Stomeo E., T. Kalganova, C. Lambert (2005) Analysis of Genotype Size for an Evolvable Hardware System. Proc. of the 2nd Int. Conf. on Computational Intelligence, ICCI 2005, August 2005, Enformatika pp.70 - 74.

Analysis of Genotype Size for an Evolvable Hardware System

Emanuele Stomeo, Tatiana Kalganova, Cyrille Lambert

***Abstract*—**The evolution of logic circuits, which falls under the heading of evolvable hardware, is carried out by evolutionary algorithms. These algorithms are able to automatically configure reconfigurable devices. One of main difficulties in developing evolvable hardware with the ability to design functional electrical circuits is to choose the most favourable EA features such as fitness function, chromosome representations, population size, genetic operators and individual selection. Until now several researchers from the evolvable hardware community have used and tuned these parameters and various rules on how to select the value of a particular parameter have been proposed. However, to date, no one has presented a study regarding the size of the chromosome representation (circuit layout) to be used as a platform for the evolution in order to increase the evolvability, reduce the number of generations and optimize the digital logic circuits through reducing the number of logic gates. In this paper this topic has been thoroughly investigated and the optimal parameters for these EA features have been proposed. The evolution of logic circuits has been carried out by an extrinsic evolvable hardware system which uses (1+λ) evolution strategy as the core of the evolution.

***Keywords*—**Evolvable hardware, genotype size, computational intelligence, design of logic circuits.

I. Introduction

Evolvable hardware (EHW) [1] techniques introduced by H. de Garis [2] almost a decade ago enable the automatic design of antennas [3], [4], electrical circuits (digital [5], [6] and analogue [7], [8], [9]), robot controllers [10] etc. Evolvable hardware is able to auto-design and self-reconfigure the functionality of hardware systems thanks to the use of evolutionary algorithms [11], [12], [13] which are generally used to solve search and optimization problems. The major components of evolutionary algorithms are the genetics operators, the mechanism of selections and the chromosome representations. The simplest evolutionary algorithm is shown in Fig. 1, while in Fig. 2 the simplest evolvable hardware system is presented. Evolvable hardware, as proposed by Torresen [14], Andersen [15] and Gordon and Bentley [16], can be classified in several classes (see Table 1), depending on: evolutionary algorithm, target technology, building block levels, fitness evaluation and evolutionary process. Extrinsic EHW refers to a system whereby the evolutionary algorithm is performed in software, or in a dedicated chip, distinct from the chip where the chromosome will be downloaded. Intrinsic EHW describes situations where the evolutionary algorithm is implemented in hardware but into a different chip than that from which the evolving design is running. In a complete hardware the evolutionary algorithm is implemented on the same chip as the evolving design and mixtrinsic evolvable hardware [17] is a hybrid combination of intrinsic and extrinsic methods.

Manuscript received July 15, 2005. This work was supported in part by the EPSRC under grant number GR/S17178/.

E. Stomeo, C. Lambert are PhD students at School of Engineering and Design, Brunel University, West London, UK. T. Kalganova is lecture at the same university. UB8 2TR, Uxbridge, Middlesex, UK. (Tel: 0044 01895 266777; e-mail: emanuele.stomeo@brunel.ac.uk).

```
procedure EA
{
   t=0;
   produce an initial population on individuals P(t)
   evaluate the fitness of the population P(t)
   while (termination conditions are met)
   {
      t=t+1;
      select fitter individuals for reproduction from P(t-1);
      reproduce individuals in P(t);
      evaluate the fitness of the new individuals P(t);

   }
}
```

Fig. 1. Classic Evolutionary algorithm

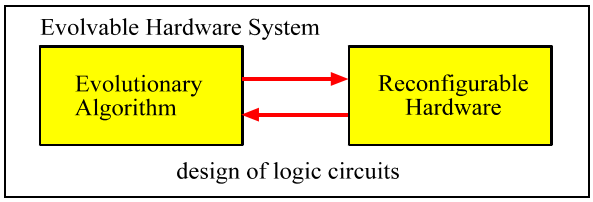

Fig. 2. Basic evolvable hardware system

Table 1. Classification in evolvable hardware

EA \ EHW	intrinsic	extrinsic	mixtrinsic	complete
chromosome representation	hardware	software	hardware software	hardware
evolutionary algorithm	software	software	software	hardware
evaluation	hardware	software	hardware software	hardware

