Applied Soft Computing xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm

Mehdi Anjomshoa [a,*], Ali Mahani [b], Salahedin Sadeghifard [a]

[a] South Pars Gas Complex, Bushehr, Iran
[b] Department of Electrical and Electronic Engineering, Shahid Bahonar University, Kerman, Iran

ARTICLE INFO

Article history:
Received 10 December 2011
Received in revised form 27 January 2014
Accepted 8 April 2014
Available online xxx

Keywords:
Evolutionary hardware
Combinational logic circuit
Imperialist Competitive Algorithm

ABSTRACT

This paper proposes a novel evolutionary approach based on modified Imperialist Competitive Algorithm for combinational logic circuits designing and optimization. The Imperialist Competitive Algorithm operates on real values and is not applicable to logic circuits optimization problems. So a modified version of ICA is proposed to overcome this shortcoming. Modification of the algorithm depends on random cell replacement between Imperialist and its colonies as assimilation policy. Also a multi-objective evaluation mechanism in the form of a weighted cost function is introduced to obtain optimized circuits in case of circuit area and propagation delay. To evaluate the effectiveness of this method some general benchmark circuits are used in which the circuits with fewer logic cells (minimized space) and lower propagation delay are obtained. The simulation results of our proposed method are compared with some conventional and heuristic methods. Simulation results show that our proposed method significantly improves the performance factor which represents both circuit area and propagation delay.

1. Introduction

Combinational logic circuits are hardware networks that their outputs are only determined by the logical function of their current input state, logic "0" or logic "1" at any given instant in time. Combinational logic circuits are made up from basic logic gates that are connected together to produce more complicated switching circuits. These logic gates are the building blocks of combinational logic circuits. Truth table provides a concise list that shows the output values in tabular form for each possible combination of input variables.

Combination logic circuits have been used in various functional units such as arithmetic and logic functions, data transmission and code converters. In recent years, most of the logic circuits have been implemented on silicon chips as integrated circuits, so it is important to have compact circuits on them. On the other hand, many circuits implemented on digital chips, especially central processing unit (CPU) of computers consisting full adder, multiplier, comparator, multiplexer, etc. are combinational logic circuits. Since these processors should have a high operating speed, the logic circuits forming them should be as fast as possible with lower propagation delay.

In this paper we try to find efficient and creative circuits that match a given truth table with less complexity. The obtained circuits are expected to be optimized both in terms of space and propagation delay. By reducing the number of logic gates and using gates with lower number of transistors, a compact circuit is obtained. Also to have efficient circuits, we should search for those ones which produce the output in a short response time. The response time of a circuit depends on the number and the complexity of the gates forming the longest path.

Designing logic circuit requires knowledge of a large collection of domain-specific rules. therefore, the process of implementing a logic circuit involves transforming the original specification into a form, suitable for the target technology, optimizing the representation with respect to a number of user defined objectives and constraints (i.e. timing, fan-in/out, power, etc.), and finally carrying out technology mapping onto the target technology [1].

Among conventional methods, many design approaches, such as Boolean algebra, Karnaugh map [2], and Quine-McCluskey [3] have been widely used in solving digital circuit optimization problem. The algebraic method applies some known algebraic theorems to minimize logic functions. Designer ability has an important role in this method. The Karnaugh map [2] is a matrix based representation of logical functions and allows minimization of up to 5-input functions.

McCluskey procedure [3] is a tabular method and it is able to minimize functions of any number of inputs. Logic functions are

* Corresponding author. Tel.: +98 7727688521.
E-mail address: Anjomshoa_m@yahoo.com (M. Anjomshoa).

http://dx.doi.org/10.1016/j.asoc.2014.04.011
1568-4946/

minimized as sum of product in Both Karnaugh map and McCluskey procedure. These two methods offer circuits with the shortest response time, but not at all the smallest size. McCluskey procedure requires an execution time that grows exponentially with the number of input signals. Furthermore, Karnaugh map and McCluskey procedure use only AND, OR and NOT gates and ignores all the rest of the gates.

Evolutionary hardware (EHW) is an automated design method of circuits based on artificial evolution of natural phenomena. EHW uses evolutionary algorithms (EA) to auto configure and optimize circuits [4,5]. By exploring a large search space, EHW may find solutions for a task which is unsolvable or difficult to solve.

The process of evolutionary circuit design is fundamentally different from the traditional design process, because it is not based on designer knowledge and experience, but on the evolutionary process. The evolutionary circuit design has fewer constraints than the design based on the designer knowledge and experience. The designers are not only limited by the technology in which the circuit will be produced, but also by their own habits (routines), imagination and creative thinking.

Many different evolutionary methods for the design of digital circuits have been created until now. Coello [6–8] introduced NGA (Genetic Algorithm with N-cardinality representation) and MGA (Multi-objective Genetic Algorithm) to evolve logic circuits at the gate level and Slowik [9,10] presented MLCEA method (Multi Layer Chromosome Evolutionary Algorithm) to optimize logic circuits with respect to transistor count. Shuguang Zhao [11] used genetic algorithm, Zhaohui Gan [12] presented improved gene expression-based clonal selection algorithm (IGE-CSA), Srivastava [13] used genetic algorithm and Xia, Xuewen [14] introduced a method based on genetic algorithm with variable-length cellular automata (CA). In [15], Nedjah presented a methodology based on genetic programming to design and optimization logic circuits and Dipayan Bahdra [16] introduced LNN method (logical neural network) which evolved logic circuits by the combination of neural network and genetic algorithm.

Atashpaz-Gargari and Lucas [17] proposed a new algorithm which was inspired from the social and political behavior of human societies and it is Imperialist Competitive Algorithm (ICA). This algorithm has been applied in different fields of science and engineering [19–21] as an optimization tool and comparisons between the results of ICA and other evolutionary algorithms clearly show the ability of this new algorithm, regarding to convergence time and optimized solution.

In [18] we proposed a new method based on Imperialist Competitive Algorithm (ICA) to design and optimize logic circuits. In [18], we just tried to minimize circuits according to the number of logic gates and circuit area.

Here, we are following two goals: first we modified Imperialist Competitive Algorithm to be suitable for designing logic circuit structures and second achieving optimized circuits regarding to the area and propagation delay to have compact and fast digital modules. This paper improves the methodology used in [18].

The rest of this paper is organized as follows. In Section 2, original version of ICA algorithm will be introduced. Section 3 describes how to encode the circuit structure and the usage of ICA as a new approach for the automatic design of an optimized circuit. Section 4 shows experiments and compares them with works which have been proposed before. Finally, Section 5 concludes the paper.

