Evolutionary Multiobjective Design of Combinational Logic Circuits

Carlos A. Coello Coello*
Laboratorio Nacional de Informática Avanzada
Rébsamen 80, A.P. 696
Xalapa, Veracruz, México 91090
ccoello@xalapa.lania.mx

Arturo Hernández Aguirre
EECS Department
Tulane University
New Orleans, LA 70118, USA
hernanda@eecs.tulane.edu

Bill P. Buckles
EECS Department
Tulane University
New Orleans, LA 70118, USA
buckles@eecs.tulane.edu

Abstract

In this paper, we propose an evolutionary multiobjective optimization approach to design combinational logic circuits. The idea is to use a population-based technique that considers outputs of a circuit as equality constraints that we aim to satisfy. A small sub-population is assigned to each objective. After one of these objectives is satisfied, its corresponding sub-population is merged with the rest of the individulas in what becomes a joint effort to minimize the total amount of mismatches produced (between the encoded circuit and the truth table). Once a feasible individual is found, all individuals cooperate to minimize its number of gates. The approach seems to reduce the amount of computer resources required to design combinational logic circuits, when compared to our previous research in this area.

1. Introduction

The problem of interest to us consists of designing a combinational circuit that performs a desired function (specified by a truth table), given a certain specified set of available logic gates. Such a combinational logic circuit contains no memory elements and no feedback paths.

In the past, we have approached this problem using a genetic algorithm (GA) with a matrix encoding scheme, and an n-cardinality alphabet (after a series of experiments, we found this n-cardinality representation scheme more robust than the traditional binary representation used with GAs) [3, 4]. Our approach presents great (coincidental) resemblance with the one proposed by Miller [17] and further developed by Miller and his colleagues [16, 13]. The 2 main differences between the two approaches are the encoding scheme and the fitness function. Regarding the encoding, Miller et al. [17] use a more compact representation that instead of considering the inputs and gates as completely separate elements in the chromosomic string (as in our case), uses a single gene to encode a complete Boolean expression.

Regarding the fitness function, in our case, it works in two stages [10]. At the beginning of the search, only validity of the circuit outputs is taken into account, and the GA is basically exploring the search space. Once a feasible solution appears, then the fitness function is modified such that any valid designs produced are rewarded for each WIRE[1] gate that they include, so that the GA tries to find the circuit with the minimum number of gates that performs the function required. It is at this stage that the GA is actually exploiting the search space, trying to optimize the solutions found (in terms of their number of gates) as much as possible. Miller's initial work emphasized generation of functional circuits, rather than optimization. It was not until recently, that Kalganova & Miller experimented with a two-stage (or multiobjective, as they call it) fitness function as the one used by us [12]. However, the use of truly multiobjective optimization techniques (e.g., based on the concept of Pareto optimality [5]) remained as an open area

---

[*] Please send all correspondence to: PO Box 60324-394, Houston, Texas 77205, USA. Phone: 011 (52) (28) 18 13 02. Fax 011 (52) (28) 18 15 08.

[1] WIRE basically indicates a null operation, or in other words, the absence of gate, and it is used just to keep regularity in the representation used by the GA that otherwise would have to use variable-length strings.

of research in combinational circuit design, as suggested by Kalganova & Miller [12].

In this paper, we propose the use of an evolutionary multiobjective optimization technique (rather than just a multiobjective fitness function) to design combinational circuits. There is some (relatively scarce) previous work on using multiobjective techniques to handle constraints. This work, however, has concentrated on numerical optimization only. Our approach, originally introduced in a recent paper [6], was probably the first attempt to use this kind of technique in the design of circuits (we presented one example of the design of a circuit in [6]). Our proposal is to handle each of the matches between a solution generated by a GA and the values specified by the truth table as equality constraints. This, however, introduces some dimensionality problems for conventional multiobjective optimization techniques (this is because checking for dominance is an $O(n^2)$ process), and therefore the idea of using a (more efficient) population-based approach similar to the Vector Evaluated Genetic Algorithm (VEGA) [22].

The remainder of this paper is organized as follows: first, we describe some of the previous related work on using multiobjective optimization techniques to handle constraints. Then, we describe our approach and give some examples of its performance. Results are compared against those produced by our previous approach (a GA with an n-cardinality alphabet and a two-stage fitness function that we will simply denote as NGA) and against designs produced by humans (using Karnaugh Maps [14] and the Quine-McCluskey Procedure [20, 15]). Then, we present our conclusions and some of the possible paths of future research.

