FUZZIFIED SIMULATED EVOLUTION ALGORITHM FOR COMBINATIONAL DIGITAL LOGIC DESIGN TARGETING MULTI-OBJECTIVE OPTIMIZATION

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

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

ABSTRACT

In this paper, we employ fuzzified Simulated Evolution (SimE) algorithm for combinational digital logic design targeting area, delay and power as objectives. This technique is considered to be an evolutionary technique in logic design compared to conventional technique which uses deterministic algorithms for logic design. The performance of the proposed algorithm is evaluated using selected ISCAS'85 benchmark circuits. The results obtained using the proposed algorithm are compared to those obtained using SIS.

1. INTRODUCTION

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

In designing a complex system, circuit designers usually have to tradeoff one design objective for another. For example, often a designer tries to find a possibly faster circuit compared to a given previously designed one. However, the number of gates used and power dissipation are strongly related to delay. Thus, in seeking a faster circuit, one may end up having a complex system or a system that has higher power dissipation. Logic synthesis attempts to provide an answer to this problem. The purpose of logic synthesis tools is to aid circuit designers reaching an optimal tradeoff. Several logic synthesis algorithms are found in the literature [2, 3, 4, 5, 6].

Circuit designers use logic synthesis tools to create digital systems of arbitrary complexity. By using a top-down approach they tend to work in a space of lower dimensionality in which they are expert. However, this method of working is somewhat constrained both by the training and experience of the designer and by the amount of domain-specific knowledge known to the designer.

On the other hand, *evolutionary algorithms* may allow designers to define the search space of circuit design in a way that is natural to both the problem and the implementation. Evolutionary algorithms have tendency to search for a solution to the circuit design problem in a much larger, and often richer, design space beyond the realms of the traditional hardware search space. Evolutionary algorithms can thus help explore the search space regions needed to reach designs that are beyond the scope of conventional methods. It may therefore be possible to use evolutionary algorithms to obtain novel designs that are difficult to discover by conventional heuristics.

Furthermore, evolutionary design approaches do not assume a prior knowledge of any particular design domain. They can be used in domains where little knowledge is available or where such knowledge is costly to obtain. It is often possible to evolve hardware that is too complex in its structure for human to design.

The first work in evolutionary design of digital circuits, Designer Genetic Algorithms (DGA), was proposed in [7]. Later, the work of Thompson [8] 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 [9, 1] claimed to build some arithmetic circuits that cannot be produced by human designer's conventional methods. Coello [10, 11] 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 [12, 13]. 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.

The Simulated Evolution (SimE) algorithm is a general search strategy for solving a variety of combinatorial optimization problems [14, 15]. The SimE algorithm starts from

an initial assignment, and then, following an evolution-based approach, it seeks to reach better assignments from one generation to the next. A cost function called ***goodness measure*** is used by SimE algorithm in order guide the algorithm in the search space. In this paper, fuzzified Simulated Evolution (SimE) algorithm for combinational digital logic design targeting area, power and delay optimization is proposed. The results from the fuzzified Simulated Evolution (SimE) algorithm using selected ISCAS'85 benchmark circuits are compared to the results of a conventional logic design technique using SIS as our synthesis tool.

2. PROBLEM FORMULATION AND CIRCUIT ENCODING

Evolutionary computation views the problem of logic design as a search task. The methodology explores a solution space larger than that of the desired function, but gradually pulls the specification of the circuit towards the target truth table. However, the design space of digital circuits is huge. There are 2^n ($C_1^{2^n}$) possible solutions that satisfy $2^n - 1$ out of 2^n truth table's pattern for an n inputs single output function. In addition to that, the number of possible structures representing each of these solutions is many. These different structures represent different design objectives and/or constraints. Exploring the whole search space is impractical. Therefore, the size of the search space explored by the algorithm has to be reduced.

In this paper, we use the structure proposed in [7]. Each cell of the $n \times m$ matrix contains the information of the gate type and its corresponding inputs. However, unlike the fixed interconnection rules used in [7], we allow the output of each cell in column j to be connected to any of the cells in column $j+1$ ($j > 0,\ j+1 < m$). Thus, it is possible that cell$(i,k),\ 0 < i < n,\ 0 < k < m$, is not connected to any of the cells in column $k+1$.