2. Imperialist Competitive Algorithm

Imperialist Competitive Algorithm (ICA) [17] is a new evolutionary algorithm in the Evolutionary Computation field based on the human's socio-political evolution. Like other evolutionary algorithms, it starts with an initial random population which is called countries. Some of the best countries in the population selected to be the imperialists and the rest form the colonies of these imperialists. In an N dimensional optimization problem, a country is a $1 \times n$ array (Eq. (1)).

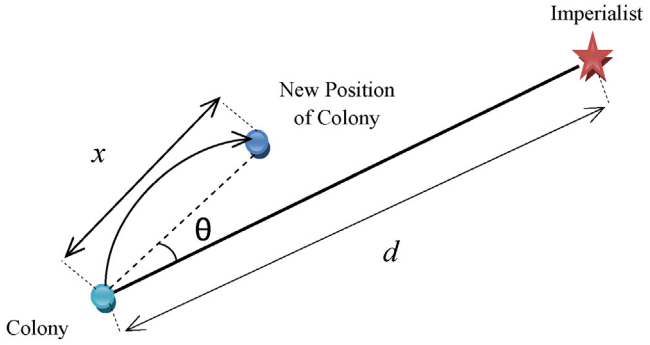

Fig. 1. Moving colonies toward their relevant Imperialist [17].

$$\text{Country} = [p_1, p_2, \ldots, p_n] \tag{1}$$

The cost of a country is found by evaluating the cost function f at the variables $(p_1, p_2, \ldots, p_n)$. Then

$$c_i = f(\text{country}_i) = f(p_1, p_2, \ldots, p_n) \tag{2}$$

The algorithm starts with N initial countries and the N_{imp} best of them (countries with minimum cost) chosen as the imperialists. The remaining countries are colonies that each belong to an empire. The initial colonies belong to the imperialists according to their powers. To distribute the colonies among imperialists proportionally, the normalized cost of an imperialist is defined as Eq. (3).

$$C_n = \max(c_i) - c_n \tag{3}$$

where c_n is the cost of nth imperialist and c_n is its normalized cost. Each imperialist that has more cost value, will have less normalized cost value. Having the normalized cost, the power of each imperialist is calculated from Eq. (4) and based on that the colonies distributed among the imperialist countries.

$$p_n = \left| \frac{c_n}{\sum_{i=1}^{N_{imp}} c_n} \right| \tag{4}$$

On the other hand, the normalized power of an imperialist is assessed by its colonies. Then, the initial number of colonies of an empire will be $NC_n = rand\{p_n \times N_{col}\}$ where NC_n is initial number of colonies of nth empire and N_{col} is the number of all colonies.

To distribute the colonies among imperialist, NC_n of the colonies is selected randomly and assigned to their imperialist. The imperialist countries absorb the colonies toward themselves using the assimilation policy. The assimilation policy shown in Fig. 1 makes the main core of this algorithm and causes the countries move toward to their minimum optima. The imperialists absorb these colonies toward themselves with respect to their power that described in Eq. (4).

The total power of each imperialist is determined by the power of its both parts, the empire power plus percents of its average colonies power.

$$T \cdot C_n = \text{Cost}\ (\text{imperialist}_n) + \xi\ \text{mean}\{\text{Cost}\ (\text{colonies of impire}_n)\} \tag{5}$$

where $T{\cdot}C_n$ is the total cost of the nth empire and ξ is a positive number which is considered to be less than one.

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

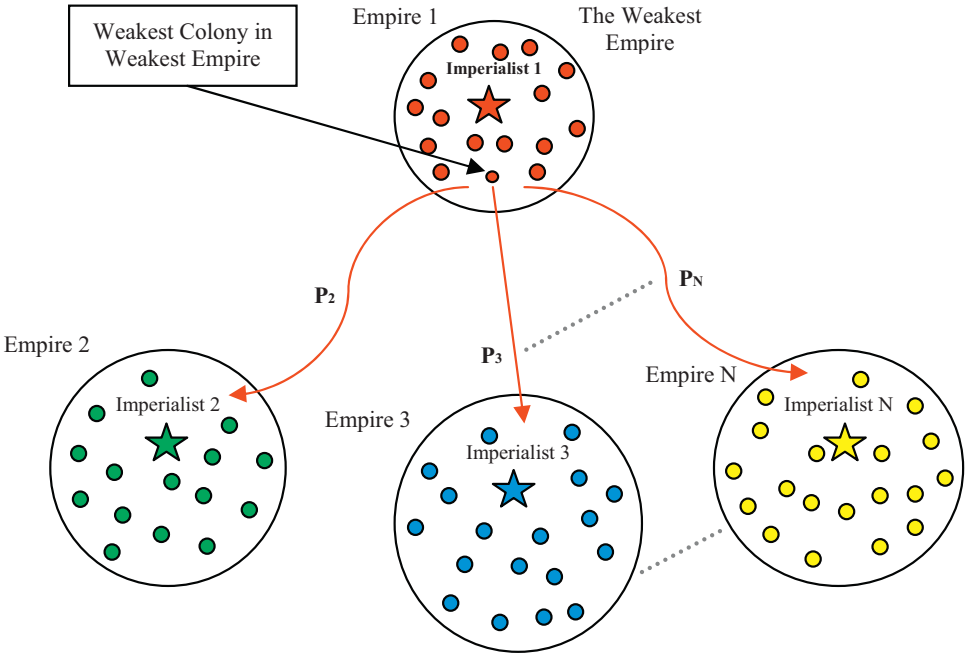

Fig. 2. Imperialistic competition. The more powerful an empire is, the more likely it will possess the weakest colony of weakest empire [17].

In the assimilation policy, the colony moves toward the imperialist by x unit. The direction of movement is the vector from colony to imperialist, as shown in Fig. 1, the distance between the imperialist and colony shown by d and x is a random variable with uniform distribution. Where β is greater than 1 and is near to 2 ($x \sim U(0, \beta \times d)$).

In ICA algorithm, to search different points around the imperialist, a random amount of deviation is added to the direction of colony movement toward the imperialist. In Fig. 1, this deflection angle is shown as θ, which is chosen randomly and with a uniform distribution ($\theta \sim U(-\gamma, \gamma)$).

While moving toward the imperialist countries, a colony may reach to a better position, so the colony's position changes according to the position of the imperialist. In each running cycle, we select some of the weakest colonies and replace them with new ones, randomly. The replacement rate is named as the revolution rate. In ICA, revolution causes a country to suddenly change its socio-political characteristics. That is, instead of being assimilated by an imperialist, the colony randomly changes its position in the socio-political axis. The revolution increases the exploration of the algorithm and prevents the early convergence of countries to local minimums.