2. Related Work

The idea of using multiobjective optimization techniques to handle constraints is not new. Some researchers have proposed to redefine the single-objective optimization of f as a multiobjective optimization problem in which we will have $m + 1$ objectives, where m is the number of constraints. Then, we can apply any multiobjective optimization technique [9, 5] to the new vector $\bar{v} = (f, f_1, \ldots, f_m)$, where $f_1, \ldots, f_m$ are the original constraints of the problem. An ideal solution $\mathbf{X}$ would thus have $f_i(\mathbf{X})$=0 for $1 \leq i \leq m$ and $f(\mathbf{X}) \leq f(\mathbf{Y})$ for all feasible $\mathbf{Y}$ (assuming minimization).

Surry et al. [24, 23] proposed the use of Pareto ranking [8] and VEGA [22] to handle constraints using this technique. In their approach, called COMOGA, the population was ranked based on constraint violations (counting the number of individuals dominated by each solution). Then, one portion of the population was selected based on constraint ranking, and the rest based on real cost (fitness) of the individuals.

Parmee and Purchase [18] implemented a version of VEGA [22] that handled the constraints of a gas turbine problem as objectives to allow a genetic algorithm to locate a feasible region within the highly constrained search space of this application. However, VEGA was not used to further explore the feasible region, and instead Parmee and Purchase [18] opted to use specialized operators that would create a variable-size hypercube around each feasible point to help the genetic algorithm to remain within the feasible region at all times.

Camponogara & Talukdar [1] proposed the use of a procedure based on an evolutionary multiobjective optimization technique. Their proposal was to restate a single objective optimization problem in such a way that two objectives would be considered: the first would be to optimize the original objective function and the second would be to minimize the total amount of constraint violation of an individual.

Once the problem is redefined, non-dominated solutions with respect to the two new objectives were generated. The solutions found defined a search direction $d = (x_i - x_j)/|x_i - xj|$, where $x_i \in S_i$, $x_j \in S_j$, and S_i and S_j are Pareto sets. The direction search d is intended to simultaneously minimize all the objectives [1]. Line search is performed in this direction so that a solution x can be found such that x dominates x_i and x_j (i.e., x is a better compromise than the two previous solutions found). Line search takes the place of crossover in this approach, and mutation is essentially the same, where the direction d is projected onto the axis of one variable j in the solution space [1]. Additionally, a process of eliminating half of the population is applied at regular intervals (only the less fitted solutions are replaced by randomly generated points).

Jim´enez and Verdegay [11] proposed the use of a min-max approach [2] to handle constraints. The main idea of this approach is to apply a set of simple rules based on constraint violation to decide the selection process (individuals with the lowest amount of constraint violation would be preferred in a binary tournament).

In the context of combinational logic circuits design, we are not aware of any work in which the direct use of a multiobjective optimization technique had been proposed, except for the single circuit solved in [6]. The idea was, however, suggested by Kalganova and Miller [12].

3. Description of the approach

The main idea behind our proposed approach is to use a population-based multiobjective optimization technique such as VEGA [22] to handle each of the outputs of a circuit as an objective. In other words, we would have an optimization problem with m equality constraints, where m is the number of values (i.e., outputs) of the truth table that we aim to match. So, for example, a circuit with 3 inputs and a

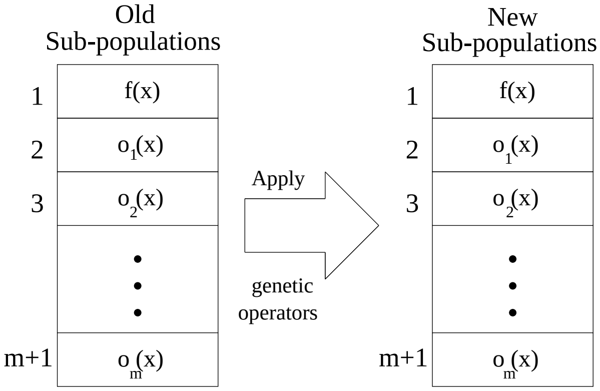

Figure 1. Graphical representation of the approach introduced in this paper.

single output, would have $m = 2^3 = 8$ values to match.

The technique may be better illustrated by Figure 1. At each generation, the population is split into $m + 1$ sub-populations, where m is defined as indicated before (we have to add one to consider also the objective function). Each sub-population optimizes a separate constraint (in this case, an output of the circuit). Therefore, the main mission of each sub-population is to match its corresponding output with the value indicated by the user in the truth table. Although the size of each sub-population may be variable, it was decided to allocate the same size to each of them in the experiments reported in this paper, but the use of different sizes remains as an open issue that requires further research.

