2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Optimization of Combinational Logic Circuits Through Decomposition of Truth Table and Evolution of Sub-Circuits

Francisco Manfrini
Federal University of Juiz de Fora
Federal Institute of Science and Technology
of the Southeast of Minas Gerais
Juiz de Fora, MG, Brazil
francisco.manfrini@ifsudestemg.edu.br

Helio J.C. Barbosa
National Lab. for Scientific Computing
Petrópolis, RJ, Brazil.
Federal University of Juiz de Fora
Juiz de Fora, MG, Brazil
hcbm@lncc.br

Heder S. Bernardino
Federal University of Juiz de Fora
Juiz de Fora, MG, Brazil
heder@ice.ufjf.br

***Abstract*—In this work, a genetic algorithm was used to design combinational logic circuits (CLCs), with the goal of minimizing the number of logic elements in the circuit. A new coding for circuits is proposed using a multiplexer (MUX) at the output of the circuit. This MUX divides the truth table into two distinct parts, with the evolution occurring in three sub-circuits connected to the control input and the two data inputs of the MUX. The methodology presented was tested with some benchmark circuits. The results were compared with those obtained using traditional design methods, as well as the results found in other articles, which used different heuristics to design CLCs.**

I. Introduction

Designing a combinational logic circuit (CLC) requires considerable knowledge and human creativity [1], and can be considered as a discrete optimization problem [2]. The design of a CLC is based on the data from a truth table that lists all possible combinations of input logic levels with the corresponding output logic level. Given a certain truth table, it is possible to identify the CLC that meets the conditions provided by the truth table using traditional techniques and metaheuristics [3]. Because the global optimum (the circuit with the lowest costs) is unknown for most logic functions, using metaheuristics is appropriate for designing CLCs [4]. Alba [2] points out that in the design of CLCs, the size of the search space grows very rapidly as the number of input variables to the circuit increases, and because all the constraints imposed by the truth table must be satisfied, the problem can be considered to have a large number of hard equality constraints.

To design a CLC and obtain a reduced Boolean function, the traditional techniques that are most used are the Karnaugh map [5] and the Quine-McCluskey algorithm [6], [3], which provide a CLC as a sum of products or product of sums. After applying these methods, it is possible to use Boolean algebra theorems to simplify the result. However, it is not obvious which theorem must be applied to produce the simplest result, since, depending on the sequence in which the theorems and postulates are applied, more than one simplified form can be obtained [2]. Therefore, there is no way to establish whether the expression is in its simplest form, and as a result, this simplification process becomes problematic. In addition, the number of input variables increases [3], the algebraic manipulation comes to depend entirely on the designers intuition due to the fact that the final solutions are rarely unique [7].

The Karnaugh map is a graphical method for finding a CLC corresponding to a truth table in a simple manner, but the usefulness of the Karnaugh map is limited to a maximum of six input variables [8]. Furthermore, different solutions can be found. The Quine-McCluskey method is theoretically useful for any number of variables, but as it is an exhaustive method, it is used in practice for systems of only a few variables. Sasao [9] implemented CLCs using only AND and XOR gates. Another traditional way of designing CLCs is building a network of MUXs through Shannon decomposition [10].

In the early 1990s, a new field of research arose called Evolutionary Hardware. This represented a paradigm shift in electronic circuit design [11] because it then became possible to design a CLC autonomously. The evolutionary design explores a much richer set of possibilities, and it is not limited by the conventional knowledge of the designer [12]. Many studies have been conducted on the design of CLCs using genetic algorithms [7], [2], [13], genetic programming [14], [15] particle swarm optimization (PSO) [3], [16] and ant colony optimization (ACO) [17], [11].

The complexity of a CLC is related to the number of circuit components, while a general measure of circuit optimization is the total number of logic gates, regardless of their type [2]. The main objective of this paper is to propose and test a new coding method for the design of CLCs that minimises the number of logic elements used. Experiments were conducted on four benchmark problems, and their output quality was compared with traditional techniques and results found in the literature that used other heuristics.

The remainder of this paper is organized as follows: Section II shows some heuristics used for the same benchmark problems addressed here; Section III provides the details of the encoding that was used and the algorithm that was

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

implemented; Section IV shows the results and compares them with the best results in the literature; and finally, in Section V the conclusions and possible future work are presented.