While examining the progress made during the last decade by the evolvable hardware community in relation to the

evolution of digital logic circuits [18], [19], [20], it may be noticed that researchers often appear to tune their EA's parameters in order to make their own EA more scalable when compared with that of others. Therefore in several publications there are sentences saying: "we used a Population size = …. Mutation rate =….Crossover rate =…. Genotype size = ……" without explaining why those values have been chosen or without presenting any statistical analysis of those parameters. Therefore, to avoid this downside, a deep analysis of how to choose the genotype size for evolving digital logic circuits of different complexity using an extrinsic evolvable hardware system is provided in this paper. Our extrinsic EHW is based on the used of (1+λ) rudimentary evolution strategy, already tested for its efficiency in [5], [21] for the evolution of digital logic circuits.

The purpose of this paper is to provide a statistical analysis of the chromosome representation for evolving digital circuits using an extrinsic evolvable hardware. Different chromosome representations are analyzed and stressed for finding the best chromosome representations that improve evolvability, reduce the number of generations and find the optimal solution (the design which gives the fewest number of logic gates). Furthermore, because an extrinsic evolvable hardware system has been used, which is based on a FPGA structure, therefore the results here reported will be useful and easy transferable to an intrinsic evolvable hardware system.

The remaining sections of this paper are organized as follows: the next section introduces the extrinsic evolvable hardware used for the evolution of logic circuits. In that section the evolutionary algorithm used, the chromosome representation and the fitness functions are also presented. Section 3 provides the initial set up for the experiment together with the experimental results. The conclusions are given in section 4.

II. Extrinsic Evolvable Hardware

In this section the extrinsic evolvable hardware system (evolutionary algorithm, fitness function and chromosome representation) used for the evolution of digital logic circuits is presented. The evolutionary algorithm chosen is the (1+λ) evolution strategy. It was decided to consider this algorithm because it has been widely and successfully used as a core for several evolutionary designs of logic circuits [5], [21], [6]. A dynamic fitness function previously tested for its performance in [5], together with Cartesian Genetic Programming [22] has also been selected.

A. Evolutionary Algorithm

All the experiments have been carried out using the (1+λ) rudimentary evolution strategy [5], [21], where λ represents the population size. Firstly all the chromosomes are randomly initialized. Secondly, the fitness function of each individual is calculated and the fittest individual is selected. The new population is created by mutating the best chromosome (see Fig. 3). The operation of mutation consists of flipping some genes of the chromosome, where in this context the genes represent the behavior and the connections between the logic gates.

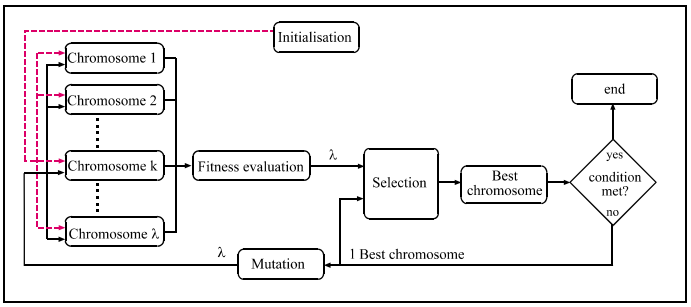

Fig. 3. (1+λ) rudimentary evolution strategy.

B. Chromosome Representation

The "design-evolution" of logic circuits has been carried out on a system which is based on a FPGA structure. Therefore the best chromosome representation is a rectangular array of logic gates which will be connected during the evolution. The unconnected logic gates are removed following the design. The logic gates in this rectangular array are: AND, OR, XOR, NOT and MUX, where MUX is a multiplexer with 2 inputs and one control signal. The chromosome is represented by a 3 level structure (see Fig. 4):

- Geometry level contains information about the *number of rows*, the *number of columns* of the rectangular array and the degree of internal connectivity, also referred to as *level-back parameter* [22]. The level-back parameter, or so called connectivity parameter, defines how many columns of cells to the left of the current column might have their outputs connected to the inputs of the current cell.