The objective function in our case is defined as in previous work [3, 4]: it is the total number of matches (between the outputs produced by an encoded circuit and the intented values defined in the truth table defined by the user). For each match, we increase the value of the objective function by one. If the encoded circuit is feasible (i.e., it matches the truth table completely), then we add one (the so-called "bonus") for each WIRE present in the solution.

Using the proposed scheme, a fraction of the population will be selected using the objective function as its fitness (i.e., will try to maximize the total number of matches); another fraction will use the match of the first output as its fitness and so on (since they are all binary values, we only check if it matches or not, without computing any extra values as required in numerical optimization). The main issue here is how to handle the different situations that could arise. Our proposal is the following:

```
if $o_j(\mathbf{X}) \neq t_j$   then   fitness($\mathbf{X}$) = 0
else if $v \neq 0$              then   fitness = $-v$
else                                   fitness = $f(\mathbf{X})$
```

where $o_j(\mathbf{X})$ refers to the value of output j for the encoded circuit $\mathbf{X}$; t_j is the value specified for output j in the truth table; and v is the number of outputs that are not matched by the circuit $\mathbf{X}$ ($\leq m$). Finally, $f(\mathbf{X})$ is the fitness function described before:

$$f(\mathbf{X}) = h(\mathbf{X}) + \begin{cases} 0 & \text{if } f(\mathbf{X}) \text{ is infeasible} \\ w(\mathbf{X}) & \text{otherwise} \end{cases} \tag{1}$$

In this equation, $h(\mathbf{X})$ refers to the number of matches between the circuit $\mathbf{X}$ and the values defined in the truth table, and $w(\mathbf{X})$ is the number of WIREs in the circuit $\mathbf{X}$.

There are a few interesting things that can be observed from this procedure. First, each sub-population associated with an output of the circuit will try to match it with the value defined in the truth table. Once this is achieved, then the fitness function will try to maximize the number of matches of the rest of the outputs. In other words, this sub-population will cooperate with the others that are having difficulties to match their outputs. If the circuit is feasible, then all the sub-populations will join efforts to maximize the number of WIREs in the circuit.

It is important to clarify that the current approach does not use dominance to impose an order on the constraints based on their violation (like in the case of COMOGA [24]) which is a more expensive process (in terms of CPU time) that also requires additional parameters. In fact, the current approach does not rank individuals, but it uses instead different fitness functions for each of the sub-population allocated (whose number depends on the number of outputs in a circuit) depending on the feasibility of the individuals contained within each of them. This is easier to implement, does not require special operators to preserve feasiblity (like in the case of Parmee and Purchase's approach [18]), makes unnecessary the use of a sharing function to preserve diversity [7] (like with traditional multiobjective optimization techniques [9]), and does not require extra parameters to control the mixture of feasible and infeasible individuals (like in the case of COMOGA [24]).

Although VEGA is known to have difficulties in multi-objective optimization problems due to the fact that it tries

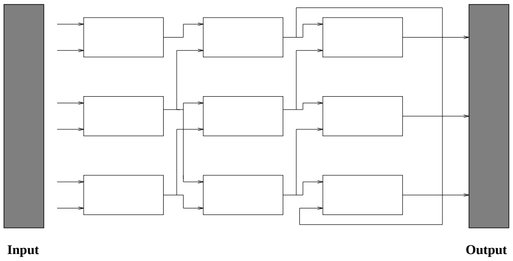

Figure 2. Matrix used to represent a circuit. Each gate gets its inputs from either of the gates in the previous column.

to find individuals that excel only in one dimension regardless of the others (the so-called “middling” problem [22, 9, 5]), that drawback turns out to be an advantage in this context, because what we want to find are precisely solutions that are completely feasible, instead of good compromises that may not satisfy one of the constraints (which are the kinds of solutions that a Pareto ranking strategy would normally produce). Also, the use of sub-populations is much more efficient than using Pareto dominance, because of the potentially high number of objectives involved (this will be illustrated in the examples shown in this paper).

4. The Genetic Algorithm Used

As in previous work [4, 3], we used a matrix to represent a circuit as shown in Fig. 2. This matrix is encoded as a fixed-length string of integers from 0 to $N-1$, where N refers to the number of rows allowed in the matrix (that is why we call it “n-cardinality alphabet).