II. Related Work

Several bio-inspired algorithms have been proposed for the design of CLCs [7]. Coello [1] implemented a genetic algorithm GA with a two-dimensional matrix encoding to represent the circuit. Each element of the matrix represents a logical element (AND, NOT, OR, XOR and WIRE) with its entries, as a result of which it is possible to represent any CLC [1]. An important point of the presented encoding was the use of an alphabet of cardinality n, where n refers to the number of rows in the matrix, resulting in this GA being named NGA. The cardinality n allowed the manipulation of shorter chromosomal chains decreasing the complexity of the coding. The GA that was implemented works in two phases; in the beginning of the research, only the validity of the outputs is considered, and the GA explores the search space. Once a solution appears, the suitability is modified such that the valid designs compensate for each included wire so that the functionality of the CLC remains. In [18] Coello proposed a multi-objective genetic algorithm (MGA) in which the outputs of the circuit were considered as equality constraints to be satisfied. In [3] a hybrid evolutionary algorithm called QEPSO was proposed, which was based on the quantum-inspired evolutionary algorithm (QEA) [19], and the technique of particle swarm optimization (PSO). This implementation was efficient and produced circuits that were equivalent to those obtained by the reference algorithm MGA.

ACO has also been applied to the design of CLC, and some examples are shown in [17] in which the results of ACO outperformed the binary GA. Mostafa [11] commented that in ACO, in addition to the agents cooperating among themselves (ants), characteristics of other heuristics can also be easily incorporated to improve the solution [11]. Moreover, he also proposed a modified ACO that has been compared to benchmark problems.

Coello [2] conducted a comparative study of serial and parallel heuristics used to design CLCs and found that hybridization of a GA with a local search algorithm (simulated annealing) is beneficial and that the parallelization not only increases speed but improves the solutions that are found. The hybrid algorithm that was implemented was called GASA2.

Genetic Programming (GP) was also applied to the design of CLCs [14]. Karakatic [14] designed benchmark circuits and compared them with the best results of the NGA and the MGA and found that, unlike GA, GP avoids premature convergence.

In addition to the logic gates, a CLC can also be designed using 2x1 MUXs that have two data inputs and a control input. Since any Boolean function can be implemented using a MUX, this is considered to be a universal module [20], One form of implementing a Boolean function using only multiplexers is through Shannon decomposition [21]. GP was applied to synthesize underlying logical functions using only MUXs [4]. Traditional design techniques do not use MUXs and logic gates simultaneously, but Miller [12] designed arithmetic CLCs using MUXs and logic gates with GA.

III. Description of circuit's approach and representation

The design of CLCs is highly sensitive to the coding that is used, as well as to the degree of interconnectivity between the gates[2]. The determination of an appropriate geometry for a Boolean problem is not trivial. A large geometry increases the search space unnecessarily, while a small geometry might not be sufficient to solve the problem [7]. A structure of variable size, as is the case with a traditional GP, is interesting, but it can generate swelling, i.e. the uncontrolled growth of the tree. A good solution has been adopted by Coello [7], who proposed a matrix wherein the number of columns and rows can be increased gradually while a feasible circuit is not found.

Various published works addressing the design of CLCs through evolutionary computing [2], [12], [18], [13], ACO [11], [17], PSO [3], [16] and hybrid algorithms [3] use a two-dimensional matrix to represent a CLC.

As already pointed out by Alba [2], the search space grows exponentially with the number of inputs of the circuit which requires the use of heuristics. As a result, the search for more efficient encodings is justified.

This paper proposes a new coding method for CLCs. Instead of using only one matrix, at the output of the matrix, we have connected a MUX that provides the output of the circuit. Thus, the system consists of three sub-circuits and a MUX, as shown in fig. 1. The matrix provides three sub-circuits, identified as circuits 1, 2, and 3, which in turn are connected to the two data inputs and the MUX control input, respectively. The choice of a MUX coupled to the output of the circuit allows, through the control inlet (circuit 3), the truth table to be split into two independent parts: when the output of the control circuit is one, the data input of MUX I_1 is activated, and circuit 1 is selected; when the output of the control circuit is zero, the data input I_0 of the MUX is selected, and circuit 2 is activated. Thus, circuit 1 only has to satisfy some restrictions of the truth table, and circuit 2 will have to attend to the other constraints. The choice of those constraints will be attended to by circuit 1, and circuit 2 is controlled by circuit 3.

