Proceedings of the 2004 IEEE
Conference on Cybernetics and Intelligent Systems
Singapore, 1-3 December, 2004

Improving EHW Performance Introducing a New Decomposition Strategy

Emanuele Stomeo
Electrical and Computing Engineering Department
Brunel University
Kingston Lane, Uxbridge, UK UB8 3PH
emanuele.stomeo@brunel.ac.uk

Tatiana Kalganova
Electrical and Computing Engineering Department
Brunel University
Kingston Lane, Uxbridge, UK UB8 3PH
tatiana.kalganova@brunel.ac.uk

***Abstract*— This paper describes a new type of decomposition strategy for Evolvable Hardware, which tackles the problem of scalability. Several logic circuits from the MCNC benchmark have been evolved and compared with other Evolvable Hardware techniques. The results demonstrate that the proposed method improves the evolution of logic circuits in terms of time and fitness function in comparison with BIE and standard EHW.**

Keywords— Evolvable hardware, evolutionary computation, logic design, problem decomposition.

I. Introduction

Evolvable hardware (EHW) [1] is a technique to automatically design circuits using methods inspired by natural evolution. The configuration is carried out under the control of evolutionary algorithms (EA). Initially evolvable hardware was introduced to be applied to real-world applications, but due to its limitations in scalability [1][2][3], to date no real world applications have been developed for relatively large applications. A number of approaches have been introduced to overcome these problems. In terms of scalability, approaches such as function-level evolution [4], bi-directional incremental evolution (BIE) [5] and the divide-and-conquer method have been introduced [6]. Function-level evolution has been proven to be successful in achieving the evolution of relatively complex task [4]. One of the main weaknesses of this approach is that it still requires human intervention to select the most appropriate functions for the relative problems. Regarding the divide and conquer method, so called *increased incremental evolution* [7] has been introduced to reduce the search space which allows the complete evolution of logic circuits up to 10 inputs (5*5 bit multiplier) [8]; but a significant weakness is also present, that is the difficulties of the definition of the fitness function for the initial stages of the evolution, which makes it less suitable for completely automatic systems. Furthermore this method gives an unconditional imposition to the system: the top-down design which does not allow the discovery of new designs. BIE evolution is a completely automatic system which does not require any knowledge from the designer which is not scalable to really large circuits due to the limitations of EHW-oriented output and Shannon decompositions [5]. Although the last two methods have been proven to be successful in the evolution of logic circuits, the scalability problem remains to be one of the main issues in the evolution of relatively large logic circuits in a reasonably short time. This paper addresses these issues.

The proposed method is based on the use of a decomposition method where the number of inputs in evolved logic circuits is reduced by the introduction of new output functions.

The paper is organised as following: the next section considers the basis of the system implemented in order to run the method proposed in this paper. Section III explains the proposed method and Section IV explores the behaviour of the proposed algorithm based on obtained experimental results. Section V concludes the work and provides the summary.

II. Extrinsic EHW

In this section an explanation of the system used to evolve combinational logic circuits is given. The evolutionary algorithms used, together with the fitness functions, chromosome representations and genetic operators are also presented.

A. Evolutionary algorithms applied to EHW

In Fig. 1 a general evolutionary process used in evolvable hardware is shown. The chromosome can be generated via software or hardware, and then converted into models by using software programs such as Spice, VHDL, C++, LISP etc., or into circuits by using reconfigurable hardware such as FPGA, FPTA, and PLA. After that, the implemented models or circuits are evaluated via software or hardware, and the fittest individuals are selected in order to reproduce a new population for a new lifecycle. The EA process terminates when the desired logic circuit is evolved, or other requirements are met, such as the maximum number of generations, maximum time, maximum memory usage etc.

B. Evolutionary algorithms

In the given extrinsic EHW approach the (1+λ) rudimentary evolutionary strategy is used [9][10], where λ represents the population size. Once the fitness function of each individual is calculated, the fittest individual is selected and duplicated for the population of the next generations and it is brought up to date by using a mutation operator.

This work is supported in part by the EPSRC under Grant GR/S17178/.