- Circuit level describes the array of cells and determines the circuit's outputs

- Gate level represents the structures of each cell in the circuit.

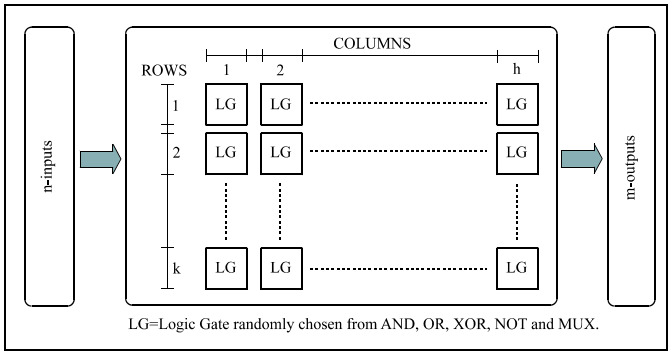

Fig. 4. Chromosome structure.

The connection between building blocks (combination of primitive logic gates) is in interactive and cascade mode. Each logic gate has up to 4 inputs. The structures of cascade and interactive building blocks are given in Fig. 5 and Fig. 6. Each of these blocks represents the combination of primitive logic gates.

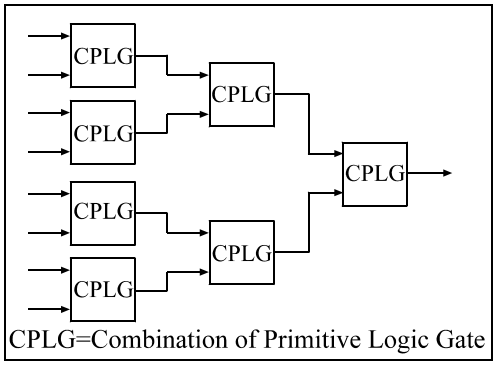

Fig. 5. Cascade building blocks.

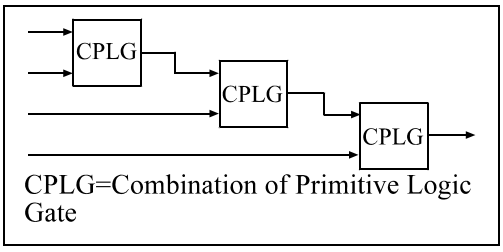

Fig. 6. Interactive building blocks.

C. Fitness Function

The fitness function is responsible for evaluating the quality of the logic circuits together with their functionality. In our experiments, a dynamic fitness function has been considered [5]. It has two main stages: first *design*, and secondly, once the circuit is fully functionally evolved, *optimization,* which leads to a reduced number of active logic gates used in the circuit configuration.

The dynamic fitness function *f* is calculated as:

$$f = \begin{cases} f_1 & f < 100 \quad \text{circuit design} \\ f_1 + f_2 & f \geq 100 \quad \text{circuit optimization} \end{cases} \tag{1}$$

where f_1 is a design criterion that defines the percentage of correct bits in the evolved circuit, f_2 is the optimization criterion for the optimization stage.

The fitness function for the functionality of the evolved circuit f_1, or so called design criterion is calculated as follows:

$$f_1 = \frac{100}{m \cdot p} \sum_{f_c}^{2^{n-1}} \sum_{i=0}^{m-1} 2^{i-1} \cdot \left| y_i - d_i \right| \tag{2}$$

where m and n are the number of outputs and the number of inputs of the given logic function respectively; p is the number of input-output combinations; y_i is the i^{th} digit of the output combination produced by the evaluation of the circuit, d_i is the desired output for the fitness case f_c. $|yi\text{-}di|$ is the absolute difference between the actual and the required outputs. The fitness function for the optimization stage is calculated as:

$$f_2 = \left(N_{\lg} - N_{a\lg}\right) \cdot N_{p\lg} \tag{3}$$

III. Experimental Results

Here, the experimental results obtained from the evolution of digital logic circuits are shown. The system used for the evolution is the extrinsic evolvable hardware model presented in the previous section. The aim of these experiments is to provide the evolvable hardware community with information regarding the circuit layout to be used in order to create better, faster and well optimized logic circuits evolved by the evolutionary algorithm. The experimental simulation was performed by taking into account the evolvability and the:

- Number of generations
- Fitness function
- Number of logic gates

required to completely evolve the functionality of the circuit by changing the circuit layout of the chromosome.

