Generalized Disjunction Decomposition for the Evolution of Programmable Logic Array Structures

Emanuele Stomeo, Tatiana Kalganova, Cyrille Lambert
School of Engineering and Design
Brunel University
Uxbridge Middlesex, United Kingdom
stomeo@ieee.org

Abstract

Evolvable hardware refers to a self reconfigurable electronic circuit, where the circuit configuration is under the control of an evolutionary algorithm. Evolvable hardware has shown one of its main deficiencies, when applied to solving real world applications, to be scalability. In the past few years several techniques have been proposed to avoid and/or solve this problem. Generalized disjunction decomposition (GDD) is one of these proposed methods. GDD was successful for the evolution of large combinational logic circuits based on a FPGA structure when used together with bi-directional incremental evolution and with (1+λ) evolution strategy. In this paper a modified generalized disjunction decomposition, together with a recently introduced multi-population genetic algorithm, are implemented and tested for its scalability for solving large combinational logic circuits based on Programmable Logic Array (PLA) structures.

1. Introduction

Evolvable hardware [1]–[3] is a technique to automatically design electronic circuits [4], robot controllers [5][6], antennas [7][8] etc. using evolutionary algorithms [9]–[11]. Since the beginning of the 1990s, several projects have been initiated and several researchers and research groups are showing an interest in this new discipline and trying to find a solution to evolutionary design problems, which are mainly the scalability [12] [13]. The term scalability has been used to describe how the size of the problem will influence the performance of algorithms [1] [14]. Evolvable hardware systems are not scalable because of:

- the genotype length, which increases with the problem size. The genotype is the genetic composition of an individual that takes part in the evolution process for the design of circuits; the bigger the desired electronic circuit, the lengthier and more complex the genotype.
- the time required for fitness evaluation, which increases rapidly with the size of the desired evolvable circuits.

Referring to the evolution of digital circuits, the length of the genotype increases with the number of logic gates used during the evolution and the permitted connectivity between logic gates. The time necessary for the fitness evaluation is not scalable because it is exponentially dependent on the number of inputs of the system that should be evolved. If the number of inputs increases linearly, the number of input-output combinations, which represents the description of the digital logic circuit's problem, increases by the power of 2. Consequently, as the number of inputs increases, the system needs more time to produce new potential solutions, to evaluate them and to select new individuals. Several approaches have been introduced in order to overcome these problems and big improvements have been obtained. For instance,

- increased complexity evolution [16]: evolve several different functions using simple building blocks, then evolve the systems using the previously evolved functions. This method was further improved with the introduction of the training vector and partioned training set [26], which allows the evolution of the 5-bits multiplier.

- bi-directional incremental evolution [17]: evolve a system by gradually decomposing it using the output and Shannon decomposition [17]. Once the simpler sub-systems are fully evolved they are merged together to recover the desired system.

- Function level evolution [18], which uses function sub-circuits as system building blocks

have been proposed as methods to improve the evolution of logic circuits and to reduce the required number of generations to obtain a fully functional solution to a given task. Although these methods have brought some benefits to the evolvable hardware field, the evolution of circuits with high numbers of inputs remains the central issue. In 2004 another technique [19], the generalized disjunction decomposition (GDD) was introduced to improve evolvability and scalability for the evolution of combinational logic circuits. GDD is a decomposition technique which can be implemented into any evolutionary algorithm used for the design of electronic circuits.

In [19] [20], GDD was extrinsically implemented into bi-directional incremental evolution (BIE) and the system "GDD+BIE" was able to design and optimize circuits based on a FPGA structure better than the design offered by similar evolutionary algorithms. Furthermore the BIE with the use of GDD was able to improve scalability.

In this paper the advantages brought by the use of GDD for the design and the optimization of logic circuits structures are shown. A modified version of GDD is here implemented together with a recently introduced genetic algorithm [21], which is designed for the evolution of PLA. A PLA structure (see Figure 1) has been chosen as it is a good structure for designing combinational logic circuits in VLSI and for its simplicity, regularity and flexibility. The chosen genetic algorithm will also be briefly described. The system implemented here: modified GDD with the GA proposed in [21] it is novel and it was never implemented before. The experimental results, which are statistically relevant since they have been obtained over several simulations, prove that the generalized disjunction decomposition can improve scalability and evolvability for the evolution of large logic circuits.