In the optimal design of circuits, one would like to minimize the total cost of fabrication. However, due to the practical difficulties in measuring all costs involved, several simplified design metrics have been adopted in the literature. One idea is to minimize the total number of transistors employed [22]. Perhaps more commonly, the total number of logical gates is adopted [2]. To replicate the same unit as much as possible, even leading to the use of more gates, is another possibility [23]. Yet another metric is to minimize the number of components making no distinction between a simple gate and a MUX [12], [24], [25]. This last approach will be adopted here. An example of a more detailed representation of the proposed coding is shown in fig. 2, in which a 3 x 3 matrix represents the three sub-circuits, and each matrix element (gene) represents a component (AND, OR, XOR, NOT and

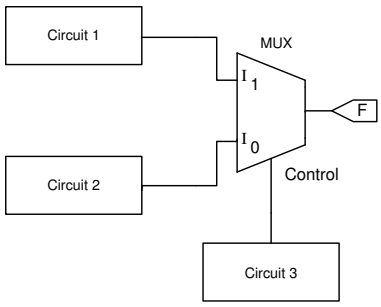

Fig. 1. Representation of circuit composed of three sub-circuits connected to MUX. Output of circuit is called F

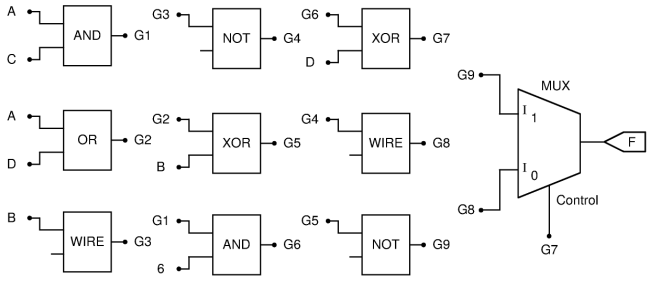

Fig. 2. Detailed representation of circuit: matrix with three subcircuits connected to MUX

WIRE) with their respective entries. The elements in the first column have the Boolean variables of the truth table (A, B, C and D) as inputs. In the second and third columns, the elements can have Boolean variables as inputs and the outputs of the gates of the previous levels (G1, G2, etc.), with one of the inputs of each gate having to be connected to an output of the previous level that was previously not connected. When a particular gene represents a NOT gate (it has only one input) or a WIRE, the input of the gate that is not connected to the previous level will be disregarded, thus guaranteeing that there will be no discontinuity in the circuit and that all of the individuals that are generated will be feasible. The inputs of the MUX (G7, G8, G9) are fixed, and the function of the MUX is to select, through the control, which output of sub-circuit G8 or G9 will be connected at the output of F.

IV. Case Study

Four benchmark examples, taken from [14], [2], [17], [18], [3], were chosen to verify the effectiveness of the adopted encoding. All the chosen examples have four input variables and one output.

The representation of the circuit is done by means of a fixed 3 x 3 matrix connected to a MUX as shown in fig. 2. A steady-state mutation-only GA with a population of 50 individuals evolving for 80,000 generations was implemented in MATLAB[1]. Within about 5 runs, it was observed that the best individuals obtained are competitive with those found in the literature.

[1]http://www.mathworks.com

The computational experiments were performed in a notebook with an AMD A6-3400M processor, 4GB RAM, and Windows 7 Home Premium 64 bits. The elapsed time of each independent run was around 20 minutes. Here, the objective is to obtain the best design (the one with the minimum number of logic elements) and, thus, no analysis is presented with respect to the computational cost.

For each experiment, the truth table is constructed considering the control input of the MUX (output of circuit 3) and its data inputs, I_0 and I_1 (outputs of circuits 1 and 2, respectively). The output of the circuit is selected by the control of the MUX, and corresponds to outputs of the sub-circuits 1 or 2. The unused sub-circuits are considered as "don't care" (-). For all experiments, the truth table, an illustration of the best obtained circuit by the proposed algorithm, and a table comparing the solution found with those from the literature are presented. Each truth table contains: the minterm (the decimal representation of the binary string of the inputs variables), the inputs (A, B, C, and D), the control of the MUX, the data inputs of the MUX (I_0 and I_1), and the output of the circuit. Notice that the truth tables include not only the input and output variables, but also information regarding the control and data inputs of the MUX.