In this algorithm, the imperialistic competition has an important role. During the imperialistic competition, the weak empires will lose their power and their colonies (Fig. 2). To model this competition, firstly we calculate the probability of possessing all the colonies by each empire considering the total cost of empire.

$$NTC_n = \max\{TC_i\} - TC_n \tag{6}$$

where TC_n is the total cost of nth empire and NTC_n is the normalized total cost of nth empire. Having the normalized total cost, the possession probability of each empire is calculated from Eq. (7).

$$P_{pn} = \left| \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \tag{7}$$

After a while all the empires, except the most powerful one will collapse and all the colonies will be under the control of this unique empire.

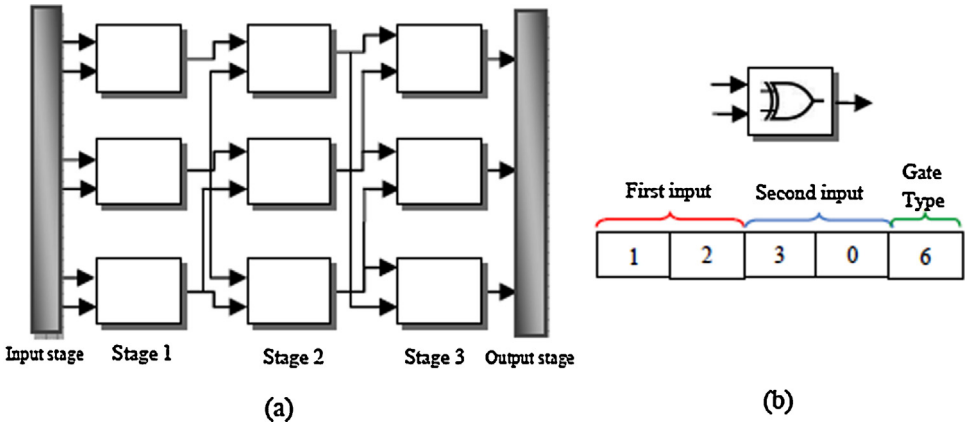

Fig. 3. (a) Two dimensional random matrix as circuit structure. (b) Cell encoding.

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

Table 1
Standard CMOS cell information from vsclib013 library [23].

Gate type code	Gate symbol	Propagation delay(ps)	Area (μm)	Number of transistors	Cell name
0	———	0	0	0	WIRE
1		109	1728	2	NOT
2		158	2880	6	AND
3		167	2880	6	OR
4		133	2304	4	NAND
5		135	2304	4	NOR
6		165	4608	9	XOR
7		174	5184	9	XNOR

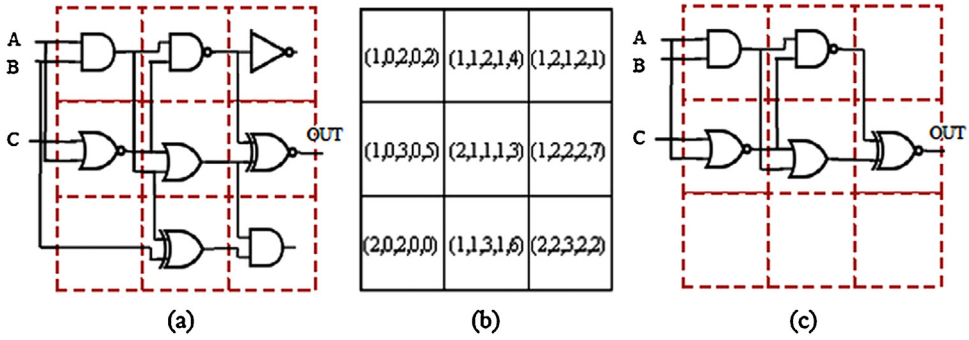

Fig. 4. Example of a circuit and its encoding: (a) circuit schematic with logic gates and interconnections. (b) Encoding of circuit. (c) Effective gates in matrix.

3. Proposed method

3.1. Encoding of circuit structure

Imperialist Competitive Algorithm starts with some random colonies and after evaluation, some of these colonies are selected as imperialist. In this work our logic circuit structures are introduced as colonies. A two dimensional matrix is a common structure that has been used in many research works which was proposed by Louis and Rawlins [22]. That structure is shown in Fig. 3. Each cell of the $m \times n$ matrix contains the information of the gate type and its corresponding inputs. However, unlike the fixed interconnection rules used in [22], the inputs of each unit can be randomly connected by any element output in previous stages and there is no feedback between cell elements. Each cell of the matrix is encoded in an array of five integer numbers. First and second number of the array is the address of the cell which the first input of the gate is connected and third and fourth number of the array is for second input and finally fifth number indicates gate type.

Different logic cells (NOT, AND, OR, NAND, NOR, XOR, XNOR and wire) used in our algorithm. Wire is assumed as a logic gate that transfers data from its input to output without any changes. Cells information which extracted from the open source standard cell library "vsclib013" [23] in 0.13 micron technology, are shown in Table 1.

Fig. 4 shows how logic cells are encoded in a matrix of circuit structure. This circuit has 3 inputs (A,B,C) and one output (OUT). As shown in Fig. 4a, 3×3 matrix is populated with random logic gates and their interconnections. In Fig. 4b we have encoded matrix and in Fig. 4c the effective gates of the circuit are shown. Effective gates are those which connect output(s) of circuit to its inputs. Cell (2,3) which attribute is (1,2,2,2,7) is an XNOR gate (according to Table 1). Output of the circuit is the output of this gate and the first input of this cell is connected to the output of the cell (1,2), which is a NAND gate, and the second input is connected to the output of an OR gate in (2,2). Cells in location (1,1), (2,1) and (3,1) are AND,NOR and wire respectively and they are connected to primary inputs A, B and C.

3.2. Circuit evaluation

After initializing random matrixes of logic cells as colonies, all circuits should be evaluated. Here, to evaluate the evolving circuits, two main issues should be taken into consideration: 1 – functionality, 2 – optimization both in terms of space and propagation delay.

As mentioned before, truth table is a table that shows the status of circuit output(s) according to different combinations of circuit inputs. It is used to investigate the functionality of the circuit. A multi-objective evaluation mechanism in the form of a weighted cost function introduced to obtain both functional and optimized circuits (Eq. (8)):

$$f = \frac{w_{\text{match}} \cdot N_{\text{match}} + w_{ad} \cdot N_{\text{area-delay}}}{w_{\text{match}} + w_{ad}} \tag{8}$$

where w_{match} is weight of N_{match} and w_{ad} is weight for the area and propagation delay.

