Combinational and Sequential Logic Optimisation using Genetic Algorithms.

Julian F. Miller and Peter Thomson

EECE Department, Napier University,
219 Colinton Road,
Edinburgh EH14 1DJ.
e-Mail: j.miller@csu.napier.ac.uk.

1. Introduction.

This paper reviews our recent work in logic optimisation using Genetic Algorithms (GAs). We have applied GAs to optimisation of combinational and sequential logic, namely: the minimisation of fixed polarity Reed-Muller (RM) expansions, the minimisation of exclusive-OR sum-of-products (ESOP) expansions, and the determination of optimal state assignments for finite or algorithmic state machines (FSMs and ASMs). We present a brief discussion of each of these problems, the manner in which we have employed GAs to tackle them, and comparisons with heuristic techniques.

2. Reed-Muller Fixed Polarity and ESOP Expansions.

Reed-Muller logic is an alternative to the traditional Boolean logic form. In Reed-Muller, the exclusive-OR (XOR) operator is used in place of the inclusive-OR of Boolean. This allows many alternative expansions of the logic function which may assist minimisation due to cancellation of product terms. The most general expansions which are possible are known as exclusive-OR sum-of products (ESOPs). It has been shown that, on average, Reed-Muller circuits are more compact than their Boolean counterparts [1]. In addition, Reed-Muller circuits are more easily tested [2]. A subset of the general Reed-Muller form is the set of so-called **fixed polarity** expansions. For a logic function with *n* input variables, there are 2^n possible fixed polarity expansions, each incurring its own cost in terms of gates (both XOR and ANDs). For *n*>=14, it is impractical to carry out an exhaustive search in order to determine the most economical expansion [3]. Consequently, much work has been carried out in developing heuristic methods to determine a favourable fixed polarity [4][5][6].

A polarity, *p*, is defined as a non-negative integer less than 2^n whose binary digits p_k are defined:- $p_k = \delta(x_k', \overline{x_k})$ where we define $x_k' = x_k$ or $\overline{x_k}$ and $\delta(a,b) = 1$ if *a*=*b*, 0 otherwise. As an example consider the simple two minterm (0 and 1) function $f = \overline{x_1}.\overline{x_0} \oplus x_1.x_0$ (where $\oplus$ is the exclusive-OR operator). This simplifies to: $f = \overline{x_1} \oplus x_0$ in polarity 2.

A genetic algorithm was developed [7] which used a binary encoded chromosome to represent the polarity of the solution under consideration. The fitness of a chromosome was taken to be the number of two-input exclusive-OR gates which would be required to implement the minimised circuit. A classical windowed roulette wheel [8] was used to perform parent selection, and single-point crossover was used to generate child chromosomes.

The GA performed better than any of the previously devised fixed polarity hill-climbing heuristics (Gains and Tabular). This is illustrated in figure 2.1.

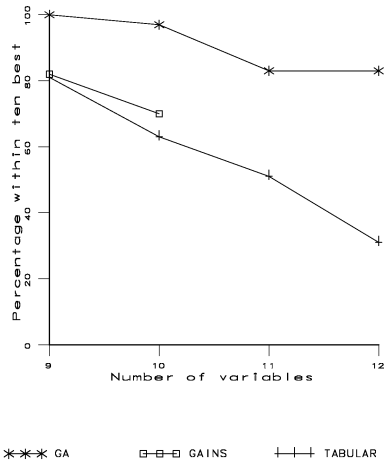

Figure 2.1 Comparative Performance of GA against Heuristic Algorithms.

Encouraged by the experience of using a GA in tackling Reed-Muller optimisation, and taking into account the drawbacks of fixed polarity - in terms of the limit it imposes upon the possible

number of expansions - we decided to widen the approach to encompass ESOP minimisation. Our first attempt used a binary chromosome once again to represent the expansion of a particular variable, however we removed the restriction that variables should be expanded throughout in the same polarity. A gene having the value 1 would cause the corresponding variable (within the sum-of-products) to be expanded according to the RM identity $x_n' = \overline{x_n'} \oplus 1$, and if the gene was zero then the variable would not be expanded. The resulting expansion was then simplified by cancellation of identical terms where they arose. This technique proved fairly unsuccessful due to the fact that there are a very large number of possible expansions ($2^{n \times m}$ where n is the number of input variables, and m is the number of minterms in the function). The number of expansions, therefore, becomes very large even for fairly trivial examples, and a good solution could not be found within a reasonable time. In addition a single bit flip on the chromosome makes little difference to its fitness as it can change by at most a single exclusive-OR gate.