Each cell of the matrix is considered to be an individual. The collection of all individuals of the matrix represents a solution. Each cell of the circuit matrix is encoded in a triplet of inputs and gate type, as illustrated in Figure 1. The first two numbers are for the inputs (input1, input2) and the third indicates the gate type. A gate at position (i,j), where i is the column number and j is the row number, can only be connected to the one at $((i-1), j')$ and j' can be any row of the previous column.

Input 1	Input 2	Gate type

Fig. 1. Representation of an individual in matrix.

Table 1 lists different types of gates and their functions, along with their code for the gene encoding.

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, Gate ID, and its corresponding Boolean function.

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.

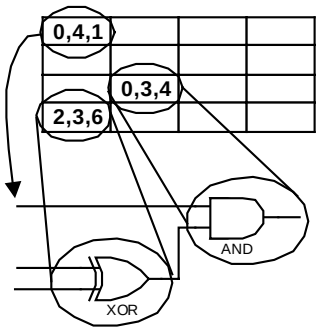

Fig. 2. Example of a circuit and its encoding

3. COST FUNCTION FORMULATION

The cost function or the fitness of a solution consists of two parts: functional fitness and objective fitness.

3.1. Functional Fitness

The functional cost measure is the correctness of the obtained logic circuit in matching the truth table of the required function. In this work, we proposed functional cost measure called ***Multilevel Logic Based Goodness Measure*** and it is based on the assumption that the higher the level of a gate in a multilevel logic circuit, the more minterms are covered at the output of that gate. Therefore, the ***goodness*** of a gate is affected by the number of minterms covered at its output and the level where the gate is located. Figure 3 illustrates this assumption. Since the number of inputs

of the circuit is 4, there are 16 patterns that should be generated at the output correctly. Initially, these patterns are distributed among the levels of the circuit evenly and progressively. Also, it is assumed that the initial number of levels is 4 since there are 4 columns in the search matrix. Therefore, a logic gate located at the second level should cover 8 patterns while a logic gate located at the first level should cover 4 patterns.

In general, for n inputs (2^n patterns) circuit, to have a goodness of 1 at a cell in level i, there should be $\lceil (2^n/n)i \rceil$ correct patterns produced at this cell. Thus, the *multilevel logic goodness measure* is formulated as follows:

$$g_i = \frac{\rho}{\lceil 2^n/n \rceil j}$$

where g_i is the *goodness* of cell i, j is the level number or column number, n is the number of inputs of the required circuit and ρ is the number of matching patterns at the output of cell i compared to the intended truth table.

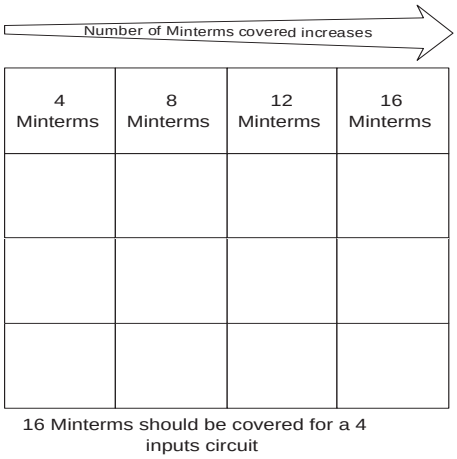

Fig. 3. Multilevel logic goodness assumption.

3.2. Objective Fitness

In this work, we are targeting area, delay and power. These are computed as follows. The cost for area of VLSI circuits is stated as follows.

$$Cost_{area} = \sum_{i \in G, i \neq WIRES} A(g_i) \quad (1)$$

Where $A(g_i)$ is the area of gate $g(i)$.

The propagation delay of signals in VLSI circuit consists of two elements, switching delay of gates and interconnect delay. If a path π consists of n gates $\{v_1, v_2, ..., v_n\}$, then, the delay T_π along π is expressed by the following equation:

$$T_\pi = \sum_{i=1}^{n-1} (CD_i + ((LF_i + R_i) \times C_i))$$

Where CD_i is the switching delay of the cell driving gate v_i. LF_i is the load factor of the driving block, R_i is the interconnect resistance of net v_i, and C_i is the load capacitance of cell i given by Equation 2. Since the value of R_i is constant, it can be neglected. The overall circuit delay is determined by the delay along the longest path (the most critical path).