A. Example I

The first example was used in [14], [2], [17], [18]. The truth table is presented in Table I and the best result obtained by the proposed algorithm is presented in fig. 3. One can see in Table II that the best circuit found by the proposed technique is composed by 5 logic gates and 1 MUX, totalizing 6 logic elements. Also, the same table shows that the best circuit from the literature is composed by 7 logic elements.

TABLE I
Truth table for the circuit of the example 1

Minterm	A	B	C	D	Control	I_0	I_1	F
0	0	0	0	0	0	1	-	1
1	0	0	0	1	0	1	-	1
2	0	0	1	0	1	-	0	0
3	0	0	1	1	1	-	1	1
4	0	1	0	0	0	0	-	0
5	0	1	0	1	0	0	-	0
6	0	1	1	0	1	-	1	1
7	0	1	1	1	1	-	1	1
8	1	0	0	0	0	1	-	1
9	1	0	0	1	0	0	-	0
10	1	0	1	0	1	-	1	1
11	1	0	1	1	1	-	0	0
12	1	1	0	0	0	0	-	0
13	1	1	0	1	0	1	-	1
14	1	1	1	0	1	-	0	0
15	1	1	1	1	1	-	0	0

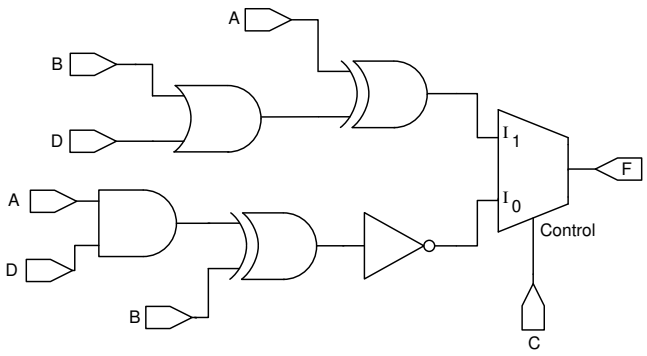

Fig. 3. Best circuit found by the proposed algorithm for example 1

TABLE II
COMPARISON OF BEST RESULTS OF EXAMPLE 1 FOUND IN THE LITERATURE USING DIFFERENT HEURISTICS

Algorithm	Size
Proposed Algorithm (AGMUX)	5 gates , 1MUX
Genetic Programming [14]	7 gates
Hybrid (GASA2) [2] Genetic Algorithm and Simulated Annealing	7 gates
Ant System [17]	7 gates
Genetic Algorithm (MGA) [18]	8 gates
Genetic Algorithm (NGA) [18]	10 gates
Human Designer 1 [18]	11 gates
Human Designer 2 [18]	12 gates
SASAO [9] Technique based ANDs, XOR	12 gates

B. Example 2

[17], [18] used this second example in their computational experiments. The truth table is presented in Table III and the best result obtained by the proposed algorithm is presented in fig. 4. It is shown in Table IV that the best circuit found by the proposed technique is composed by 4 logic gates and 1 MUX, totalizing 5 logic elements. Notice that the best result from the literature is composed by 7 logic elements.

TABLE III
TRUTH TABLE FOR THE CIRCUIT OF THE EXAMPLE 2

Minterm	A	B	C	D	Control	I_0	I_1	F
0	0	0	0	0	0	1	-	1
1	0	0	0	1	1	-	0	0
2	0	0	1	0	0	0	-	0
3	0	0	1	1	1	-	0	0
4	0	1	0	0	0	1	-	1
5	0	1	0	1	1	-	1	1
6	0	1	1	0	0	1	-	1
7	0	1	1	1	1	-	1	1
8	1	0	0	0	1	-	1	1
9	1	0	0	1	0	1	-	1
10	1	0	1	0	1	-	1	1
11	1	0	1	1	0	0	-	0
12	1	1	0	0	1	-	0	0
13	1	1	0	1	0	1	-	1
14	1	1	1	0	1	-	0	0
15	1	1	1	1	0	1	-	1

