2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

A Novel Genetic Algorithm for Evolvable Hardware

Emanuele Stomeo, *Member IEEE*, Tatiana Kalganova, Cyrille Lambert

Abstract—**Evolutionary algorithms are used for solving search and optimization problems. A new field in which they are also applied is evolvable hardware, which refers to a self-configurable electronic system. However, evolvable hardware is not widely recognized as a tool for solving real-world applications, because of the scalability problem, which limits the size of the system that may be evolved. In this paper a new genetic algorithm, particularly designed for evolving logic circuits, is presented and tested for its scalability. The proposed algorithm designs and optimizes logic circuits based on a Programmable Logic Array (PLA) structure. Furthermore it allows the evolution of large logic circuits, without the use of any decomposition techniques. The experimental results, based on the evolution of several logic circuits taken from three different benchmarks, prove that the proposed algorithm is very fast, as only a few generations are required to fully evolve the logic circuits. In addition it optimizes the evolved circuits better than the optimization offered by other evolutionary algorithms based on a PLA and FPGA structures.**

I. Introduction

THE evolutionary design of electronic circuits [1], [2] is limited by the difficulties in evolving large circuits in a reasonably short time. This is due to the scalability problems [2]–[5] and stalling effects during the fitness function calculation. The scalability problem limits the size of the system that may be designed using evolutionary design techniques [2]. The stalling effects appear during the evaluation of possible solutions and are as a result of the complexity of the fitness functions taken into consideration. Usually, an evolvable hardware system is not scalable because of the genotype length which increases with the problem size [6], with the number of elements (logic gates, transistors or other discrete elements, functional blocks, etc.) used during the evolution and also with the permitted connectivity between those elements. To tackle these issues and to design larger circuits, several techniques have been introduced, such as gate and function level evolution [7], increased complexity evolution [8], bi-directional incremental evolution [9], Generalized Disjunction Decomposition [2] etc. Several researchers have used different circuit structures as a base from which to evolve electrical circuits, the most common are the:

- Field Programmable Gate Array (FPGA) based structure used by: Tyrrell [11] and Thompson [12] to evolve robot controllers, and by Vinger [13] and Gwaltney [14] to design digital filters.
- Field Programmable Transistor Array (FPTA) structure used by Stoica et al. to design fault tolerant very large scale integrated (VLSI) circuits [15].
- PLA structure used by: Arslan et al. [17] to evolve digital filters and by Stomeo et al. [18] to design logic circuits.

In this paper a new genetic algorithm, particularly designed for the evolution of logic circuits based on a PLA (Programmable Logic Array) structure is presented. A PLA was chosen for its structure, which is good for designing combinational logic circuits in VLSI and for its simplicity, regularity and flexibility. Another reason for having chosen a PLA instead of FPGA as structure for the evolution of digital logic circuits is because the permitted connectivity between logic gates within a PLA is smaller; therefore a reduction of the genotype length could be easily achieved. This helps the evolution of digital circuits. However the presented algorithm could be implemented into FPGA, initial work is given in [19]. The efficiency of the proposed algorithm has been examined using three test benches: the adders and the multipliers, both commonly used within the evolvable hardware community, and the MCNC benchmark traditionally used in logic design. This algorithm has proven itself to be very fast during the design process, as fewer generations are required to design the circuit. It has also demonstrated the ability to optimize the evolved circuits. Based on the results obtained for the evolved circuits, the proposed method also outperforms human design in terms of design cost. The simulation results of this new method have shown that it is able to design circuits faster than the fastest existing methods (know to the authors), which are Embedded Cartesian Genetic Programming (ECGP) [16] and Traceless Genetic Programming [20]. In the following sections the proposed genetic algorithm, together with the initialization, evaluation, selection and reproduction mechanisms are presented. This is followed by an explanation of the fitness value calculation, where a numeric example is also supplied. In Section III, the optimization within the proposed algorithm is outlined. In Section IV the experimental results obtained for the designed logic circuits together with an analysis of the initial data used for the simulations are presented. In Section V an analysis of the results found and the benefits of the proposed genetic

This work is supported in part by the EPSRC under Grant GR/S17178.
The authors are with the School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH U K (stomeo@ieee.org).

algorithm are highlighted and compared with other similar techniques. Section VI concludes and summarizes the content of the paper.

II. A Novel Genetic Algorithm

In this section an introduction to the proposed method is firstly given, which outlines the motivation for the development of this new genetic algorithm. The initial section contains the description of the terms used to illustrate the proposed algorithm. The following sections explain the algorithm, together with chromosome representations, selection, reproduction mechanisms and fitness value calculations.