For the simulations multipliers and adders of different complexities, have been used. It has been decided to consider these tasks because they are widely used within the evolvable hardware community, hence an easier comparison with other evolutionary algorithm can be made.

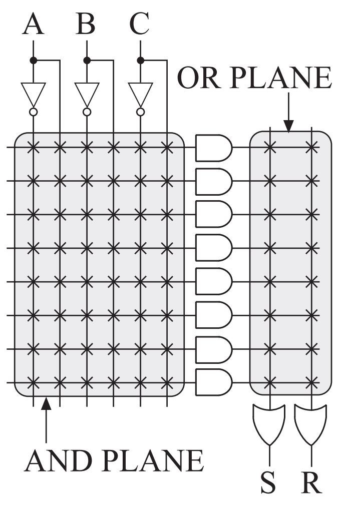


Figure 1. Example of a Programmable Logic Array (PLA) with 3 inputs and 2 outputs. In a PLA the AND plane and OR plane are programmable.

The paper is organized as follows: the next section considers the implemented genetic algorithm and the generalized disjunction decomposition which are both used for the evolution of logic circuits based on a PLA structure. Section 3 describes the system setup used for the simulations. Section 4 shows the experimental results. Section 5 concludes this paper and provides a summary of the key conclusions.

2. Methods used for the Design of PLA

In this section the generalized disjunction decomposition together with the chosen genetic algorithm for the evolution of combinational logic circuits are discussed. The first sub-section gives a brief description of the multi-population genetic algorithm which uses two different mechanisms to generate new individuals [21]. The second sub-section gives a concise introduction on the generalized disjunction decomposition; a complete description of the GDD is given in [20].

2.1 Evolutionary Algorithm for the Design of PLA

The genetic algorithm briefly described in this section is intended for the design of combinational logic circuits based on a PLA structure. The algorithm makes use of a multi population system (see Figure 2) and it also shares some characteristics with:

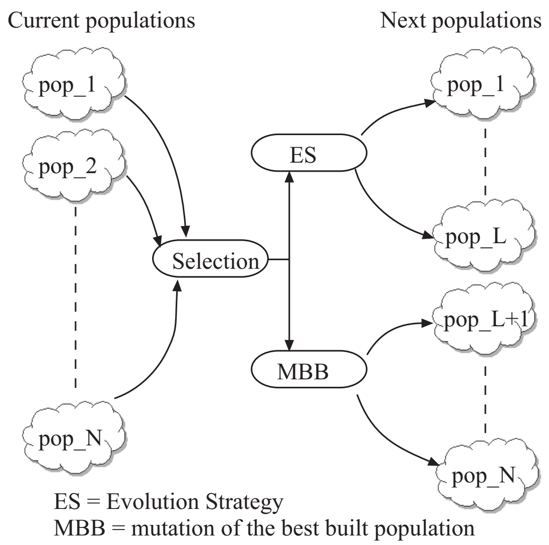


Figure 2. Basic schema of the multi population genetic algorithm [21].

- the Cooperative Coevolution Architecture [22] introduced by Potter and De Jong. This architecture models an ecosystem consisting of two or more species, which are genetically isolated and working together to reach the final goal. The species are in cooperation and at the same time in competition. This is because each species receives credit for the achievement of a final goal (therefore different species cooperate together) and also in relation to how good they are in comparison with other species (the species compete with each other to obtain more credits).
- (μ/ρ, λ) evolution strategy [23][24]; where the letter μ means the total number of parents within the population, ρ refers to the number of parents which will be taken into consideration for the reproduction of other individuals for the future generation, and λ is the number of offspring. During the evolutionary process μ individuals are tested for their efficiency and the best ρ are chosen to create a new population of other λ individuals that are used to replace the previous population. Therefore, the new population is generated from the best individuals of the previous generation using the mutation operator. Mutation consists of flipping some genes of the individual's chromosome.

Furthermore, the chosen genetic algorithm has got something completely new: the construction of the chromosome from the elitism's pool (which contains the best chromosome of each population). A more detailed description of this algorithm is given in [21]. A particularity of the algorithm used is the reproduction mechanism that is depicted in Figure 2. Some of the populations are replaced using the (μ/ρ, λ) evolution strategy [23][24] and some by mutating the best built population, which is the population created by collecting the best chromosomes from each region of the solution space.