Stomeo E. and T. Kalganova (2004) Improving EHW performance introducing a new decomposition strategy. Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS'2004). Singapore. Published by IEEE SMC Society Singapore Chapter and IEEE R&A Society Singapore Chapter. pp. 439 - 444

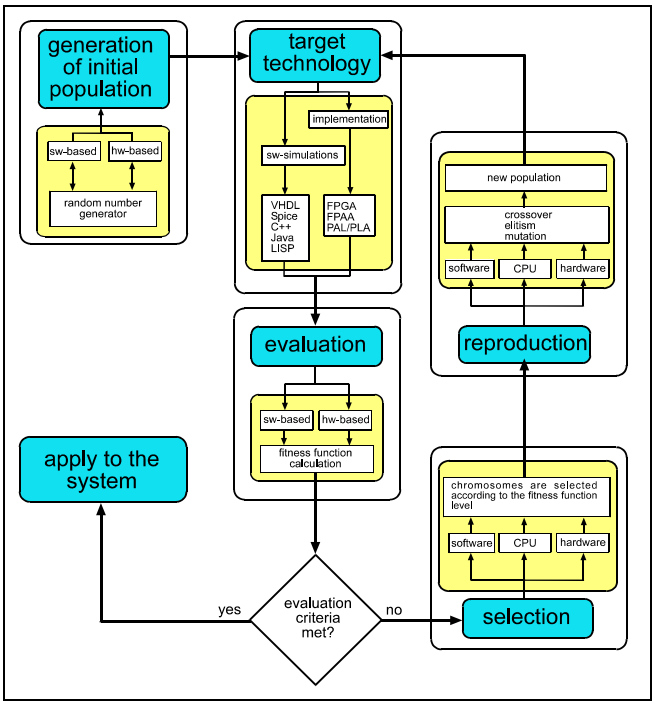

Figure 1 Generic evolutionary process used in EHW

C. *Chromosome encoding*

The chromosome defines the connection in the logic circuits between the inputs and the outputs. The chromosome encoding used takes into account the aspects of any combinational logic network: cell functionality and inter-connectivity of the cells between the inputs and outputs of the circuit [9]. In our approach the logic circuit is presented as a rectangular array of logic gates. Each logic cell in this array is uncommitted and can be removed from the network if it is redundant. All the logic functions are chosen from the set of AND, OR, EXOR, NOT and MUX. The chromosome is represented by a 3 level structure: geometry, circuit and gate. At the first level the global characteristics of the circuit are defined: the internal connectivity, number of rows and columns of the rectangular array. At the second level the array of cells is created and the circuit's outputs are determined; and the third level represents the structures of each cell in the circuit [9].

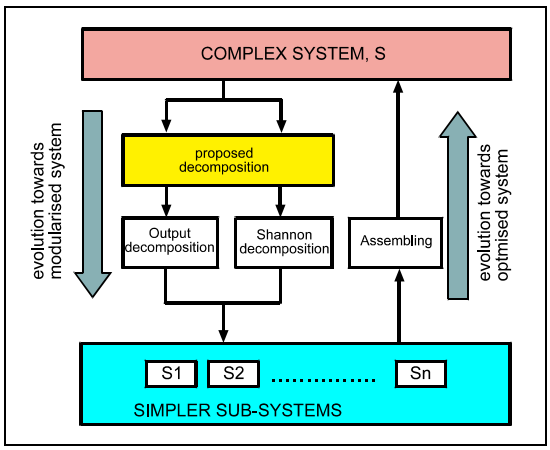

Figure 2 System used for evolving logic circuits

D. *Dynamic Fitness Function*

The fitness function is responsible for measuring the evaluation of the process. In our experiment we took into consideration a dynamic fitness function, which has two main criteria: first *functionality*, the fully functional circuit is evolving and second, only after the circuit is completely evolved, *optimization* which allows us to reduce the number of active gates and improve the quality of the evolved fully functional circuit. That way the dynamic fitness function f_{tot} is calculated as follows:

$$f_{tot} = \begin{cases} f_1 & f_{tot} \le 100 \quad design \\ f_1 + f_2 & f_{tot} \le 100 \quad optimization \end{cases} \tag{1}$$

Where the functionality of the evolved circuit 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 inputs of the evolved logic function respectively; p is the number of ON and OFF sets in the Boolean function (if $p=2^n$, then the Boolean function is completely specified, else the function is incompletely specified); y_i is the i^{th} digit of the output combination produced by evaluation of the circuit, di is the desired output for the fitness case fc. $|yi\text{-}di|$ is the absolute difference between the actual and the required outputs.