More formally, we can say that any circuit can be represented as a bidimensional array of gates $S_{i,j}$, where j indicates the *level* of a gate, so that those gates closer to the inputs have lower values of j. (Level values are incremented from left to right in Fig. 2). For a fixed j, the index i varies with respect to the gates that are “next” to each other in the circuit, but without being necessarily connected. Each matrix element is a gate (there are 5 types of gates: AND, NOT, OR, XOR and WIRE) that receives its 2 inputs from any gate at the previous column as shown in Fig. 2. Although our GA implementation allows gates with more inputs and these inputs might come from any previous level of the circuit, we limited ourselves to 2-input gates and restricted the inputs to come only from the previous level.

A chromosomic string encodes the matrix shown in Fig. 2 by using triplets in which the 2 first elements refer to each of the inputs used, and the third is the corresponding gate as shown in Fig. 3.

Input 1	Input 2	Gate Type

Figure 3. Encoding used for each of the matrix elements that represent a circuit.

Table 1. Truth table for the circuit of the first example.

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Our goal was then to produce a fully functional design (i.e., one that produces all the expected outputs for any combination of inputs according to the truth table given for the problem) which maximizes the number of WIREs.

5. Examples

We have used several circuits of different degrees of complexity to test our approach. For the purposes of this paper, 4 examples were chosen to illustrate our approach (called multiobjective genetic algorith, or MGA for short), and the results produced were compared with those generated by human designers and by our previous GA-based implementation (called NGA) [3, 4].

5.1. Example 1

Our first example has 3 inputs and one output, as shown in Table 1. In this case, the matrix used was of size 5×5, and the length of each string representing a circuit was 75. Results are compared on Tables 2, 3 and 4. Human Designer 1 used Karnaugh Maps plus Boolean algebra identities to simplify the circuit, whereas Human Designer 2 used the Quine-McCluskey Procedure. In both cases, they produced solutions with more gates than the GA.

In previous work [4] we had been able to find a solution with 4 gates (fitness value of 29) for this circuit. To choose the size of each sub-population in the MGA, we started with 10, and performed ten runs. To make a fair comparison, we

NGA	**MGA**
$F = (Z + Y)(Y \oplus (X \oplus Z))'$	$F = (X + Y)Z \oplus (XY)$
5 gates	4 gates
1 AND, 1 OR, 2 XORs, 1 NOT	2 ANDs, 1 OR, 1 XOR

Table 2. Comparison of the best solutions found by the n-cardinality GA (NGA) and a our multiobjective genetic algorithm (MGA) for the circuit of the first example. In both cases, a population size of 90 was used.

Human Designer 1
$F = Z(X \oplus Y) + Y(X \oplus Z)$
5 gates
2 ANDs, 1 OR, 2 XORs

Table 3. Results produced by the first human designer for the circuit of the first example

used the same parameters for both GAs used. If a feasible solution was not found, then we would increase the size by 10 and would perform ten more runs. This process was repeated until a suitable sub-population size was found (i.e., such that at least one third of the results produced from the ten runs were feasible).

In this case, due to the small size of the circuit, a sub-population size of 10 was enough. Since the circuit has 8 outputs, there were 9 objectives. Therefore, the total population size was set to 90. We arbitrarily set the maximum number of generations to 300. The same representation scheme and the same genetic operators (uniform crossover with a probability of 0.5, and uniform mutation with a probability of 0.5/l, where l is the length of the chromosome) were used for both the MGA and the NGA (see [4] for more details). These genetic operators and their parameters (except for the population size) were also used in the other examples).

The MGA consistently found a solution with fitness value of 29 (6 out of 10 times; the 4 remaining solutions had a fitness of 28). The graphical representation of this solution is depicted in Fig. 4. In all runs of the MGA, the circuits produced were feasible.

On the other hand, the best solution that the NGA could find using the same population size had a fitness of 28 (i.e., a feasible circuit with 5 gates). This solution appeared only 4 times in the 10 runs performed, and in one case, the best solution found was infeasible. To find a solution with a fitness of 29 with the NGA, a population size of at least 700 individuals was required.

Human Designer 2
$F = X'YZ + X(Y \oplus Z)$
6 gates
3 ANDs, 1 OR, 1 XOR, 1 NOT

Table 4. Results produced by a second human designer for the circuit of the first example

Z	**W**	**X**	**Y**	**F**
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Table 5. Truth table for the circuit of the second example.

