Use of a Population-based Evolutionary Multiobjective Optimization Technique to Design Combinational Logic Circuits

Carlos A. Coello Coello[1] and Arturo Hernández Aguirre[2]

[1]CINVESTAV-IPN
Depto. de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. 07300, MEXICO
ccoello@cs.cinvestav.mx
[2]Department of Electrical Engineering and Computer Science
Tulane University
New Orleans, LA 70118, USA
hernanda@eecs.tulane.edu

Abstract. This paper introduces a population-based evolutionary multiobjective optimization approach to design combinational circuits. The approach divides the computational effort among several sub-populations, and then uses a selection mechanism that combines these partial results, reducing considerably the computational cost required by a genetic algorithm to produce compact designs.

1 Introduction

The genetic algorithm (GA) has been widely used for optimization tasks [2] and is known to be a very powerful tool in certain domains. The most interesting aspect of evolutionary design, however, is the possibility of studying its emergent patterns [5, 1]. The goals are, therefore, different when we design circuits using evolutionary algorithms. First, we aim to optimize circuits (using a certain metric) in a different way, and intuitively, we can think of producing novel designs (since there is no human intervention). Such novel designs have been shown in the past [5, 1]. Second, it would be extremely useful to extract design patterns from such evolutionary-generated solutions. This could lead to a practical design process in which a small (optimal) circuit is used as a building block to produce complex circuits. Such a divide-and-conquer approach has also been suggested in the past [10, 5]. However, in the previous work on evolutionary design of combinational circuits, efficiency has been an important issue. The main approaches reported so far in the literature require a significant amount of fitness function evaluations. The motivation of this work was precisely to conceive an approach that could reduce the amount of fitness function evaluations, while keeping the capabilities of a GA to generate novel (and compact) designs. This does not mean that we claim that our approach will solve the scalability problem that has characterized evolvable hardware [9, 5]. Nevertheless, we believe that approaches such as the one presented in this paper, may contribute to the development of alternative techniques that could improve the performance of a GA at least when solving relatively small circuits (under the assumption that they could be used as building blocks to produce larger circuits).

2 Related Work

Although there are several constraint-handling techniques based on evolutionary multiobjective optimization approaches (see for example [6]), 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. The idea was, however, suggested by Kalganova and Miller [3]. Nevertheless, evolutionary multiobjective optimization approaches have been used by several researchers to solve some other related problems. For example, Wilson and Macleod [11] used Pareto ranking [2] to design multiplierless IIR filters and Zebulum et al. [12] used a GA with a target vector approach (with adaptive weights) for the synthesis of low-power operational amplifiers.

3 The Genetic Algorithm Used

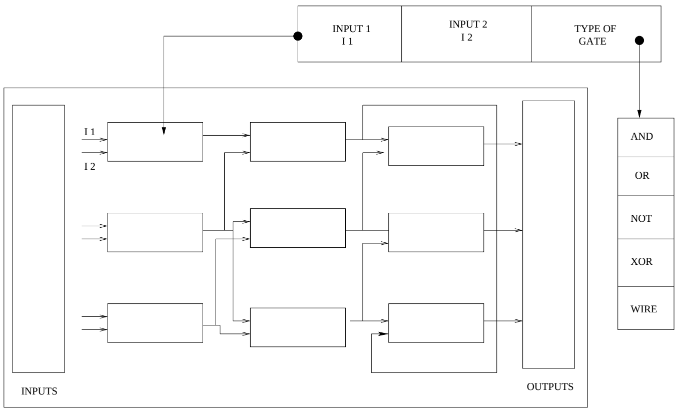

Fig. 1. Matrix used to represent a circuit. Each gate gets its inputs from either of the gates in the previous column. Note the encoding adopted for each element of the matrix as well as the set of available gates used.

We used a matrix to represent a circuit also adopted in previous work [1], as shown in Fig. 1. 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 (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. 1). 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[1].) that receives its 2 inputs from any gate at the previous column as shown in Fig. 1.

