Experimental Comparison of Logic Circuit Synthesis Methods

Maksim Vershkov
MIEM
HSE
Moscow, Russia
mdvershkov@edu.hse.ru

Alexey Yagzhov
MIEM
HSE
Moscow, Russia
aayagzhov@edu.hse.ru

Nikita Romanov
MIEM
HSE
Moscow, Russia
nsromanov_1@edu.hse.ru

Egor Znatnov
Faculty of Computer Science, HSE
Moscow, Russia
epznatnov@edu.hse.ru

Anna Fedotova
Faculty of Computer Science, HSE
Moscow, Russia
aafedotova_6@edu.hse.ru

***Abstract*—This paper presents the results of an experimental comparison of methods for the synthesis of combinational logic circuits that implement specified Boolean functions. The comparison was based on an estimate of power, delay and area of synthesized logic circuits. The following methods were considered: the method of Akers, bi-decomposition, the methods of cascades, Minato-Morreale, Reed-Muller and DSD-decomposition.**

Keywords—Boolean functions, logic optimization, electronic design automation

I. Introduction

The technology-independent optimization (optimization) of logic circuits is one of the most important steps in the logic synthesis process with a significant impact on the quality of the digital systems being designed.

Different methods of synthesis of logic circuits implementing given Boolean functions (synthesis methods) are widely used in logic optimization approaches. For instance, these methods can be used in optimization that is based on rewriting that is the replacement of subcircuits of the original logic circuit with other subcircuits that more effectively implement the same functions [1].

There are three main criteria for logic optimization (metrics for evaluating these criteria in a logic net are given in parentheses):

1) power (switching activity);
2) delay (the longest path from input to output);
3) area (the number of logic gates).

The aim of this study was to perform an experimental comparison of logic circuit synthesis methods and to identify the best one for each optimization criterion.

Other published papers also present the results of comparisons between different synthesis methods. In [2], the considered methods are as follows: sum-of-products (SOP) and product-of-sums (POS) two-level expressions, MUX-based expressions, Quine-McCluskey [3] method and different types of XOR expressions. The comparison was made for all Boolean functions of 3 and 4 variables and for a thousand random generated functions of 5 variables. The comparison was based on estimating the delay and area of synthesized logic circuits without technology mapping.

In [4], a new approach to bi-decomposition of Boolean functions is described and compared with the best algorithms in logic synthesis tools such as FBDD [5], SIS [6], ABC [7]. Sixteen circuits from the MCNC [8], ISCAS [9] and IWLS [10] benchmark suites were used for the comparison by power, delay and area after the process of technology mapping.

In [11], a comparison was made between the size (number of products in the SOP) of the minimal SOP of a Boolean function and the size of the SOP obtained by Minato-Morreale [12] method. The study was considered for different Boolean functions with the number of variables ranging from 3 to 20.

In this article an experimental comparison of methods of synthesis of logic circuits implementing given Boolean functions was carried out. The following methods were considered: the method of Akers [13], bi-decomposition [14], the methods of cascades [15], Minato-Morreale [12], Reed-Muller [16] and DSD-decomposition [17]. The circuits were synthesized in bases of different logic gates without technology mapping processing. Power, delay and area of the circuits were estimated for the comparison. Boolean functions of size from 4 to 10 variables were considered in the experiment. The choice of a benchmark may have a significant impact on the results of experiments. In this study we decided to generate a set of tests based on information about the frequency of occurrence of NPN-equivalence classes of Boolean functions of four variables.

This paper is organized as follows: Section II provides a brief overview of considered methods. Section III describes the methodology of the tests carried out. Section IV presents the results of this study and Section V is a conclusion.

II. Methods Overview

Akers method [13] is used to create majority-based circuits. It is the iterative algorithm that implies the manipulations of the table created from the given truth table. The columns of the converting table correspond to gates arguments. At each step of the algorithm, all sets of triples of the columns are iterated and a truth table of a majority gate is evaluated. Received truth tables are compared with each other, after which the best one is selected according to some heuristics. It is inserted into the table as a column. The algorithm terminates after obtaining the truth table that is equal to the given one.

Bi-decomposition [14] is based on extracting the superposition of two Boolean functions from a single source function:

$$f = \varphi(g_1(z_1), g_2(z_2)), \quad (1)$$

where f is a source function, g_1 and g_2 are Boolean functions with sets of arguments z_1 and z_2 respectively, φ is a given boolean function of two arguments.

