International Journal of Electronics
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713599654

Using a genetic algorithm for optimizing fixed polarity Reed-Muller expansions of boolean functions

Julian F. Millert [a]; Henri Luchian [b]; Peter V. G. Bradbeeru [c]; Peter J. Barclay [c]
[a] Department of Electrical, Electronic and Computer Engineering, Napier University, Edinburgh, U.K
[b] Faculty of Computer Science, 'Al. I. Cuza' University, Iasi, Roumania
[c] Department of Computer Studies, Napier University, Edinburgh, U.K

Online Publication Date: 01 April 1994
To cite this Article: Millert, Julian F., Luchian, Henri, Bradbeeru, Peter V. G. and Barclay, Peter J. (1994) 'Using a genetic algorithm for optimizing fixed polarity Reed-Muller expansions of boolean functions', International Journal of Electronics, 76:4, 601 - 609
To link to this article: DOI: 10.1080/00207219408925956
URL: http://dx.doi.org/10.1080/00207219408925956

INT. J. ELECTRONICS, 1994, VOL. 76, NO. 4, 601-609

Using a genetic algorithm for optimizing fixed polarity Reed-Muller expansions of boolean functions

JULIAN F. MILLER†, HENRI LUCHIAN‡§,
PETER V. G. BRADBEER‖ and PETER J. BARCLAY‖

The use of a genetic algorithm is presented which determines good sub-optimum fixed polarity Reed-Muller expansions of completely specified boolean functions. The algorithm performs better than previous techniques which find a good fixed polarity by non-exhaustive search.

1. Introduction

It is now well known that boolean logic functions may be expressed in terms of boolean algebra (inclusive-OR and AND gates), or in terms of modulo-2 algebra (exclusive-OR gates and AND gates) (Green 1986, Almaini 1989). The latter is more commonly known as Reed-Muller algebra.

Implementation of logic functions in Reed-Muller algebra can be advantageous for two reasons. Firstly, Reed-Muller circuits are readily testable (Reddy 1972). The trend towards increasing circuit complexity has now made the testability of circuits an important issue. Secondly, it is often the case that logic functions which do not minimize well in the boolean domain can be reduced substantially if implemented in Reed-Muller logic.

Reed-Muller logic functions may be expressed in two main ways. Firstly there are mixed-polarity expansions which allow logic variables to be complemented and uncomplemented in the same expansion. Secondly there are fixed-polarity or canonical (RMC) expansions in which logic variables must be either complemented or uncomplemented throughout the expansion. In this paper we deal only with the latter. The techniques for conversion of boolean functions in minterm form are now well established (Tran 1987, Almaini *et al.* 1991); in addition, many papers have been written on the minimization of RMC expansions (Even *et al.* 1967, Bioul *et al.* 1973, Wu *et al.* 1982, Besslich 1983, Zhang and Rayner 1984, Green 1987). A number of exhaustive search techniques have been devised (Besslich 1983, Harking 1990, Almaini *et al.* 1991, Lui and Muzio 1991, Miller and Thomson 1994). The most efficient exhaustive searches have time complexities of $O[3^n]$ and are presently impractical for more than 14 variables. In response to this, there have been a number of algorithms developed that find a sub-optimum polarity in a much reduced time (Wu *et al.* 1982, Habib 1990, Almaini *et al.* 1991, McKenzie *et al.* 1993). McKenzie *et al.* (1993) compared the performances of a number of algorithms with each other

Received 6 September 1993; accepted 6 December 1993.

†Department of Electrical, Electronic and Computer Engineering, Napier University, 219 Colinton Road, Edinburgh, EH14 1DJ, U.K.

‡Faculty of Computer Science, 'Al. I. Cuza' University, Iasi, Roumania.

§At present visiting the Department of Computer Studies, Napier University, 219 Colinton Road, Edinburgh, EH14 1DJ, U.K.

‖Department of Computer Studies, Napier University, 219 Colinton Road, Edinburgh, EH14 1DJ, U.K.

and found an original algorithm (referred to as the Gains algorithm) to be the most effective. The testing of algorithm performance in this area is often carried out on bench-mark examples (see, for example, Sarabi and Perkowski 1992). These bench-mark examples form a minute fraction of available logic functions. Another method of evaluating algorithm performance is to carry out an exhaustive search for collections of randomly chosen logic functions and then to grade the sub-optimum algorithms accordingly. This method is only feasible for up to 12 variables at present. However, this is sufficient to give a reliable indication of comparative algorithm performance for much larger numbers of variables, since performance in general decreases smoothly and with the same power dependence with the number of variables for all available algorithms.