A. Settings: Initial Data

The initial data for the experiments are reported in Table 2; where the number of generations refers to the number of cycles of the EA run, the population size is the number of initial individuals which have been randomly generated, elitism and mutation rate are the genetic operators used. The number of runs refers to the number of times a single circuit has been evolved.

Table 2. Initial data for the experiments carried out using (1+λ) rudimentary evolution strategy.

Number of generations	5000
Population size	5
Elitism is applied	
Mutation rate	5%
Number of runs	50

The logic circuits evolved were randomly generated and fully defined by truth tables. The truth tables used to describe the logic circuits are compatible with the Berkeley format, see Fig. 7. Where .i specifies the number of inputs, .o the number of outputs, .p the number of product or input-output combinations and .e the end of file.

```
.i 3
.o 3
.p 8
000 010
001 011
010 101
011 101
100 011
101 110
110 101
111 011
.e
```

Fig. 7. Example of truth table (in Berkley format) used for the evolution of logic circuit with (1+λ) evolution strategy.

B. Results: Evolvability

In Fig. 8 the evolvability (the capability of the EA to fully functional evolve a given task) of a logic circuit with 3 inputs and 5 outputs is reported. From those results one may notice that the percentage of fully evolved logic circuits is lower when the circuit layout used for the evolution is a wider rectangular array. By varying the chromosome representation from 1 x100 to 10 x 10 the evolvability increases from 70% to 86%. Therefore a rectangular array of 10 rows by 10 columns performs better in terms of evolvability.

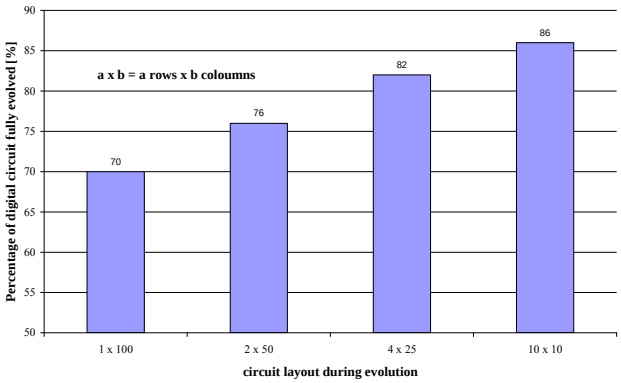

Fig. 8. Evolvability. In this figure the percentages of fully evolved digital logic circuit in relation with the change of the chromosome representations is given

C. Results: Number of Generations

In Fig. 9 the number of generations with different chromosome's sizes (rectangular array) required in order evolve a circuit with 3 inputs and 5 outputs are reported. In that figure it is observable that by widening the shape of the rectangular array (note that the number of total logic gates does not change, it is always 100) the number of generations required to fully evolve the circuit (fitness function reaches 100%) increases. For that particular circuit, the number of generations has been reduced from circa 4500 generations when the circuit layout is 1 x 100 (1 rows by 100 columns) to around 1700 generations when the circuit layout is 10 x 10.