A. *Prologue*

The genetic algorithm described in this paper is intended for the design of combinational logic circuits based on a PLA structure. The key motivation for designing this algorithm is to have a fast configuration of the combinational logic circuits. The main characteristics of the proposed method are the use of several populations in parallel, and the construction of one population from the elitism's pool (which contains the best chromosomes of each population). In order to describe the proposed algorithm, the evolution of a simple circuit is considered. Supposing that a circuit based on a PLA structure with 4 inputs should be evolved. Therefore the evolutionary algorithm should set the correct connections in the AND plane (for each interconnection the possible choices in the AND plane are "connect to 1", "connect to 0" or "not connect") and in the OR plane (the choices are "connected" or "not connected"). For the proposed algorithm (as reported in Figure 1) two different types of chromosomes have been considered:

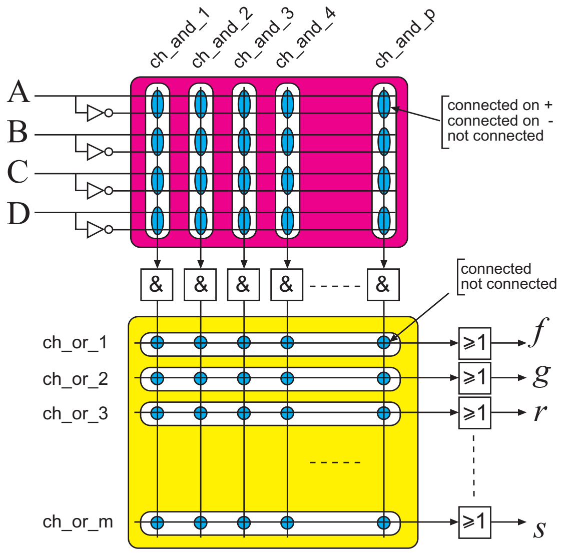


Figure 1. Example of chromosomes in the PLA. Two types of chromosomes are noticeable, the **ch_and** for the AND plane and the **ch_or** for the OR plane.

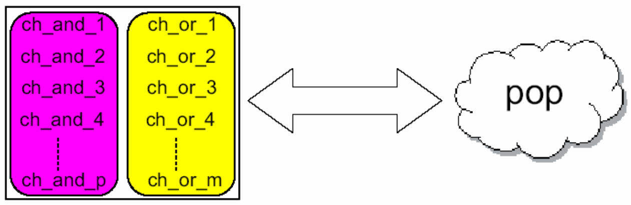


Figure 2. One population contains all the chromosomes (**ch_and** and **ch_or**) of the PLA.

- **ch_and**, which represents a single column of possible connections inside the AND_PLANE.
- **ch_or**, which represents a single row of possible connections inside the OR_PLANE

Each population contains all the chromosomes of the AND plane and all the chromosomes of the OR plane (see Figure 2) of the Programmable Logic Array.

B. *Structure of the Proposed Algorithm*

The proposed algorithm makes use of a multi-population of individuals. All the chromosomes of each population will be evaluated by the proposed algorithm and new populations of chromosomes will be created. The old populations will be replaced by those newly generated populations. The proposed genetic algorithm consists of the same mechanisms of the simple genetic algorithm [23][24] which are: initialization, evaluation, selection and generation. All these mechanisms are described in the next sections.

C. *Initialization and Evaluation Mechanism*

Supposing that, for a particular experiment, N populations have been taken into account. At the initialization stage, all the chromosomes of the N populations have been randomly initialized. For the evaluation stage, each chromosome of each population will be individually evaluated. The evaluation is performed by analyzing and comparing the chromosome's output with the truth table, which describes the digital circuit. The fitness value is calculated according to the quality of the evaluated chromosome. See Section II.F for more details on how the fitness value is calculated. During the evaluation stage a table, called the *fitness's value table*, which contains the fitness's value of each chromosome that is participating in the evolution of the digital circuit, is generated (see Figure 3). The table in Figure 3 refers to the fitness evaluation case of only the AND_PLANE. The *fitness's value table* for the OR_PLANE is not given because it is similar. Each chromosome of each population is evaluated and its fitness's value is stored in the *fitness's value table*.

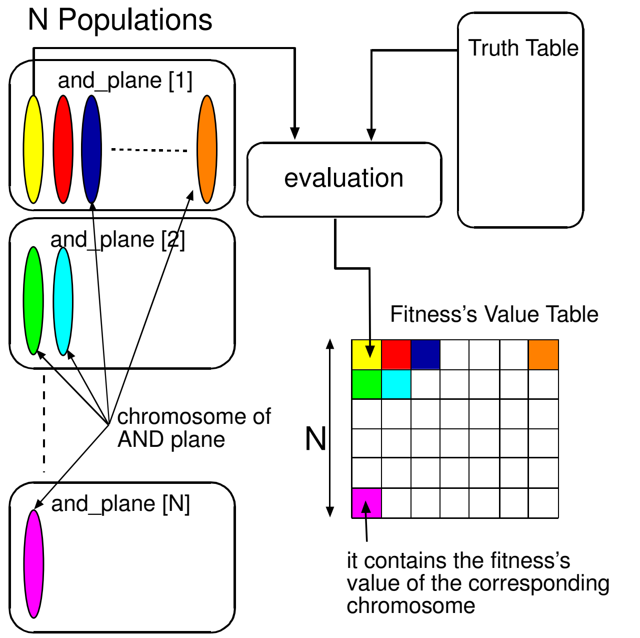


Figure 3. Evaluation mechanism for the proposed genetic algorithm. Each chromosome of all the populations will be evaluated and its fitness value is stored in the *fitness's value table*.