The fitness function for the optimization stage is calculated as:

$$f_2 = \sum N_{\lg} \cdot N_{p\lg} - \sum N_{u\lg} \cdot N_{p\lg} \tag{3}$$

Where N_{lg} is the number of all the logic gates, N_{plg} is the number of the primitive logic gates and N_{ulg} is the number of used logic gate. Fig. 3 shows the progress of the fitness function during the evolution of the functionality for the following function:

$$f = \lceil sqrt(x) \rceil \tag{4}$$

with 4 inputs and 3 outputs. Once the circuit is completely evolved (100% functionality is achieved), the optimization stage starts to improve the overall quality of the circuit: it can seen that the number of primitive active gates is reducing in each generation.

E. *Genetic operators*

The genetic operators used are: gene mutation, tournament selection and elitism [11].

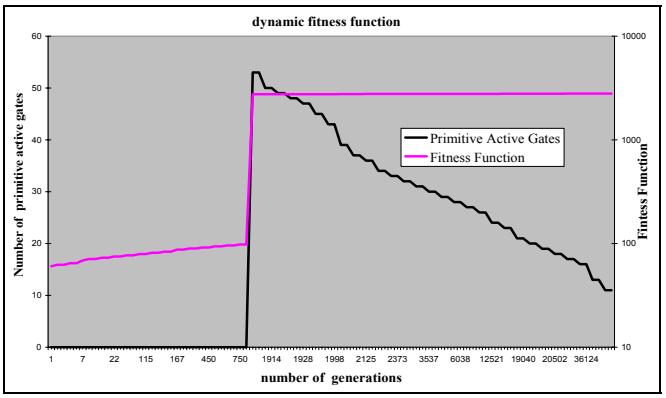

Figure 3 This graphic shows the effect of the use of the dynamic fitness function during the evolution of logic circuits. When the fitness function reaches 100% (functionality evolved), the optimization process begins.

III. PROPOSED METHOD

In this section the proposed method, which speeds up the evolutionary process and optimizes the logic circuit is explained. The proposed method improves the stalling effect and scalability for evolving logic circuits.

A. Limitations of EHW evolution

In order to identify the limitations of the previous systems, a number of experiments have been carried out. The purpose of these experiments was to quantify how the performance of the evolutionary process is dependant on the complexity of the tasks used. The logic circuits were evolved using the extrinsic EHW approach with (1+5) rudimentary evolutionary strategy described in detail in Section II. The system set-up used for evolving the logic circuits is in Table 1. The obtained results were classified according to the number of inputs and outputs of the logic functions. In Fig. 4 the relationship between the dimension of the circuits and the required number of generations in order to evolve them is considered. In Fig. 5 the average of the redundancy *r* for the evolved circuit is given. The redundancy has been calculated as:

$$r = 1 - \frac{N_{a\,lg}}{N_{lg}} \tag{5}$$

where N_{alg} is the number of active gates required for the evolved circuit and N_{lg} is the number of the total logic gates. The summarized experimental results presented in Fig. 4 show that the number of generations required in order to evolve a logic circuit is mainly dependent on the number of inputs. The system set-up together with the EHW algorithm used is able to evolve only the circuit for which the result is given on the Fig. 4. The system considered is not able to evolve more complex logic circuits. Based on the obtained results one may conclude that there is a need for the development of a method that would concentrate on the input decomposition for EHW systems. The experimental results prove that such a method will produce better scalability results than the methods focused on the output decomposition. This paper is devoted to proposing one such method.

