World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:1, No:12, 2007

Feasibility of the Evolutionary Algorithm using Different Behaviours of the Mutation Rate to Design Simple Digital Logic Circuits

Konstantin Movsovic, Emanuele Stomeo, and Tatiana Kalganova

***Abstract*—**The evolutionary design of electronic circuits, or evolvable hardware, is a discipline that allows the user to automatically obtain the desired circuit design. The circuit configuration is under the control of evolutionary algorithms. Several researchers have used evolvable hardware to design electrical circuits. Every time that one particular algorithm is selected to carry out the evolution, it is necessary that all its parameters, such as mutation rate, population size, selection mechanisms etc. are tuned in order to achieve the best results during the evolution process. This paper investigates the abilities of evolution strategy to evolve digital logic circuits based on programmable logic array structures when different mutation rates are used. Several mutation rates (fixed and variable) are analyzed and compared with each other to outline the most appropriate choice to be used during the evolution of combinational logic circuits. The experimental results outlined in this paper are important as they could be used by every researcher who might need to use the evolutionary algorithm to design digital logic circuits.

***Keywords*—**Evolvable hardware, evolutionary algorithm, digital logic circuit, mutation rate.

I. Introduction

EVOLVABLE hardware [1] – [3] is a technique to automatically design electronic circuits, where the circuit configuration is carried out by evolutionary algorithms. Several evolutionary algorithms, such as genetic algorithm [4], evolution strategy [5], genetic programming [6], have been used for the evolution of digital circuits. Furthermore, numerous researchers have altered evolutionary algorithms (by changing for example the selection mechanisms, the chromosome evaluation etc.) to improve its performance in terms of the number of generations, fitness value reached, computational time etc. leading to the introduction of several other evolutionary algorithms such as Strength Pareto Evolutionary Algorithm [14], Traceless Genetic Programming [15], Embedded Cartesian Genetic Programming [16] etc. Other researchers, alternatively, have introduced decomposition strategies such as the increased complexity evolution [11], bi-directional incremental evolution [10], generalized disjunction decomposition [7] to improve the scalability and the evolvability, which are the main issues that limit the use of evolvable hardware for real world applications. The scalability limits the size of the circuit that may be evolved. Evolvability, as defined by Altenberg in [8], is the ability of the genetic operator/representation scheme to produce offspring that are fitter than their parents. What is clear from all those papers is that every time a researcher uses different evolutionary methods, it is necessary to tune its parameters, such as mutation rate, population size, and fitness evaluation, in order to achieve the final goal more efficiently: the design of the circuits. In this paper, we are analyzing the behavior of the evolutionary algorithm for designing digital circuits based on PLA structures when the mutation rate is changing dynamically. Six different mutation rates (fixed and variable) are analyzed and compared with each other in order to find general solutions. In [12][13] the behavior of the mutation rate was studied for designing and optimizing digital logic circuits based on FPGA structures. The mutation operator is very important since it brings diversity into the population of the possible solutions. If the mutation rate is very high, the genetic search will be transformed into random searches but it also helps to reintroduce lost genetic material [9]. Therefore a correct value for the mutation rate should be investigated.

This paper is organized as follow: Section 2 illustrates the evolutionary algorithm, from the description of the chromosome to the explanation of the fitness calculation chosen for the evolution of combinational logic circuits. Section 3 describes the six different mutation rates used for the evolution processes and Section 4 shows the experimental results, based on the number of successful evolutions, of the designed circuits. Section 5 concludes this paper and outlines the significance of the results obtained.

II. Extrinsic Evolvable Hardware

In this section extrinsic evolvable hardware is described. All the genetic mechanisms from the initialization to the fitness function calculation are explained. As both the genetic algorithm and the fitness function are simulated in software, the implemented system is called extrinsic evolvable hardware system.

Manuscript received February 15, 2006. This work was supported in part by the EPSRC Grant GR/S17178/.

K. Movsovic and E. Stomeo are undergraduate and PhD students at School of Engineering and Design, Brunel University, West London, UK. (e-mail: Konstantin.Movsovic; emanuele.stomeo; tatiana.kalganova@brunel.ac.uk). T. Kalganova is lecture at the same university. UB8 2TR, Uxbridge, Middlesex, UK.

