ICM 2003, Dec. 9-11, Cairo, Egypt.

Ant Colony Algorithm for Evolutionary Design of Arithmetic Circuits

Mostafa Abd-El-Barr, *Senior Member, IEEE,* Sadiq M. Sait, *Senior Member, IEEE,* and Bambang A. B. Sarif

***Abstract*— Evolutionary computation is a new field of research in which hardware design is pursued by deriving inspiration from biological organisms. This new paradigm is expected to radically change the synthesis procedures in a way that allows discovering novel designs and/or more efficient circuits. In this paper, a multi objective optimization strategy for design of arithmetic circuits based on Ant Colony optimization algorithm is presented. Results are compared with those obtained using other techniques.**

***Index Terms*— Logic Design, Evolutionary Computation, Ant Colony Optimization, Multiobjective Optimization, Fuzzy Logic.**

I. Introduction

DESIGN of digital circuits is a process to assemble a collection of components to realize a specified function using a target technology. Typically, the behavior of each component of the designed circuit is well known. The difficulty lies in predicting how an assembly of such components will behave.

Unfortunately, current design systems tend to depend on domain-specific knowledge, which is somewhat constrained both by the training and experience of the designer. On the other hand, non-deterministic iterative heuristics, with little domain knowledge, may allow us to define a search space, make some assumptions and use domain-independent operators for generating candidate solutions in the design space. Iterative heuristics have tendency to search for solutions in a much larger, and often richer, design space beyond the realms of the conventional techniques. It may therefore be possible to use them to obtain novel designs that are difficult to find using conventional methods.

It was Hugo de Garris who made the first move to investigate the design of evolving circuits. In his paper [1], de Garris suggested the establishment of a new field of research called Evolvable Hardware (EHW). At about the same time, the first work in evolutionary design of digital circuits was carried out by Louis [2]. A complete review and taxonomy of the field is described in [3]. The work of Thompson [4] that produced a tone discriminator circuit without input clock has shown the emergence of this new way of designing circuits.

In a recent development, much attention is given to the evolutionary design of arithmetic circuits. Such effort has resulted in the development of arithmetic circuits that range from a simple sequential adder structure to the more complex 3-bit multiplier. Some of the recent work can be found in [5], [6], [7], [8]. Unfortunately, majority of the published work attempts to obtain optimized circuits in terms of gate count only, and overlook other major issues such as delay and power consumption. In this paper, a multi objective evolutionary logic design based on Ant Colony Optimization (ACO) for arithmetic circuits is proposed. The goal is to find optimized circuits in terms of area, delay and power.

II. Ant Colony Optimization Algorithm

Ant Colony Optimization (ACO) algorithm [9] is a new meta-heuristic that combines distributed computation, autocatalysis (positive feedback) and constructive greedy heuristic in finding optimal solutions for combinatorial optimization problems. Unlike Genetic Algorithms (GAs), which are blind, ACO involves cooperating agents (ants).

The ACO algorithm has been inspired by the behavior of real ants. It was observed that real ants were able to select the shortest path between their nest and food resource, in the existence of alternate paths between the two. The search is made possible by an indirect communication known as *stigmergy* amongst the ants. While traveling their way, ants deposit a chemical substance, called *pheromone*, on the ground. When they arrive at a decision point, they make a probabilistic choice, biased by the intensity of pheromone they smell. When they return back, the probability of choosing the same path is higher (due to the increase of pheromone). Then, new pheromone will be released on the chosen path. This behavior has an autocatalytic effect because the very fact of choosing a path will increase the amount of pheromone on the corresponding path, which in turn will make it more attractive for future ants to follow. Shortly, all ants will select the shortest path. Figure 1 illustrates this phenomenon.

In ACO algorithm, the optimization problem is formulated as a graph $G = (C, L)$, where C is the set of components of the problem, and L is the possible connection or transition among the elements of C. The solution is expressed in terms of feasible paths on the graph G, with respect to a set of given constraints.

III. Fitness Function Calculation

The fitness of a solution contains two parts, namely functional fitness and objective fitness.

A. Functional Fitness

The functional fitness deals with the functionality of the solution, i.e., how good the solution is in satisfying the truth table of the intended Boolean function. Several functional fitness (FF) function calculations are reported in the literature [3]. The most commonly used one is the ratio of the number

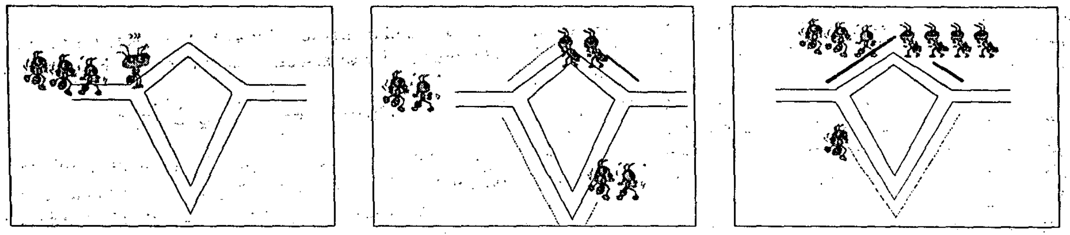

Fig. 1. Illustration of ants finding the shortest path in ACO algorithm

of hits to the length of the truth table. This can be formulated as follows.