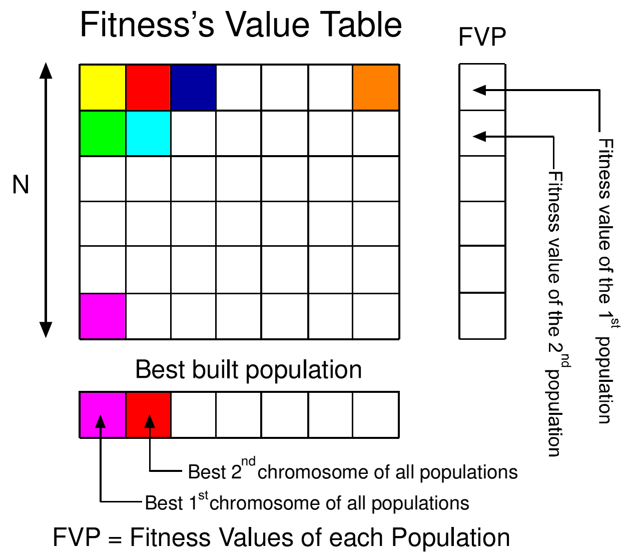

Fitness's Value Table
FVP
N
Fitness value of the 1st population
Fitness value of the 2nd population
Best built population
Best 2nd chromosome of all populations
Best 1st chromosome of all populations
FVP = Fitness Values of each Population

Figure 4. Selection mechanism.

D. *Selection Mechanism*

In genetic algorithms, the selection mechanism is usually based on choosing the best individuals of a population. The generations of a new population is based on mating the chosen individuals using genetic operators. The most common genetic operators are the mutation, the crossover and the elitism.

In evolvable hardware one single individual is generally not able to configure the entire circuit, especially for complex tasks. Usually, for the design of large circuits, several chromosomes should be linked together, since one chromosome represents a circuit configuration, which could be part of the entire design. In this novel genetic algorithm the selection of individuals is based on two stages. In the first stage the *best populations* (BPs) will be selected for the reproduction. The *best populations* are the populations with highest fitness's value. The fitness value of one population is the average of all chromosomes' fitness's value. In order to accomplish this first selection a FVP (Fitness Value of the entire Population) vector is generated (see Figure 4) which contains the fitness value of each evaluated population. The second stage is performed in order to avoid chromosomes being allocated to solve the same regions of the solution space. This is possible because of the nature of the chromosome's encoding used for the design of the PLA (see chromosome's structure in Figure 1). By applying the proposed selections a new population is created as shown in Figure 4. This population is referred to here as: *best built population,* which is the population created by collecting the best chromosome from each region of the solution space.

E. *Reproduction Mechanism*

The reproduction of new populations is obtained by applying the mutation operator to the selected individuals. The new populations are generated in the following way (supposing that N is the total number of populations that are involved in the evolution):

- N/D (with D>0. D is an integer and it will be called divider) populations are generated from the *best built population* using the mutation operator (see Figure 5).

- N-N/D (with D>0) populations are generated using the (μ/ρ, λ) evolution strategy [21][22]. Where μ is equal to N. ρ is the *best population*. λ represents the number of new populations generated and it is equal to N-N/D.

The proposed reproduction mechanism is shown in Figure 5.

F. *Fitness Function Calculation*

Since the proposed algorithm deals with the design and optimization of combinational logic circuits a multi-objective fitness function has to be used. First object is the design and second the optimization.

During the first phase of the evolution (the design of the circuit) the fitness value (FV) of each chromosome is calculated based on what outputs the chromosome is able to produce. A value in percentage terms is given to each chromosome and this value is stored in the *fitness value's table*.