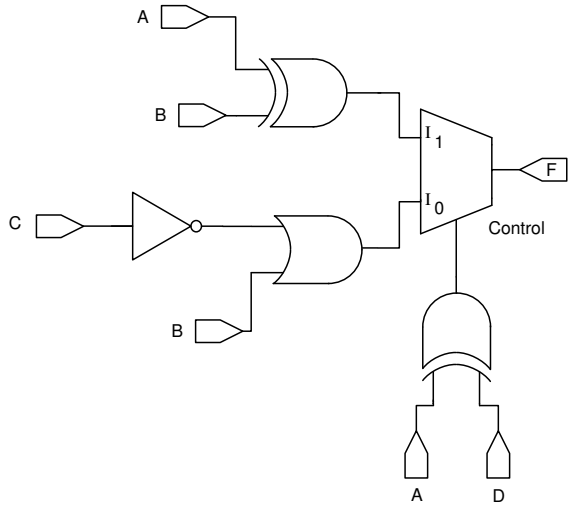

Fig. 4. Best circuit found by the proposed algorithm for example 2

TABLE IV
COMPARISON OF BEST RESULTS OF EXAMPLE 2 FOUND IN THE LITERATURE USING DIFFERENT HEURISTICS

Algorithm	Size
Proposed Algorithm (AGMUX)	4 gates , 1MUX
Ant System [17]	7 gates
Genetic Algorithm (MGA) [18]	7 gates
Genetic Algorithm (NGA) [18]	7 gates
Human Designer 1 [18]	9 gates
Human Designer 2 [18]	10 gates

C. Example 3

The third example was used in [3]. The truth table is displayed in Table V, and the best result obtained by the proposed algorithm is presented in fig. 5. One can see in Table VI that the best circuit found by the proposed technique is composed by 3 logic gates and 1 MUX; that is, 4 logic elements. Also, the same table shows that the best circuit from the literature is composed by 5 logic elements.

TABLE V
TRUTH TABLE FOR THE CIRCUIT OF THE EXAMPLE 3

Minterm	A	B	C	D	Control	I_0	I_1	F
0	0	0	0	0	0	0	-	0
1	0	0	0	1	0	0	-	0
2	0	0	1	0	0	1	-	1
3	0	0	1	1	0	1	-	1
4	0	1	0	0	1	-	0	0
5	0	1	0	1	1	-	0	0
6	0	1	1	0	1	-	0	0
7	0	1	1	1	1	-	0	0
8	1	0	0	0	0	1	-	1
9	1	0	0	1	1	-	1	1
10	1	0	1	0	0	0	-	0
11	1	0	1	1	1	-	1	1
12	1	1	0	0	1	-	1	1
13	1	1	0	1	0	1	-	1
14	1	1	1	0	1	-	1	1
15	1	1	1	1	0	0	-	0

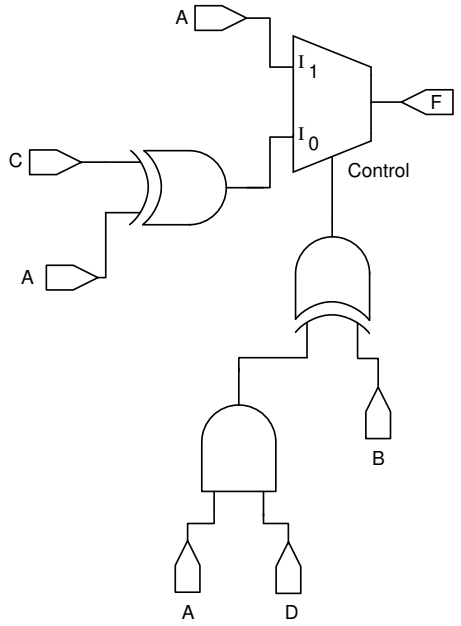

Fig. 5. Best circuit found by the proposed algorithm for example 3

TABLE VI
COMPARISON OF BEST RESULTS OF EXAMPLE 3 FOUND IN THE LITERATURE USING DIFFERENT HEURISTICS

Algorithm	Size
Proposed Algorithm (AGMUX)	3 gates , 1MUX
Hybrid (QEPSO) [3] Quantum Evolution and Particle Swarm	5 gates
Particle Swarm(PSO) [3]	5 gates
Genetic Algorithm (MGA) [3]	5 gates

D. Example 4

This fourth example was also solved in [3]. The truth table is presented in Table VII, and the best result obtained by the proposed algorithm is illustrated in fig. 6. It is shown in Table VIII that the best circuit found by the proposed technique is composed by 4 logic gates and 1 MUX, totalizing 5 logic elements. Notice that the best result from the literature is composed by 6 logic elements.