A. Evolutionary Algorithm

The evolutionary algorithm chosen is the well known (1+λ) evolution strategy [17][18]. It has been decided to use this algorithm because was extensively tested for its efficiency for the design of digital logic circuits [19]. The implemented evolution strategy is shown in Fig. 1. At the beginning all the λ individuals are randomly initialized. After this first step, they are evaluated and then selected. The individual with the highest fitness function value will be selected in order to reproduce a new population, using the mutation operator only. In the next generation all the individuals are evaluated and then again selected. This continues to run until all the conditions are met. The conditions are usually a certain number of generations or a fitness value that does not increase anymore. The second condition happens when the stalling effects occur [7]. However, at this generation the best individual is going to be selected within (1+λ) indivuduals. This evolutionary algorithm is very simple and could be easily implemented into hardware and tested in an intrinsic environment.

B. Genotype Representation

Since the evolutionary algorithm chosen is to be used for designing logic circuits based on PLA, and in the PLA both planes are programmable (the AND plane and the OR plane), we have decided that each chromosome should contain the required information to make the evolution of the connections in the AND plane and in the OR plane possible. In Fig. 2 an example of a PLA, with the AND and the OR planes, together with the chromosomes representations is shown. Each connection-point in the AND plane could be: connected to the positive (direct to the input signal), connected to the negative (connected to the output of the NOT gate of the input signal) or not connected. The connection-point of the OR plane could be connected or not connected to the output of the AND gates. As a result, to encode the possible connections in the AND plane 2 bits are required. For the OR plane only 1 bit is needed for each possible connection.

C. Initialization

In the implemented evolutionary algorithm all the chromosomes are randomly initialized.

D. Fitness Evaluation

The fitness is a measure of the quality of the evolved individual. During the evolutionary process each individual (described by its chromosome) is evaluated. The fittest individuals are usually selected for the reproduction of a new generation that will replace the old one. The fitness function is the function used to evaluate those individuals. The fitness function for the evaluation of the PLA is very simple: each evolved PLA is stimulated with all the possible input

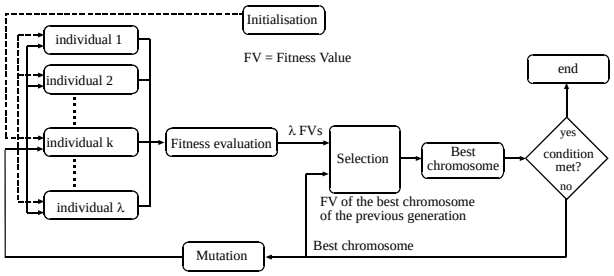

Fig. 1 Implemented algorithm for the design and optimization of digital logic circuits. Figure taken from [7]

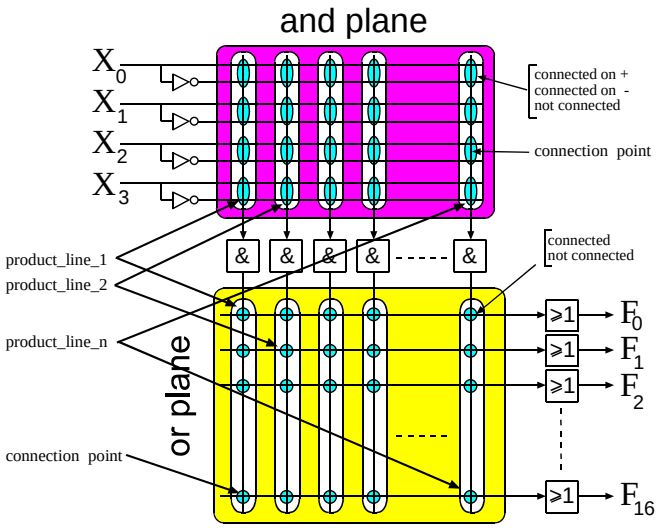

Fig. 2 Schema of the PLA with representation of the chromosome

combinations and the obtained results are compared with the truth table of the desired digital logic circuits. A value in percentage, based on the quality of the evaluated individual, is assigned to each PLA. The PLA with the highest fitness value is selected for the reproduction. λ new individuals are then generated applying the mutation operator to the selected PLA.