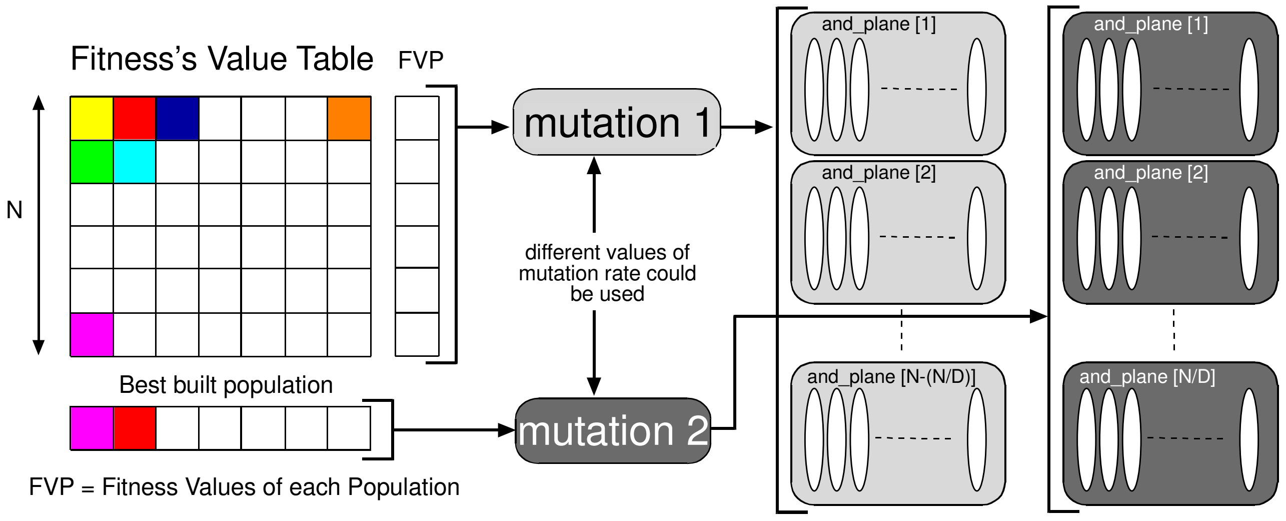


Figure 5. Reproduction mechanism. The new populations are reproduced from the *best population* (the population with the highest value of FVP) and from the *best built population* using only the mutation operator.

The fitness for the design stage is calculated as:

$$f_{design} = \frac{100}{2^i} \cdot \sum_{j=0}^{2^i-1} \left(1 - \left| x_j - d_j \right| \right) \tag{1}$$

where i is the number of inputs of the system, x_j and d_j are the obtained and the desired output of each evaluated chromosome.

Once the combinational logic circuit is fully evolved the second phase of the algorithm starts. The fitness value of this second phase is incremented by one unit every time a logic gate of the designed logic circuit is found redundant and it is eliminated from the design, see Equation 2.

$$f_{optimization} = 100 + \sum_{j=0}^{max\text{-}1} 1 \ \left(if \ x_r = x_d\right) \tag{2}$$

where x_r is the obtained output of the PLA when the randomly chosen r^{th} logic gates is eliminated from the PLA; x_d is the desired output of the PLA and *max* is the maximum number of generations set for the optimization stage of the evolution.

An example of the fitness value calculation for one chromosome that belongs to the OR plane of the PLA is given in Figure 6. The same procedure is applied to calculate the fitness value for the chromosomes of the AND plane. Referring to Figure 6, the letters A, B, C, …, H are the outputs of the AND plane, and they are designed and evaluated as reported in previous sections. The fitness value of the evaluated chromosome reported in Figure 6 is 62.5% because 5 out 8 alleles of the evaluated chromosome are correct.

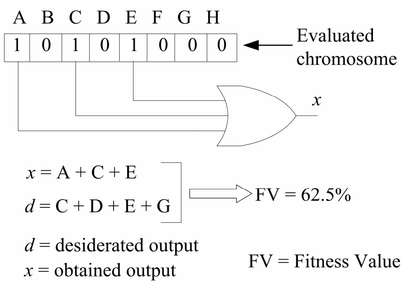


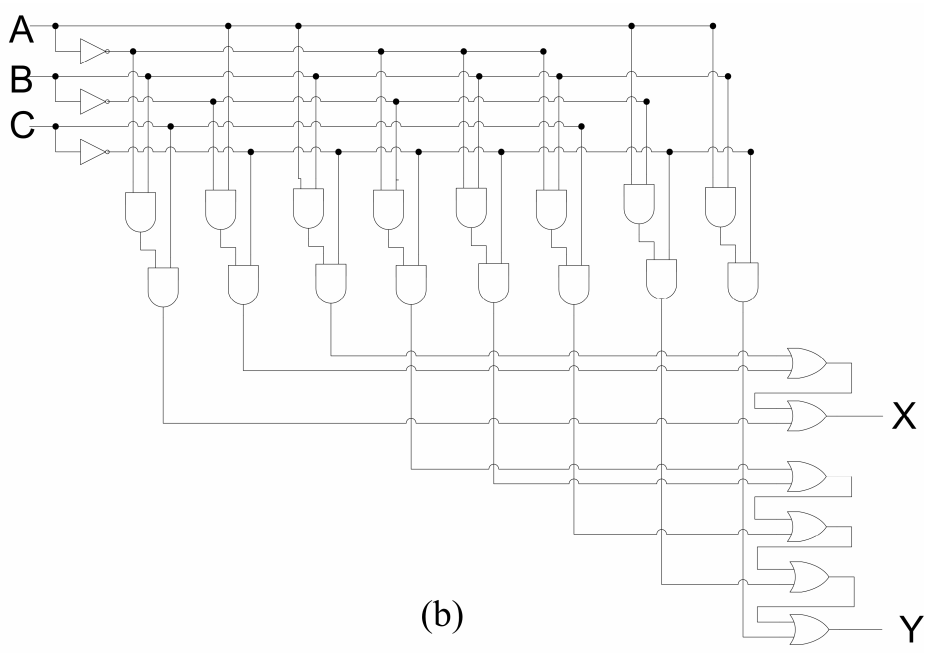
Figure 6. Example of fitness function's value calculation.