2. Genetic algorithms

Genetic algorithms (GAs) present a non-deterministic approach to problem-solving; but they are by no means a random search. The approach is based on Holland's pioneering study of adaptation in natural and artificial systems (Holland 1975).

The basic technique is to create a population (breeding pool) of potential solutions to a problem. These solutions are encoded as 'chromosomes' (abstract representations of the solution), and each chromosome is subjected to an evaluation function that assigns it a 'fitness' depending on how well the solution it encodes solves the problem at hand.

Existing solutions are recombined by a process called 'cross-over' or 'breeding'; the rationale for this is that good solutions will contain good building-blocks (partial solutions), re-arrangement of which may produce even better solutions. Further, a 'mutation' process makes random changes in a few randomly selected chromosomes, preventing premature convergence by maintaining the diversity of the population. It is also possible that these operations may produce worse solutions; however, the population is iterated through generations, each individual's survival depending on its fitness; hence, the best 'genetic material' tends to be preserved, and the average fitness of the population increases over time.

In the classical form of GAs, chromosomes are bitstrings, and crossover and mutation are cut-and-splice and bit-flipping operations, respectively; however, many non-classical representations and operations are now in widespread use.

GAs seem to provide a highly effective balance between exploitation (using known good partial solutions) and exploration (trying new solutions). Hence, the evolution of better solutions to the problem by the 'artificial selection' of the genetic algorithm models the way in which organisms better fitted to their environment evolve through the processes of (biological) genetics under natural selection.

Genetic algorithms have now been successfully applied to a large number of varied applications. Good introductions to genetic algorithms may be found in the work of Goldberg (1989), Davis (1991) and Michaelowicz (1992); current research is documented in the *Proceedings of the Foundations of Genetic Algorithms* and the *Proceedings of the International Conference on Genetic Algorithms.*

3. Genetic algorithm for the polarity problem

The fixed polarity problem lends itself to a straightforward representation by means of genetic algorithm techniques. Indeed, as required for a classical genetic

algorithm approach, a polarity is easily encoded by a string of bits (genes) of length n, where n is the number of variables (the meaning attached to a chromosome $b_n b_{n-1} \ldots b_1$ is that, in the corresponding polarity, variable x_i will be complemented or not depending on whether b_i is 1 or 0). Such a chromosome is evaluated by an algorithm that calculates, for a given function and a given polarity, the number of XOR operations needed to express that function in modulo-2 algebra.

Since the genetic algorithm only evaluates the fitness of a fixed number of polarities, the time complexity is $O[2^n]$ (the evaluation algorithm actually used is that previously devised by Almaini *et al.* 1991). For an n variable function, the search space (the set of all possible polarities) has 2^n elements.

For any optimization problem, there is a suitable evaluation function that shows how good a candidate solution is; the actual solution is the best candidate solution with respect to this evaluation (fitness) function. In general, a genetic algorithm starts with an initial sample population or breeding pool of chromosomes (candidate solutions); the chromosomes are individually evaluated using the fitness function. Next, a selection mechanism is applied to the breeding pool (classically a roulette-wheel-like scheme, that gives a proportionately greater chance of being selected to better fitted chromosomes—see Goldberg 1989). The chromosomes that were selected survive in the next generation (of candidate solutions). Now the genetic operators crossover and mutation are applied as defined below.

3.1. *Crossover*

One application of the classical crossover operator takes two randomly selected chromosomes (parents) and creates two offspring in the following way:

(*a*) a number k between 1 and $n-1$ is randomly generated;

(*b*) the parents are cut after the kth bit;

(*c*) then the first chunk from the first parent is paired with the second chunk of the second parent and vice versa. This is illustrated in Fig. 1.

Thus two offspring are obtained which, after all crossover operations have been completed, replace the parents in the breeding pool.

3.2. *Mutation*

The most common mutation operator randomly generates a few numbers which represent positions of genes (bits) in the entire breeding pool; these selected genes are then complemented (as logical values).

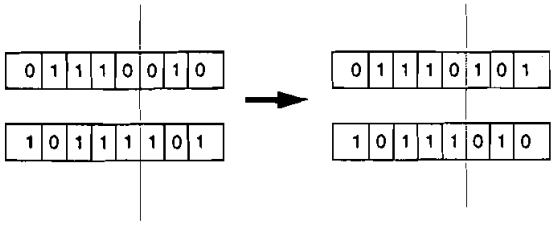

Figure 1. One-point crossover.