E. Selection Mechanism

The selection mechanism is based on elitism, meaning that only the best individual is selected for reproduction. All the other individuals are to be replaced with newly generated.

F. Reproduction Mechanism

In the implemented (1+λ) evolution strategy, the reproduction mechanism is limited to the mutation operator. An example of the mutation is given in Fig. 3. Since it has been decided to use the (1+λ) evolution strategy the mutation points are to be changed λ times and the same parent is going to produce λ different offspring. How the mutation points change during the evolution process is described in the next section.

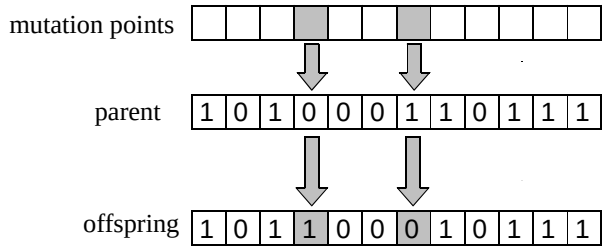

Fig. 3 Example of a mutation operator. To obtain a new individual some genes of the chromosome's parent are changed

III. Behavior of the Mutation Operators

This section illustrates the different behaviors of the mutation operator that have been taken into account for the simulations of the described (1+λ) evolution strategy. In order to have a general analysis, six different mutation rates have been tested for their efficiency to design logic circuits. The chosen mutation rates are:

- (Task 1). Fixed mutation rate at 5% for the evolution of the AND plane and the OR plane. It means that the 5% of all chromosomes within the PLA are changed during the evolution.
- (Task 2). Mutation rate fixed at 3.75% for the AND plane and 1.25% for the OR plane.
- (Task 3). Mutation rate fixed at 3.75% for the OR plane and 1.25% for the AND plane.
- (Task 4). Variable mutation rate (see Equation 1, where *y* is the mutation rate to be chosen and *x* is the number of generations used) within the range of 0 – 5%. For the AND plane $\alpha=5/N_{gen}$ and $\beta=0$. For the OR plane $\alpha=-(N_{gen}/5)$ and $\beta=5$.
- (Task 5). Inverted Task 4.
- (Task 6). Mutation rate fixed at 5% applied to all the chromosomes.

For all the tasks, except for Task 6, the following constraint has been applied to the system: the part of the chromosome corresponding to the AND plane is fixed in such a way that each input will be considered for the evolution process. However, for Task 6, could happen that one or more inputs may not take part during the evolution.

$$y = \alpha x + \beta \quad (1)$$

IV. Experimental Results

In this section the experimental results obtained using a 2-bit multiplier, the 4- and 5-bit even parity circuits are presented. Since the two bit multiplier is widely used within the evolvable hardware community it has been decided to use it as a benchmark for the simulation. The chosen test benches also include the even parity circuits. It is difficult to evolve these circuits since a change in the value of any of its arguments toggles the value of the function. The parity functions are often used to check the accuracy of the stored or transmitted binary data in computers. The intention of these experiments is to show how the choice of a particular mutation rate will influence the evolvability for the selected digital circuits. The initial data for the simulations is the number of generations for the evolutionary process together with the number of product lines per each PLA. They are reported at the left side of each table. For the experiments, the described (1+λ) evolution strategy with λ=5 (thus in total 6 PLAs are involved in the evolution) has been implemented into an extrinsic environment using C++ and tested with a desktop PC with the following configuration: 2,4Ghz Pentium 4, 512 MB RAM. Table I reports the average (based on 100 runs) of the successful evolution for the evolution of the 2-bit multiplier, when the mutation rate is chosen according to the given task. All the tasks are described in the previous section. In Table II and Table III the experimental results for the evolution of the 4- and 5- bit even parity bit circuits is shown. In all the given tables the best results are highlighted.

From Table I is clear that if we centre attention only on the design part, the best behavior of the mutation operator is the one given with the Task 3 and Task 4. Using those two tasks it is noticeable that a higher value of the successful evolution is reached. Nevertheless, it is also true that the given solutions are not very well optimized since they are using a higher number of product lines. The more the product lines are used during the evolution the higher the number of the required logic gates in the final configuration of the PLA. Instead, if we focus our attention on the optimal solutions based on the number of logic gates, we notice that they are obtained when a fixed mutation rate (equal to 5%) is applied to the system. Regarding the experimental results obtained for the evolution of even parity functions, that are shown in Table II and Table III, it is evident that the mutation rate that gives better results is equal to the one described in Task 1 (mutation rate fixed at 5% for the evolution process).