4 Description of the approach

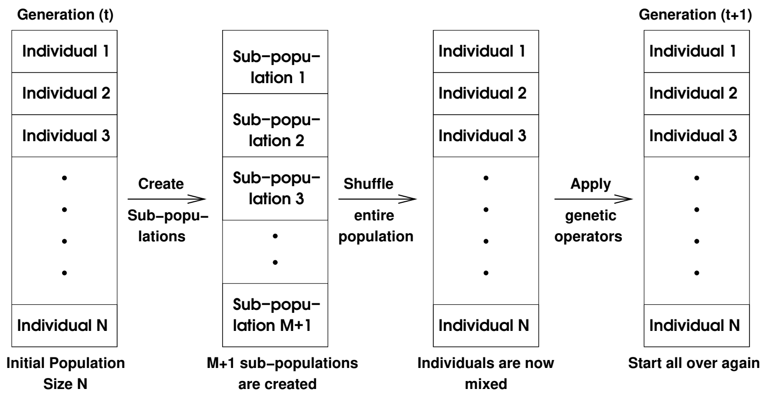

Fig. 2. Graphical representation of the approach proposed in this paper.

The main idea behind our proposed approach is to use a population-based multiobjective optimization technique such as VEGA [8] 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 single output, would have $m = 2^3 = 8$ values to match. The technique may be better illustrated by Fig. 2. 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 is on charge of optimizing a constraint of the problem (in this case, an output of the circuit). Therefore, the main goal of each sub-population is to match its corresponding output with the value indicated by the user in the truth table.

[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.

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 subpopulation sizes is also possible.

The objective function in our case 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., it 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. We use the following scheme:

$$\begin{array}{lll}\textbf{if } o_j(\boldsymbol{x}) \neq t_j & \textbf{then} & \text{fitness}(\boldsymbol{x}) = 0 \\ \textbf{else if } v \neq 0 & \textbf{then} & \text{fitness}(\boldsymbol{x}) = -v \\ \textbf{else} & & \text{fitness}(\boldsymbol{x}) = f(\boldsymbol{x})\end{array}$$

where $o_j(\boldsymbol{x})$ refers to the value of output j for the encoded circuit $\boldsymbol{x}$; 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 $\boldsymbol{x}$ ($\leq m$). Finally, $f(\boldsymbol{x})$ is the fitness function described before:

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

In this equation, $h(\boldsymbol{x})$ refers to the number of matches between the circuit $\boldsymbol{x}$ and the values defined in the truth table, and $w(\boldsymbol{x})$ is the number of WIREs in the circuit $\boldsymbol{x}$. Although VEGA is known to have difficulties in multiobjective optimization problems due to the fact that it tries to find individuals that excel only in one dimension regardless of the others, that drawback turns out to be an advantage in this context, because what we want to find are precisely circuits that are fully functional, instead of good compromises that may not satisfy one of the outputs (which are the kinds of solutions that a Pareto ranking strategy would normally produce).

5 Comparison of Results

We have used several circuits of different degrees of complexity to test our approach. For the purposes of this paper, three examples were chosen to illustrate our approach (called multiobjective genetic algorithm, or MGA for short), and the results produced were compared with those generated by human designers and by our previous n-cardinality GA (called NGA) [1]. Regarding the size of the matrix used, it was normally the case that for small circuits a matrix of

5×5 was sufficient. However, in some cases, it is necessary to reach larger matrix sizes (e.g., 6×7 was reached in our third example). We used an iterative approach to determine the proper matrix size in each case. To choose the size of each sub-population in the MGA, we started with 10, and performed 20 runs. If we did not find feasible solutions in at least one fourth of our runs, we would increase the sub-population size by 10 and would perform 20 more runs. This process was repeated until a suitable sub-population size was found. We used a crossover rate of 50% and a mutation rate such that each string had a 50% probability of being mutated at a certain position.

5.1 Example 1

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

Our first example has 4 inputs and one output, as shown in Table 1. A matrix of size 5×5 was used. The comparison of the results produced by the MGA, the NGA, a human designer using Karnaugh Maps, and Sasao's approach [7] are shown in Table 2.