$$R = \frac{Number\ of\ hits}{Length\ of\ truth\ table} \tag{1}$$

The number of hits is defined as the number of correct matchings between the output patterns obtained from the solution and the truth table of the intended function. The solution has to be 'inverted' if the value of R is less than 0.5. Therefore, the formulation below is applied.

$$FF = Max\{R, 1 - R\} \tag{2}$$

B. Objective Fitness

The objective fitness (OF) is the measure of the quality of solution in terms of optimization objectives such as area, delay, gate count and power consumption. It contains two aspects: constraints satisfaction and multi objective optimization. In this paper, fuzzy logic is used to represent the cost function for area, delay and power. In order to build the membership function, the lower bound and upper bound of the cost function must be determined [10].

In order to guide the search intelligently, the maximum value must be carefully estimated. For this purpose, SIS tools [11] are used to obtain circuits with minimum area. In this context, *rugged.script* is used to generate the circuits' netlist files. These files are then fed to our own tool to obtain the estimated value for area, delay and power consumption. The reason behind this is twofold. Firstly because the delay optimization in SIS does not consider switching delay. Secondly, SIS does not consider power optimization.

Since we want to obtain circuits better than SIS, these values (area, delay, and power) are used as the target values. In the case of area as optimization objectives, the target area is equal to the area of circuits obtained by SIS and denoted as tg_{area1} (see Figure 2).

In order to guide the search intelligently, the maximum value must be carefully estimated. For this purpose, SIS tool [11] is used to estimate the minimum area and minimum delay of the target circuits.

The estimated lower bound of maximum area (called $target_{area}$) is associated with a specific degree of membership called target membership (μ_{target}). The shape of the membership function is depicted in Figure 2. The shape of the membership function is depicted as the bold line shown in Figure 2.

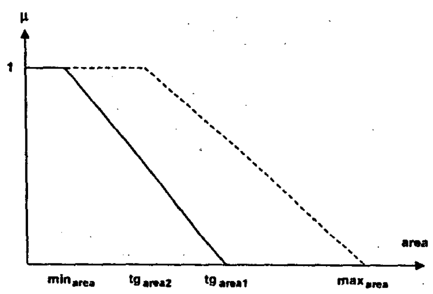

Fig. 2. Membership function for area

In case of area as constraint, the area of circuit obtained from SIS is used as target value. For this purpose, the max_{area} and tg_{area2} should be defined. The following settings are applied, $tg_{area2} = k_1 \times tg_{area1}$ and $max_{area} = k_2 \times tg_{area1}$, k_1, $k_2 \in \Re$, $0 < k_1 \leq 1$, $k_2 \geq 1$. The shape of the membership function is depicted as dashed line shown in Figure 2.

The membership function for delay and power are built using similar rules (see [12] for further details). These three membership functions will be aggregated into one unit (the objective fitness) using OWA operator [13].

C. Overall Fitness Calculation

The overall fitness is then can be formulated as follows.

$$Fitness = Wf \cdot FF + (1 - Wf) \cdot OF \tag{3}$$

Where Wf is the weight for functional fitness. The value of Wf must be large enough in order to have better functionality of the circuit, because at the end functionally correct circuits are the only solutions accepted. However, Wf should not be too large in order to get better quality solutions in terms of design objectives.

IV. Circuit Encoding

A circuit is modelled as a matrix M of size $n \times m$. Each cell in the matrix containts a triplet of attributes. The first two numbers are for the inputs (input 1, input 2) and the third indicates the gate type. The value of input 1 and input 2 indicates the row from which the current cell is getting its input from. The value of the gate type indicates the type of the gate being assigned to that cell from a predetermined set of gate types. A gate at position (i, j), where i is the row number and j is the column number, can only be connected to the one at $(i', (j-1))$.

There are 10 types of gate available. Table I shows these gates.

Gate ID	Gate	Output
0	WIRE1	a
1	WIRE2	b
2	NOT1	$\overline{a}$
3	NOT2	$\overline{b}$
4	AND	$a \cdot b$
5	OR	$a + b$
6	XOR	$a \oplus b$
7	NAND	$\overline{a \cdot b}$
8	NOR	$\overline{a + b}$
9	XNOR	$\overline{a \oplus b}$

TABLE I
GATE TYPES USED, CONSIDERING INPUT a AND b.