TABLE I
Average (based on 100 of Runs) of the Successful Evolution for the 2-Bit Multiplier

Initial data		Behavior of the mutation rate chosen according with the					
Num. of generations	Number of Product lines	Task 1 %	Task 2 %	Task 3 %	Task 4 %	Task 5 %	Task 6 %
5000	9	12	0	0	0	0	0
5000	10	28	0	6	6	6	0
5000	11	35	4	6	6	11	0
5000	12	29	12	32	30	10	0
5000	13	47	5	24	31	12	0
5000	14	33	5	39	41	3	1
5000	15	36	4	42	44	5	0
5000	16	32	2	48	35	1	0
5000	17	22	3	38	43	1	0
5000	18	13	3	41	41	1	0
5000	19	12	0	36	23	1	0
5000	20	5	0	24	27	0	0
5000	21	6	1	17	17	0	0
5000	22	3	0	24	14	0	0

TABLE II
AVERAGE (BASED ON 100 RUNS) OF THE SUCCESSFUL EVOLUTION FOR THE 4-BIT EVEN PARITY CIRCUITS

Initial data		Behavior of the mutation rate chose according with the					
Num. of generations	Number of Product lines	Task 1 %	Task 2 %	Task 3 %	Task 4 %	Task 5 %	Task 6 %
5000	7	0	0	0	0	0	0
5000	8	47	10	40	39	36	0
5000	9	82	14	82	67	70	0
5000	10	89	20	95	83	90	0
5000	11	95	49	99	92	92	0
5000	12	99	65	100	95	95	0
5000	13	100	78	100	98	99	0
5000	14	100	76	100	99	100	0
5000	15	100	83	100	100	100	0
5000	16	100	98	98	100	100	0
5000	17	100	97	100	100	100	0
5000	18	100	96	100	100	99	0
5000	19	100	95	99	100	100	0

TABLE III
AVERAGE (BASED ON 100 OF RUNS) OF THE SUCCESSFUL EVOLUTION FOR THE 5-BIT EVEN PARITY CIRCUITS. ALL THE TASKS EXCEPTS THE TASK 6, HAVE BEEN ANALYZED. TASK 6 HAS NOT BEEN TAKEN INTO ACCOUNT BECAUSE IT DOES NOT GIVE ANY VALUABLE RESULTS

Initial data		Behavior of the mutation rate chose according with the				
Num. of generations	Number of Product lines	Task 1 %	Task 2 %	Task 3 %	Task 4 %	Task 5 %
5000	35	67	18	59	28	45
5000	36	69	26	67	42	56
5000	37	77	15	55	33	53
5000	38	77	26	67	40	55
5000	39	78	18	64	39	52
5000	40	67	17	76	48	57
5000	41	88	13	72	35	53
5000	42	77	21	59	46	56
5000	43	78	23	64	33	56
5000	44	80	18	64	36	63
5000	45	81	24	71	40	62
5000	46	82	25	71	47	62
5000	47	79	24	80	47	67
5000	48	73	23	74	36	65
5000	49	81	10	75	40	66
5000	50	87	22	68	37	47
5000	51	75	24	67	39	
5000	52	68	20	68	44	59

V. CONCLUSION

This paper has exploited several behaviours of the mutation operator to be used for the evolution of digital logic circuits based on PLA structures. To have statistically relevant results each analyzed logic circuit has been evolved 100 times and the average, based in terms of successful evolution has been shown and discussed. The mutation operators analyzed were with fixed and linearly variable mutation rates. The mutation rate was changed dynamically during the evolution process. Regarding the evolution of multipliers, the experimental results have shown that with the use of fixed mutation rates the successful evolution rate was quite low, however the few obtained solutions were well optimized. Contrary to this, when the mutation rate was dynamically changed the number of required generations to fully evolve the combinational circuits was drastically reduced. For the even parity circuits was noticed that the best solution are obtained when the mutation rate was fixed at 5%. Future work would consider a wider range for the mutation rate to be used for the evolutionary process in order to design and optimize logic circuits larger circuits.