III. Optimization of the Evolved Logic Circuits

Once the logic circuit is fully evolved (i.e. for all input combinations stimuli the designed logic circuit gives the expected response), the proposed genetic algorithm starts with the optimization of the evolved circuit. This step is performed in the following way: a primitive logic gate will be selected at random from the evolved PLA, it could be an AND, OR or NOT gate. The selected logic gate will be checked if it is redundant or not. If it is, it would be eliminated from the PLA. This simple method is very efficient as it reduces the evolved logic circuit. Experimental results in Section IV.B have proven its ability to achieve this reduction. As an example, a simple logic circuit, for which the truth table is shown in Figure 7a, will be evolved. The genetic algorithm presented here proposes a solution shown in Figure 7b at the end of the design stage, which makes use of 25 primitive logic gates. This circuit is fully evolved and for each input combination it will produce the expected outputs. At this point the optimization stage starts and the minimization of the circuit take places. The design developed at the end of the second stage is illustrated in Figure 8. The optimized logic circuit uses only 7 primitive logic gates.

ABC	XY
000	01
001	00
010	01
011	11
100	11
101	00
110	11
111	00

(a)

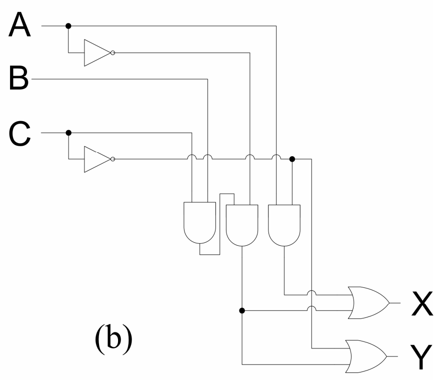


(b)

Figure 7. (a) Truth table of a logic circuit. (b) Logic circuit based on a PLA designed by the proposed genetic algorithm.

ABC	XY
--0	01
011	11
1-0	10

(a)



(b)

Figure 8. Logic circuit optimized by the proposed genetic algorithm.

IV. Experimental Results

In this section the experimental results of the proposed method are given. Three different benchmarks have been considered: the multiplier and the adder, of differing complexities, and the MCNC benchmark [25]. It was decided to use these benchmarks because they are widely used within the evolvable hardware community.

The proposed genetic algorithm has been firstly tested in order to find the best initial set up. An analysis for choosing the best value of divider (D) to be used for the design of combinational logic circuits has been performed and the results have been shown in the next section. The proposed method has been implemented in C++ and tested on a PC with the following characteristics: Pentium 4 at 3.00 GHz and 768 MB of RAM.

A. *Analysis of the parameter D*

For finding the best value of the divider D to be used during the evolution of digital logic circuits, experiments have been carried out. The value of D is directly related to the reproduction mechanism, since N/D populations (with N the total number of populations) are generated using the *best built population* and (N-N/D) are generated using the (μ/ρ, λ) evolution strategy. While investigating the number of generations required to fully evolve the circuit, it has been noticed that by decreasing the value of D (so increasing the number of populations that have generated using the *best built population,* see sections II.D-E) the number of generations is decreased. Figure 9 reports the evolution results of the circuit misex1.pla (8 inputs, 7 outputs chosen from the MCNC benchmark). From that graphic it can be see that by reducing the value of the divider (D) from 12 to 2, the number of required generations to fully evolve the logic circuit is decreased from 221 to 15 respectively. Using those results, a correct value of the divider (D) to be used during the evolution of the analyzed benchmarks has been extrapolated.

B. *Evolution of Digital Logic Circuits*

In this section the experimental results of the evolved logic circuits are presented. The circuits chosen are three multipliers (with 6, 8 and 10 inputs), three adders (with 6, 8 and 10 inputs) and three logic circuits taken from the MCNC benchmark, these circuits are the 9sym.pla (with 9 inputs), the Z5xp1.pla and the con1.pla (both with 7 inputs). The experiments have been carried out using the initial value exposed in Table 1 and each logic circuit has been fully evolved several times. In Table 2 the average of the results are shown. In that table "name" refers to the name of the circuit, "in" to the number of inputs, "out" the number of outputs, "pro" to the number of products of the truth table.