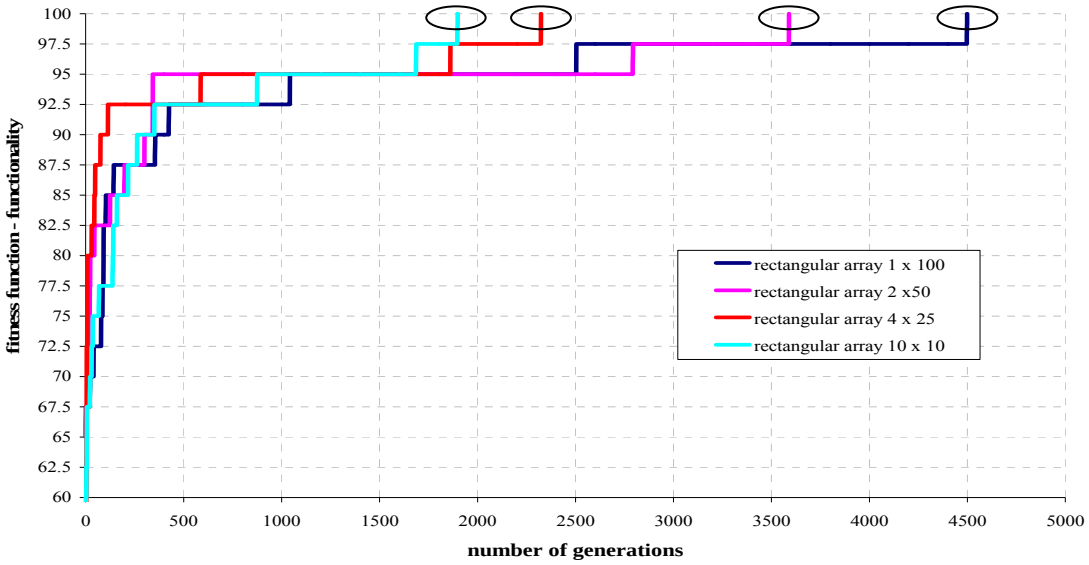

Fig. 9. Relationship between the number of generations and the fitness function for the evolution of a digital circuit with 3 inputs and 5 outputs for different chromosome representations is shown.

In Table 3 the average of 50 experiments (per each kind of logic circuit evolved and per each different circuit layout used) of the number of generations required to completely evolve the functionality of the circuit is shown. In Table 3 n, m and p refer to the number of inputs, outputs and input-output combinations respectively. In that table the best results, the evolution with fewer generations, is highlighted. From these results we recognize that the best circuit layout for the evolution of simple logic circuits is a rectangular array of 4 rows by 10 columns and 10 rows by 10 columns. Furthermore it appears to be clear that it is better to avoid evolution using a wider rectangular array or a string.

Table 3. Number of generations required to evolve logic circuits of different complexity with different chromosome representation.

Number of generations for fully functional evolving logic circuits							
Logic circuit				**Chromosome representations**			
Name	**n**	**m**	**p**	**1 x 100**	**2 x 50**	**4 x 25**	**10 x 10**
2-2	2	2	4	40	43.82	35.68	44.12
2-3	2	3	8	75.30	85.62	63.56	62.84
2-4	2	4	16	105.62	116.96	83.56	86.98
2-5	2	5	32	160.80	155.78	159.92	101.44
3-2	3	2	4	178.58	233.76	183.20	175.20
3-3	3	3	8	565.92	557.74	483.40	526.04
3-4	3	4	16	1567.90	1545.57	1352.40	1384.71
3-5	3	5	32	2439.29	2176.13	2111.51	2102.20

A. Results: Fitness Function

In this section the fitness function of the evolved logic circuits before and after the optimization stage, in relation to the different chromosome representations is presented. The evolved circuits have been optimized using equation 3. The highlighted results are the best obtained. The results again show that the rectangular array of 10 rows by 10 columns is superior to any other during evolution. After the optimization stage a big difference between different chromosome configurations cannot be found

Table 4. Average out of 50 experiments of the fitness function value obtained during the evolution of logic circuits of different complexity.

Fitness function for the fully functional solution							
Logic circuit				**Chromosome representations**			
Name	**n**	**m**	**p**	**1 x 100**	**2 x 50**	**4 x 25**	**10 x 10**
2-2	2	2	4	2749	2754	2745	2762
2-3	2	3	8	2735	2737	2739	2749
2-4	2	4	16	2729	2729	2730	2739
2-5	2	5	32	2717	2720	2726	2735
3-2	3	2	4	2751	2749	2747	2754
3-3	3	3	8	2732	2735	2735	2744
3-4	3	4	16	2724	2729	2727	2736
3-5	3	5	32	2712	2719	2723	2727

Table 5. Average out of 50 experiments of the fitness function value obtained after the optimization stage.