After completion of the genetic operators, a new generation (breeding pool) has emerged and the cycle is iterated. The general scheme of a genetic algorithm is presented below.

Step 1. Generate the initial breeding pool;

Step 2. evaluate each chromosome;

Step 3. number_of_generations: = 1;

Step 4. while (maximum_number_of_generations not exceeded) do

```
begin
    select a new breeding pool;
    apply crossover;
    apply mutation;
    evaluate the new chromosomes;
    number_of_generations: = number_of_generations + 1
end.
```

The best chromosome in the final generation is the solution given by the genetic algorithm to the optimization problem being undertaken; fundamental theoretical results state that better fitted chromosomes have exponentially greater chances of survival (Goldberg 1989, Michaelowicz 1992, Radcliffe 1990, 1991, 1992).

Many variations of the above-stated features of genetic algorithms have been developed. In one such variation, elitism, the genetic algorithm promotes to the next generation the best chromosome it actually found in the current generation. It is also worth noting that a genetic algorithm has many parameters to be fine-tuned for the problem at hand. These include

(*a*) the breeding pool size;
(*b*) the mutation rate: the percentage of genes in the entire breeding pool that will be complemented; and
(*c*) the crossover rate: the number of chromosomes that will breed in one generation.

The genetic algorithm we built for the fixed polarity problem is a classical one: the selection mechanism is based upon the classical roulette-wheel approach; one-point crossover is used; mutation is performed at the gene level rather than at the chromosome level. The algorithm uses elitism.

3.3. *Results*

In our experiments with the genetic algorithm we have also separately run an exhaustive search algorithm (Miller and Thomson 1994) and also the Tabular and Gains algorithms (Almaini *et al.* 1991, McKenzie *et al.* 1993). We could thus compare relative performances (Figs 2-4).

In early experiments with the GA, we included the solution predicted by the Tabular algorithm as individuals in the breeding pool. The inclusion of existing techniques is known as a hybrid algorithm, and from our discussion of the experimental results it can be seen that the hybrid did not perform better for larger numbers of variables. The GA is therefore not critically sensitive to the contents of the initial breeding pool.

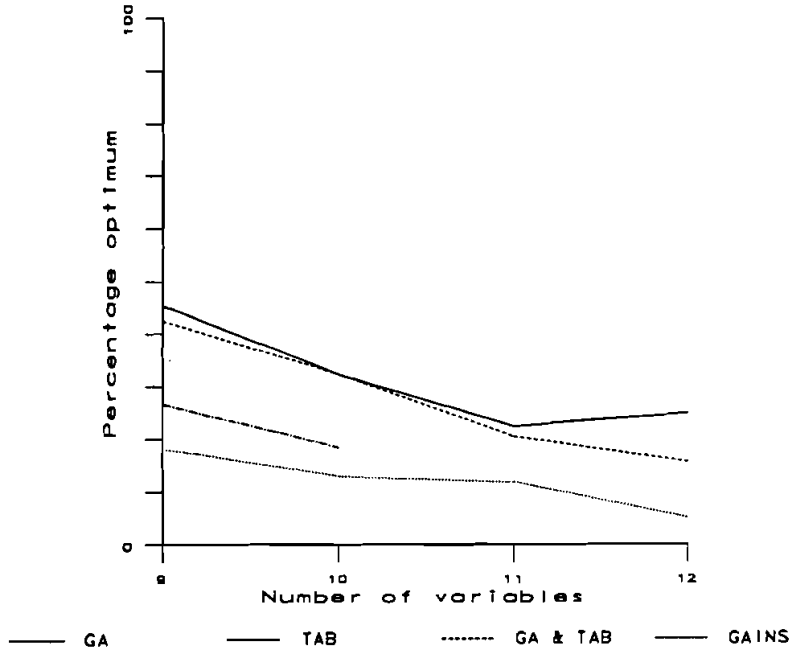

Figure 2. Comparative percentages of cases where method finds an optimum polarity.