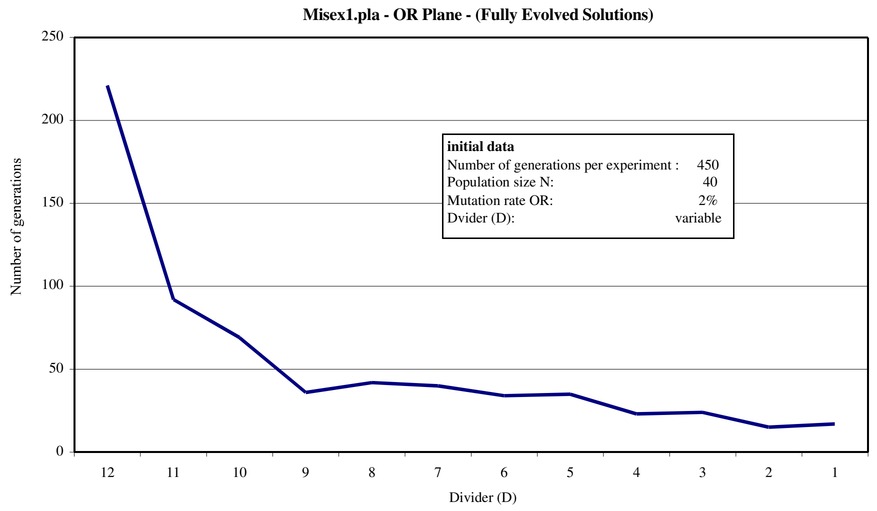


Figure 9. Number of generations required to fully evolve the OR plane of the Misex1.pla logic circuit (8 inputs and 7 outputs) using different values of the divider (D).

TABLE 1. INITIAL DATA USED FOR THE EXPERIMENTS

Name	Number of runs	Population size (N)	Max number of generations during the design stage	Divider (D)	Mutation Rate	
					AND PLANE	OR PLANE
Mult3	30	40	250	N/2	2%	2%
Mult4	30	40	500	N/2	2%	2%
Mult5	20	40	500	N/2	2%	2%
Adder3	50	40	250	N/2	2%	2%
Adder4	50	40	500	N/2	2%	2%
Adder5	25	40	500	N/2	2%	2%
9sym	50	40	450	N/2	2%	2%
Con1	50	40	450	N/2	2%	2%
Z5xp1	50	40	450	N/2	2%	2%

Table 2. Experimental results, three different benchmarks have used for testing the proposed algorithm "in" is the number of inputs, "out" the number of outputs "pro" the number of products or input-output combinations. FV stands for fitness value and Avg. stands for average.

Info circuit					**Design stage (FV=100)**		**Optimization stage**			
Name	in	out	pro	Plane	Avg. NG	Avg. Time [s]	Average of			
							NG	FV	Time [s]	pro
Mult3	6	6	64	AND	37.26	0.12	3,706.2	414.6	93.1	33.6
				OR	15.6	0.08				
Mult4	8	8	256	AND	59.26	0.87	25,542.4	1,916.1	1,785.1	137.0
				OR	25.5	0.57				
Mult5	10	10	1024	AND	90.5	7.15	49,982.9	8,266.0	36,503	607.6
				OR	38.0	4.23				
Adder3	6	4	64	AND	36.24	0.10	3,012.7	428.0	34.4	31.0
				OR	8.32	0.03				
Adder4	8	5	256	AND	60.2	0.85	9,980.9	2,170,3	335.83	70.6
				OR	16.36	0.33				
Adder5	10	6	1024	AND	84.12	5.68	49,979.5	11,425.3	11,135.4	167.6
				OR	23.72	2.22				
9sym	9	1	512	AND	72.48	0.86	9,991.9	3,829.1	359.4	102.3
				OR	1.00	0.01				
Con1	7	2	128	AND	47.06	0.14	4,841.0	1,001.5	33.8	9
				OR	1.00	0.01				
Z5xp1	7	10	128	AND	46.8	0.46	9,919.2	1,135.1	199.3	73.5
				OR	29.88	0.38				

"Avg. NG" refers to the average number of generations required to fully design the examined logic circuits; while "Avg. Time [s]" is the average time for the simulation in seconds. "FV" refers to the fitness value reached during the evolution stage. The experimental results show that the proposed algorithm is very efficient in terms of the number of generations. For example the average number of generations for evolving a circuit with 8 inputs (256 combinations) and 9 inputs (512 combinations) is less than 80. For the design of circuits with 10 inputs the number of required generations is less than 130. Furthermore the proposed algorithm also produces exceptionally optimized logic circuits.

V. Analysis of the Experimental Results and Benefits of the Proposed Genetic Algorithm

As proven by the experimental results, the benefits of the proposed method lie in producing the desired functionality of the logic circuits in a few generations (see Table 2) and it also optimizes the evolved circuits in terms of number of products more effectively. In Table 3 the reduction expressed as a percentage of the number of products of the evolved logic circuits is given. The best result is achieved with the evolution of the combinatorial logic circuit “con1”; a reduction of the number of products of 92% is reached. Comparing these results (number of generations to fully evolve the combinational logic circuits and number of input-output combinations at the end of the optimization stage) with the results of other similar evolution algorithms used for the evolution of digital logic circuits as:

- Evolution of PLA using (1+λ) evolution strategy with dynamic mutation [18] (the evolution of several logic circuits is achieved. The best optimization in terms of number of products is 13% for the 3 bit multiplier. For the same circuit the proposed method reaches a reduction of 40%).

- Cartesian Genetic Programming with (1+λ) evolution strategy (successfully used to evolve a 4 bit multiplier after 643,274,721 generations. The evolution was performed based on a FPGA structure [26]).