And

$$N_{\text{match}} = \frac{\text{the number of correct outputs from circuit}}{\text{the number of outputs from truth table}} \tag{9}$$

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

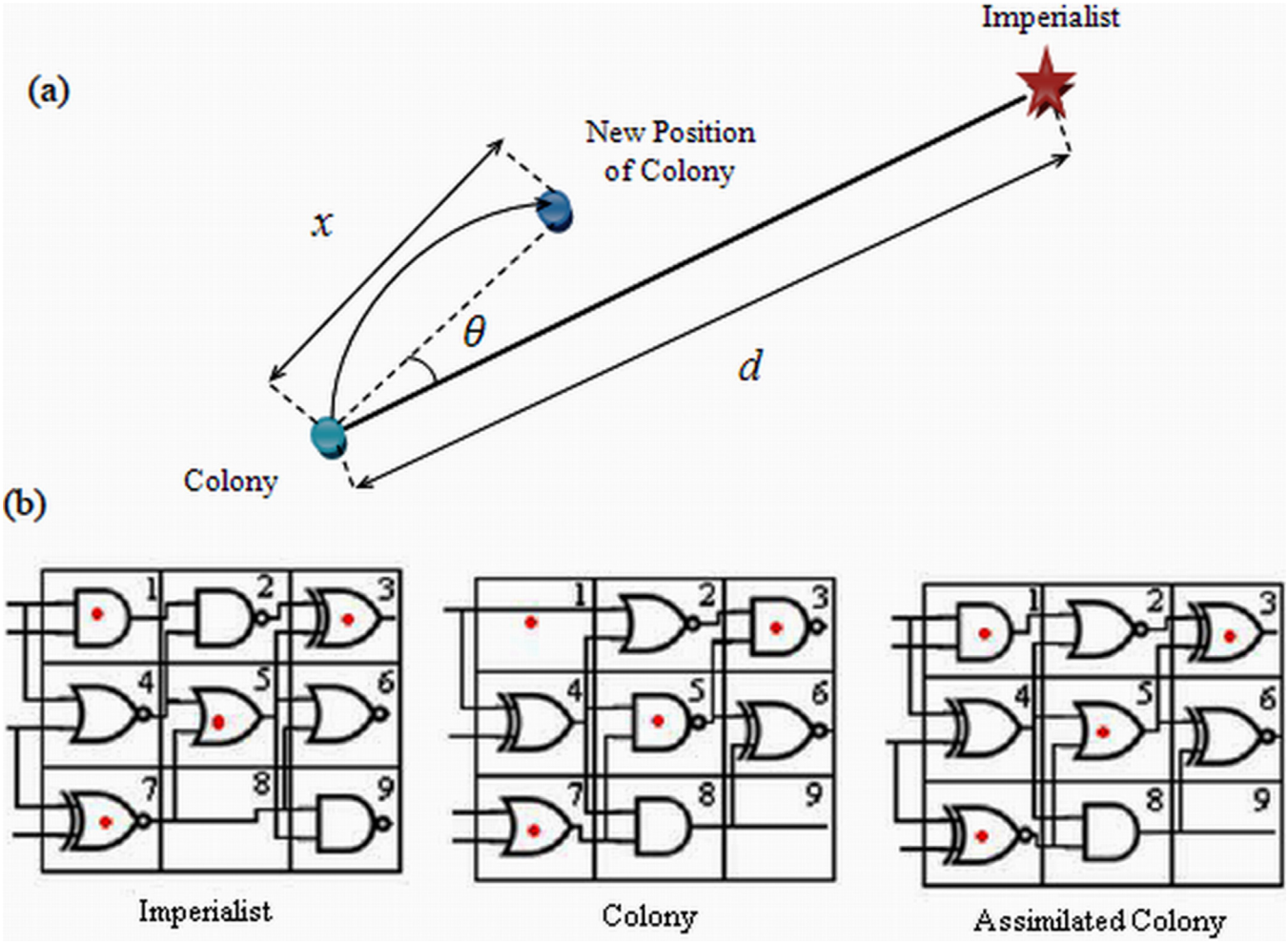

Fig. 5. (a) Moving colonies toward their relevant Imperialist [17]. (b) Modification of moving colonies toward Imperialist.

And

$$N_{\text{area-delay}} = \sum_{i=1}^{m\times n} \text{Area}(\text{Gate}_i) \times (\text{Delay of critical path}) \quad (10)$$

Area of a circuit is calculated by the effective area of logic gates without considering interconnections. When the input signal to a logic gate changes (i.e. from "0" to "1"), there is a finite time delay before the output changes. This is called the propagation delay of a logic gate and it differs from one gate to another. To evaluate delay of logic circuits, critical path should be considered and it is the path from input to output with the highest total propagation delay.

Since, in this work circuit functionality is prior to optimization, therefore we consider $w_{\text{match}} = 10\, w_{ad}$.

During search process, we encounter some circuits which their output does not match truth table output or N_{match} is less than 50%. For these cases we invert the output gate of the circuit (e.g. XNOR instead of XOR) and change N_{match} (Eq. (11)):

$$N_{\text{match}} = \max\{N_{\text{match}}, 1 - N_{\text{match}}\} \quad (11)$$

3.3. Modification of assimilation policy

The original version of the Imperialist Competitive Algorithm operates on real values [17]. Imperialists tend to improve their colonies behavior and this fact has been modeled by moving all colonies toward their Imperialist. As shown in Fig. 5a, colonies move toward their relevant imperialist along a line between them or some time with a random deviation angle.

In logic circuit designing with ICA, we deal with matrix structures as countries and moving these matrix structures toward each other is nonsense. So we proposed a modification for assimilation operator to overcome this shortcoming and it is as follows:

(1) Select some random cells (according to assimilation rate) from imperialist matrix structure.
(2) Replace selected cells of the imperialist matrix with the same cells in colonies matrix structure.

Fig. 5b shows modified assimilation on circuit structures. In this figure, the left matrix is imperialist and the middle one is colony. Some random cells (i.e. 1,3,5,7) are selected and replaced in colony structure. Finally right matrix is obtained by assimilation policy. This modification causes circuit structures of an empire (colonies) approach to their relevant imperialist circuit. For revolution operator, we select some colonies (circuits) in each imperialist and replace them with random circuit structures.

4. Experiments and results

In this section, we compare the evolutionary circuits obtained by our proposed algorithm to those designed by human (Karnaugh map) as well as those evolved by Coello's genetic algorithm [7] and with

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

Table 2
Truth table of examples.