5.2. Example 2

Our second example has 4 inputs and one output, as shown in Table 5. A matrix of the same size as before was used (i.e., 5×5).

The comparison of the results produced by the MGA, the NGA, a human designer using Karnaugh Maps, and Sasao's approach [21] are shown in Tables 6 and 7. Sasao has used this circuit to illustrate his circuit simplification technique based on the use of ANDs & XORs. His solution uses, however, more gates than the circuit produced by the NGA or the MGA.

In previous work [4] we had been able to find a solution with 10 gates (fitness value of 31) for this circuit. Since this example has 16 outputs, there are 17 objectives for the MGA. After performing 10 runs using a sub-population size of 10 (i.e., total population size of 170) neither the MGA nor the NGA found a feasible solution. With a sub-population size of 20 (i.e., total population size of 340), the MGA was able to find 7 feasible solutions. Interestingly, one of these

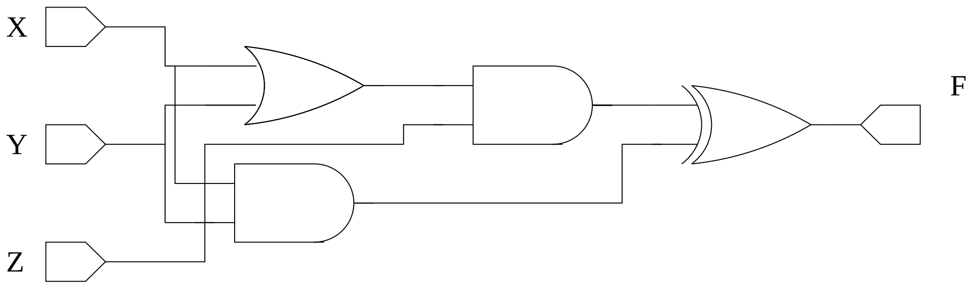

Figure 4. Graphical representation of the optimum circuit for the first example.

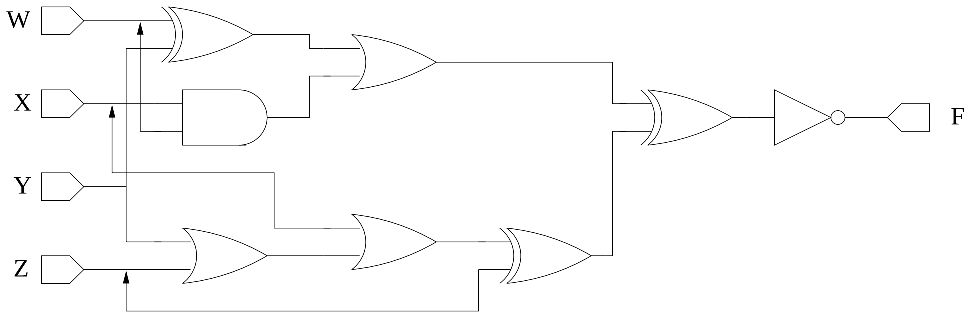

Figure 5. Circuit produced by our MGA for the second example.

MGA	**NGA**
$F = (((W \oplus WX) \oplus ((Z + X + Y) \oplus Z))'$	$F = (WYX' \oplus ((W + Y) \oplus Z \oplus (X + Y + Z)))'$
8 gates	10 gates
1 AND, 3 ORs, 3 XORs, 1 NOT	2 ANDs, 3 ORs, 3 XORs, 2 NOTs

Table 6. Best circuit found by the MGA (using a population size of 340) and the NGA (using a population size of 900) for the second example.

Human Designer 1	**Human Designer 2**
$F = ((Z'X) \oplus (Y'W')) + ((X'Y)(Z \oplus W'))$	$F = X' \oplus Y'W' \oplus XY'Z' \oplus X'Y'W$
11 gates	12 gates
4 ANDs, 1 OR, 2 XORs, 4 NOTs	3 XORs, 5 ANDs, 4 NOTs

Table 7. Results produced by one human designer (using Karnaugh Maps) and Sasao for the second example.

A	**B**	**C**	**D**	**F**
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Table 8. Truth table for the circuit of the third example.

solutions had a fitness of 33 (i.e., a circuit with only 8 gates). The maximum number of generations in this case (for both the MGA and the NGA) was arbitrarily set to 400.