- Embedded Cartesian Genetic Programming ECGP [16] (the evolution of an 8 inputs circuit is accomplished after an average of 135,056 generations).

- Bi-directional Incremental Evolution [9] (successfully used for the evolution of logic circuits taken from the MCNC benchmark. The most complex circuit evolved was the Z5xp1 with 7 inputs; the number of required generations was 5,000,000).

- Tournament selection genetic algorithm [28]. (In [29] the evolution of a 4 bit multiplier, using an average of half million of generation, is shown).

- Traceless Genetic Programming [20] (the evolution of a 7 inputs circuit with 4918 generations is shown in [20]).

- Increased Complexity Evolution [8] together with partitioned training vector and partitioned training set [27] (Successfully evolved a 5-bit multiplier using more than one million of generations. The exact number of required generations is not given in [27]).

reveal that the proposed algorithm allows a faster evolution and provides better optimized logic circuits. Other researchers have used genetic algorithms to design digital filters, for example Zhang uses Cartesian Genetic Programming Reconfigurable Architecture with 5,000 generations for the evolution of digital filter, see [31]. However a comparison with filter design is not given because of the different complexities and characteristics of the evolved circuits. In this paper a comparison of evolvable hardware techniques used for the evolution of multipliers and adders is given, other evolved circuits such as filters, even parity circuits etc is not given, since the proposed algorithm is used here to design multipliers and adders. If it was used to design digital filters its results would have been compared with other evolved filters. It should be noted that the evolved logic circuits, as the 5 bit multipliers, the 5 bit adders are the most complex circuits ever evolved using a stand alone genetic algorithm. Only Stomeo in [2] was able to evolve larger circuits such as the 6 bit multiplier using the GDD for FPGA structures. A comparison with traditional synthesis approaches, which are very effective and produce impressive results, is not given since the aims of the paper are to present a new genetic algorithm applied for the evolution of combinational logic circuits, to show that is possible to have cost free design (this is because the designer should only provide the truth table of the desired circuit) and to compare the obtained designs with other automatic and self-reconfigurable methods. It is true that now traditional approaches are able to design circuits with thousand of inputs, but it is also true that those designs are very expensive and only very well prepared designers are able to optimize them, which is impossible and/or very expensive when human intervention is impractical. Moreover traditional hardware is notorious for its inflexibility, which makes it impossible to change the hardware structure and its functionality once designed. Otherwise, using the proposed approach everybody is able to design logic circuits, even users with no electronic knowledge are able to design and optimize those circuits, which makes it particularly interesting in applications where different hardware structures are required over time.

TABLE 3. REDUCTION OF THE NUMBER OF PRODUCTS (PRO) FOR THE OPTIMIZED LOGIC CIRCUITS

Name	in	out	Avg. pro design	Average pro optimization	Reduction [%]
Mult3	6	6	64	33.6	47.50
Mult4	8	8	256	136.1	46.84
Mult5	10	10	1024	607.6	40.66
Adder3	6	4	64	31.0	51.56
Adder4	8	5	256	102.6	59.92
Adder5	10	6	1024	618.5	39.60
9sym	9	1	512	102.3	80.02
Con1	7	2	128	9.0	92.97
Z5xp1	7	10	128	73.5	42.58

VI. Conclusion

In this paper a new genetic algorithm for the design of digital logic circuits based on a programmable logic array structure has been illustrated. The paper also describes the fitness function calculation, together with all the phases for generating and optimizing logic circuits. The proposed algorithm has been extrinsically tested with three different benchmarks, the multiplier and the adder, mostly common within the evolutionary computation community, and the MCNC benchmark, traditionally used in logic design area. The experimental results reveal that the proposed algorithm is able to design the desired logic circuits within a few generations. It also produces well optimized solutions. As an example, the design of the 5 bit multiplier is accomplished within an average of 138 generations and the analysis of the optimized design reveals that the number of products is reduced by 40%. The benefits of the proposed method are highlighted and a comparison with other methods is also given, showing that the proposed algorithm allows more rapid circuits design.

Acknowledgment

The authors thank the BIIS research group at Brunel University, United Kingdom and A. M. Walsh.

References

[1] 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.

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

[3] V. K. Vassilev, J. F. Miller "Scalability problems of digital circuit evolution evolvability and efficient designs" Proceedings of The Second Proceedings of The Second NASA/DoD Workshop on Evolvable Hardware, 2000. Los Alamitos, CA: IEEE Computer Society. 13-15 July 2000. Pages: 55 – 64.

[4] J. F. Miller, P. Thomson. "A Developmental Method for Growing Graphs and Circuits". *Fifth International Conference on Evolvable Systems: From Biology to Hardware*, Trondheim, March 17-20 2003.

[5] T. G. W. Gordon and P. J. Bentley. "Development Brings Scalability to Hardware Evolution". *Proceeding of 2005 NASA/DoD Conference on Evolvable Hardware*. 29-01 June 2005. Washington DC, USA. IEEE Computer Society. Pages: 272 – 279.