Function φ is typically represented by logic operations such as OR (NOR), AND (NAND), and XOR. Different constraints can be imposed on the arguments of new functions, such as requiring subfunctions to have disjoint support. Alternatively, these arguments can be given before bi-decomposition. In this article the method of heuristic decomposition was considered. This means that the arguments of g_1 and g_2 are not provided, and there is only one constraint for their arguments: the number of arguments must be less than the number of arguments of the source function. Nevertheless, the probability of existence of decomposition of this kind is law, especially for completely specified functions. If decomposition is impossible, when the source function is broken down into two functions, one or both of them have the same number of variables. The circuit synthesis algorithm using bi-decomposition is recursive. The two Boolean functions obtained at each step of recursion are decomposed in the same manner.

The method of cascades [15] is based on the Shannon [18] decomposition, which represents the source function as the sum of two sub-functions:

$$f(x_1, x_2, \dots, x_n) = x_n \cdot f(x_1, x_2, \dots, x_{n-1}, 1) \cup \bar{x}_n \cdot f(x_1, x_2, \dots, x_{n-1}, 0), \quad (2)$$

where f is a source Boolean function, x_i is a i argument of this function. This method is also recursive, and two sub-function are decomposed in the same way.

The Minato-Morreale [12] method is a technique for obtaining an irredundant sum-of-product (ISOP) of Boolean function. An ISOP is a sum-of-products (SOP) of Boolean function that cannot have any literal or product removed without losing equivalence to the source function. The study considered two variations of the method: with and without algebraic factoring. Algebraic factoring is an algorithm that transforms SOP, reducing the number of literals and products. This method does not guarantee obtaining the ISOP of a source Boolean function with the minimum number of products [11].

The Reed-Muller method [16] aims to obtain modulo-2 sums of products for a source Boolean function (polynomial). There are only positive or negative literals for each variable in this polynomial. In this article we considered expressions where all literals are positive (Zhegalkin polynomial [19]).

DSD-decomposition (Disjoint Support Decomposition) [17] is based on decomposition of a source Boolean function into several subfunctions with disjoint support:

$$f = k(a_1, a_2, \dots, a_n), \quad (3)$$

where f is a source Boolean function, $a_1, a_2, \dots, a_n$ are subfunctions in decomposition, k is a function that links these subfunctions.

In this method the number of subfunctions is equal to or greater than 2 (depends on the source function). AND, OR, XOR or the Prime function (Boolean function of 3 variables or greater that cannot be further decomposed) can be used as k. The method is recursive, new functions are decomposed in the same way.

The bases of synthesized circuits for each considered method are presented in Table I.

TABLE I. BASIS OF SYNTHESIZED CIRCUITS

Method	**Logic gates**
Akers	MAJ, NOT, 0
Bi-decomposition	AND, NOT
Cascades	AND, OR, NOT, 0, 1
Minato-Morreale	AND, NOT
Minato-Morreale with factoring	AND, NOT
DSD-decomposition	AND, OR, XOR, NOT, 1, 0
Reed-Muller	AND, OR, XOR, 1

III. METHODS

The synthesis methods were implemented in C++. The kitty [20] and STACCATO [21] libraries were used for Minato-Morreale and DSD-decomposition methods, respectively. The comparison of circuits synthesized by the methods was carried out at logic level only without technology mapping.

To compare the methods with each other, a set of test cases was written in which a certain number of truth tables of Boolean functions over four to ten arguments were generated with a probability that was calculated based on the information about the frequency of occurrence of NPN-equivalence classes of Boolean functions of four variables. The circuits from the OpenABC-D [22] test set were used for obtaining this statistic. The collection process was organized as follows: all cuts of the size four [1] were iterated at each circuit, then the algorithm identified a NPN-equivalence class of the cut function. As a result, the frequency of occurrence of each class was achieved. To receive a function over more than four arguments a concatenation of the truth tables was used. The truth table over four variables was obtained after generating a function of NPN-equivalence class according to the received probabilities, then four variables swapping, four input flipping and output flipping with a 50 percent probability were made.

The number of generated truth tables of Boolean functions over a particular number of arguments was 1000. These truth tables were supplied as input arguments to the synthesis methods, which constructed circuits consisting of logic gates.

In the first comparison, the arity of the gates in synthesized circuits was limited to two, therefore the Akers algorithm was not participated. Afterwards, using three-input gates was allowed and all the algorithms took part in the second comparison.

The following characteristics were used to compare circuits: the number of function arguments, area, delay, and switching activity. Additionally, the runtime of the methods was also taken into account. The switching activity of a circuit was calculated as the sum of the switching probabilities of all its gates:

$$Z = \sum_{i=0}^{n} \frac{s_i}{t}, \quad (4)$$

where Z is the switching activity of the logic circuit, s_i is the number of cells switching (from 1 to 0 and from 0 to 1), i is the cell index of the logic circuit, t is the number of simulations.