Input				Example 1,2,3			Example 4			Example 5			
x_3	x_2	x_1	x_0	Y	Y	Y	y_2	y_1	y_0	p_3	p_2	p_1	p_0
0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0	1	0	0	0	0	0
0	1	0	0	0	1	1	0	0	1	0	0	0	0
0	1	0	1	0	0	1	1	0	0	0	0	0	1
0	1	1	0	1	1	1	0	1	0	0	0	1	0
0	1	1	1	1	1	1	0	1	0	0	0	1	1
1	0	0	0	1	1	1	0	0	1	0	0	0	0
1	0	0	1	0	1	1	0	0	1	0	0	1	0
1	0	1	0	1	1	1	1	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	1	1	0
1	1	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	1	1	1	1	0	0	1	0	0	1	1
1	1	1	0	0	1	0	0	0	1	0	1	1	0
1	1	1	1	0	1	1	1	0	0	1	0	0	1

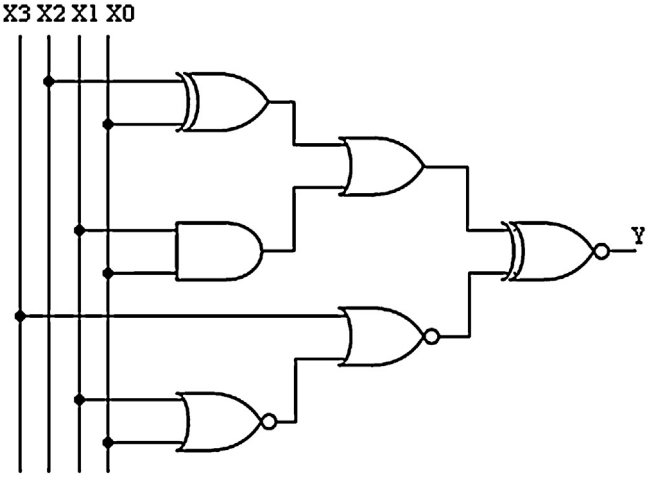

Fig. 6. Evolved circuit for example 1.

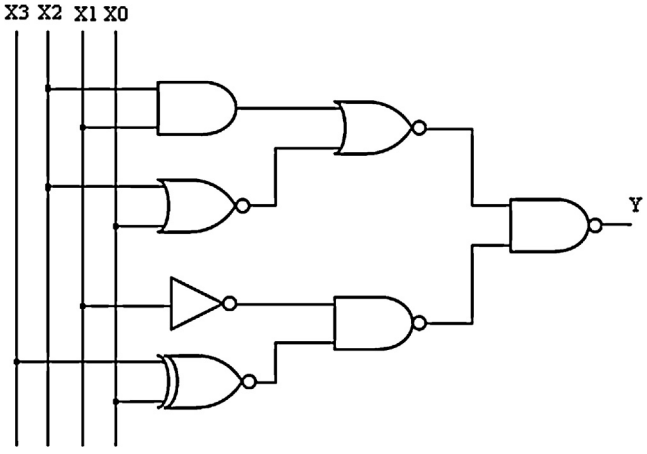

Fig. 7. Evolved circuit for example 2.

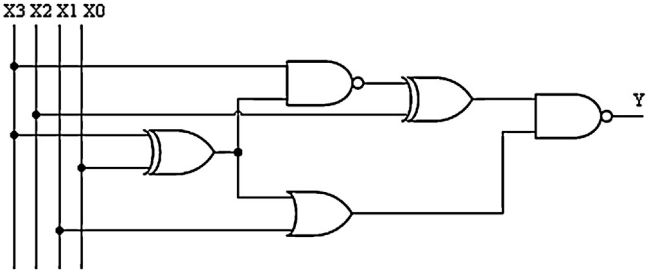

Fig. 8. Evolved circuit for example 3.

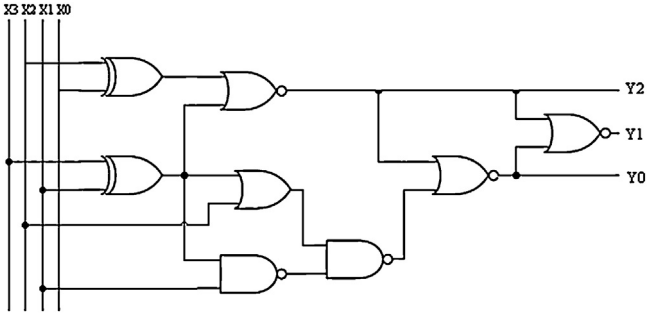

Fig. 9. Evolved circuit for example 4.

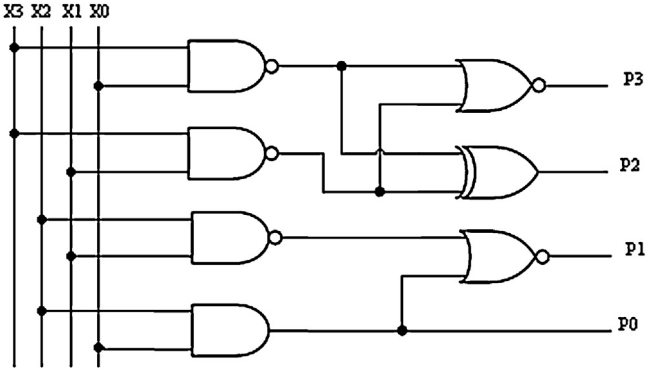

Fig. 10. Evolved circuit for example 5.