The total capacitance C_i of gate i consists of the interconnect capacitance at the output node of gate i and the sum of the capacitances of the input nodes of the gates driven by gate i.

$$C_i = C_i^r + \sum_{j \in M_i} C_j^g \quad (2)$$

Where C_j^g is the capacitance of the input node of a gate j driven by gate i and C_i^r represents the interconnect capacitance at the output node of cell i.

The total power consumption can be approximated by the following equation [16].

$$P_t \simeq \sum_{i \in M} \frac{1}{2} \cdot C_i \cdot V_{DD}^2 \cdot f \cdot S_i \cdot \beta \quad (3)$$

Where P_t is the total power consumption, V_{DD} is the supply voltage, S_i is the switching probability at the output node of cell i, i.e., the average number of transitions per clock cycle at the output of gate i, f is the clock frequency and β is a technology dependent constant.

The cost of the overall power consumption in VLSI circuits can then be estimated as follows.

$$Cost_{power} = \sum_{i \in M} S_i \cdot C_i \quad (4)$$

The objective fitness (F_o) is the measure of the quality of solution in terms of optimization of area, delay 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 [17].

In order to guide the search intelligently, the maximum value must be carefully estimated. For this purpose, SIS tool [5] 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 4

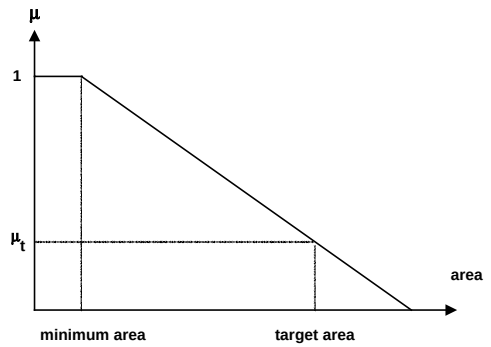

Fig. 4. Membership function for area as optimization objective

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 [18].

4. PROPOSED SIMULATED EVOLUTION (SIME)

Assume there are a set L of $|L|$ distinct locations and a set M of n elements of all logic gates as in Table 1 where $|L| \leq n$. SimE algorithm proceeds as follows. Initially, a population[1] is created at random from all populations satisfying the environmental constraints of the problem. Therefore, L locations (cells) of set L will be filled randomly by different logic gates. The algorithm has one main loop consisting of three basic steps, *Evaluation*, *Selection*, and *Allocation*. The algorithm evaluates every location (cell) of the set L using the ***Multilevel Logic Based Goodness Measure***. Some cells will be selected by the *Selection* step according to their goodness. If a cell has high goodness, it has less probability being selected. Next, in *Allocation* step, selected cells will be assigned different gate types in order to improve their goodness. The three steps are executed in sequence until the population average *goodness* reaches a maximum value, or no noticeable improvement to the population *goodness* is observed after a number of iterations. Another possible stopping criterion could be to run the algorithm for a prefixed number of iterations (see Figures 5 and 6).

5. 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.

[1] In SimE terminology, a population refers to a single solution. Individuals of the population are components of the solution; they are the movable elements.

ALGORITHM *Simulated_Evolution(E, ,L, Stopping-Criteria)*;
$INITIALIZATION$;
Repeat
  $EVALUATION$
  $SELECTION$
  $ALLOCATION$
Return ($BestSolution$);
End *Simulated_Evolution*.

Fig. 5. Simulated Evolution algorithm.

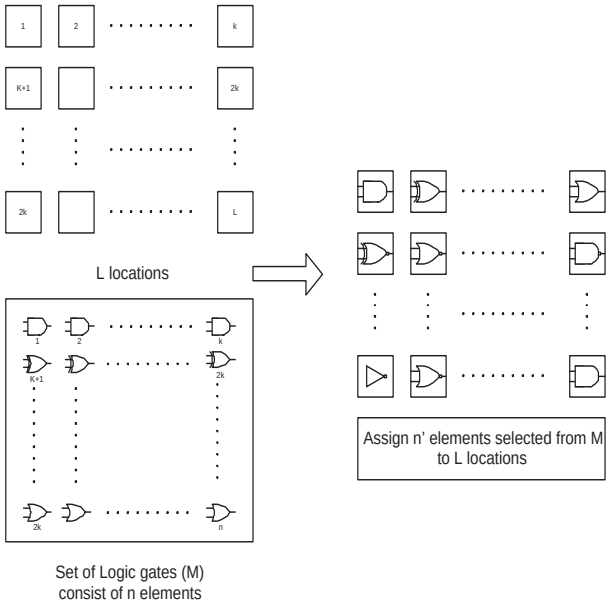

Fig. 6. Representation of digital logic design problem.

