COMBINATIONAL LOGIC CIRCUITS DESIGN THROUGH ANT COLONY OPTIMIZATION ALGORITHM

Mostafa Abd-El-Barr, Sadiq M. Sait, Bambang A. B. Sarif, Uthman Al-Saiari

Computer Engineering Department
KFUPM, Dhahran-31261
{mostafa, sadiq, sarif, saiarios}@ccse.kfupm.edu.sa

ABSTRACT

With the increasing demand for high quality, more efficient and less area circuits, the problem of circuit design has become a multi-objective optimization problem. Therefore, there should evolve new methodologies for designing logic circuits. The new paradigm is expected to radically change the synthesis procedures in a way that can help discovering novel designs and/or more efficient circuits. In this paper, a multiobjective optimization of logic circuits based on a modified Ant Colony (ACO) algorithm is presented. The results obtained using the proposed algorithm are compared to those obtained using SIS in terms of area, delay and power.

1. INTRODUCTION

In conventional logic design, circuit designers begin with a precise specification in the form of truth tables or Boolean expressions. These expressions are manipulated by applying logic synthesis algorithms, such as factorization and kernel extraction to minimize circuit representations. Therefore, the outcome of logic synthesis algorithms will always be in the space of all logically correct representations. These will be either in two-level, or multi-level or Reed Muller representation.

Iterative heuristics work on a larger space that may not represent the desired function. Through the process of assemble and test, candidate solutions are built and evaluated. At the end, the optimum solution could evolve from this process.

The first work in evolutionary design of digital circuits, Designer Genetic Algorithms (DGA), was proposed in [1]. Later, the work of Thompson [2] that produced a tone discriminator circuit without input clock showed the emergence of a new way of designing circuits. In a recent development, much attention is given to the evolutionary design of arithmetic circuits as they provide the essential building blocks needed for larger DSP applications. Such effort has resulted in the development of arithmetic circuits that range from a simple sequential adder to the more complex 3-bit multiplier. The work of Miller [3, 4] claimed to build some arithmetic circuits that cannot be produced by human designer's conventional methods. Coello [5, 6] proposed a similar approach to evolve a circuit, which they claimed was better than that of Miller's. A complete review and taxonomy of the field could be found in [7, 8]. Unfortunately, these published work tries to find the optimized circuits in terms of gate count only. Nevertheless, power consumption has become one of the major criteria in modern circuit design.

Ant Colony Optimization (ACO) algorithm [9] is a new meta-heuristic algorithm with a combination of distributed computation, auto-catalysis (positive feedback) and constructive greedy heuristic in finding optimal solutions for combinatorial problems. Unlike Genetic Algorithms (GAs), which is a blind search heuristic, ACO is an optimization of co-operating agents (ants) algorithms. In this paper, a multi-objective evolutionary logic design based on Ant Colony Optimization (ACO) is proposed. The goal is to find optimized circuits in terms of area, delay and power.

2. ANT COLONY OPTIMIZATION ALGORITHM

The ACO algorithm has been inspired by 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. This behavior has an autocatalytic effect because of the very fact that choosing a path will increase the probability that it will be chosen again by future ants. When they return back, the probability of choosing the same path is higher (due to the increase of pheromone). New pheromone will be released on the chosen path, which makes it more attractive for future ants. Shortly, all ants will select the shortest path.

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.

3. PROPOSED APPROACH

Consider the Boolean function $f = \overline{x}yz + x\overline{y}z + xy\overline{z}$. Figure 1 shows a graph of some possible paths connecting literal x to the intended function f. Assume that the ants start the tour from literal x. The ant will traverse the paths by selecting the edges through a probabilistic process. Assume that the goal is to find the shortest path to represent function f. Therefore, the ants that found the path $x \rightarrow (x+y) \rightarrow (x+y)(xy \oplus z)$ would return the best representation for function f.

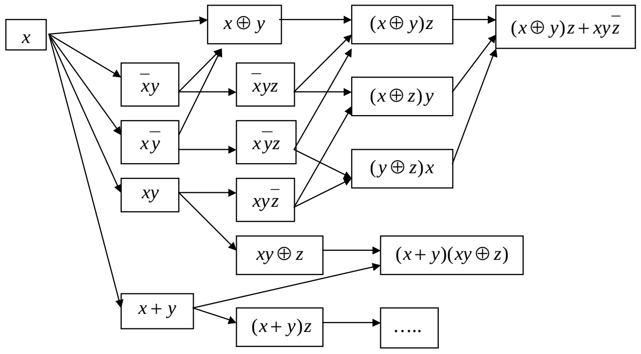

Fig. 1. Some of the possible paths in the function f.