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

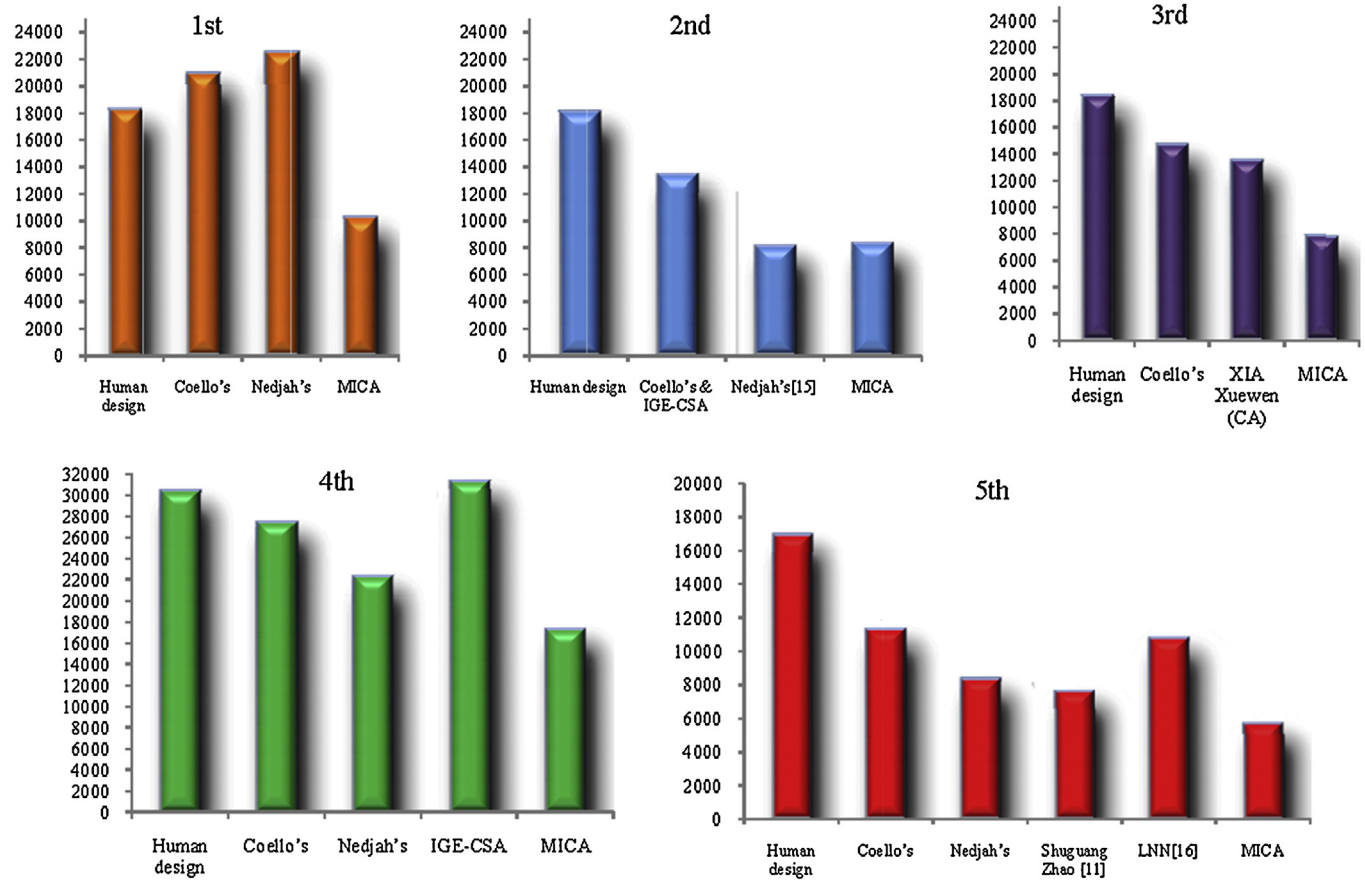

Fig. 11. Graphical comparisons for different methods according to performance factor.

some recently proposed works [12,14–16]. We used 5 popular examples for experiment.

Examples 1, 2 and 3 need a 4-bit input signal $X = \langle x_3x_2x_1x_0 \rangle$ and a single-bit output signal Y. Example 4 requires a 4-bit input signal X and a 3-bit output signal $Y = \langle y_2y_1y_0 \rangle$ and finally example 5 requires a 4-bit input signal X and a 4-bit output signal $P = \langle p_3p_2p_1p_0 \rangle$. The truth tables of the 5 examples are summarized in Table 2. Note that example 5 is a simple 2-bit multiplier of $\langle x_1x_0 \rangle$ times $\langle x_3x_2 \rangle$.

Simulation was done in MATLAB 2009 and 20 independent experiments (runs) were performed for each circuit. Parameters of the algorithm are: number of countries (300), number of imperialists (30), zeta (0.05), revolution rate (0.2) and assimilation rate (0.4). The size of the matrix, which was used as the circuit structure for examples 1,2,3,5 is 4×4 and for 4th example is 5×5.

For the first example, a human designer offered a circuit with 4 AND gates, 1 OR gate, 2 XOR gates and 4 NOT gates. Coello's genetic algorithm [7] evolved the circuit with 1 AND gate, 3 OR gates, 3 XOR gates and 1 NOT gate.

The circuit obtained by our proposed algorithm for the first example is shown in Fig. 6.

The specification of this circuit is given in following signal assignment (Eq. (12)).

$$y = ((x_2\text{XOR}x_0)\text{OR}(x_1\text{AND}x_0))\text{XNOR}(x_3\text{NOR}(x_1\text{NOR}x_0)) \quad (12)$$

For the second example, a human designer obtained circuit that requires 9 gates. 4 AND gates, 3 OR gates and 2 NOT gates. Coello's genetic algorithm [7] evolved the circuit that requires 2 AND gates, 2 OR gates, 2 XOR gates and 1 NOT gate.

The circuit obtained by our proposed algorithm for second example is shown in Fig. 7.

The specification of this circuit is given in the following signal assignment (Eq. (13)).

$$y = ((x_1\text{AND}x_2)\text{NOR}(x_2\text{NOR}x_0))\text{NAND}((\text{NOT}x_1)\text{NAND}(x_3\text{XNOR}x_0)) \quad (13)$$

For the third example, a human designer obtained circuit that requires 10 gates. 4 AND gates, 2 OR gates and 4 NOT gates and Coello's genetic algorithm [7] evolved the circuit that requires 1 AND gate, 2 OR gates, 3 XOR gates and 1 NOT gate.

The circuit obtained by our proposed algorithm for third example is shown in Fig. 8.

The specification of this circuit is given in the following signal assignment (Eq. (14)).

$$y = (x_2\text{XOR}(x_3\text{NAND}(x_3\text{XOR}x_0)))\text{NAND}(x_1\text{OR}(x_3\text{XOR}x_0)) \quad (14)$$

For example 4, a human designer engineered a circuit that requires 13 gates: 2 XOR gates, 2 OR gates, 4 AND gates and 5 NOT gates. For the same example, the Coello's genetic algorithm [7] evolved the circuit that requires 11 gates, which consist of 2 AND gates, 3 OR gates, 3 XOR gates and 3 NOT gates.

The circuit obtained by our proposed algorithm for 4th example is shown in Fig. 9.

The specification of this circuit is given as follows:

$$y_0 = y_2\text{NOR}((x_2\text{OR}(x_1\text{XOR}x_3))\text{NAND}(x_1\text{NAND}(x_1\text{XOR}x_3)));$$

$$y_1 = y_0\text{NOR}y_2;$$

$$y_2 = (x_0\text{XOR}x_2)\text{NOR}(x_1\text{XOR}x_3);$$

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

Table 3
Comparison of results for the five examples due to different methods.

Circuit	Method	Number of gates	Area (μm)	Propagation delay (ns)	Performance factor	Improvement of performance factor
Example 1	Human design	11	30,528	0.599	18,286.2	44.2%
	Coello's [7]	8	27,072	0.773	20,926.6	51.2%
	Nedjah's [15]	7	23,616	0.954	22,529.6	54.7%
	Our method (MICA)	6	20,160	0.506	10,200.9	–
Example 2	Human design	9	23,616	0.768	18,137	53.8%
	Coello's & IGE-CSA [7,12]	7	22,464	0.599	13,456	37.8%
	Nedjah's [15]	4	14,976	0.544	8147	No improvement
	Our method (MICA)	7	19,008	0.440	8363.5	–
Example 3	Human design	10	24,192	0.759	18,361.7	57.6%
	Coello's [7]	7	24,192	0.608	14,708.7	47.1%
	XIA Xuewen (CA) [14]	7	27,072	0.499	13,509	42.5%
	Our method (MICA)	5	16,704	0.465	7767.3	–
Example 4	Human design	13	35,136	0.868	30,498	43.0%
	Coello's [7]	11	33,408	0.822	27,461.3	36.7%
	Nedjah's [15]	13	31,680	0.704	22,302.7	22.1%
	IGE-CSA [12]	10	38,016	0.827	31,439.2	44.7%
	Our method (MICA)	8	23,616	0.735	17,357.7	–
Example 5	Human design	12	35,136	0.483	16,970.6	66.6%
	Coello's [7]	7	23,616	0.481	11,359.3	50.1%
	Nedjah's [15]	9	25,920	0.325	8424	32.7%
	Shuguang Zhao [11]	7	23,616	0.323	7628	25.7%
	LNN [16]	7	19,008	0.568	10,796.5	47.5%
	Our method (MICA)	7	19,008	0.298	5664.3	–