This lack of success caused us to re-evaluate the way in which we were attempting to model this problem. Taking a different approach, we visualised the ESOP problem as if it were to be minimised using an Ordered Binary Decision Diagram (OBDD) which is a binary tree where selected variables are simplified, and then cannot be selected again. An unsimplified BDD for the Boolean function with minterms 1, 3, 4, 5 and 6 is shown in fig 2.2.

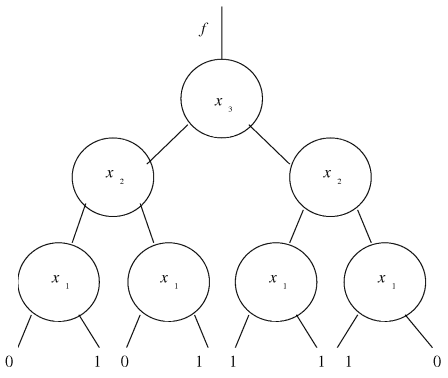

Figure 2.2 Example OBDD

This technique attempts to minimise logical problems by simplifying nodes, corresponding to logical sub-functions, in a binary tree structure. The major problem with this type of model is establishing the variable order which will give the best minimisation opportunities. This is where the GA may be appropriately used. The chromosome representation used here is constructed to avoid the possibility of creating "child" chromosomes which are invalid as problem solutions. To overcome this difficulty, we have used a "pick lists". For example, a chromosome was encoded as the "pick list" 2 3 2 1 1, then this would map to a variable ordering of 1 3 2 0 4. Crossover may now be freely employed without the problem of creating invalid chromosomes.

Provided one has a reliable method of evaluating the effects of a particular variable ordering (by actually minimising the tree, for example), then the GA may be used to “breed” a near optimal solution. This GA, along with bottom-up simplification rules appropriate to Reed-Muller minimisation (BDDs are usually used for Boolean optimisation), has been implemented in our XORGA (exclusive-OR Genetic Algorithm) software and executed on an HP-720 workstation.

It was found that the one drawback in using XORGA was that only functions expressed in minterms were able to be minimised. This problem was addressed in a further re-definition and refinement of the technique in which ternary rather than binary decision model were employed. This led to a simplification of the rule set, and more importantly, allowed the minimisation of functions presented in implicant (groups of simplified minterms) rather than minterm form. This implementation is called XORGA1, and compares favourably with EXMIN2 [9], the best previously devised heuristic for dealing with ESOP minimisation. A set of comparative results for XORGA1 against EXMIN2 is presented in table 2.1. All the functions shown are either MCNC or ESPRESSO benchmarks used to compare the effectiveness of minimisation algorithms.

Table 2.2 shows the results obtained for running XORGA1 on a number of larger benchmark functions - which, to date, no other Reed-Muller ESOP algorithm has been able to tackle. In using both XORGA and XORGA1 algorithms we have found that we are able to achieve these good solutions with relatively small populations (typically of between 10 and 20 chromosomes), and number of generations (usually no more than 20). The GA parameters are normally set at: breeding rate = 60% and mutation rate = 10%. We have found by experience that these parameters give

reasonable results for these applications, though experiments with a meta-GA may provide a more exact parameter set. The results demonstrate the effectiveness of using a GA to find a favourable variable ordering for decision diagram type models in combinational logic optimisation.