The number of paths in Figure 1 is more than eleven. Traversing all possible paths is however, impractical. We need to modify the ACO algorithm to handle this huge search space.

3.1. Circuit Encoding and Representation

A circuit is modelled as a matrix M of size $n \times m$. The content of matrix M is dynamically filled. There are 10 types of gate available. Table 1 shows these gates.

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

Table 1. Gate types used

Consider the example shown in Figure 2. Cell(2,2) whose attribute is (0,3,4) is an AND gate (according to Table. 1). 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(3,1). Note that, the first column of the matrix that contains primary inputs is not shown in the figure.

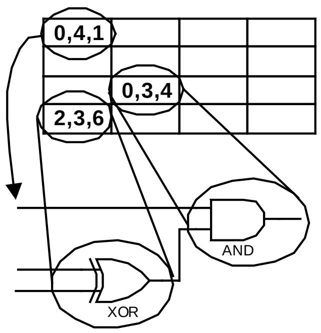

Fig. 2. Example of a circuit and its encoding

3.2. Fitness Function Calculation

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

3.2.1. *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 (F_f) function calculation are reported in the literature [8]. The most commonly used one is the ratio of the number of hits to the length of the truth table. This can be formulated as follows.

$$NH = \frac{Number\ of\ hits}{Length\ of\ truth\ table}$$

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.

$$F_f = Max\{NH, 1 - NH\} \qquad (1)$$

3.2.2. *Objective Fitness*

The objective fitness (F_o) 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 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 3

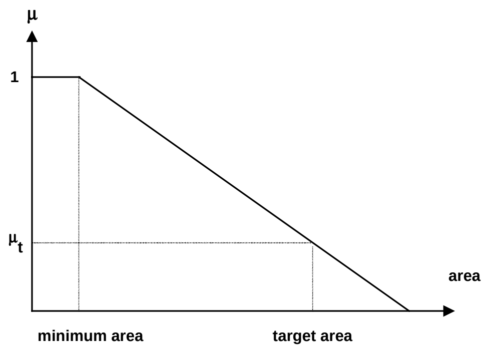

Fig. 3. Membership function for area as optimization objective

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

3.2.3. Overall Fitness Calculation

The overall fitness is then can be formulated as follows.

$$Fit = W_f \cdot F_f + (1 - W_f) \cdot F_o \qquad (2)$$

Where W_f is the weight for functional fitness. The value of these weights must be chosen intelligently. The value of W_f 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, it should not be too large in order to get better quality solutions in terms of design objectives.

3.3. Solution Construction

At first, matrix M is filled with randomly generated cells. Then, each ant will traverse the matrix. These ants originate from a dummy cell called *nest* (see Figure 4), and traverse each state (a cell in a column) until it reaches the last column or a cell that has no successor.

The selection edges to traverse is determined by a stochastic probability function. It depends on the pheromone value

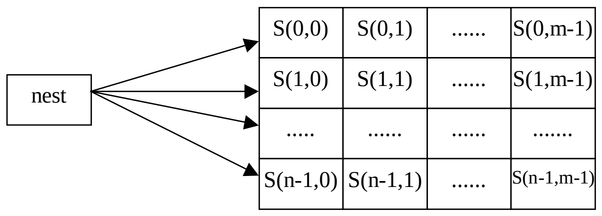

Fig. 4. Nest cell and matrix M for ant to be traversed.

(τ) and heuristic value (η) of the edge (or the next cell). The probability of selecting next node is formulated below:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \aleph_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \qquad (3)$$

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 F_f values between cells. The distance d between cells is then formulated as follows.

$$d = F_f(i+1) - F_f(i) \qquad (4)$$

$$\eta = d + 0.5 \qquad (5)$$

The addition of 0.5 in the calculation of η is meant to normalized 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:

$$\tau(t) = (1 - \rho) * \tau(t) + \lambda \cdot Fit(t) \qquad (6)$$

where $Fit(t)$ denotes the overall fitness of the solution that the ants built, ρ is pheromone evaporation rate and λ is a constant.

When all ants finish their movement, 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 be then filled up again in the beginning of the next iteration. If it has not reached the maximum iteration, the procedure will be cycled again. Otherwise, the best solution is returned.