Consider the example shown in Figure 3. Cell(1,2) whose attribute is (0,3,4) is an AND gate (according to Table I). The first input of the AND gate of this cell is connected to the output of cell(0,1), which is a WIRE, and the second input is connected to the output of cell(2,1).

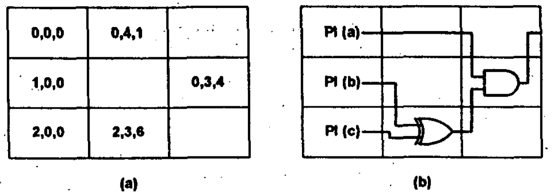

Fig. 3. Example of a circuit and its encoding.

At first, the matrix is filled up with randomly generated cells. Then, each ant will traverse the matrix. These ants are originated from a dummy cell called *nest*, and traverse each state (a cell in a column) until they reach the last column or a cell that has no successor. After the ants finish their tour, the matrix M is checked to see which cells of the matrix that are worth to be kept. The cells that are not included in the best solution in the current iteration will be removed. These empty cells will then be filled up again in the beginning of the next iteration. If it has not reached the maximum number of iterations, the procedure will be cycled again. Otherwise, the best solution is returned.

V. Pheromone Trail Calculation

The selection of which edge to traverse is determined by a stochastic probability function. It depends on the pheromone value (τ) and heuristic value (η) of the edge (or the next cell). The probability of selecting next node is formulated below:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} \cdot [\eta_{il}]^{\beta}} \qquad (4)$$

The value of α and β imply the preference of the search, whether it depends more on pheromone value or heuristic value respectively. Every newly created cell will be given an initial and small amount of pheromone value. This value will be updated every iteration by the ant.

The heuristic value (η) depends on the distance of FF values between cells. The distance d between cells is formulated as follows.

$$d = FF(j) - FF(i) \qquad (5)$$

$$\eta = d + 0.5 \qquad (6)$$

Where i is the current cell and j is the next cell visited by the ants.

The addition of 0.5 in the calculation of η is meant to normalize the value of η into [0,1]. A decrease in functional fitness means that the value of η is in the range of [0,0.5), while an increase of functional fitness makes the value of η in the range of (0.5, 1]

When all ants finish their tour, pheromone update is performed. The pheromone update is performed using the following equation:

$$\tau(t) = \tau(t) + \lambda \cdot OvF(t) \qquad (7)$$

where $OvF(t)$ denotes the overall fitness of the solution that the ants built and λ is a constant.

VI. Experiments and Results

Table II shows the results obtained using the proposed algorithm for area and delay minimization for some arithmetic circuits. The table shows that the percentage of improvement in area, delay and power of circuits obtained using delay minimization over area minimization varies. However, it can be seen clearly that the improvement in delay is always less than or equal to zero. This means that using delay minimization, the proposed algorithm successfully find circuits with less delay compared to the circuits obtained using area minimization.

In order to compare the results of applying our algorithm with known published results, some arithmetic circuits are tested and compared to the results reported in [6], [7]. These circuits include 2-bit adder, 2-bit multiplier and 3-bit multiplier. The comparison of results is shown in Table III. However, since the technique in [6], [7] do not incorporate delay and power, the comparison is performed only for gate count and area. The parameters used for the algorithms is obtained from MOSIS .25 μ library [14].

The table show that the proposed algorithm produced the best circuit in terms of area for 2-bit multiplier circuit. It also produced better results for 3-bit multiplier. For 2-bit adder circuit, the technique proposed in [6] produced better results. The reason behind this is that it uses MUX in addition to two input gates, while the proposed algorithm uses only two input gates.

Circuit	Area Optimization			Delay Optimization			% Improvement		
	Area	Delay	Power	Area	Delay	Power	Area	Delay	Power
majority	13851	4.57	5.06	16038	4.19	5.02	-15.79	8.32	0.79
xor8	20655	5.9	9.32	20655	5.9	9.32	0.00	0.00	0.00
xor9	23328	8.84	10.65	27216	8.84	11.48	-16.67	0.00	-7.79
add2	24300	11.48	9.96	31347	8.957	11.463	-29.00	21.98	-15.09
mul2	12636	3.56	4.66	18225	2.96	5.99	-44.23	16.85	-28.54
add3	49086	21.96	18.474	53703	12.979	21.484	-9.41	40.90	-16.29
mul3	59292	15.03	17.541	74358	13.138	21.645	-25.41	12.59	-23.40

TABLE II

RESULTS OBTAINED USING THE PROPOSED ALGORITHM FOR AREA AND DELAY OPTIMIZATION

Circuit	Proposed ACO		Coello [7]		Miller [6]	
	# Gate	Area	# Gate	Area	# Gate	Area
add2	11	24300	NA	NA	10*	19440
mul2	8	14823	7	17253	7	16281
mul3	32	59292	NA	NA	24	60264

* Assuming that a MUX is equivalent to 3 simple 2-input gates
NA Results are not available

TABLE III

COMPARISON WITH THE EXISTING TECHNIQUES

VII. CONCLUSION

In this paper, we have proposed an ACO-based evolutionary logic design technique. Performance of the proposed approach and comparison with existing techniques are shown. The proposed approach has shown that it is capable of producing optimized arithmetic circuits and has shown some promising results.