REFERENCES

[1] N. Forbes. "Evolution on a chip: evolvable hardware aims to optimize circuit design". *Computing in Science & Engineering [see also IEEE Computational Science and Engineering]*. Volume 3, Issue 3, May-June 2001 Page(s):6 – 10.

[2] G. W. Greenwood, "On the practicality of using intrinsic reconfiguration for fault recovery." *IEEE Transactions on Evolutionary Computation.* Volume 9, Issue 4. Pages: 398 – 405.

[3] X. Yao, T. Higuchi. "Promises and challenges of evolvable hardware" IEEE Trans. Systems, Man and Cybernetics, Part C, vol. 29, Pages. 87 - 97, February 1999.

[4] J. Holland. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[5] I. Rechenberg, "Evolution Strategy", in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life*, 1994, pp. 147-159.

[6] J. R Koza. Genetic Programming: On the Programming of Computers by Means of Natural selection. ISBN 0-262-11170-5. MIT Press, 1992.

[7] E. Stomeo, T. Kalganova, C. Lambert. "Generalized Disjunction Decomposition for Evolvable Hardware" *IEEE Trans. Systems, Man and Cybernetics, Part B. 2006* (Accepted for publication).

[8] Lee Altenberg "The Evolution of Evolvability in Genetic Programming". Chapter 3 in Advances in Genetic Programming, ed. Kenneth Kinnear. pp. 47-74. MIT Press, Cambridge, 1994.

[9] M. Srinivas, L. M. Patnaik; "Genetic algorithms: a survey". *IEEE JNL Computer*, Volume: 27, Issue: 6, June 1994. Pages: 17 – 26.

[10] T. Kalganova; "Bidirectional incremental evolution in extrinsic evolvable hardware". *Proc. of the Second NASA/DoD Workshop on Evolvable Hardware. IEEE Computer Society*, 13-15 July 2000. Pages:65 – 74

[11] J. Torresen, "Increased complexity evolution applied to evolvable hardware", *ANNIE'99*, November 1999, St. Louis, USA.

[12] E. Stomeo, T. Kalganova, C. Lambert. "Mutation Rate for Evolvable Hardware". *International Conference on Computational Intelligence - ICCI 2005* August 26-28, 2005. Prague, Czech Republic. Pages: 117 - 124.

[13] E. Stomeo, T. Kalganova, C. Lambert "Chose the Right Mutation Rate for Better Evolve Combinational Logic Circuits". *International Journal of Computational Intelligence (IJCI)* (accepted for publication).

[14] S. Bleuler, M. Brack, L. Thiele, E. Zitzler. "Multiobjective genetic programming: reducing bloat using SPEA2". *Proceedings of the 2001 Congress on Evolutionary Computation, 2001.Volume 1*, 27-30 May 2001. Page: 536 – 543.

[15] M. Oltean. "Solving even-parity problems using traceless genetic programming". *Congress on Evolutionary Computation, 2004. CEC2004. Volume 2.* 19-23 June 2004 Page: 1813 – 1819.

[16] A. James Walker and Julian F. Miller, "Evolution and Acquisition of Modules in Cartesian Genetic Programming", *Proceedings of EuroGp2004. Lecture Notes in Computer Science*, Volume 3003 / 2004. Pages: 187 – 197. Maarten Keijzer, Una-May O'Reilly, Simon M. Lucas, Ernesto Costa, Terence Soule (Eds.).

[17] T. Bäck, F. Hoffmeister, and H. P. Schwefel. "A survey of evolutionary strategies". *In R. Belew and L. Booker, editors, Proceedings of the 4th International Conference on Genetic Algorithms*, San Francisco, CA, 1991. Morgan Kaufmann. Pages 2–9.

[18] H.-P. Schwefel. Numerical Optimization of Computer Models. John Wiley & Sons, Chichester, UK, 1981.

[19] E. Stomeo, T. Kalganova, C. Lambert, N. Lipnitsakya, Y. Yatskevich. "On Evolution of Relatively Large Combinational Logic Circuits". *The IEEE 2005 NASA/DoD Conference on Evolvable Hardware.* June 29 - July 1, 2005, Washington DC, USA. IEEE Computer Society. Pages 59 – 66.