Since this example has 16 outputs, there are 17 objectives for the MGA. A population size of 170 was enough to solve this circuit. The maximum number of generations in this case (for both the MGA and the NGA) was set to 400. The MGA found a solution with a fitness value of 34 (i.e., a circuit with 7 gates) 15% of the time, and solutions with 8 gates were found 25% of the time. The MGA produced feasible circuits 100% of the time. The average fitness of the 20 runs performed was 32.1, with a standard deviation of 1.252366. The best solution

MGA	NGA	HD 1	Sasao
$F = ((W + XY)$ $\oplus((X + Y)$ $(X \oplus Z)))'$	$F = (WYX'$ $\oplus((W + Y)$ $\oplus Z \oplus (X$ $+Y + Z)))'$	$F = ((Z'X)$ $\oplus(Y'W'))$ $+((X'Y)$ $(Z \oplus W'))$	$F = X' \oplus Y'W'$ $\oplus XY'Z'$ $\oplus X'Y'W$
7 gates	10 gates	11 gates	12 gates
2 ANDs, 2 ORs, 2 XORs, 1 NOT	2 ANDs, 3 ORs, 3 XORs, 2 NOTs	4 ANDs, 1 OR, 2 XORs, 4 NOTs	3 XORs, 5 ANDs, 4 NOTs

Table 2. Comparison of the best solutions found by the n-cardinality GA (**NGA**), our multiobjective genetic algorithm (**MGA**), a human designer using Karnaugh Maps (**HD 1**), and **Sasao** for the circuit of the first example. A population size of 170 was used with both GAs.

that the NGA could find using the same population size had a fitness of 31 (i.e., a circuit with 10 gates), and it appeared only once in the 20 runs performed. Also, 95% out of the 20 runs performed, the best solution found was infeasible. The average fitness of these 20 runs was 15.55, with a standard deviation of 3.677456. Our MGA only required 68,000 evaluations of the fitness function to reach the best known solution to this circuit. In contrast, the NGA required 500,000 fitness function evaluations to reach equivalent results.

5.2 Example 2

A_1	A_0	B_1	B_0	C_3	C_2	C_1	C_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Table 3. Truth table for the 2-bit multiplier of the second example.

Our second example has 4 inputs and 4 outputs, as shown in Table 3. A matrix of the same size as before was used (i.e., 5×5). Since this example has 64 outputs, there are 65 objectives for the MGA. A sub-population size of 10 (i.e., total population size of 650), was sufficient for the MGA. The maximum number of generations in this case (for both the MGA and the NGA) was set to 500. The MGA found a solution with a fitness value of 82 (i.e., a circuit with 7 gates) 15% of the time, and it produced feasible circuits 100% of the time. The average fitness of the 20 runs performed was 80.4, with a standard deviation of 1.142481141. On the other hand, the best solution that the NGA could find using the same population size of 650 had a fitness of 80 (i.e., a circuit with 9 gates). This solution appeared only twice in the 20 runs performed. In most cases (70% out of the 20 runs performed), the best solution found was infeasible. The average fitness of these 20 runs was 66.65, with a standard deviation of 7.638372657.

MGA	**NGA**	**HD 1**	**HD 2**	**MIL**
$C_0 = A_0B_0$	$C_0 = A_0B_0$	$C_0 = A_0B_0$	$C_0 = A_0B_0$	$C_0 = A_0B_0$
$C_1 = A_0B_1$ $\oplus A_1B_0$	$C_1 = A_1A_0$ B_0B_1 $\oplus(A_0B_1 + A_1B_0)$	$C_1 = A_0B_1$ $\oplus A_1B_0$	$C_1 = (B_1 + B_0)$ $(A_1 + A_0)$ $((A_1A_0) \oplus (B_1B_0))$	$C_1 = A_1B_0$ $\oplus A_0B_1$
$C_2 = A_1B_1$ $\oplus(A_0B_0A_1B_1)$	$C_2 = (A_0B_0$ $+A_1B_1) \oplus A_0B_0$	$C_2 = A_1B_1$ $(A_0B_0)'$	$C_2 = A_1B_1$ $(A_0B_0)'$	$C_2 = (A_0B_0)'$ (A_1B_1)
$C_3 = A_0B_0$ A_1B_1	$C_3 = A_1B_1$ A_0B_0	$C_3 = A_1A_0$ B_1B_0	$C_3 = A_1B_1$ A_0B_0	$C_3 = (A_1B_0$ $\oplus A_0B_1)'(A_1B_0)$
7 gates	9 gates	8 gates	12 gates	9 gates
5 ANDs, 2 XORs	5 ANDs, 2 ORs, 2 XORs	6 ANDs, 1 XOR, 1 NOT	8 ANDs, 2 ORs, 1 XOR, 1 NOT	6 ANDs, 1 XOR, 2 NOTs