5.1. 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 delay optimization for both techniques. The table shows that the highest improvements are obtained at cm82a and mul3 circuits by 58.25% and 51.63 % respectively.

Circuit	SIS			Proposed Algorithm			% Improvement		
	Area	Delay	Power	Area	Delay	Power	Area	Delay	Power
mul2	18225	6.587026	5.561531	12636	3.56	4.66	44.23	84.98	19.35
mul3	112752	43.385843	37.745321	74358	13.14	21.65	51.63	230.23	74.38
cm42a	40824	8.864164	13.648234	40824	8.86	13.64	0.00	0.00	0.00
cm82a	39609	19.539984	14.879328	25029	11.84	9.24	58.25	64.98	61.12
b1	10206	3.225844	3.994219	11215	2.91	2.78	-8.93	10.85	43.68
c17	9963	3.559452	3.64207	9963	3.55	3.64	0.00	0.00	0.00
con1	31590	8.637996	11.21212	30233	6.90	14.23	4.49	25.19	-21.21
majority	14823	6.276723	5.405396	13851	4.57	5.06	7.02	37.32	6.93

Table 2. Comparison with SIS in area optimization

Circuit	SIS			Proposed Algorithm			% Improvement		
	Area	Delay	Power	Area	Delay	Power	Area	Delay	Power
mul2	25272	4.331142	7.157387	12636	3.56	4.66	100	21.66	53.59
mul3	174231	31.663494	47.161334	74358	13.14	21.65	51.63	230.23	74.38
cm42a	43740	8.46137	12.233066	40824	8.86	13.64	0	0	0
cm82a	64638	19.011371	18.936375	28552	9.34	9.1	38.73	109.21	63.51
b1	10206	3.225844	3.994219	11215	2.91	2.78	-8.93	10.85	43.68
c17	9963	3.559452	3.64207	9963	3.55	3.64	0	0	0
con1	31590	8.637996	11.21212	30233	6.9	14.23	4.49	25.19	-21.21
majority	14823	6.276723	5.405396	13851	4.57	5.06	7.02	37.32	6.93

Table 3. Comparison with SIS in delay optimization

5.2. 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.

Table. 2 shows the results for delay optimization for both techniques. It can be seen that the results obtained from SimE algorithm for area and delay optimization are the same except cm82a improved in the delay with larger area. The table shows that the highest improvements are obtained at cm82a and mul3 circuits.

6. CONCLUSION

In this paper, the use of Simulated Evolution (SimE) algorithm in logic design is being proposed. A goodness measure to guide SimE algorithm through the search space of digital logic design is suggested. Comparison of the proposed approach with SIS is shown. The proposed approach has shown better results in most cases compared to SIS considering area optimization and delay optimization.

7. REFERENCES

[1] 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.

[2] R. Brayton, G. D. Hachtel, C. T. McMullen, and A.L. Sangiovanni-Vincentelli, *Logic Minimisation Algorithms for VLSI Synthesis*, Kluwer Academic Publisher, 1984.

[3] R. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli, and A. Wang, "MIS: A Muliple-Level Logic Optimisation System," *IEEE Trans. on Computer-Aided Design*, vol. CAD-6, pp. 1062–1081, Nov. 1987.

[4] R. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis," *Proceeding of the IEEE*, vol. 78, pp. 264–300, Feb. 1990.

[5] 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.

[6] D. Bostick et al., "The Boulder Optimal Logic Design System," *Proceeding of the International Conference on CAD*, pp. 62–65, Nov. 1987.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] J. F. Miller and P. Thomson, "A Developmental Method for Growing Graphs and Circuits," *Fifth International Conference on Evolvable Systems: From Biology to Hardware*, vol. 2606, pp. 93–104, Mar 2003.

[15] R. M. Kling and P. Banerjee, "ESP: A New Standard Cell Placement Package using Simulated Evolution," *Proceeding of 24th Design Automation Conference*, pp. 60–66, 1987.

[16] Srinivas Devadas and Sharad Malik, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits," *32nd ACM/IEEE Design Automation Conference*, 1995.

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

[18] 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.