TABLE 1 INITIAL DATA FOR THE EXPERIMENTS CARRIED OUT WITH (1+λ)ES

Number of generations		800.000
Population size		5
Number of runs or each experiments		>50
Elitism is applied		
Cell mutation rate		0.05
Selection pressure		1
Circuit layout	Rows	10
	Columns	10
	Level Back	10

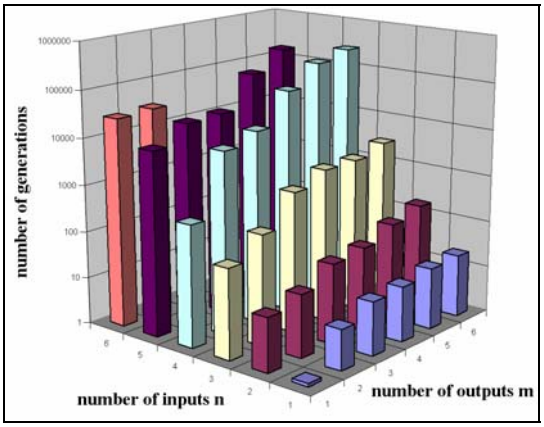

Figure 4 Average of the number of generations required to evolve logic circuits with n inputs and m outputs.

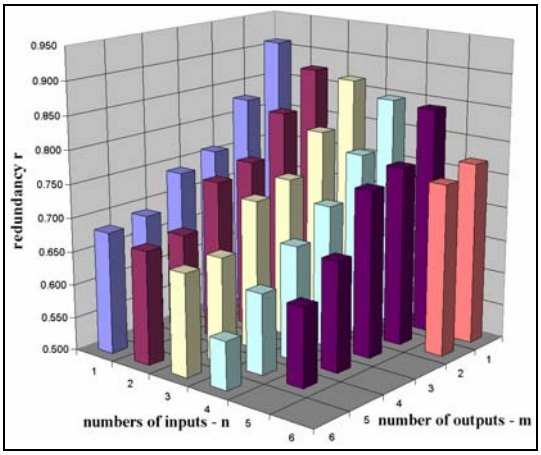

Figure 5 Average values of the redundancy for evolved logic circuits. All the experiments have been carried out with the configuration given in Table 1.

Therefore, a new system, which is capable of reducing the number of required generations and at the same time improving the fitness functions and evolving larger logic circuit is considered.

B. The proposed method

Let us assume that the following system with *n* inputs and *m* outputs, see Fig. 6a, should be evolved using either the extrinsic EHW approach described in section II or using BIE. The functionality of this system can be described by the truth table given in Fig. 6b, where $p=2^n$ is the number of products (or so called number of input-output combinations). The system depicted in Fig. 6a can be decomposed into two sub-systems as shown in Fig. 7a.

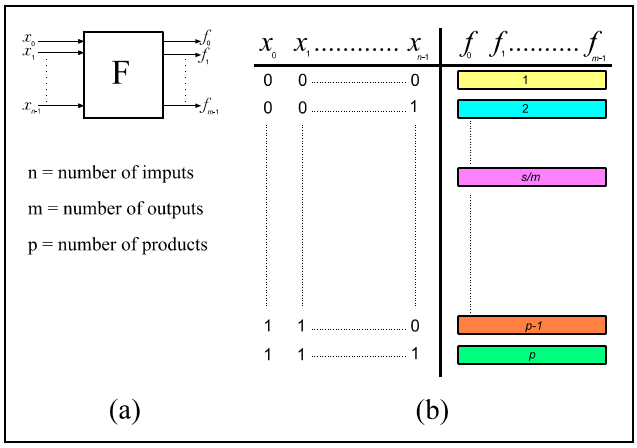

Figure 6 The general description of evolved circuit (a) the schemata of the evolved system; (b) the truth table for the evolved system

The sub-system G with r inputs and s outputs represents the evolvable part of the newly created system, where:

$$q = 2^r \tag{6}$$

$$s = m \cdot 2^{n-r} \tag{7}$$

The sub-system H with $(s+n-r)$ inputs and m outputs represents the fixed part of the circuit that is mainly generated using multiplexers. This part does not participate in the evolutionary process. The structure of this sub-circuit depends on the number of used inputs and outputs.