Table 4. Comparison of the best solutions found by the n-cardinality GA (NGA), our multiobjective genetic algorithm (MGA), two human designers (**HD 1** & **HD 2**), and Miller et al. (**MIL**) for the circuit of the second example. A population size of 650 was used with both the MGA and the NGA.

The comparison of the results produced by the MGA, the NGA, two human designers, and Miller et al. [4] are shown in Table 4. It should be mentioned that Miller et al. [4] considered their solution to contain only 7 gates because of the way in which they encoded their Boolean functions (the reason is that they encoded NAND gates in their representation). However, since we considered each gate as a separate chromosomic element, we count each of them, including NOTs that are associated with AND & OR gates. Regardless of that fact, it is more important to point out that Miller et al. [4] found their solution with runs of 3,000,000 fitness function evaluations each, whereas in our case, we performed runs of only 325,000 fitness function evaluations each. In contrast, our NGA required 600,000 fitness function evaluations to reach equivalent results.

5.3 Example 3

A	B	C	D	F1	F2	F3
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

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

Our third example has 4 inputs and 3 outputs, as shown in Table 5. In this case, the matrix used was of size 6×7, and the chromosomic length was 126 ($r = 6, q = 7, t = 6 \times 7 = 42, l = 3 \times t = 126$). The cardinality $c = max(r, g) = 6$ The size of the intrinsic search space for this problem is $c^l = 6^{126} \approx 1.1 \times 10^{98}$. The comparison of the results produced by the MGA, the NGA, and two human designers are shown in Table 6. Since this example has 48 outputs, there are 49 objectives for the MGA. A sub-population size of 10 (i.e., total population size of 490), was sufficient for the MGA. The maximum number of generations in this case (for both the MGA and the NGA) was set to 2000. The MGA found a solution with a fitness value of 81 (i.e., a circuit with 9 gates) 15% out of the 20 runs performed, and it produced feasible circuits 100% out of the 20 runs performed (in 55% of the runs, the MGA found better solution than the best found by the NGA). The average fitness of the 20 runs performed was 78.9, with a standard deviation of 1.020835571. On the other hand, the best solution that the NGA could find using the same population size of 490 individuals had a fitness of 78 (i.e., a circuit with 12 gates). This solution appeared only once in the 20 runs performed. In most cases (80% of the runs performed), the best solution found was infeasible. The average fitness of these 20 runs was 52.15, with a standard deviation of 11.92641915. Our MGA required 980,000 evaluations of the fitness function to reach the best known solution to this circuit. In contrast, the NGA required 5,600,000 fitness function evaluations to reach equivalent results.