On the other hand, the best solution that the NGA could find using the same population size of 340 had a fitness of 15 (i.e., it was an infeasible circuit). This solution consistently appeared in the 10 runs performed (except for one in which the maximum fitness was 14). To find feasible circuits, the NGA required population sizes of at least 500 individuals. To find a solution with a fitness of 31 (i.e., with 10 gates) the NGA required a population size of at least 900 individuals.

5.3. Example 3

Our third example has 4 inputs and one output, as shown in Table 8. A matrix of the same size as before was used (i.e., 5×5).

The comparison of the results produced by the MGA, the NGA and two human designers (the first, using Karnaugh Maps and the second using the Quine-McCluskey Procedure) are shown in Tables 9 and 10. In previous work [4] we had been able to find a solution with 7 gates (fitness value of 34) for this circuit.

Since this example has 16 outputs, there are 17 objectives for the MGA. Using a sub-population size of 10 (total population size of 170), the MGA found 8 feasible solutions. One of them had a fitness of 34 (i.e., it had 7 gates). The graphical representation of this circuit is shown in Fig. 6. The maximum number of generations in this case (for both the MGA and the NGA) was arbitrarily set to 400.

On the other hand, the best solution that the NGA could find using the same population size of 170 had a fitness of 15 (i.e., it was an infeasible circuit). This solution consistently appeared in the 10 runs performed (except for two cases in which there were solutions with a fitness of 13 and 14, respectively). To find feasible circuits, the NGA required population sizes of at least 400 individuals. To find a solution with a fitness of 34 (i.e., with 7 gates) the NGA required a population size of at least 800 individuals.

$\mathbf{A}_1$	$\mathbf{A}_0$	$\mathbf{B}_1$	$\mathbf{B}_0$	$\mathbf{X}_2$	$\mathbf{X}_1$	$\mathbf{X}_0$
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Table 11. Truth table for the circuit of the fourth example.

5.4. Example 4

Our fourth example has 4 inputs and 3 outputs, as shown in Table 11. A matrix of the same size as before was used (i.e., 5×5).

The comparison of the results produced by the MGA, the NGA, and one human designer (one using Karnaugh Maps) are shown in Tables 12 and 13.

Since this example has 48 outputs, there are 49 objectives for the MGA. With a sub-population size of 10 (i.e., total population size of 490), the MGA was able to find 5 feasible solutions, from which one of them had a fitness of 66 (i.e., a circuit with 7 gates). The graphical representation of this circuit is shown in Fig. 7. The maximum number of generations in this case (for both the MGA and the NGA) was arbitrarily set to 400.

The best solution that the NGA could find using the same population size of 490 had a fitness of 45 (i.e., it was an in-

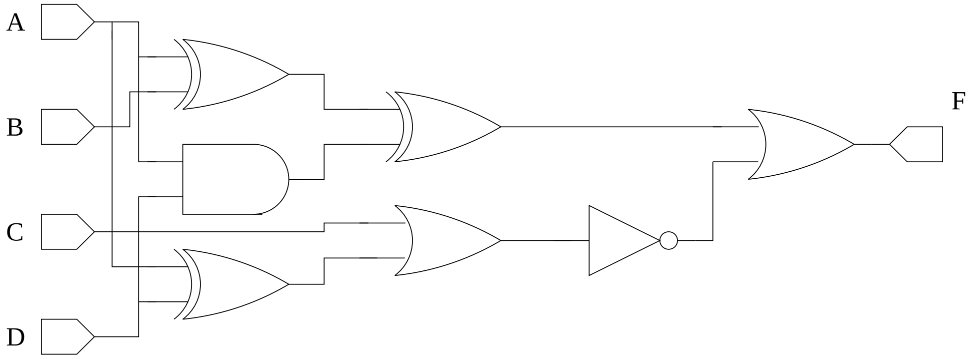

Figure 6. Circuit produced by our MGA for the third example.

MGA	NGA
$F = ((A \oplus B) \oplus AD) + (C + (A \oplus D))'$	$F = ((B \oplus A) \oplus AD) + (C + (D \oplus A))'$
7 gates	7 gates
1 AND, 2 ORs, 3 XORs, 1 NOT	1 AND, 2 ORs, 3 XORs, 1 NOT

Table 9. Best circuit produced by our MGA (using a population size of 170) and by the NGA (using a population size of 800) for the third example.

Human Designer 1	Human Designer 2
$F = ((A \oplus B) \oplus ((AD)(B + C))) + ((A + C) + D)'$	$F = A'B + A(B'D' + C'D)$
9 gates	10 gates
2 ANDs, 4 ORs, 2 XORs, 1 NOT	4 ANDs, 2 ORs, 4 NOTs

Table 10. Results produced by two human designers for the third example.