This sub-system G can be evolved using either the traditional EHW approach or any other scalable approach such as divide-and-conquer, bi-directional incremental evolution, etc. The complexity of the evolutionary process will depend on the type of method used. In the traditional EHW methods, the entire truth table is always used to evaluate the quality of evolved circuits.

Let us consider the process of generating the truth table for sub-system G. Let us assume that r should be always less than n ($r<n$), where r is the number of inputs in the G sub-system and n is the number of inputs in the initial system. The new truth table, shown in Fig. 7b, is calculated by applying the following procedure:

- Generation of all the input-output combinations of the truth table G.
- Identify the s/m input-output combinations, where the inputs of the truth table G match in sequence the inputs combination of the initial system.
- The outputs of the initial truth table relative to the previously identified inputs become the outputs of the reduced truth table, wherever a matching input has been identified.

C. *Case study*

Let us consider the process of the generation of a truth table for sub-system G based on the simple example. Let us consider the truth table corresponding to the function (4) with 4 inputs and 3 outputs. The truth table of this function is shown in Fig. 8a. Supposing that a sub-system G with only 2 inputs is to be generated. In order to do so, first we identify the subset of input variables used in sub-system G. Considering that $\{x_0,x_1,x_2,x_3\}$ is the set of input variables for the initial system, then the sub-set $\{x_0,x_1\}$ can be considered as a set of input variables for sub-system G. Note, that at the moment the set of input variables chosen to be inputs for sub-system G is carried out randomly. Some EHW-oriented algorithms should be developed to accommodate the optimal choice of input sub-set from all available input variables. Once the input sub-set for G is identified, the next step is to generate all the input combinations. The next step is to identify the input combination just generated in the initial truth table. For example, let us generate the input-output combination for $\{x_0,x_1\}=\{0,0\}$. In this case the truth table F is analyzed and the output of all the s/m input combinations that include $\{0,0\}$ for $\{x_0,x_1\}$ are considered as output for sub-circuit G. Once the outputs are generated for input combination $\{0,0\}$, the input combination $\{0,1\}$ is considered. The generated truth table for sub-system G is given in Fig. 8b.

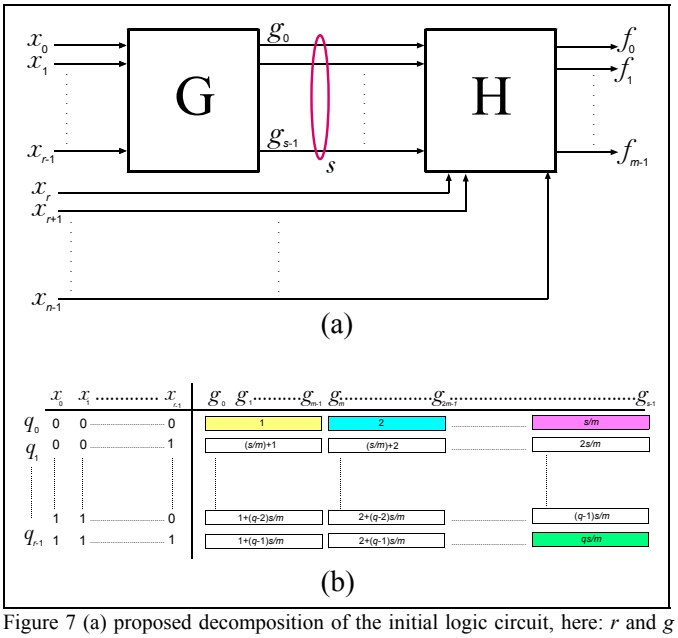

Figure 7 (a) proposed decomposition of the initial logic circuit, here: r and g refer to the number of inputs and outputs of the reduced sub-system respectively. (b) truth table of the evolved part of the proposed sub-system

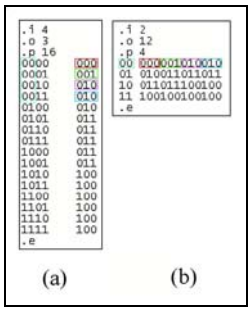

Figure 8 (a) the truth table of the function F; (b) the newly generated truth table for sub-system G, where .i is the number of inputs, .o is the number of outputs; .p is the number of products