In the experiment, the number of simulations of each logic circuit was 1024.

The implementations of Akers and bi-decomposition methods were tested on truth tables of Boolean functions over four to seven and four to eight arguments, respectively. The reason is the high asymptotic complexity of these methods [23, 14].

IV. Results

The achieved results of the methods comparison are described in Tables II and III. Table II illustrates the information about the runtime of the methods and characteristics of the synthesized circuits composed of two-input gates only, whereas Table III shows the same but for circuits that may include three-input gates. The cells of the tables contain the averages of the results obtained.

The part of Table II, which corresponds to the statistics about switching activity of synthesized circuits, reveals that the leader in optimizing this parameter for functions over four variables is DSD-decomposition method, whereas for functions over five and six arguments is Minato-Morreale method with factoring and for functions over seven to ten arguments is the original Minato-Morreale method. Bi-decomposition method comes fourth in terms of power optimization for functions over four and five arguments, comes third for functions over six arguments, for the other functions this method reduces its effectiveness and exhibits the worst results. The method of cascades is the second from the end for optimization of functions over four to eight arguments and the worst one for others. Reed-Muller method takes the last place in optimization of functions over four to six variables but improves its position with an increase in the number of variables and becomes the third by functions over eight arguments.

According to Table II, Minato-Morreale method optimizes the delay of circuits containing from five to ten inputs better than the other methods. This method is followed by Reed-Muller algorithm and then Minato-Morreale algorithm with factoring. DSD-decomposition method is the best for optimizing functions over four arguments but comes fourth in terms of delay optimization for the other number of arguments. Bi-decomposition and cascades methods are the least preferred for optimizing circuit delay.

The part of Table II *Area* illustrates that the leader for reducing the number of logic gates for circuits from five to nine inputs is Minato-Morreale method with factoring. DSD-decomposition is the first in terms of this optimization for functions over four and ten arguments, the method only the second and the third for functions over eight and nine arguments and over five and six arguments respectively. Minato-Morreale method without factoring comes second for functions over five and six arguments. Bi-decomposition and Reed-Muller methods are the worst in terms of optimizing circuit area.

The fastest algorithm is the method of cascades, while Minato-Morreale methods require a little more time for execution, these algorithms have a comparable runtime. DSD-decomposition and Reed-Muller methods are moderately slower than the leader. The most time consuming method is bi-decomposition.

Table III demonstrates the similar ranking to Table II in terms of switching activity optimization with the exception of Minato-Morreale method without factoring, which works more efficiently using three-input logic gates than a similar one with factoring. Akers method becomes consistently second, shifting the previous rating.

The *Delay* of Table III reveals that Akers method is not preferred for delay optimization, despite the fact that it becomes second for functions over five and six arguments. Minato-Morreale method without factoring is the absolute leader for optimization of circuit delay with allowing using tree-input gates.

A slightly different result in contrast to Table II is achieved in *Area* optimization. Akers algorithm comes first for functions over five to seven arguments and the original Minato-Morreale becomes second for functions over eight to ten arguments. Otherwise, the results are similar to those obtained in Table II.

The runtime of the algorithms in Table III does not reveal any new data except that Akers algorithm is the longest-running method.

V. Conclusion

The results of the work of seven methods for the synthesis of logic circuits implementing specified Boolean functions have been analyzed. The comparison of the obtained circuits has been carried out according to the three main criteria: the number of logic gates (area), depth (delay) and switching activity (power). The analysis of the results has shown that DSD-decomposition (only for functions over four arguments) and the both Minato-Morreale methods are the best choice for power optimization. The similar results have been demonstrated for delay optimization of logic circuits. Area optimization with a restriction on the arity of logic gates equal to two is better to carry out using DSD-decomposition (only for functions over four and ten arguments) and Minato-Morreale method with factoring. Akers algorithm is the leader in optimization of a number of three-input gates in circuits from five to seven inputs and for other circuits, the leader is Minato-Morreale algorithm without factoring. Minato-Morreale and cascades methods have demonstrated a compatible and minimal runtime, whereas Akers and bi-decomposition have shown the worst performance.