Benchmark	Inputs/ Outputs	EXMIN2		XORGA1	
		XOR	AND	XOR	AND
5xp1	7 10	61	125	50	110
clip	9 5	113	404	84	226
life	9 1	54	361	23	51
rd53	5 3	19	41	16	21
adr4	8 5	40	122	21	24
mlp4	8 8	84	311	121	246
sao2	10 4	62	246	53	153
t3	12 8	43	166	19	91
rd73	7 3	56	165	28	46
rd84	8 4	67	263	56	90
z4	7 4	34	111	21	25
sym10	10 1	84	667	44	71
9sym	9 1	53	380	35	55
add6	12 7	140	732	59	97
addm4	9 8	133	521	156	335
b12	9 8	40	124	63	273
ex7	16 5	81	520	144	416
m181	15 9	41	128	64	291
m4	9 8	295	488	156	335
max512	9 6	136	560	147	319
t481	16 1	13	40	88	229
vg2	25 8	188	1804	84	401

Table 2.1 Comparison of XORGA1 with EXMIN2.

Benchmark	Inputs/ Outputs	Products	XORGA1	
		(disjoint)	XOR	AND
bc0	26 11	430	369	1297
duke2	22 29	226	191	855
ibm	48 17	1046	354	1364
in3	35 29	308	213	919
in5	24 14	348	319	1250
in6	33 23	86	84	490
mark1	20 31	22	40	712
misex2	25 18	29	11	146
misj	35 14	48	30	63
x6dn	39 5	262	250	1185

Table 2.2 Performance of XORGA1 on Large Benchmark Functions.

3. State Assignment.

The other major logic optimisation problem that we have tackled using GA techniques is the state assignment issue in the design of synchronous state machines [10] When one is designing such finite state machines (FSMs), it is important to take care in the allocation of the binary patterns which represent the states as this can radically affect the subsequent minimisation of the steering logic (the circuits which derive next state conditions). Careful choice of assignment may therefore result in a greatly reduced implementation.

Once again, the difficulty facing the designer is the large number of possible assignments that exist, even for fairly small design examples. An FSM with m states requires a minimum of s state variables for the assignment where $s = \lceil \log_2 m \rceil$, where $\lceil g \rceil$ gives the smallest integer equal to or greater than g. The number of logically unique [9] assignments for an FSM, N, is given by:- $N = \frac{(2^s - 1)!}{(2^s - m)!\,s!}$. It is difficult for the designer to predict which of these assignments will produce the most minimal solution. The GA is used to effectively search for this by breeding, over a number of generations, populations of legitimate assignments. As before, the state assignment problem has a high probability of creating, due to crossover, child chromosomes which are not valid state assignments. To avoid this, a "pick list" representation was again used. The algorithm has been implemented, and run on an HP-720 workstation. The GA population size, and number of generations are still relatively small, and allow the algorithm to be run within a reasonable period of time. In fact, all the parameters are comparable with those used in the OBDD ESOP problem.

KISS2 file	Number of states	SAGA	NOVA
donfile	24	104 / 75 / 32	207 / 160 / 60
bbtas	6	15 / 9 / 6	31 / 20 / 8
dk14	7	39 / 26 / 19	78 / 59 / 26
dk15	4	19 / 12 / 8	21 / 14 / 6
dk16	27	223 / 176 / 89	279 / 225 / 92
dk27	7	11 / 5 / 6	19 / 12 / 5
dk512	14	24 / 14 / 11	53 / 38 / 19
modulo12	12	15 / 7 / 5	31 / 19 / 10
shiftreg	8	3 / 0 / 0	18 / 11 / 7
tav	4	2 / 0 / 0	5 / 2 / 1
test	8	11 / 5 / 3	23 / 15 / 7
table1	16	22 / 13 / 7	75 / 55 / 24
table2	9	14 / 7 / 5	37 / 25 / 13
table3	6	19 / 12 / 4	36 / 24 / 13
swma1	8	9 / 4 / 2	17 / 10 / 5
swma2	8	7 / 3 / 2	14 / 8 / 4
swma3	16	67 / 46 / 20	115 / 88 / 41

Table 3.1 Comparison of SAGA with NOVA.

Table 3.1 presents results for our algorithm, SAGA (State Assignment Genetic Algorithm) [11] as against NOVA [12], a heuristic used in commercial synthesis tools such as Mentor Graphics. It should be noted that the fitness of a chromosome representing a particular state assignment is obtained by creating a combinational truth table (PLA file), and the

using a Boolean logic minimiser (ESPRESSO [13]) to determine the number of gates required for implementation. We hope to be able to replace the use of ESPRESSO with our own GA based algorithm e.g. XORGA1.
The table shows the number of states each benchmark possesses (benchmarks presented in standard KISS2 format for ASMs), followed by a summary for both SAGA and NOVA of the number of two-input gates required to implement the machine. The breakdown of results is thus:- number of literals/ number of AND gates/ number of OR gates respectively.
The table demonstrates that the GA technique is capable of finding more favourable assignments than NOVA, which, like EXMIN2 in the ESOP problem, is the best previously devised heuristic for state assignment.