IV. EXPERIMENTAL RESULTS

In this section, the results of the logic circuits evolved by using the proposed approach are shown. The aim of the experiments is to prove that the proposed method improves scalability for designing logic circuits in EHW in comparison with existing methods. A number of logic circuits taken from the MCNC benchmark [12] have been evolved. The obtained results have been compared with the performance of BIE technique, because this technique allows one to evolve relatively large logic circuits. The obtained results have not been compared with standard EHW approach due to its limitation in evolution of relatively large logic circuits. Several attempts to evolve the logic circuits using standard EHW approach have been made, but almost none of them were successful for the logic circuits considered in this paper. In other words, no fully functional solutions have been achieved for the chosen MCNC benchmark functions during evolution using the standard EHW approach. The initial data used for the experiments are given in Tables 2 and 3. The experimental results obtained for all evolved logic circuits demonstrated that the number of generations required to evolve the circuit with the proposed method is decreased by up to 15.09%. The total time required is also reduced by up to 8%. Furthermore the total fitness functions are increased by up to 741%, which means that the evolved circuit with the proposed method accomplish better optimization. All the improvements are exposed in Table 4, which shows number of inputs and outputs of the evolved circuit; the reduction (expressed in percentage) of the number of generations and the reduction of the time obtained with the proposed method compared with BIE. In the same table the improvement of the fitness function values is also given. Fig. 9-12 show the relationship between the fitness functions, the number of generations and the time spent for each experiment. Let us call "*reduced-circuit*" a circuit which is obtained by applying the proposed method to the truth table of the original circuit. Each graph compares the evolution results between the original logic circuits taken from the MCNC benchmark and two different *"reduced circuits"*. For all the evolved logic circuits, it can be noticed that a smaller amount of generations are required for evolving the *"reduced circuits"* and at the same time better values of fitness function are achieved. The same results for the time spent for each experiment are found. For any of the given graphs it can be noticed that when the number of inputs is reduced the fitness function value for each experiment is increased and the time and the number of generations is reduced. In Table 5 the average and the standard deviation of the evolved circuits are given. For the circuit "t841.pla", only the results of the proposed method are given, this is because the BIE approach is not able to evolve it.

TABLE 2 INITIAL DATA FOR THE EXPERIMENTS CARRIED OUT WITH BIE

Population size	5
Number of runs or each experiments	≥ 100
Cell mutation rate	0.05
Selection pressure	1

TABLE 3. INITIAL DATA: DIMENSION SIZE AND LEVEL BACK OF THE CIRCUIT LAYOUT USED DURING SIMULATIONS.

Name	Circuit layout used		
	Rows	Columns	Level Back
9sym	3	80	80
add2_7c	10	10	10
addm4	3	80	80
co14	3	80	80
con1	3	80	80
rd84	3	80	80
t841	4	100	100

TABLE 4. STATISTICAL RESULTS FOR THE EVOLUTION OF MCNC BENCHMARKS FOR MORE THAN 100 RUNS. THE TABLE SHOWS THE IMPROVEMENTS IN PERCENTAGE OF THE PROPOSED METHOD COMPARED WITH BIE SOLUTIONS.

Name	in	out	Number of generations	Time	Fitness function
9sym	9	1	15.09	11.29	175.48
add2_7c	8	4	26.49	19.34	436.32
addm4	9	8	78.79	45.81	178.78
co14	14	1	27.50	8.80	262.30
con1	7	2	74.31	47.74	741.38
rd84	7	4	43.74	32.03	174.64