TABLE II. EXPERIMENT RESULTS OF CIRCUITS CONTAINING TWO-INPUT GATES ONLY

Arguments number	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Switching activity							
Bi-decomposition	4.54	8.71	17.66	35.36	70.31		
Method of cascades	5.11	9.78	18.57	34.06	61.29	109.68	197.85
Minato-Morreale	4.35	7.52	12.88	**20.89**	**33.11**	**49.41**	**72.61**
Minato-Morreale with factoring	4	**7.16**	**12.73**	22.63	41.74	75.22	134.7
DSD-decomposition	**3.69**	9.18	18.25	33.57	59.8	106.14	189.89
Reed-Muller	6.62	12.13	20.69	33.99	54.71	86.72	136.4

Delay							
Bi-decomposition	3.7	7.61	15.47	31.45	65.9		
Method of cascades	4.89	7.7	9.96	12	14	16	18
Minato-Morreale	3.11	**4.87**	**6.51**	**7.76**	**8.89**	**10.16**	**11.92**
Minato-Morreale with factoring	3.11	5.64	7.97	10.4	12.84	15.27	17.73
DSD-decomposition	**2.82**	6.84	9.71	11.93	14	16	18
Reed-Muller	4.38	6.11	7.77	9.03	10.06	11.26	12.96
Area							
Bi-decomposition	5.92	16.5	42.81	102.69	237.23		
Method of cascades	7.26	17.79	38.44	75.42	140.7	256.73	468.3
Minato-Morreale	5.97	15.52	35.91	75.78	155.87	301.82	574.24
Minato-Morreale with factoring	4.23	**11.26**	**25.37**	**53.92**	**114.1**	**231.55**	463.79
DSD-decomposition	**3.43**	15.88	36.77	77.45	134.61	244.54	**444.29**
Reed-Muller	12.8	30.26	64.77	134.16	272.82	548.99	1096.01
Runtime (ms)							
Bi-decomposition	4.07	7.41	16.39	44.22	167.53		
Method of cascades	2.65	**2.91**	**3.56**	**5.22**	**9.44**	**21.75**	**62.09**
Minato-Morreale	2.69	3.02	3.73	5.52	9.85	22.71	63.07
Minato-Morreale with factoring	2.65	3.01	3.68	5.47	9.94	22.85	63.55
DSD-decomposition	7.2	8.06	9.04	11.26	16.39	30.69	74.56
Reed-Muller	**2.64**	3.05	4.16	7.42	16.54	44.13	127.76

TABLE III. EXPERIMENT RESULTS OF CIRCUITS CONTAINING TWO-INPUT AND THREE-INPUT GATES

Arguments number	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Switching activity							
Akers	3.81	6.2	10.41	21.07			
Bi-decomposition	4.42	8.32	15.95	28.87	58.87		
Method of cascades	4.81	9.37	18.2	33.65	60.85	109.1	197.04
Minato-Morreale	3.57	**5.23**	**7.83**	**11.98**	**19.3**	**29.96**	**45.14**
Minato-Morreale with factoring	3.81	6.64	11.65	20.04	37.53	68.05	123.81
DSD-decomposition	**3.46**	9.17	18.25	33.57	59.8	106.14	189.89
Reed-Muller	5.43	9.39	15.56	25.5	40.87	64.69	101.25
Delay							
Akers	3.32	5.06	7.57	13.61			
Bi-decomposition	3.6	7.44	15.36	31.35	65.79		
Method of cascades	4.58	7.53	9.93	12	14	16	18
Minato-Morreale	**2.29**	**3.7**	**4.65**	**5.22**	**6.32**	**7.49**	**8**
Minato-Morreale with factoring	2.89	5.34	7.72	10.1	12.54	14.98	17.44
DSD-decomposition	2.56	6.83	9.71	11.93	14	16	18
Reed-Muller	3.26	4.57	5.57	6.28	7	8.18	9
Area							
Akers	4.67	**9.18**	**17.29**	**38.93**			
Bi-decomposition	5.66	15.55	38.62	90.02	213.62		
Method of cascades	6.57	16.77	37.56	74.41	139.63	255.25	466.14
Minato-Morreale	4.09	9.74	22.39	48.57	**102.94**	**206.14**	**409.41**
Minato-Morreale with factoring	3.78	10.02	22.8	48.74	104.58	215.87	441.36
DSD-decomposition	**2.92**	15.86	36.77	72.45	134.61	244.54	444.29
Reed-Muller	9.38	21.7	46.96	98.86	201.3	405.41	808.42
Runtime (ms)							
Akers	3.35	6.25	14.75	82.48			
Bi-decomposition	4	7.28	16.14	43.62	165.79		
Method of cascades	2.57	**2.83**	**3.46**	**5.06**	**9.2**	**21.1**	60.8
Minato-Morreale	2.6	2.9	3.55	5.2	9.28	21.49	**60.59**
Minato-Morreale with factoring	2.58	2.94	3.6	5.29	9.62	22.19	62.21
DSD-decomposition	7.3	7.8	8.7	10.75	15.63	29.31	71.96
Reed-Muller	**2.55**	2.92	3.96	7.05	15.92	43.48	127.38

REFERENCES

[1] H. Riener, A. Mishchenko, and M. Soeken, “Exact DAG-aware rewriting,” in Proc. Design, Autom. Test Europe Conf. Exhibit., 2020, pp. 732–737.