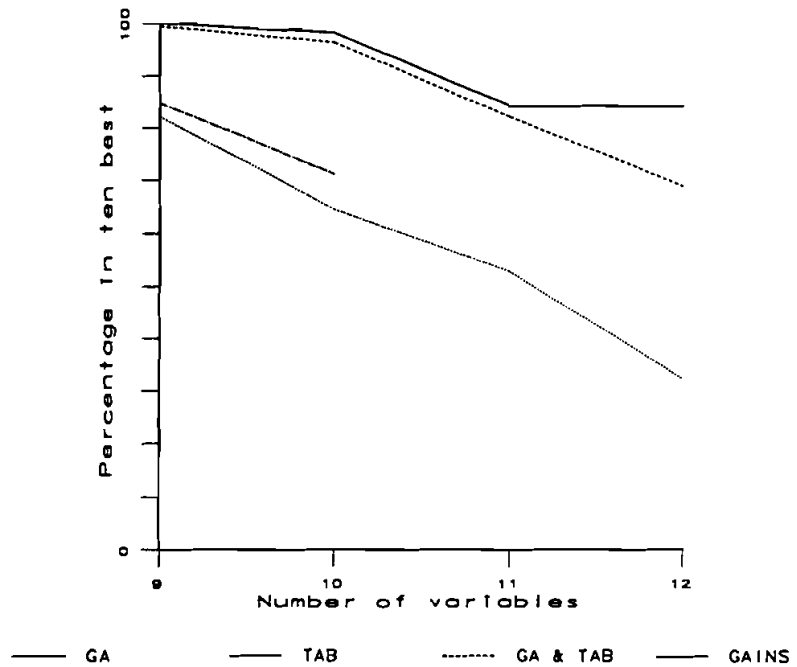

Figure 3. Comparative percentage of cases where method finds a polarity in the first ten optimum polarities.

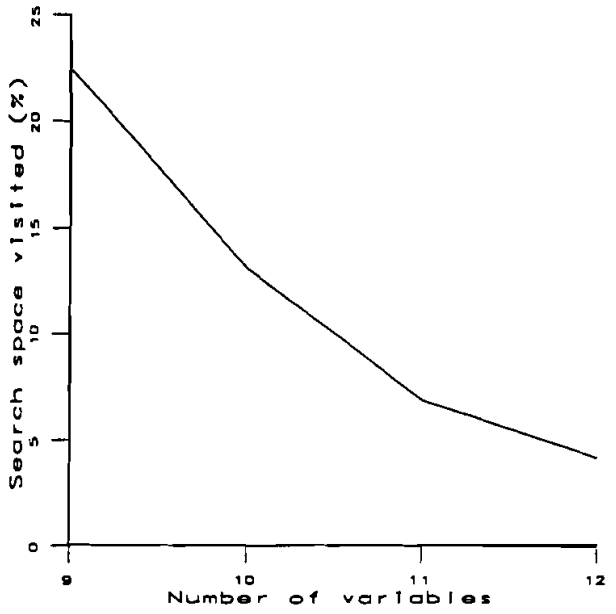

Figure 4. Percentage of search space visited.

As regards the parameters, our experimental results have shown that small breeding pool sizes (6 to 10 chromosomes) give very good results (even for more than 10 variables): in many cases, the global best was found in less than 20 generations; with few exceptions, a better solution than the Tabular and Gains algorithms was found. The genetic algorithm performed equally well with breeding pool sizes of six or much greater. It turned out that a mutation rate of 6% and a crossover rate of 60% gave good experimental results over a wide range of functions of various variables and minterms.

4. Algorithm performance

The performance of the genetic algorithm is presented in Figs 2 and 3. It is clear that for larger numbers of variables the pure GA performs at least as well as the 'hybrid GA' (GA with the initial breeding pool seeded with the Tabular method prediction). This may be due to the Tabular method directing the GA to local maxima. Both GAs perform considerably better than the Tabular method or the Gains method. The steepness of the decline in performance with the number of variables can be altered by changing the number of GA evaluations of each function and the number of generations. The gradient reduces with increasing values of both these quantities but with an increased time penalty.

The figures from which the graphs are drawn are based on testing the algorithm performances for three sets of 200 random functions with minterm numbers given by 1/4, 1/2 and 3/4 of those available (MacKenzie *et al.* 1993). The average performance is then computed. The GA is performed three times for each function and the best polaritiy chosen overall. Data was collated to ascertain the number of polarity evaluations carried out by the GA averaged over three runs. This data was then used

to show how the percentage of the total search space visited by the GA varied with the number of variables (Fig. 4).