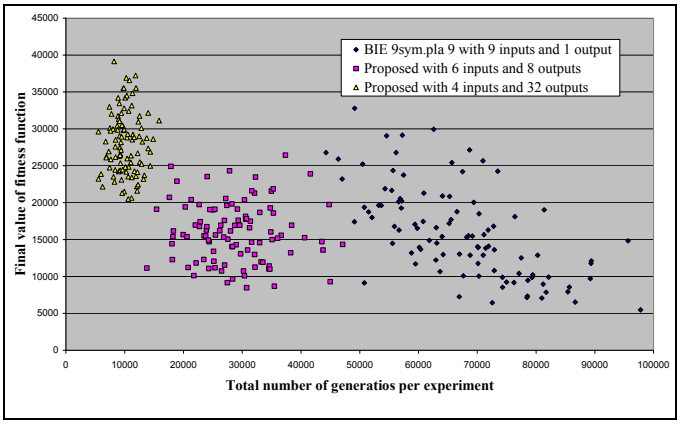

Figure 9. Relationship between the final fitness function reached at the end of each experiment and the number of generations required to evolve the circuit 9sym.pla

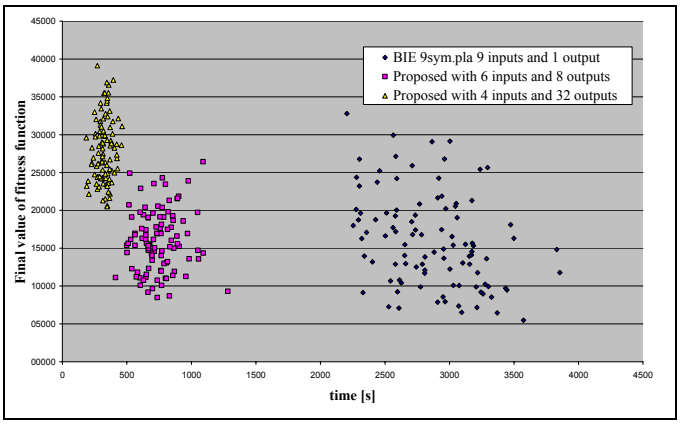

Figure 10. Relationship between the final fitness function reached at the end of each experiment and the CPU time required to evolve the circuit 9sym.pla

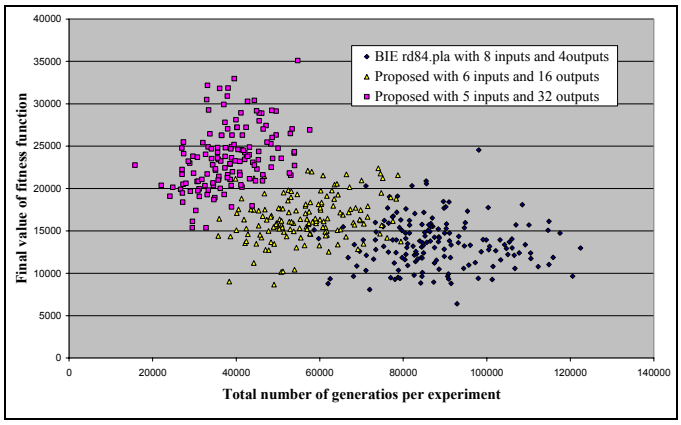

Figure 11 Relationship between the final fitness function reached at the end of each experiment and the number of generations required to evolve the circuit rd84.pla

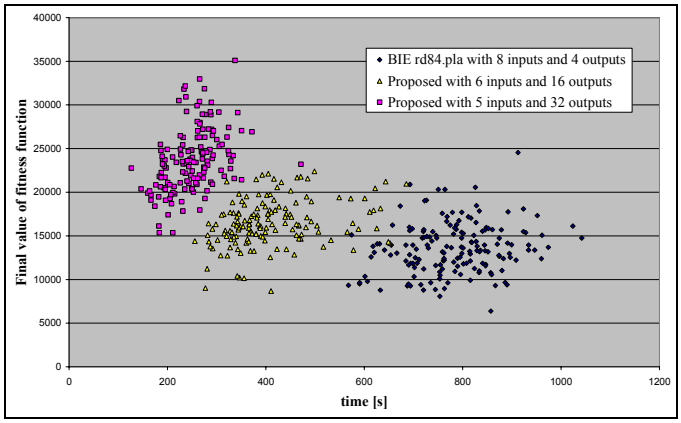

Figure 12. Relationship between the final fitness function reached at the end of each experiment and the CPU time required to evolve the circuit rd84.pla