Circuit	SIS			Proposed Algorithm			% Improvement		
	Area	Delay	Power	Area	Delay	Power	Area	Delay	Power
majority	13851	4.57	5.06	14823	6.28	5.41	6.56	27.18	6.48
xor8	20655	5.90	9.32	27945	27.69	10.82	26.09	78.70	13.89
xor9	23328	8.84	10.65	33048	33.25	12.65	29.41	73.40	15.83
add2	24300	11.48	9.96	29889	17.22	11.38	18.70	33.31	12.48
mul2	12636	3.56	4.66	18225	6.59	5.56	30.67	45.94	16.21
add3	49086	21.96	18.474	42282	24.99	15.68	-16.09	12.13	-17.79
mul3	59292	15.03	17.541	112752	43.39	37.75	47.41	65.36	53.53

Table 2. Comparison with SIS in area optimization

Circuit	SIS			Proposed Algorithm			% Improvement		
	Area	Delay	Power	Area	Delay	Power	Area	Delay	Power
majority	16038	4.19	5.02	18711	7.53	5.40	14.29	44.34	7.11
xor8	20655	5.90	9.32	32805	9.53	11.65	37.04	38.11	20.04
xor9	27216	8.84	11.48	41067	15.42	14.15	33.73	42.64	18.85
add2	31347	8.957	11.463	50787	11.77	14.63	38.28	23.90	21.64
mul2	18225	2.96	5.99	25272	4.33	7.16	27.88	31.57	16.30
add3	53703	12.979	21.484	118827	19.20	35.21	54.81	32.40	38.98
mul3	74358	13.138	21.645	174231	31.66	47.16	57.32	58.51	54.10

Table 3. Comparison with SIS in delay optimization

4. EXPERIMENTS AND RESULTS

In this section, comparison of the proposed algorithm with an existing conventional technique is given. For this purpose, SIS tools is used. However, SIS does not consider capacitance load in their delay calculation and does not have power optimization. Therefore, the results obtained from SIS are in the form of *netlist* file. These netlist file will be used as input to the cost function calculation procedures of the proposed algorithm to determine the area, delay and power of the circuits.

4.1. Area Optimization

The results from SIS are the area optimized circuits obtained by executing *rugged.script* script, mapped for area minimization. Both SIS and the proposed algorithm use the same gate library.

Table. 2 shows the results for area optimization for both techniques. The table shows that the highest improvements are obtained at 8-bit and 9-bit odd parity circuits. The parity circuits consist of XOR (XNOR) gates only. Unfortunately, SIS is unable to perform XOR decomposition. Thus, the parity circuits obtained by SIS will require larger area as compared to the ones obtained by the proposed algorithm.

For multiple-output circuits, the improvement in area varies. The highest improvements are observed at multiplier circuits circuits.

4.2. Delay Optimization

For delay optimization, the results from SIS are obtained by executing *delay.script* script, mapped for delay minimization. Both the proposed algorithm and SIS used the same gate library for the experiments. The test cases used are the same circuits used for area optimization in the previous section.

As can be seen from the table above, in contrast with area optimization, the results of delay optimization is very positive. The reason behind this is the following. ACO can be easily modeled as a shortest path finding problem. Since delay can be said proportional to the length of the path, ACO algorithm, which is the basic of the proposed algorithm, provides a good solution for delay optimization problem.

5. CONCLUSION

In this paper, we have proposed an ACO-based evolutionary logic design technique. Comparison of the proposed approach with SIS is shown. The proposed approach has shown that it is capable of producing optimized combinational circuits and has shown some promising results.

6. REFERENCES

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

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

[3] 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*, pp. 105–131, 1998.

[4] 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*, vol. 1, no. 1, pp. 8–35, 2000.

[5] C. A. Coello, A. D. Christiansen, and A. H. Aguirre, “Towards Automated Evolutionary Design of Combinational Circuits,” *Computers and Electrical Engineering, Pergamon Press*, vol. 27, no. 1, pp. 1–28, Jan. 2001.

[6] Coello Coello, Carlos A. and Hernndez Aguirre, Arturo,, “Evolutionary Multiobjective Design of Combinational Logic Circuits,” *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, pp. 161–170, Jul 2000.

[7] R. S. Zebulum, M. A. Pacheco, and Marley Vellasco, “Evolvable Systems in Hardware Design: Taxonomy, Survey and Applications,” *Evolvable System: From Biology to Hardware. Proceeding of the First International Conference, ICES 96 Tsukba, Japan, Lecture Notes in Computer Science*, vol. 1259, pp. 344–358, Oct. 1997.

[8] 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.

[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] Ronald R. Yager, “On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making,” *IEEE Transaction on Systems, MAN, and Cybernetics*, vol. 18, no. 1, January 1988.