MGA	**NGA**	**HD 1**	**HD 2**
$F1 = ((B \oplus D)$ $+(A \oplus C))'$	$F1 = ((B \oplus D)$ $+(A \oplus C))'$	$F1 = (A \oplus C)'$ $(B \oplus D)'$	$F1 = (A \oplus C)'$ $(B \oplus D)'$
$F3 = ((B \oplus D)$ $+(A \oplus C))$ $(((A \oplus C)$ $+(A \oplus B)) \oplus C))$	$F3 = ((B \oplus D)$ $+(A \oplus C))$ $((D + (A \oplus$ $C))' + (A' + C)')$	$F3 = BD'(A$ $+ C')$ $+AC'$	$F3 = (F1 + F2)'$
$F2 = F3 \oplus ((B \oplus D)$ $+(A \oplus C))$	$F2 = ((B \oplus D) + (A$ $\oplus C)) \oplus ((B \oplus D) + (A$ $\oplus C))((D + (A \oplus C))'$ $+(A' + C)')$	$F2 = B'D(A' + C)$ $+A'C$	$F2 = A'C+$ $(A \oplus C)'(B'D)$
9 gates	12 gates	19 gates	13 gates
3 XORs, 3 ORs, 2 ANDs, 2 NOTs	3 XORs, 4 ORs, 1 AND, 4 NOTs	2 XORs, 4 ORs, 7 ANDs, 6 NOTs	2 XORs, 2 ORs, 4 ANDs, 5 NOTs

Table 6. Comparison of the best solutions found by the n-cardinality GA (**NGA**), our multiobjective genetic algorithm (**MGA**), and two human designers (**HD 1** and **HD 2**) for the circuit of the third example. A population size of 490 was used with both GAs.

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 two human designers, and a previous GA developed by us which uses and n-cardinality alphabet and a two-stage fitness function. Our approach, called MGA, consistently found better solutions than the human designers, and was able to find the same or even better solutions than our previous GA (called NGA), using a lower number of fitness function evaluations. The proposed approach seems very suitable for parallelization, and that will probably be a path of research that we will explore in the near future. Also, we are interested in coupling this approach with another system based on genetic programming that we have recently developed. We aim to benefit from a more powerful chromosomic representation while keeping an efficient selection mechanism.

Acknowledgements

The first author acknowledges partial support from CINVESTAV through project JIRA'2001/08, and from CONACyT through project 34201-A. The second author acknowledges support for this work in part by DoD EPSCoR and the Board of Regents of the State of Louisiana under grant F49620-98-1-0351, and from the NASA Jet Propulsion Laboratory, under contract No. 1230282.

References

1. Carlos A. Coello Coello, Alan D. Christiansen, and Arturo Hernández Aguirre. Use of Evolutionary Techniques to Automate the Design of Combinational Circuits. *International Journal of Smart Engineering System Design*, 2(4):299–314, June 2000.
2. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing, Reading, Massachusetts, 1989.
3. Tatiana Kalganova and Julian Miller. Evolving more efficient digital circuits by allowing circuit layout and multi-objective fitness. In Adrian Stoica, Didier Keymeulen, and Jason Lohn, editors, *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, pages 54–63, Los Alamitos, California, 1999. IEEE Computer Society Press.
4. 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.
5. Julian F. Miller, Dominic Job, and Vesselin K. Vassilev. Principles in the Evolutionary Design of Digital Circuits—Part I. *Genetic Programming and Evolvable Machines*, 1(1/2):7–35, April 2000.
6. 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.
7. Tsutomu Sasao, editor. *Logic Synthesis and Optimization*. Kluwer Academic Press, 1993.
8. J. David 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.
9. Adrian Thompson, Paul Layzell, and Ricardo Salem Zebulum. Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution. *IEEE Transactions on Evolutionary Computation*, 3(3):167–196, September 1999.
10. Jim Torresen. A Divide-and-Conquer Approach to Evolvable Hardware. In Moshe Sipper, Daniel Mange, and Andrés Pérez-Uribe, editors, *Proceedings of the Second International Conference on Evolvable Systems (ICES'98)*, pages 57–65, Lausanne, Switzerland, 1998. Springer-Verlag.
11. P. B. Wilson and M. D. Macleod. Low implementation cost IIR digital filter design using genetic algorithms. In *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, pages 4/1–4/8, Chelmsford, U.K., 1993.
12. R. S. Zebulum, M. A. Pacheco, and M. Vellasco. A multi-objective optimisation methodology applied to the synthesis of low-power operational amplifiers. In Ivan Jorge Cheuri and Carlos Alberto dos Reis Filho, editors, *Proceedings of the XIII International Conference in Microelectronics and Packaging*, volume 1, pages 264–271, Curitiba, Brazil, August 1998.