5 / 4	4	0 / 0 / 0
case02	38	12 / 4	4	0 / 0 / 0
case03	271	41 / 32	3	0 / 0 / 0
case04	59	12 / 4	4	6 / 0 / 0
case05	1620	178 / 123	32	123 / 0 / 0
case06	704	86 / 26	10	0 / 4 / 0
case07	77	14 / 8	4	0 / 0 / 2
case08	227	36 / 7	5	3 / 2 / 2
case09	13003	256 / 120	47	13 / 9 / 6
case10	1550	207 / 108	19	19 / 11 / 13

VI. Conclusions

We extended the Boolean Matching problem to consider the input/output ports grouping information. The grouping information is caused by the buses or datapaths in the modern digital IC design. Total 20 public and hidden testcases are generated from ISCAS'85 and a subset of industrial designs. And the evaluation function quantifies the performance by the number of correctly matched output ports. We expect the bus information can greatly reduce the complexity of Boolean matching problem.

VII. Acknowledgments

The authors acknowledge to Prof. Takashi Sato, Prof. Chun-Yao Wang, Prof. Tsung-Wei Huang, Prof. Yu-Guang Chen to hold the contest. We also acknowledge to Prof. Mark Po-Hung Lin for reviewing this problem.

VIII. References

[1] C. Wu, C. Hsu and K. Khoo, "ICCAD-2016 CAD contest in non-exact projective NPNP boolean matching and benchmark suite," Proc. ICCAD, 2016.

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

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

[4] 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.

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

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

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

[8] 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.

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

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

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

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

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

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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

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

[21] 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.