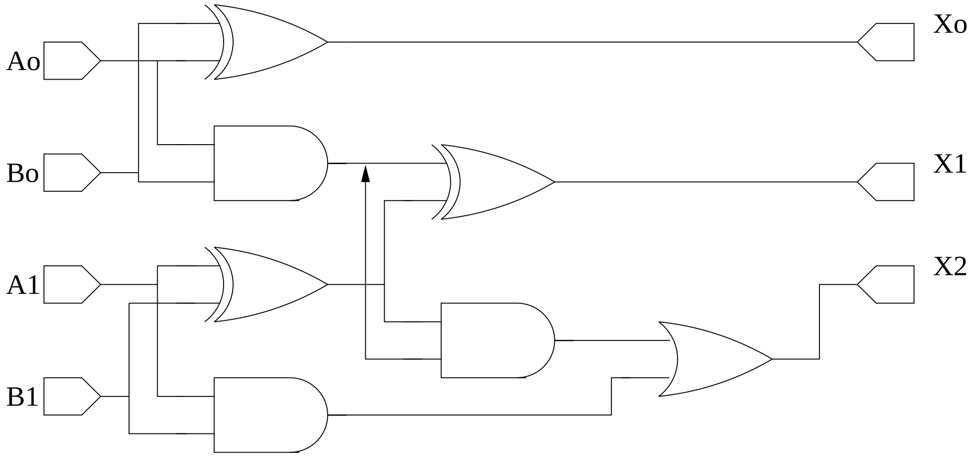

Figure 7. Circuit produced by our MGA for the fourth example.

MGA	NGA
$X_0 = A_0 \oplus B_0$	$X_0 = A_0 \oplus B_0$
$X_1 = A_0B_0 \oplus (A_1 \oplus B_1)$	$X_1 = A_0B_0 \oplus (A_1 \oplus B_1)$
$X_2 = A_1B_1 + A_0B_0(A_1 \oplus B_1)$	$X_2 = (A_0B_0)(A_1 \oplus B_1) + (A_1B_1)$
7 gates	7 gates
3 ANDs, 2 ORs, 2 XORs	3 ANDs, 1 OR, 3 XORs

Table 12. Best solution found by the MGA (using a population size of 490) and the NGA (using a population size of 1000) for the fourth example.

Human Designer 1
$X_0 = A_0 \oplus B_0$
$X_1 = (A_1 \oplus B_1)B_0' + ((A_1 \oplus B_1) \oplus A_0)B_0$
$X_2 = A_1B_1 + A_0B_0(A_1 + B_1)$
12 gates
5 ANDs, 3 ORs, 3 XORs, 1 NOT

Table 13. Results produced by a human designer (using Karnaugh Maps) for the fourth example.

feasible circuit). This solution consistently appeared in the 10 runs performed (except for two in which the maximum fitness was 44). To find feasible circuits, the NGA required population sizes of at least 600 individuals. To find a solution with a fitness of 66 (i.e., with 7 gates) the NGA required a population size of at least 1000 individuals.

6. Conclusions and Future Work

We have proposed a multiobjective optimization technique to design combinational logic circuits. The proposed approach uses a population-based technique to split the search task among several (small) sub-populations. The approach compared well with respect to a previous GA developed by us which uses and n-cardinality alphabet and a two-stage fitness function. Our approach, called MGA, was able to find the same or even better solutions than the previous one (called NGA), using a lower number of fitness function evaluations.

We believe that the good performance obtained with this algorithm is mainly due to an emergent behavior obtained from the cooperation of the different sub-populations aiming to satisfy a simple goal. This line of thought is consistent with the recent work by Potter & DeJong [19], according to which the resolution of complex problems with evolutionary algorithms requires a cooperative effort. Additionally, the current technique can also be considered a variation of a divide-and-conquer approach to evolvable hardware suggested by Torresen [25]. In this approach, a system is evolved through its smaller components. Only that in our case, these smaller components happen to be individual outputs of a circuit. Torresen [25] also showed that a scheme of this sort could substantially reduce the computational power required to evolve a system.

We want to study this approach in more detail, solving more complex circuits, and analyzing the interactions between the parameters of the GA (including the sizes of each sub-population) and its performance. Finally, we also intend to develop a parallel version of the algorithm.

7. Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments that helped them improve this paper. The first author acknowledges partial support from REDII-CONACyT, and from CONACyT project number I-29870 A. The second and third authors acknowledge support for this work to DoD EPSCoR and the Board of Regents of the State of Louisiana under grant F49620-98-1-0351.

References

[1] E. Camponogara and S. N. Talukdar. A Genetic Algorithm for Constrained and Multiobjective Optimization. In J. T. Alander, editor, *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, pages 49–62, Vaasa, Finland, August 1997. University of Vaasa.

[2] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making: Theory and Methodology*. Systems Science and Engineering. North-Holland, 1983.

[3] C. A. Coello, A. D. Christiansen, and A. H. Aguirre. Automated Design of Combinational Logic Circuits using Genetic Algorithms. In D. G. Smith, N. C. Steele, and R. F. Albrecht, editors, *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 335–338. Springer-Verlag, University of East Anglia, England, April 1997.

[4] C. A. Coello, A. D. Christiansen, and A. H. Aguirre. Use of Evolutionary Techniques to Automate the Design of Combinational Circuits. *International Journal of Smart Engineering System Design*, 2000. (in press).

[5] C. A. C. Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.

[6] C. A. C. Coello. Treating Constraints as Objectives for Single-Objective Evolutionary Optimization. *Engineering Optimization*, 32(3):275–308, 2000.

[7] K. Deb and D. E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.

[8] C. M. Fonseca and P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[9] C. M. Fonseca and P. J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[10] A. Hernández-Aguirre, C. A. Coello-Coello, and B. P. Buckles. A Genetic Programming Approach to Logic Function Synthesis by means of Multiplexers. In D. K. A. Stoica and J. Lohn, editors, *The First NASA/DoD Workshop on Evolvable Hardware*, pages 46–53. IEEE Computer Society, 1999.

[11] F. Jiménez and J. L. Verdegay. Evolutionary techniques for constrained optimization problems. In *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, Aachen, Germany, 1999. Springer-Verlag.

[12] T. Kalganova and J. Miller. Evolving more efficient digital circuits by allowing circuit layout and multi-objective fitness. In A. Stoica, D. Keymeulen, and J. Lohn, editors, *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, pages 54–63, Los Alamitos, California, 1999. IEEE Computer Society Press.

[13] T. Kalganova, J. Miller, and T. Fogarty. Some Aspects of an Evolvable Hardware for Multiple-Valued Combinational Circuit Design. In M. Sipper, D. Mange, and A. Pérez-Uribe, editors, *Proceedings of the Second International Conference on Evolvable Systems (ICES'98)*, pages 78–89, Lausanne, Switzerland, 1998. Springer-Verlag.

[14] M. Karnaugh. A Map Method for Synthesis of Combinational Logic Circuits. *Transactions of the AIEE, Communications and Electronics*, 72 (I):593–599, November 1953.

[15] E. J. McCluskey. Minimization of Boolean Functions. *Bell Systems Technical Journal*, 35 (5):1417–1444, November 1956.

[16] J. Miller, T. Kalganova, N. Lipnitskaya, and D. Job. The Genetic Algorithm as a Discovery Engine: Strange Circuits and New Principles. In *Proceedings of the AISB Symposium on Creative Evolutionary Systems (CES'99)*, Edinburgh, UK, 1999.

[17] J. F. Miller, P. Thomson, and T. Fogarty. Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*, pages 105–131. Wiley, Chichester, England, 1997.

[18] I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pages 97–102, Plymouth, UK, 1994. University of Plymouth.

[19] M. Potter and K. DeJong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[20] W. V. Quine. A Way to Simplify Truth Functions. *American Mathematical Monthly*, 62 (9):627–631, 1955.

[21] T. Sasao, editor. *Logic Synthesis and Optimization*. Kluwer Academic Press, 1993.

[22] J. D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[23] P. D. Surry and N. J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3), 1997.

[24] P. D. Surry, N. J. Radcliffe, and I. D. Boyd. A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method. In T. C. Fogarty, editor, *Evolutionary Computing. AISB Workshop. Selected Papers*, Lecture Notes in Computer Science, pages 166–180. Springer-Verlag, Sheffield, U.K., 1995.

[25] J. Torresen. A Divide-and-Conquer Approach to Evolvable Hardware. In M. Sipper, D. Mange, and A. Pérez-Uribe, editors, *Proceedings of the Second International Conference on Evolvable Systems (ICES'98)*, pages 57–65, Lausanne, Switzerland, 1998. Springer-Verlag.