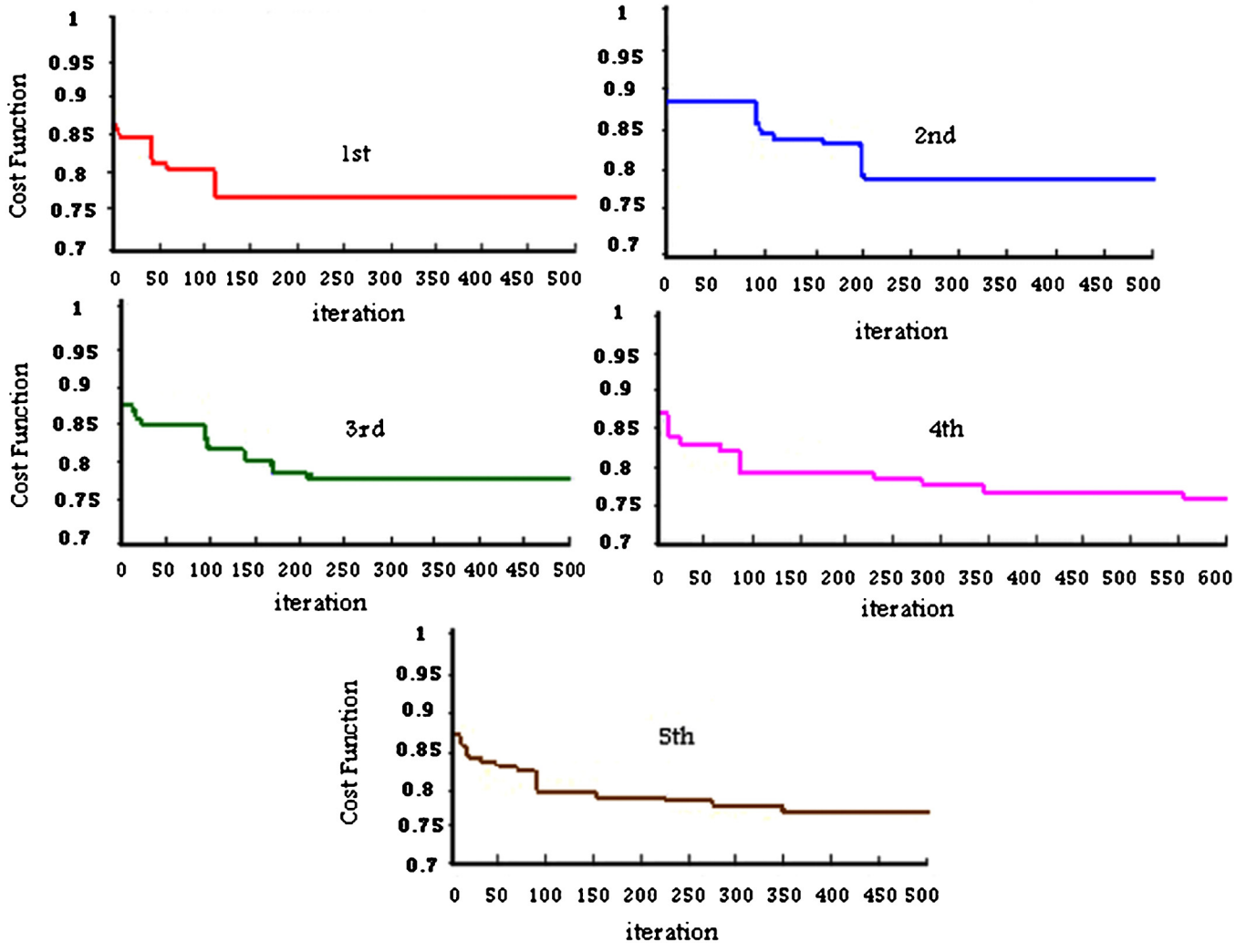

Fig. 12. Convergence characteristics of MICA for the best solutions.

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

Table 4
Summary of the Algorithm results after 20 runs for 5 examples.

	Best iteration	Worst iteration	Average	Standard deviation	Cost function of the best run
1st	115	416	211.5	86.02	0.7625
2nd	200	642	381.5	142.67	0.7875
3rd	210	743	460.5	164.63	0.775
4th	560	1429	845.5	271.7	0.7648
5th	350	781	528.5	131.99	0.7653

Finally for example 5, a human designer obtained a circuit that requires 12 gates: 8 AND gates, 2 OR gates, 1 XOR gate and 1 NOT gate. The Coello's genetic algorithm [7] evolved the circuit that requires an area of 7 gates: 5 AND gates and 2 XOR gates.

The schematic of the circuit we evolved is given in Fig. 10.

The specification of this circuit is given as follows:

$$p_0 = (x_0 \text{AND} x_2);$$

$$p_1 = (x_0 \text{AND} x_2)\text{NOR}(x_1 \text{NAND} x_2);$$

$$p_2 = (x_1 \text{NAND} x_3)\text{XOR}(x_0 \text{NAND} x_3);$$

$$p_3 = (x_1 \text{NAND} x_3)\text{NOR}(x_0 \text{NAND} x_3);$$

Nedjah [15] introduced a performance factor (area × propagation delay) and used this factor as a comparison criterion between her proposed method and the others. This factor shows the effectiveness of the optimized circuit in both area and propagation delay. Here, we used this performance factor to evaluate our proposed method with Nedjah's and the other proposed methods.

Table 3 shows a comparison between the results of different methods for each example according to the number of gates. Area and propagation delay and also performance factor and improvement of this factor in our proposed method to the others.

Fig. 11 shows a graphical comparison between different methods for 5 examples according to their performance factor. In examples 1, 3, 5 our evolved circuit is better than the others both in case of area and propagation delay. In example 2, Nedjah obtained circuit with fewer gates but our evolved circuit is faster and in example 4, our evolved circuit has fewer gates but Nedjah's circuit is better in case of propagation delay.