2.2 Generalized Disjunction Decomposition

Generalized disjunction decomposition [19][20] was introduced as an independent method (independent because it can be implemented within different evolutionary algorithms) which enhances the performance, based on number of generations and evolution time, of the evolutionary algorithm used for the design of electronic circuits. GDD also improves the scalability and allows the design of large circuits never before evolved, as the 17-bit even parity circuit, the 6-bit multiplier and the ALU4 which is a circuit with 14 inputs taken from the MCNC library [25]. The scalability problem limits the size of the circuit that may be evolved. In [20] the aims of the GDD have been proven. In order to show how GDD works, supposing that the circuit in Figure 3 with n inputs and m outputs needs to be evolved. The number of required generations for the design of that circuit is mainly dependent on the number of inputs rather than the number of outputs [19]. Therefore to improve the scalability the system in Figure 3 is decomposed into two subsystems, as shown in Figure 4; the sub-system G with *r* inputs and *s* outputs (see Equation 1) and the subsystem H, which is made of multiplexers. After the GDD decomposition, both subsystems could be evolved using any evolutionary technique. The ease of evolving the sub-systems G and H depends on the evolutionary algorithm chosen.

$$s = m \cdot 2^{n-r} \tag{1}$$

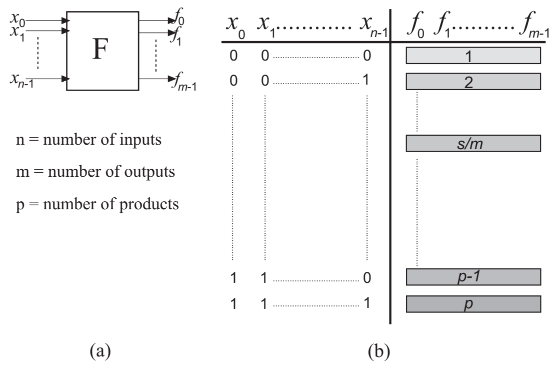
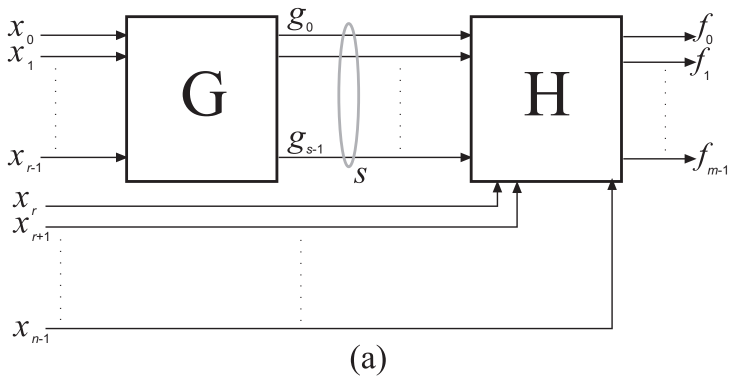


Figure 3. General description of a logic circuit. (a) the schemata of the evolved system; (b) the truth table for the evolved system.



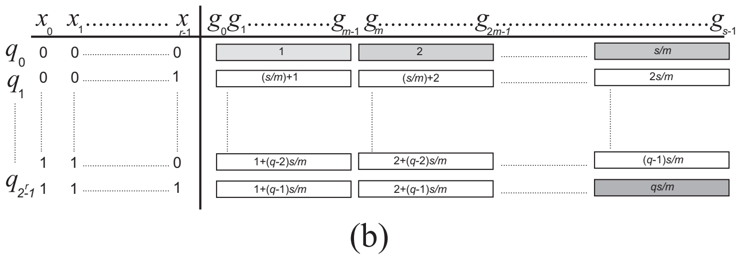


Figure 4. Generalized disjunction decomposition (a) Connections between the two subsystems. (b) Truth table of the subsystem G. Picture taken from [20].

3 System Setup and Initial Data

The system used for the experiments is the generalized disjunction decomposition together with the briefly described genetic algorithm. This system is implemented in C++ and tested in an extrinsic environment using a desktop PC with the following characteristics: Pentium 4 at 3.00 GHz and 768 MB of RAM. The initial data for the simulations are shown in Table 1, where "Logic circuits" refers to the circuit's name, then the number of runs is illustrated. In order to have statistically relevant results each circuit has been evolved 15 times. In this table the number of generations for the evolution process as well as the mutation rate and the number of populations selected for the experiments are also shown. The values for the mutation rate and the population size are selected after a primary testing phase of the algorithm; the chosen values are tuned in order to gain the best performance.