ACKNOWLEDGMENT

We would like to acknowledge the continued support for our research from King Fahd University of Petroleum & Minerals. This work was supported under project entitled "Iterative Heuristics for the Design of Combinational Logic Circuits".

REFERENCES

[1] Hugo de Garis. Evolvable Hardware: Genetic Programming of a Darwin Machine. *Proceedings of the International Conference in Innsbruck, Austria*, pages 441–449, Springer-Verlag, 1993.

[2] Sushil J. Louis. *Genetic Algorithms as a Computational Tool for Design.* PhD thesis, Department of Computer Science, Indiana University, Aug 1993.

[3] R. S. Zebulum and M. A. Pacheco and Maria Vellasco. *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms.* CRC Press, 2002.

[4] Adrian Thompson. Silicon Evolution. *Proceedings of the First Annual Conference on Genetic Programming*, pages 444–452, MIT Press, 1996.

[5] J. F. Miller, T. Fogarty, and P Thomson. Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study. *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science, John Wiley and Sons, Chichester*, pages 105–131, 1998.

[6] J. F. Miller, D. Job, and Vassilev V. K. Principles in the Evolutionary Design of Digital Circuits - Part I. *Journal of Genetic Programming and Evolvable Machines*, 1(1):8–35, 2000.

[7] C. A. Coello, A. D. Christiansen, and A. H. Aguirre. Ant Colony System for the Design of Combinational Logic Circuits. *Evolvable Systems: From Biology to Hardware, Edinburgh, Scotland*, pages 21–30, April Springer Verlag, 2000.

[8] C. A. Coello, A. D. Christiansen, and A. H. Aguirre. Towards Automated Evolutionary Design of Combinational Circuits. *Computers and Electrical Engineering, Pergamon Press*, 27(1):1–28, Jan. 2001.

[9] M. Dorigo, M. Maniezzo, and A. Colorni. The Ant Systems: An Autocatalytic Optimizing Process. Revised 91-016, Dept. of Electronica, Milan Polytechnic, 1991.

[10] S. Sait and H. Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems.* IEEE, 1999.

[11] E. M. Sentovic, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, University of California, Berkeley, May 1992.

[12] Bambang Ali Basyah Sarif. Modified ant colony optimization algorithm for combinational logic circuits design. Master's thesis, King Fahd University of Petroleum and Minerals, Dhahran, KSA, November 2003.

[13] Ronald R. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making. *IEEE Transaction on Systems, MAN, and Cybernetics*, 18(1), January 1988.

[14] Standard Cell Library for MOSIS CMOS.

Dr. Mostafa Abd-El-Barr obtained his Ph.D. from the University of Toronto, Canada, in the area of Computer Engineering in 1986. In July 1986 he joined the Faculty of the Department of Computer Science, University of Saskatchewan, Saskatoon, Canada. He was promoted to the Full Professor rank in July 1993. He has long experience in the area of Fault-Tolerant Computing and Testability, VLSI design and implementation of algorithms, System-level specifications of VLSI-based designs, Multiple-valued logic design and CMOS testability. He has taught both graduate and undergraduate courses in these areas in Canada, Kuwait and Saudi Arabia. He is the author and/or co-author of more than 100 scientific papers in the above research areas. Dr. Abd-El-Barr is a senior member of the IEEE Computer Society, a member of the Circuit and Systems Society, and a member of the ACM. Dr. Abd-El-Barr is a registered 'Professional Engineer' with the Province of Ontario, Canada.

Dr. Sadiq M. Sait has major interests in VLSI Design automation, and, in engineering and applications of computers. He has published several papers in the area of VLSI physical design automation. He has co-authored two books, which are directly related to the project: (a) *VLSI Physical Design Automation: Theory and Practice*, McGraw-Hill Book Co., Europe, December 1994. Also Co-published by IEEE Press, USA, January 1995 and, (b) *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems.* December 1999, IEEE Computer Society Press, California.

Bambang Ali Basyah Sarif obtained his BSc. degree from the Institut Teknologi Bandung, Indonesia, in the area of Electrical Engineering in 2000. In November 2003, he obtained his MSc. degree from the Department of Computer Engineering, King Fahd University of Petroleum & Minerals, KSA. He has major interest in the area of logic synthesis, evolutionary computation, evolvable hardware and fault tolerant system.