TABLE 5. EXPERIMENTAL RESULTS

Info circuit				Result					
				Number of generations performed		Total time spent		Final fitness function	
name		in	out	average	standard deviation	average	standard deviation	average	standard deviation
9sym	BIE	9	1	67199	11451	2866	367	15976	6177
	Proposed	6	8	28741	6857	745	151	15971	4010
		4	32	10142	2131	323	59	28034	4208
add2_7	BIE	7	4	28121	9334	269	77	3036	882
	Proposed	5	16	11665	3897	90	80	7465	1930
		4	32	7448	1847	52	13	13248	2370
addm4	BIE	9	8	168053	16621	10713	1623	40847	8201
	Proposed	7	32	132414	12261	4908	658	73027	8332
co14	BIE	14	1	184476	22448	70877	5380	5024	1824
	Proposed	10	16	50733	18467	6240	1116	13179	5088
con1	BIE	7	2	6584	3075	286	99	3015	1401
	Proposed	5	8	7092	2384	212	60	10036	2844
		3	32	4893	1229	136	30	22358	3902
rd84	BIE	8	4	87752	12777	781	91	13571	2867
	Proposed	6	16	56764	10180	410	91	16473	2661
		5	32	38379	7691	250	50	23701	3650
t841	BIE	16	1	Impossible to evolve					
	Proposed	9	128	570496	87952	19098	4809	399486	34315

Based on the results found one may conclude that the main advantages of this method, as proven from the simulation are:

- Less computational requirements
- Fewer number of generations required to evolve the system
- Better optimization of the evolved circuit
- The possibility of evolving larger circuits
- The system is completely automatic and does not require any information from the user

V. CONCLUSION

In this paper a new decomposition method for the evolution of logic circuits has been introduced. The performance of the proposed algorithm has been tested based on the evolution of standard logic functions from the MCNC benchmark library. The performance of the algorithm has been compared with the performance of bi-directional incremental evolution. The experimental results confirmed that the proposed approach requires significantly less time, fewer generations to evolve fully functional solutions and the evolved circuits reach higher values of fitness function. In other words the experimental results confirmed that the proposed method produces better optimization, even though the work done is concentrated only on the evolution of fully functional circuits rather than in the optimization of evolved circuit. Further work will be concentrated on the development of algorithms to identify the optimal set of inputs used in sub-system G.

ACKNOWLEDGMENT

E. Stomeo thanks A.M. Walsh for her help and support and the BIIS research group at Brunel.

REFERENCES

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

[2] V. K. Vassilev, J. F. Miller "Scalability problems of digital circuit evolution" *Proc. of the 2nd NASA/DOD Workshop on Evolvable Hardware*, pp. 55-64. Los Alamitos, CA: IEEE Computer Society

[3] C. A. Coello, A. D. Christiansen and A. A. Hernández, "Towards automated evolutionary design of combinational circuits", *Computers and Electrical Engineering,* Pergamon Press, Vol. 27, No. 1, pp. 1-28, January 2001

[4] T. Higuchi, M. Murakawa, M. Iwata, I. Kajitani, Weixin Liu, M. Salami, "Evolvable hardware at function level"; *IEEE International Conference on Evolutionary Computation*, pp. 187 - 192, April 1997

[5] T. Kalganova, "Bidirectional incremental evolution in evolvable hardware", *Proc. of The Second NASA/DoD Workshop on Evolvable Hardware*. IEEE Computer Society. Palo Alto, California, USA.

[6] J. Torresen, "A divide-and-conquer approach to evolvable hardware", *Evolvable Systems: From Biology to Hardware. Second International Conference,* ICES 98, volume 1478 of Lecture Notes in Computer Science, pp 57-65. Springer-Verlag, 1998.

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

[8] 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, pp. 228-237, March 2003

[9] T. Kalganova, J. Miller, "Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness". *Proc. of the First NASA/DoD Workshop on Evolvable Hardware.* IEEE Computer Society, pp. 54–63. July 1999

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

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

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