[2] G. Ammes, W. Lau, and R. P. Ribas. (Aug. 2018). Comparative analysis of different Boolean function synthesis Methods. Presented at Microelectronics Students Forum 2018, Rio Grande do Sul, Brazil [Online]. Available: https://sbmicro.org.br/eventos/sforum/volume-18

[3] E. J. McCluskey, “Minimization of Boolean functions,” *Bell Syst. Tech. J.*, vol. 35, no. 6, pp. 1417–1444, Nov. 1956.

[4] M. Choudhury and K. Mohanram, “Bi-decomposition of large Boolean functions using blocking edge graphs,” *in Proc. of the International Conference on Computer-Aided Design.*, 2010, pp. 586–591.

[5] D. Wu and J. Zhu, “FBDD: A folded logic synthesis system,” *in International Conference on ASIC*, Shanghai, China, Oct. 2005, pp. 746–751.

[6] E. M. S. K. J. Singh, L. L. C. M. R. Murgai, and R. K. B. A. SangiovanniVincentelli, “Sis: A system for sequential circuit synthesis,” *University of California, Berkeley*, vol. 94720, p. 4, 1992.

[7] ABC. [Online]. Available: https://github.com/berkeley-abc/abc

[8] S. Yang, “Logic Synthesis and Optimization Benchmarks User Guide”, *Tech. Rep. 1991-IWLS-UG-Saeyang*, Jan. 1991.

[9] F. Brglez, D. Bryan, K. Kozminski, “Combinational Profiles of Sequential Benchmark Circuits*,” in Proc. of the International Symposium of Circuits and Systems*, 1989, pp. 1929–1934.

[10] C. Albrecht, "IWLS 2005 Benchmarks,” *in International Workshop on Logic Synthesis*, June 2005.

[11] T. Sasao and J. T. Butler, “Worst and best irredundant sum-ofproducts expressions”, *IEEE Trans. Comp*, vol. 50, no. 9, pp. 935–948, Sept. 2001.

[12] S. Minato: “Fast generation of prime-irredundant covers from binary decision diagrams,” *IEICE Trans. Fundamentals*, vol. E76-A, no. 6, pp. 967-973, June 1993.**S.**

[13] B. Akers, “Synthesis of combinational logic using three-input majority gates,” *in Proc. 3rd Annu. Symp. Switch. Circuit Theory Logical Design*, 1962, pp. 149–157.

[14] Y. V. Pottosin, “Synthesis of combinational circuits by means of bi-decomposition of Boolean functions,” *Informatics*, vol. 19, no. 1, pp. 7–18, 2022.

[15] G. N. Povarov, “A method for synthesis of computing and controlling contact circuits,” (in Russian), *Automatika i Telemekhanika*, vol. 18, no. 2, pp. 145–162, 1957.

[16] B. Harking, “Efficient algorithm for canonical ReedMuller expansions of Boolean functions”, *IEE Proc., Part E*, vol. 137, no. 5, pp. 366–370, 1990.

[17] V. Bertacco, “Scalable Hardware Verification with Symbolic Simulation,” Springer, 2005.

[18] C. E. Shannon, “The synthesis of two-terminal switching circuits,” *The Bell System Technical Journal*, vol. 28, no. 1, pp. 59–98, 1949.

[19] I. I. Zhegalkin, “On the technique of calculating propositions in symbolic logic”, (in Russian and French), *Mat. Sb*, vol. 34, no. 1, pp. 9-28, 1927.

[20] kitty. [Online]. Available: https://github.com/kovidgoyal/kitty

[21] STACCATO. [Online]. Available: https://web.eecs.umich.edu/staccato/

[22] OpenABC-D. [Online]. https://github.com/NYU-MLDA/OpenABC

[23] S.-Y. Lee, H. Riener, and G. De Micheli, “Logic resynthesis of majoritybased circuits by top-down decomposition,” *in Proc. 24th Int. Symp. Design Diagnost. Electron. Circuits Syst.*, 2021, pp. 105–110.