In addition, sets of random polarities were generated with numbers of elements equal to the average number of polarity evaluations carried out by the GA over three runs. The best polarities were selected from these sets for 200 random functions and the results plotted as a reference (Figs 5 and 6). It is clear from this that for the same number of polarity evaluations the GA performs considerably better than if the polarities had been selected at random. This demonstrates very clearly how the GA 'breeds' much fitter polarities.

5. Conclusions

A genetic algorithm has been developed to obtain a good sub-optimum polarity for minimizing RMC expansions of boolean functions. This, to the authors' knowledge, is one of the first times that GAs have been used in this area of electronics. The results presented show the GA to perform very well.

In comparison to earlier deterministic techniques that always produce the same result irrespective of fitness, this technique is particularly useful in that one can obtain different results each time the GA is run, thus the potential for improvement is great. In addition, better results might be obtained if the number of generations for which the GA is allowed to run is increased. The performance of the genetic algorithm depends on the initial parameters used; although hard to predict, this dependence does not seem to be critical within broad ranges. Ongoing experiments suggest that good choices of parameters improve the performance of the GA, and that further substantial improvements may be obtained by the use of non-classical operators, based on logical operations. This is still currently under investigation.

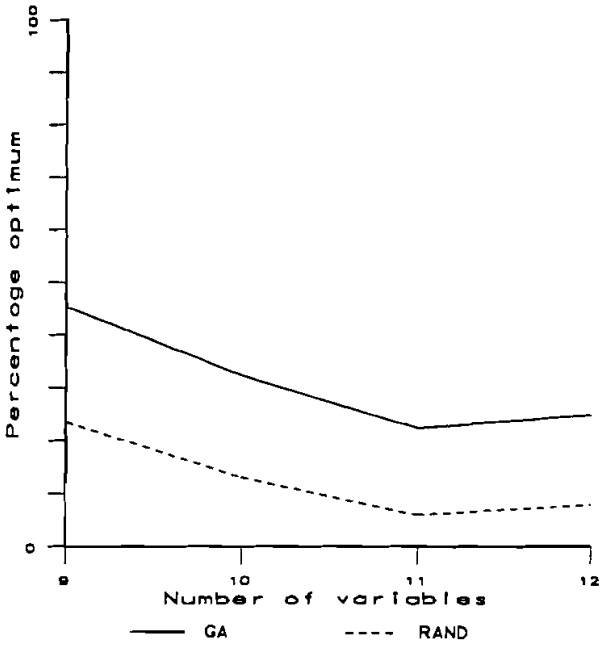

Figure 5. Comparative percentages for genetic and random searches (optimum polarity).

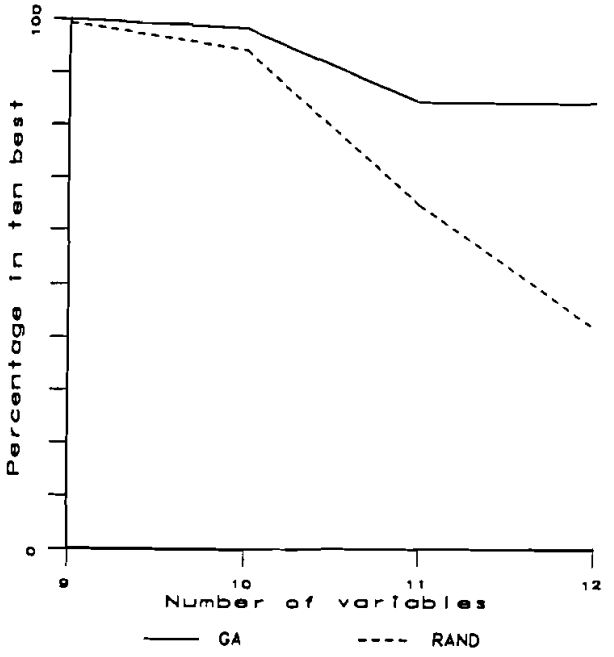

Figure 6. Comparative percentages for genetic and random searches (ten best).

Further results relating to technical aspects of the GA will be reported in a GA specialist publication. Work is now underway into the application of GAs to other computationally inaccessible problems in electronics.

However, irrespective of such improvements, the limit of the GA time performance is still set by the evaluation function, which is of time complexity $O(2^n)$. Hence it is hoped that further, more dramatic improvements may be gained by evaluating all the chromosomes in a population in parallel, perhaps delegating this task to custom hardware. This is currently under investigation.