TABLE VII
TRUTH TABLE FOR THE CIRCUIT OF THE EXAMPLE 4

Minterm	A	B	C	D	Control	I_0	I_1	F
0	0	0	0	0	0	1	-	1
1	0	0	0	1	0	1	-	1
2	0	0	1	0	1	-	0	0
3	0	0	1	1	1	-	0	0
4	0	1	0	0	0	1	-	1
5	0	1	0	1	0	1	-	1
6	0	1	1	0	1	-	1	1
7	0	1	1	1	1	-	1	1
8	1	0	0	0	0	0	-	0
9	1	0	0	1	0	0	-	0
10	1	0	1	0	1	-	1	1
11	1	0	1	1	1	-	0	0
12	1	1	0	0	0	0	-	0
13	1	1	0	1	0	0	-	0
14	1	1	1	0	1	-	0	0
15	1	1	1	1	1	-	1	1

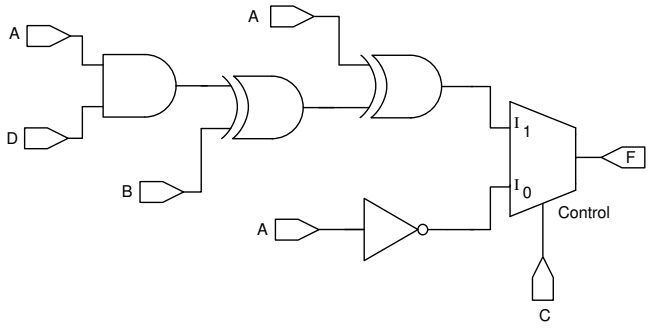

Fig. 6. Best circuit found by the proposed algorithm for example 4

TABLE VIII
COMPARISON OF BEST RESULTS OF EXAMPLE 4 FOUND IN THE LITERATURE USING DIFFERENT HEURISTICS

Algorithm	Size
Proposed Algorithm (AGMUX)	4 gates , 1MUX
Hybrid (QEPSO) [3] Quantum Evolution and Particle Swarm	6 gates
Particle Swarm(PSO) [3]	6 gates
Human Designer [3]	12 gates

V. CONCLUSIONS AND FUTURE WORK

This paper presented a new encoding for automatic design of combinational logic circuits (CLCs). The inclusion of a multiplexer allowed the expansion of the search space in a targeted manner because two individuals with a low fitness due to not meeting all the restrictions imposed by the truth table can complement each other according to the MUX control

and have a good fitness. Therefore, this encoding allows for a relaxation in the restrictions imposed by the truth table. The results obtained using the proposed encoding are better than those from the literature, when the number of logic elements is to be minimized.

An approach to improving these results is to use the proposed encoding within other optimization techniques that have provided good results in the design of CLCs, such as genetic programming and ant colony optimization algorithms.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their comments and suggestions that helped to improve the final version of the paper. The authors acknowledge the support from CNPq (grant 308317/2009-2).

REFERENCES

[1] Carlos A Coello Coello, Alan D Christiansen, and Arturo Hernández Aguirre. Use of evolutionary techniques to automate the design of combinational circuits. *International Journal of Smart Engineering System Design*, 2:299–314, 2000.

[2] Enrique Alba, Gabriel Luque, Carlos A Coello Coello, and Erika Hernández Luna. Comparative study of serial and parallel heuristics used to design combinational logic circuits. *Optimisation Methods and Software*, 22(3):485–509, 2007.

[3] Phillip W Moore and Ganesh K Venayagamoorthy. Evolving digital circuits using hybrid particle swarm optimization and differential evolution. *International Journal of neural systems*, 16(03):163–177, 2006.