4 Experimental results

In this section the experimental results of the combinational logic circuits evolved by using two different systems are shown. For the first system the circuits are evolved using the briefly described genetic algorithm only (see Section 2.1), a larger description is given in [21].

Table 1. Initial data for the experimental results

Logic circuits	Num. of runs	Num. of generations	Mutatio n rate	Populatio n
Mult4	15	25,000	2.0%	40
Mult5	15	50,000	2.0%	40
Mult6	15	250,000	2.0%	40
Adder4	15	10,000	2.0%	40
Adder5	15	50,000	2.0%	40
Adder6	15	250,000	2.0%	40

The second system is made by decomposing the initial circuit using the generalized disjunction decomposition and then applies the described genetic algorithm, thus GDD+GA. The aims of these experiments are to demonstrate that the proposed system requires less time and provides better optimized logic circuits (this is based on the number of required products, or input-output combinations). Since the system is able to fully design and optimize larger logic circuits it also improves scalability. In Table 2 the experimental results are summarized. As can be seen from that table, each circuit has been evolved using two different methods. One using genetic algorithm (GA) and the other using GA implemented into the generalized disjunction decomposition (GDD+GA). For the circuits Mult4, Mult5, Adder4 and Adder5 it is noticeable that the system GDD+GA requires less time to fully evolve the logic circuits. Furthermore, for those circuits the number of products at the end of the simulation is much smaller, which results in better optimized solutions. To better understand the quality of the evolved logic circuits, the evolution of the 5-bit multiplier using the GA and the system GDD+GA is considered and depicted in Figure 5. The initial circuit (see Figure 5.a) is described by a truth table with 1024 input combinations (also called products). After the evolution of several (exactly 15) runs the designed 5-bit multiplier contains an average of 607.6 products (see Figure 5.b). Each product represents one AND logic gate when implemented into the PLA. The initial circuit is now decomposed into two subsystems using the generalized disjunction decomposition (see Figure 5.c). At this point the newly created subsystems are both evolved using the described GA (see Figure 5.d). Table 2 reports the results of the evolution of the subsystem G; instead Table 3 shows the results of the subsystems H (the multiplexer part). Regarding the subsystem H, only 3 different multiplexers, with one, two and three control signals respectively, have been evolved. This because they are the only multiplexers used during the decomposition of the initial systems F. As can be seen from those results, only few logic gates are required to fully describe the behavior of the multiplexers. The multiplexers with 1 and 2 control signals are easily evolvable; therefore for the evolution of those circuits only the multi population genetic algorithm has been used. Since the evolution of MUX3 (a multiplexer with 3 control signals), with the use of the only genetic algorithm, requires lots of time (an average of 11,300 seconds), it has been decided to evolved them using the system GDD plus GA. This solution has brought a significantly reduction of evolution time (the new system requires and average of 1,100 seconds, it means a reduction of 89.7% of the computational time), see last two rows of the Table 3. From the experimental results (Table 2 and Table 3) can be noticed that

- a fast evolution of logic circuit is possible when the briefly described genetic algorithm [21] is used, few generations are required to fully evolve the desired circuit. Thus, it enhances the evolvability.
- with the use of the GDD, the evolved circuits are better optimized; consequently less logic gates are required. Furthermore, it can be seen that the system GDD+GA is able to fully design and optimizes the circuits Mult6 and Adder6, which are not evolved by the standalone genetic algorithm. Thus, the decomposition used improves scalability for the evolution of combinational logic circuits based on a PLA structure.