[6] A. Thompson, I. Harvey, and P. Husbands. "Unconstrained evolution and hard consequences," in Toward Evolvable Hardware: The Evolutionary Engineering Approach, vol. 1062. E. Sanchez and M. Tomassini, Eds. Berlin, Germany: Springer-Verlag, 1996. Pages: 136 – 165.

[7] T. Higuchi, M. Iwata, I. Kaijitani, M. Murakawa, S. Yoshizawa, and T. Furuya, "Hardware evolution at gate and function level," *Proc. Int. Conf. Biologically Inspired Autonomous Syst.: Computation, Cognition Action*, Durham, NC, 1996.

[8] J. Torresen. "A Divide-and-Conquer Approach to Evolvable Hardware". *Second International Conference on Evolvable Hardware (ICES98)*, Springer LNCS 1478, 1998, Lausanne, Switzerland.

[9] T. Kalganova, "Bidirectional incremental evolution in evolvable hardware". *Proceedings of The Second NASA/DoD Workshop on Evolvable Hardware, 2000.* Los Alamitos, CA: IEEE Computer Society. 13-15 July 2000. Pages: 65 – 74.

[10] J. Miller. "An empirical study of the efficiency of learning Boolean functions using a Cartesian genetic programming approach" *In Proc. of the Genetic and Evolutionary Computation Conference,* volume 1, Orlando, USA, July 1999. Pages: 1135 – 1142.

[11] A. M. Tyrrell, R. A. Krohling, Y. Zhou. "Evolutionary algorithm for the promotion of evolvable hardware". *IEE Proceedings of Computers and Digital Technique.* Volume 151, Issue 4, 18 July 2004. Pages: 267 – 275.

[12] A. Thompson, P. Layzell, R. S. Zebulum. "Explorations in design space: unconventional electronics design through artificial evolution". *IEEE Transactions on Evolutionary Computation*, Volume 3, Issue 3, Sept. 1999. Pages: 167 – 196.

[13] K. A. Vinger, J. Torresen. "Implementing Evolution of FIR-Filters Efficiently in an FPGA". *In proc. of 2003 NASA/DoD Conf. on Evolvable Hardware.* Chicago, USA. July, 2003. Pages: 26 – 29.

[14] D. A. Gwaltney, K. Dutton. "A VHDL core for intrinsic evolution of discrete time filters with signal feedback". Proceedings of 2005 NASA/DoD Conference on Evolvable Hardware. 29 June-1 July 2005. Pages: 43 – 50.

[15] D. Keymeulen, R. S. Zebulum, Y. Jin, A. Stoica. "Fault-tolerant evolvable hardware using field-programmable transistor arrays". *IEEE Transactions on Reliability*, Volume 49, Issue 3, Sept 2000. Pages: 305 – 316.

[16] A. J. Walker and J. 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] B. L. Hounsell, T. Arslan. "Evolutionary design and adaptation of digital filters within an embedded fault tolerant hardware platform". *Proceedings of The Third NASA/DoD Workshop on Evolvable Hardware, 2001.* 12-14 July 2001. Pages: 127 – 135.

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

[19] C. Lambert, T. Kalganova, E. Stomeo. "FPGA-based Systems for Evolvable Hardware". International Conference on Computer Science, ICCS'06. Vienna, Austria. March 29-31, 2006. Pages: 114 – 117.

[20] 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.

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

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

[23] D. E. Goldberg. *Genetic algorithm in search, optimization and machine learning*. Addison-Wesley Publishing Company, Incorporated, Reading, Massachusetts, 1989.

[24] M. D. Vose. *The simple Genetic Algorithm: Foundations and Theory.* MIT Press, Cambridge, MA, 1999.

[25] S. Yang. "Logic synthesis and optimisation benchmark user guide version 3.0, MCNC, 1991".

[26] D. Job, V. K. Vassilev, J. Miller. "Towards the automatic design of more efficient digital circuits". *Proceedings of The Second NASA/DoD Workshop on Evolvable Hardware, 2000.* IEEE Computer Society. 13-15 July 2000. Pages: 151 – 160.

[27] J. Torresen, "Evolving multiplier circuits by training set and training vector partitioning". *In proc. of Fifth Int. Conf. on Evolvable Hardware*, Springer LNCS 2606. March 2003. Pages: 228 – 237.

[28] Miller, B.L., and Goldberg, D.E.: 'Genetic algorithms, tournament selection, and the effects of noise', Complex Syst., 1995, 9, Pages: 193 – 212.

[29] M. Hartmann, P. C. Haddow. "Evolution of fault-tolerant and noise robust digital designs". *IEE Proceedings Computers and Digital Techniques*, Volume 151, Issue 4, 18 July 2004. Pages: 287 – 294.

[30] Y. Zhang, S. L. Smith, A. M. Tyrrell. "Digital circuit design using intrinsic evolvable hardware". *Proceeding of 2004 NASA/DoD Conference on Evolvable Hardware. Seattle, Washington, USA. IEEE Computer Society*. 24-26 June 2004. Pages: 55 – 62.