[4] Arturo Hernández Aguirre and Carlos A Coello Coello. Using genetic programing and multiplexers for the synthesis of logic circuits. *Engineering Optimization*, 36(4):491–511, 2004.

[5] Maurice Karnaugh. The map method for synthesis of combinational logic circuits. *American Institute of Electrical Engineers, Part I: Communication and Electronics, Transactions of the*, 72(5):593–599, 1953.

[6] Edward J McCluskey. Minimization of boolean functions. *Bell Systems Techinical Journal*, 35(5):1417–1444, 1956.

[7] Enrique Alba, Gabriel Luque, Carlos A Coello Coello, and Erika Hernández Luna. Comparative study of serial and parallel heuristics used to design combinational logic circuits. *Optimisation Methods and Software*, 22(3):485–509, 2007.

[8] Ronald J Tocci. *Digital Systems: principles and applications*. Pearson Education India, 2011.

[9] Tsumomu Sasao. *Logic synthesis and optimization*, volume 212. Springer, 1993.

[10] Claude Shannon et al. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949.

[11] Mostafa Abd-El-Barr, Sadiq M Sait, Bambang AB Sarif, and Uthman Al-Saiari. A modified ant colony algorithm for evolutionary design of digital circuits. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 708–715. IEEE, 2003.

[12] Julian F Miller, Dominic Job, and Vesselin K Vassilev. Principles in the evolutionary design of digital circuitspart i. *Genetic programming and evolvable machines*, 1(1-2):7–35, 2000.

[13] Tatiana Kalganova and Julian Miller. Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness. In *Evolvable Hardware, 1999. Proceedings of the First NASA/DoD Workshop on*, pages 54–63. IEEE, 1999.

[14] S Karakatic, V Podgorelec, and M Hericko. Optimization of combinational logic circuits with genetic programming. *Electronics & Electrical Engineering*, 19(7), 2013.

[15] John R Koza. *Genetic Programming: vol. 1, On the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[16] Carlos A Coello Coello, Erika Hernández Luna, and Arturo Hernández Aguirre. Use of particle swarm optimization to design combinational logic circuits. In *Evolvable Systems: From Biology to Hardware*, pages 398–409. Springer, 2003.

[17] Benito Mendoza García and Carlos A Coello Coello. An approach based on the use of the ant system to design combinational logic circuits. *Mathware & soft computing*, 9(3):235–250, 2002.

[18] Carlos A Coello Coello, Alan D Christiansen, and Arturo Hernández Aguirre. Use of evolutionary techniques to automate the design of combinational circuits. *International Journal of Smart Engineering System Design*, 2:299–314, 2000.

[19] Kuk-Hyun Han and Jong-Hwan Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *Evolutionary Computation, IEEE Transactions on*, 6(6):580–593, 2002.

[20] Milos D Ercegovac, Jaime H Moreno, and Tomas Lang. *Introduction to digital systems*. John Wiley & Sons, Inc., 1998.

[21] Glen G Langdon Jr. A decomposition chart technique to aid in realizations with multiplexers. *Computers, IEEE Transactions on*, 100(2):157–159, 1978.

[22] A Słowik and M Białko. Evolutionary design and optimization of combinational digital circuits with respect to transistor count. *TECHNICAL SCIENCES*, 54(4), 2006.

[23] Arturo Hernández Aguirre, Carlos A Coello Coello, and Bill P Buckles. A genetic programming approach to logic function synthesis by means of multiplexers. In *Evolvable Hardware, 1999. Proceedings of the First NASA/DoD Workshop on*, pages 46–53. IEEE, 1999.

[24] Julian F Miller, Dominic Job, and Vesselin K Vassilev. Principles in the evolutionary design of digital circuitspart ii. *Genetic programming and evolvable machines*, 1(3):259–288, 2000.

[25] CK Vijayakumari, Dileep Lukose, P Mythili, and Rekha K James. An improved design of combinational digital circuits with multiplexers using genetic algorithm. In *Emerging Research Areas and 2013 International Conference on Microelectronics, Communications and Renewable Energy (AICERA/ICMiCR), 2013 Annual International Conference on*, pages 1–5. IEEE, 2013.