Fig. 12 illustrates the Convergence characteristics of our evolutionary process. Y axis shows the cost function of the best circuit in each iteration. The best algorithm iteration for the 1st, 2nd, 3rd, 4th and 5th examples were obtained in 115, 200, 210,560 and 350 respectively and all of them were evolved in minutes. Table 4 shows the results of the algorithm after 20 runs.

5. Conclusion

In this paper, a new heuristic algorithm, named Modified Imperialist Competitive Algorithm (MICA) which is compatible for designing combinational logic circuits introduced. The circuits obtained by this method are optimized in term of area and propagation delay. To have fair comparisons, a performance factor (area × propagation delay) which highlights the circuits with low area and low propagation delay, is used. The simulation results of the MICA illustrate approximately 44% improvement for the performance factor in comparison with other methods. By this creative method, efficient logic circuits can be designed which save space on digital chips and make them as fast as possible.

References

[1] J.F. Miller, D. Job, V.K. Vassilev, Principles in the evolutionary design of digital circuits. Part I Genetic Programming and Evolvable Machines, vol. 1, Kluwer Academic Publishers, Netherland, 2000, pp. 7–35.
[2] M. Karnaugh, A map method for synthesis of combinational logic circuits, Trans. AIEE Commun. Electron. 72 (1953) 593–599.
[3] E.J. McCluskey, Minimisation of Boolean functions, Bell Syst. Tech. J. 35 (1956) 1417–1444.
[4] X. Yao, T. Higuichi, Promises and challenges of evolvable hardware, IEEE Trans. Syst. Man Cybern. C 29 (1999) 87–97.
[5] C.Y. Chen, R.C. Hwang, A new variable topology for evolutionary hardware design, Expert Syst. Appl. 36 (2009) 634–642.
[6] C.A. Coello, A.H. Aguirre, B.P. Buckles, Evolutionary multiobjective design of combinational logic circuits, in: Proceedings of the Second NASA/DoDWorkshop on Evolvable Hardware, Los Alamitos, CA, 2000, pp. 161–170.
[7] C.A. Coello, A.D. Christiansen, A.H. Aguirre, Towards automated evolutionary design of combinational circuits, Comput. Electr. Eng. 27 (2001) 1–28.
[8] C.A. Coello, A.H. Aguirre, Design of combinational logic circuits through an evolutionary multiobjective optimization approach, Artif. Intell. Eng. Des. Anal. Manuf. 16 (2002) 39–53.
[9] A. Słowik, M. Białko, Evolutionary design and optimization of combinational digital circuits with respect to transistor count, Bull. Pol. Acad. Sci. Tech. Sci. 54 (4) (2006).
[10] A. Slowik, M. Bialko, Evolutionary design of combinational digital circuits: state of the art, main problems, and future trends, in: 1st International Conference on IT, 2008.
[11] S. Zhao, J. Zhao, L. Jiao, Adaptive genetic algorithm based approach for evolutionary design and multi-objective optimization of logic circuits, in: Proceedings of the 2005 NASA/DoD Conference of Evolution Hardware (EH'05), 2005.
[12] Z. Gan, T. Shang, G. Shi, M. Jiang, Evolutionary design of combinational logic circuits using an improved gene expression-based clonal selection algorithm, in: Fifth International Conference on Natural Computation, 2009.
[13] A.K. Srivastava, M.C. Srivastava, A combinatorial digital circuit with evolutionary algorithm for evolvable hardware software codesign, in: IEEE Power, Control and Embedded Systems (ICPCES), 2010.
[14] X. Xia, Z. Li, L. Gui, Design of Combination Logic Circuits based on Variant-Length Cellular Automata, IEEE Multimedia Technology (ICMT), 2011.
[15] N. Nedjah, L.D.M. Mourelle, Multi-objective CMOS-targeted evolutionary hardware for combinational digital circuits, Informatica 29 (2005) 309–319.
[16] D. Bhadra, T.A. Tarique, An encoding technique for design and optimization of combinational logic circuit, in: Proceedings of 13th International Conference on Computer and Information Technology (ICCIT 2010), 23–25 December, 2010.
[17] E. Atashpaz-Gargari, C. Lucas, Imperialist Competitive Algorithm: an algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation (CEC), 2007, pp. 4661–4667.
[18] M. Anjomshoa, A. Mahani, M. Esmaeil beig, Evolutionary design and optimization of digital circuits using imperialist competitive algorithm, Int. J. Comput. Appl. (IJCA) 32 (2011) 14–19.
[19] C. Lucas, Z. Nasiri-Gheidari, F. Tootoonchian, Application of an imperialist competitive algorithm to the design of a linear induction motor, Energy Convers. Manage. 51 (2010) 1407–1411.
[20] H. Mozafari, B. Abdi, A. Ayob, Optimization of transmission conditions for thin interphase layer based on imperialist competitive algorithm, (IJCSE) Int. J. Comput. Sci. Eng. 2 (2010) 2486–2490.
[21] M.H. Sayadnavard, A.T. Haghighat, M. Abdechiri, Wireless sensor network localization using Imperialist Competitive Algorithm, in: 3rd IEEE International Conference on Computer Science and Information Technology, 2010.
[22] S.J. Louis, G.J.E. Rawlins, Designer genetic algorithms: genetic algorithms in structure design, in: Fourth Int. Conf. Genet. Algorithms, Morgan Kauffman, San Mateo, CA, 1991, pp. 53–60.
[23] G. Petley, ASIC Standard Cell Library Design, 2011, http://www.vlsitechnology.org/ (accessed 05.10.11).

Mehdi Anjomshoa (Born 1978) received his B.S. in Electronic Engineering from Shahid Bahonar University, Kerman, Iran and his M.Sc. in 2011 in Electronic Engineering from Islamic Azad University, Bushehr Branch, Bushehr, Iran. Now he is working for South Pars Gas Complex as Instrument senior engineer and his current research interests include artificial intelligence, evolutionary electronics, and automatic control systems.

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011

Ali Mahani (Born 1978) received the Ph.D. in Electrical engineering from Iran University of Science and Technology (IUST) in 2009. Since then he has been with the Department of Electrical and Electronic Engineering of Shahid Bahonar University, where he is currently an assistant professor. His interests focus on Fault tolerant digital design and Performance evaluation of Communication Networks.

Salahedin Sadeghifard (Born 1977) received his B.S. in 1999 in Electronic Engineering from the Urmia University, Urmia, Iran and his M.Sc. in 2010 in Electronics Engineering from Islamic Azad University, Bushehr Branch, Bushehr, Iran. He is the teaching assistant of Electrical Engineering Department of Tabnak University. And now he works on Artificial Electronic Nose.

Please cite this article in press as: M. Anjomshoa, et al., A new automated design and optimization method of CMOS logic circuits based on Modified Imperialistic Competitive Algorithm, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.011