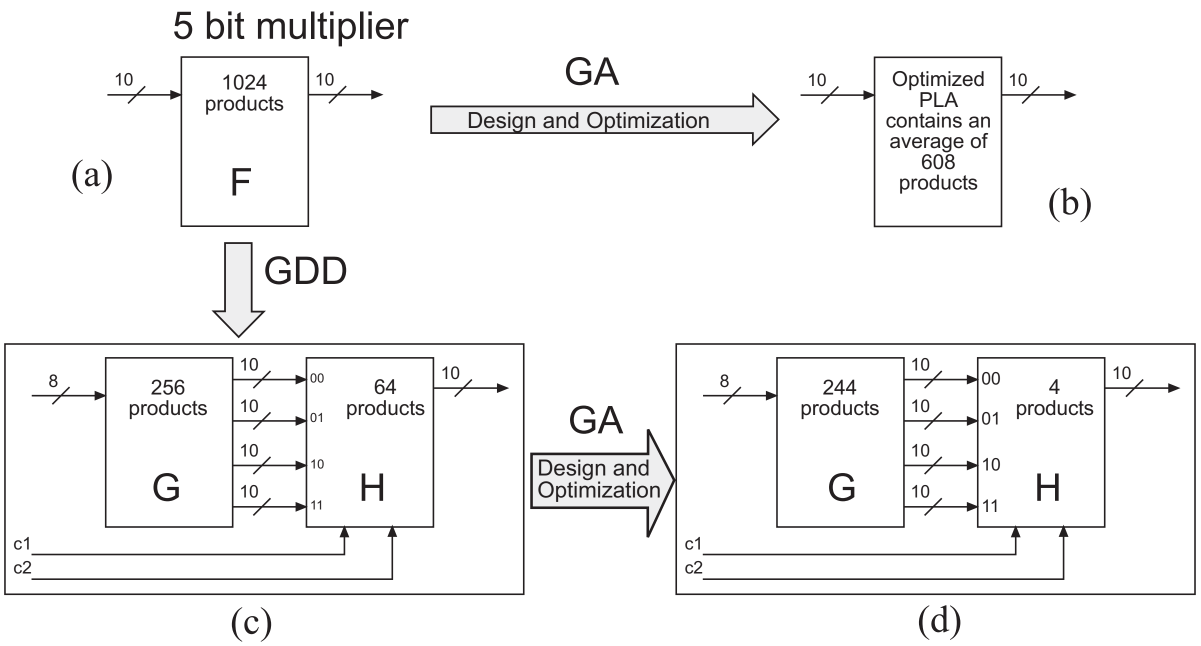


Figure 5. The evolution of the 5-bit multiplier using two different methods. GA stands for genetic algorithm and GDD for generalized disjunction decomposition. (a) Initial system with 1024 product lines, or input-combinations. Each product line represents and AND gate in the PLA. (b) PLA optimized by the genetic algorithm. (c) 5 bit multiplier after the GDD decomposition. (d) Optimization of the two subsystems using the genetic algorithm.

Table 2. Experimental results of the selected tasks. Each circuit has been evolved 15 times.

Task	Info circuits – Initial data before evolution					After evolution. Average of		
	Name	Method	Num. of Input	Num. of Output	Num. of Product	Num. of Generations	Time [s]	Num. ot Product
Multipliers	Mult4	GA	8	8	256	25,542.6	1,785.1	137.0
		GDD+GA	6	32	64	13,104.6	366.0	59.0
	Mult5	GA	10	10	1,024	49,982.9	8,266.3	607.6
		GDD+GA	8	40	256	49,729.1	5,362.7	244.1
	Mult6	GA	12	12	4,096	Not evolved		
		GDD+GA	9	96	512	47,816	364,752	504.0
Adders	Adder4	GA	8	5	256	9,980.9	2,170.3	75.4
		GDD+GA	6	20	64	9,870.1	637.0	56.8
	Adder5	GA	10	6	1,024	49,979.5	1,1425.9	167.6
		GDD+GA	7	48	128	49,899.5	2,098.5	121.1
	Adder6	GA	12	7	4,096	Not evolved		
		GDD+GA	9	56	512	249,891.4	93,925.1	368.0

Table 3. Evolution of the required multiplexers to be used as subsystem H when the generalized disjunction decomposition is used.

Info circuits – Initial data before evolution					After evolution. Average of		
Name	Method	Num. of Input	Num. of Output	Num. of Product	Num. of Generations	Time [s]	Num. of Product
MUX1	GA	3	1	8	103.0	1.0	2.0
MUX2	GA	6	1	64	2,506.7	24.1	4.0
MUX3	GA	11	1	2,048	49,992.1	11,334.5	9.7
	GDD+GA	9	4	512	41,226.2	1,159.6	8.0

5 Conclusion

This paper has shown the implementation of the generalized disjunction decomposition (GDD) for evolvable hardware into a multi-population genetic algorithm (GA). The proposed system has been used for the evolution of large combinational logic circuits based on a programmable logic array structure. Multipliers and adders of different complexities have been selected to be used for testing and analyzing the proposed method. The implemented system, as proven from the experimental results, has improved the evolution of all the tested combinational logic circuits in terms of processor time and in terms of the number of logic gates required for the system. Furthermore the presented method, multi population genetic algorithm with generalized disjunction decomposition, improves scalability: the 6-bit multiplier and 6-bit adder have been evolved several times. Every attempt to evolve those circuits was successful, i.e. 100% of the evolutions achieved high-quality results. Those circuits were only previously evolved thanks to the use of GDD implemented together with BIE for the evolution of FPGA structures. No other proposed evolutionary algorithms and/or decomposition strategies were able to produce such results in terms of computational time and overall size of the circuits.