Fitness function for the fully functional solution after optimization							
Logic circuit				**Chromosome representations**			
Name	**n**	**m**	**p**	**1 x 100**	**2 x 50**	**4 x 25**	**10 x 10**
2-2	2	2	4	2796	2796	2796	2797
2-3	2	3	8	2795	2796	2796	2796
2-4	2	4	16	2794	2794	2795	2795
2-5	2	5	32	2794	2794	2794	2796
3-2	3	2	4	2792	2793	2793	2793
3-3	3	3	8	2789	2788	2789	2790
3-4	3	4	16	2772	2773	2776	2779
3-5	3	5	32	2752	2760	2764	2771

B. Results: Number of Active Logic Gates during Evolution and After Optimization Stage

This section shows the average (out of 50 experiments) number of logic gates required to fully functionally evolve the digital circuits. The experimental results found and reported in this section were obtained during the evolution stage, Table 6, and after the optimization stage, Table 7. The best results are highlighted. From those results again, we found that the best chromosome representation is a rectangular array with 10 rows and 10 columns.

Table 6. Average out of 50s experiment of the number of required logic gates during the evolution stage.

Number of active logic gates for the fully functional solution							
Logic circuit				**Chromosome representations**			
Name	**n**	**M**	**p**	**1 x 100**	**2 x 50**	**4 x 25**	**10 x 10**
2-2	2	2	4	23.66	21.44	25.38	17.78
2-3	2	3	8	29.10	28.80	27.52	23.52
2-4	2	4	16	32.94	31.90	31.78	27.50
2-5	2	5	32	37.94	36.66	33.72	29.40
3-2	3	2	4	21.64	22.94	23.52	20.52
3-3	3	3	8	30.92	29.92	29.26	25.20
3-4	3	4	16	34.81	31.69	32.28	28.65
3-5	3	5	32	40.00	36.97	36.07	33.19

Table 7. Average out of 50 experiments of the number of required logic gates after the optimization stage.

Number of active logic gates for the fully functional solution after optimization							
Logic circuit				**Chromosome representations**			
Name	**n**	**M**	**p**	**1 x 100**	**2 x 50**	**4 x 25**	**10 x 10**
2-2	2	2	4	3.32	3.22	3.24	3.06
2-3	2	3	8	3.76	3.56	3.42	3.30
2-4	2	4	16	4.74	4.96	4.62	4.40
2-5	2	5	32	5.14	5.24	4.88	4.30
3-2	3	2	4	4.92	4.92	4.78	4.46
3-3	3	3	8	7.70	8.36	7.38	6.76
3-4	3	4	16	15.10	14.69	13.63	12.23
3-5	3	5	32	24.71	20.97	19.51	16.20

IV. CONCLUSION

In this paper a deep analysis on how to choose the genotype size for an evolvable hardware system based on a FPGA structure for evolving digital logic circuits of different complexity has been carried out. The goals of the simulations were to find the best chromosome representation in order:

- to improve the evolvability of the system
- to reduce the number of generations of the evolutionary algorithm
- to optimize the size of evolved logic circuits.

The experimental results show that our aims are reachable if wider rectangular arrays as circuit layouts are avoided. For instance a circuit layout with 10 rows and 10 columns performs much better than a circuit layout with 2 rows and 50 columns, even though the number of logic gates and the search space is still the same.

ACKNOWLEDGMENT

Author thank to Bio-Inspired Intelligent Group at Brunel University, UK. The first author also thanks Hemantha Kodikara Arachchi for his contributions.

REFERENCES

[1] X. Yao, T. Higuchi; "Promises and challenges of evolvable hardware" *IEEE Trans. Systems, Man and Cybernetics*, Part C, volume 29, pp. 87 - 97, February 1999.