4. Conclusions.

It is important to logic designers - especially those dealing with larger designs - to have available within their synthesis tools, reliable and efficient algorithms for optimising circuit implementations. Clearly, in the problem areas we have considered in this paper - Reed-Muller ESOPs and FSM state assignment - there are fairly sophisticated heuristic methods available. However, as we have demonstrated, the use of genetic search techniques proves to be an even more attractive option. Additionally, the genetic algorithm methods are not limited to always producing the same result for these optimisation problems; the designer may, if desired, allow the GA to develop over greater numbers of generation, thus increasing the scope of the search. Further, with different emerging GA techniques, such as alternative parent selection and crossover mechanisms, then the effectiveness of the method may yet be improved still further.
The GA is an extremely flexible technique when applied to optimisation applications in engineering in that the fitness function may be easily modified to accommodate new design criteria. For example, it is our intention to further improve XORGA1 by addressing, via such modifications, the timing delay and fan-out issues that inevitably pervade multi-level logic implementation.

References.

[1] T. Sasao, "Logic Synthesis and Optimization," in *Kluwer Academic Publishers, Mass. Chapter 13,* 1993.

[2] S. M. Reddy, "Easily Testable Realizations for Logic Functions," in *IEEE Trans. Comput.,* vol. C-21, pp. 1183-1188, 1972.

[3] J. F. Miller and P. Thomson, "Highly efficient exhaustive search algorithm for optimizing canonical Reed-Muller expansions of Boolean functions.", *International Journal of Electronics,* No. 76, Vol. 1, pp37-56, 1994.

[4] A. E. A. Almaini, P. Thomson and D. Hanson, "Tabular techniques for Reed-Muller logic.", *International Journal of Electronics,* No. 70, Vol. 1, pp23-34, 1991.

[5] L. McKenzie, A. E. A. Almaini, J. F. Miller and P. Thomson, "Optimization of Reed-Muller logic functions.", *International Journal of Electronics,* No. 75, Vol. 3, pp451-466, 1993.

[6] A. Sarabi and M. A. Perkowski, "Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/XOR Canonical Networks.", *29th ACM/IEEE Design Automation Conference*, No. 3.1, pp 30-35, 1992.

[7] J. F. Miller, H. Luchian, P. V. G. Bradbeer and P. J. Barclay, "Using a genetic algorithm for optimizing fixed polarity Reed-Muller expansions of boolean functions.", *International Journal of Electronics,* No. 76, Vol. 4, pp601-609, 1994.

[8] P. J. Hancock, "An empirical comparison of selection methods in evolutionary algorithms," in *AISB Workshop on Evolutionary Computing.*, at Leeds University, UK, April 1994.

[9] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-OR sum-of-products expressions for multiple-valued-input two-valued-output functions.", *IEEE Trans on CAD of integrated Circuits and Systems,* Vol. 12, No.5, pp621-632, 1993.

[10] A. E. A. Almaini, "Electronic Logic Systems," 3rd ed., *Prentice-Hall*, 1994.

[11] A. E. A. Almaini, J. F. Miller, P. Thomson and S. Billina, "State assignment of finite state machines using a genetic algorithm.", *IEE Trans, Computers and Digital Techniques,* accepted March, 1995.

[12] T. Villa and A. Sangiovanni-Vincentelli, "NOVA: State assignment of finite state machines for optimal two level logic implementation.", *IEEE Trans*, C-9, pp905-924, 1990.

[13] R. K. Brayton, G. D. Hachtel, C. T. McMullen and A. L. Sangiovanni-Vincentelli, "ESPRESSO_II: a new logic minimiser for programmable logic arrays.", *IEEE Custom Integrated Cicuits Conference*, Rochester, New York, pp. 370-376.