6 Acknowledgement

The authors gratefully acknowledge the Engineering and Physical Sciences Research Council (EPSRC) for the financial support grant GR/S17178.

7. 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 Pages: 6 – 10.

[2] 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.

[3] L. Sekanina. *Evolvable Components: From Theory to Hardware Implementations.* Natural Computing Series. Springer-Verlag, 2004. ISBN 3-540-40377-9. Pages: 194.

[4] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, N. Salami, N. Kajihara, N. Otsu. “Real-world applications of analog and digital evolvable hardware” *IEEE*

Transactions on Evolutionary Computation, Vol. 3 Issue: 3, Sept. 1999. Pages: 220 – 235.

[5] 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.

[6] R. J. Terrile, H. Aghazarian, M. I. Ferguson, W. Fink, T. L. Huntsberger, D. Keymeulen, G. Klimeck, M. A. Kordon, L. Seungwon , P. von Allmen. “Evolutionary Computation Technologies for the Automated Design of Space Systems”. *Proceeding of 2005 NASA/DoD Conference on Evolvable Hardware*. 29-01 June 2005. Washington DC, USA. IEEE Computer Society. Pages: 131 – 138.

[7] S. V. Hum, M. Okoniewski, R. J. Davies. “An Evolvable Antenna Platform Based on Reconfigurable Reflectarrays”. *Proceeding of 2005 NASA/DoD Conference on Evolvable Hardware*. 29 - 01 June 2005. Washington DC, USA. IEEE Computer Society. Pages: 139 – 146.

[8] J. D. Lohn, G. S.Hornby, D. S. Linden. “An evolved antenna for deployment on NASA's Space Technology 5 Mission”. *Genetic Programming Theory and Practice II. Boston: Kluwer Academic Publishers.* Chapter 18.

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

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

[11] D. B Fogel. “What is evolutionary computation?” *Spectrum, IEEE* Volume 37, Issue 2, Feb. 2000. Pages: 26, 28 – 32.

[12] 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.* IEEE Computer Society. 13-15 July 2000. Pages: 55 – 64.

[13] C. A. Coello, A. D. Christiansen, A. A. Hernández, “Towards automated evolutionary design of combinational circuits”, *Computers and Electrical Engineering,* Pergamon Press, Vol. 27, No. 1. January 2001. Pages: 1–28.

[14] S. Xian-He, D. T. Rover. “Scalability of parallel algorithm-machine combinations”. *IEEE Transactions on Parallel and Distributed Systems*, Volume 5, Issue 6, June 1994. Pages: 599 – 613.

[15] Lee Altenberg. *The Evolution of Evolvability in Genetic Programming*. Chapter 3 in Advances in Genetic Programming, ed. Kenneth Kinnear. MIT Press, Cambridge, 1994. Pages: 47-74.

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

[17] 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.

[18] 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.

[19] E. Stomeo and T. Kalganova. “Improving EHW performance introducing a new decomposition strategy.” *2004 IEEE Conference on Cybernetics and Intelligent Systems*. Singapore 1-3 December 2004. Publisher IEEE Inc., New York, NY 10016-5997, United States. Pages 439 – 444.

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

[21] E. Stomeo, T. Kalganova, C. Lambert. “A Novel Genetic Algorithm for Evolvable Hardware”. *IEEE World Congress on Computational Intelligence.* IEEE CEC 2006. (Accepted for publication).

[22] Mitchell A. Potter Kenneth A. De Jong. *Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. Evolutionary Computation.* 8 (1): Pages: 1-29, 2000, The MIT Press.

[23] 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.

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

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

[26] J. Torresen, “Evolving multiplier circuits by training set and training vector partitioning”. *In proc. of Fifth Int. Conf. on Evolvable Hardware (ICES03)*, Springer LNCS 2606. March 2003. Pages: 228-237.