[2] H. de Garis. "Evolvable Hardware: Principles and Practice". *Communications of the Association for Computer Machinery (CACM Journal)*. August 1997

[3] J.D. Lohn, D.S. Linden, G.S. Hornby, W.F. Kraus, A. Rodriguez-Arroyo, S.E. Seufert. "Evolutionary design of an X-band antenna for NASA's space technology 5 mission". *NASA/DoD Conference on Evolvable Hardware, 2003*.Page(s):155 – 163

[4] S. V. Hum, M. Okoniewski, R. J. Davies. "An Evolvable Antenna Platform Based on Reconfigurable Reflectarrays". *The 2005 NASA/DoD Conference on Evolvable Hardware*. June 29 - July 1, 2005, Washington DC, USA. IEEE Computer Society. Pages 139 – 146

[5] 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

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

[7] S.Zhao, L. Jiao, Y. Wang. "Evolutionary Design of Analog Circuits with a Uniform Design Based Multi-Objective Adaptive Genetic Algorithm". *The 2005 NASA/DoD Conference on Evolvable Hardware.* June 29 - July 1, 2005, Washington DC, USA. IEEE Computer Society. Pages 26 – 29

[8] A.H. Aguirre, R. Zebulum, C. Coello Coello. "Evolutionary multiobjective design targeting a Field Programmable Transistor Array". *NASA/DoD Conference on Evolvable Hardware, 2004*. 24-26 June 2004 Page(s):199 – 205

[9] A. Stoica, D. Keymeulen, T. Arslan, Vu Duong, R. Zebulum, I. Ferguson, Xin Guo "Circuit self-recovery experiments in extreme environments". *NASA/DoD Conference on Evolvable Hardware, 2004*. 24-26 June 2004 Page(s):142 - 145

[10] A. M. Tyrrell, R. A. Krohling and Y. Zhou. "Evolutionary algorithm for the promotion of evolvable hardware". *Computers and Digital Techniques, IEE Proceedings*- Volume 151, Issue 4, 18 July 2004 Page(s):267 – 275

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

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

[13] M. D. Vose. "The Simple Genetic Algorithm". MA: MIT Press 1999.

[14] Jim Torresen. "Two-Step Incremental Evolution of a Prosthetic Hand Controller Based on Digital Logic Gates". *4th Int. Conf. on Evolvable Hardware (ICES2001),* October 2001, Tokyo, Japan.

[15] P. Andersen. Evolvable Hardware: Artificial Evolution of Hardware Circuits in Simulation and Reality, M.Sc. Thesis, University of Aarhus, Denmark.

[16] Timothy G. W. Gordon and Peter J. Bentley. "On Evolvable Hardware". *In Ovaska, S. and Sztandera, L. (Ed.) Soft Computing in Industrial Electronics.* Physica-Verlag, Heidelberg, Germany, pp. 279-323.

[17] A. Stoica, R. Zebulum, and D. Keymeulen, "Mixtrinsic evolution," in *International Conference on Evolvable Systems*, Edinburgh, U.K., Apr. 2000, pp. 208–217.

[18] T. Kalganova and J. Miller. "Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness". *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, 19-21 July 1999 Page(s):54 – 63

[19] Jim Torresen. "Evolving Multiplier Circuits by Training Set and Training Vector Partitioning". *In proc. of Fifth International Conference on Evolvable Hardware (ICES03)*, Springer LNCS 2606, pp. 228-237, March 2003, Trondheim, Norway

[20] Higuchi, T.; Iwata, M.; Keymeulen, D.; Sakanashi, H.; Murakawa, M.; Kajitani, I.; Takahashi, E.; Toda, K.; Salami, N.; Kajihara, N.; Otsu, N.; "Real-world applications of analog and digital evolvable hardware" *IEEE Transactions on Evolutionary Computation* , Vol.: 3 Issue: 3 , Sept. 1999 Page(s): 220 -235.

[21] 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, pp. 1135–1142, Orlando, USA, July 1999.

[22] J. F. Miller and P. Thomson. "Cartesian genetic programming". In Riccardo Poli, Wolfgan Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin and Terence C. Forgaty, editors. *Genetic Programming, Proceedings of EuroGP 2000*. Vol. 1802 of LNCS, pages 121-132, Edinburg, 16 April 2000. Springer-Verlag.