Another advantage of GAs is that the basic algorithm is not problem-specific, and technical knowledge of the function to be optimized only enters in the chromosome's encoding stage and in the chromosome-fitness evaluation. It should be noted, however, that such knowledge is decisive in designing an effective GA application.

Acknowledgments

The authors wish to acknowledge the contributions of Peter Thomson of the Electrical, Electronic and Computer Engineering Department, Napier University, Edinburgh, U.K., Ben Paechter of the Department of Computer Studies, Napier University, Edinburgh, U.K., and Laurent Faedda of the Faculty of Computer Science, I.U.T. d'Orsay, Paris, France.

Henri Luchian wishes to acknowledge the support of E.C.C. Grant PECO 8335.

REFRENCES

ALMAINI, A. E. A., 1989, *Electronic Logic Systems*, second edition (Englewood Cliffs, NJ: Prentice-Hall).

ALMAINI, A. E. A., Thomson, P., and HANSON, D., 1991, Tabular techniques for Reed-Muller logic. *International Journal of Electronics*, **70,** 23-34.

BESSLICH, PH. W., 1983, Efficient computer method for EX-OR logic design. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **130,** 203-206.

BIOUL, G., DAVIO, M., and DESCHAMPS, J., 1973, Minimisation of ring-sum expansions of boolean functions. Phillips Research Reports 28, pp. 17-36.

DAVIS, L., (editor), 1991, *Handbook of Genetic Algorithms* (Van Nostrand Reinhold).

EVEN, S., KOHAVI, I., and PAZ, A., 1967, On minimal modulo-2 sums of products for switching functions. *IEEE Transactions on Computers*, **16,** 671-674.

GOLDBERG, D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning* (Reading, Mass: Addison Wesley).

GREEN, D. H., 1986, *Modern Logic Design* (Wokingham, U.K.: Addison-Weslcy); 1987, Reed-Muller expansions of incompletely specified functions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **134,** 228-236.

HABIB, M. K., 1990, Boolean matrix representation for the conversion of minterms to Reed-Muller coefficients and the minimisation of EX-OR switching functions. *International Journal of Electronics*, **68,** 493-506.

HARKING, B., 1990, Efficient algorithm for canonical Reed-Muller expansion of boolean functions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **137,** 366-370.

HOLLAND, J. H., 1975, *Adaptation in Natural and Artificial Systems* (Ann Arbor, Michigan: University of Michigan Press).

LUI, P. K., and MUZIO, J. C., 1991, Boolean matrix transforms for the parity spectrum and minimisation of modulo-2 canonical expansions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **138,** 411-417.

MCKENZIE, L., ALMAINI, A. E. A., MILLER, J. F., and THOMSON, P., 1993, Optimization of Reed-Muller logic functions. *International Journal of Electronics*, **75,** 451-466.

MICHAELOWICZ, Z., 1992, *Genetic Algorithms + Data Structures = Evolutionary Programs* (Berlin: Springer-Verlag).

MILLER, J. F., and THOMSON, P., 1994, A highly efficient exhaustive search algorithm for optimising canonical Reed-Muller expansions of boolean functions. *International Journal of Electronics*, **76,** 37-56.

RADCLIFFE, N. J., 1990, Equivalence class analysis of genetic algorithms. Edinburgh Parallel Computing Centre, Report TR90-03; 1991, Forma analysis and random respectful recombination. Edinburgh Parallel Computing Centre, Report TR91-02; 1992, The algebra of genetic algorithms. Edinburgh Parallel Computing Centre, Report TR92-11.

REDDY, S. M., 1972, Easily testable realisations for logic functions. *IEEE Transactions on Computers*, **21,** 1183-1188.

SARABI, A., and PERKOWSKI, M. A., 1992, Fast exact and quasi-minimal minimization of highly testable fixed polarity AND/XOR canonical networks. *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 30-35.

TRAN, A., 1987, Graphical method for the conversion of minterms to Reed-Muller coefficients and the minimisation of EX-OR switching functions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **134,** 93-99.

WU, X., CHEN, X., and HURST, S. L., 1982, Mapping of Reed-Muller coefficients and the minimisation of EX-OR switching functions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **129,** 15-20.

ZHANG, Y. Z., and RAYNER, P. J. W., 1984, Minimisation of Reed-Muller polynomials with fixed polarity. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **131,** 177-186.