On Evolution of Relatively Large Combinational Logic Circuits

E. Stomeo[1], T. Kalganova[1], C. Lambert[1], N. Lipnitsakya[2], Y. Yatskevich[2]
Brunel University UK[1], Belarusian State University[2]
emanuele.stomeo@brunel.ac.uk

Abstract

Evolvable hardware (EHW) [1] is a technique introduced to automatically design circuits where the circuit configuration is carried out by evolutionary algorithms. One of the main difficulties in using EHW to solve real-world problems is the scalability. Until now, several strategies have been proposed to avoid this problem, but none of them completely tackle the issue. In this paper three different methods for evolving the most complex circuits have been tested for their scalability. These methods are Bi-directional incremental evolution (SO-BIE) [2]; generalised disjunction decomposition (GD-BIE) [3] and evolutionary strategies (ES) with dynamic mutation rate [4]. In order to achieve the generalised conclusions the chosen approaches were tested using multipliers, traditionally used in EHW, but also logic circuits taken from MCNC [5] benchmark library and randomly generated circuits. The analysis of the approaches demonstrated that PLA-based ES is capable of evolving logic circuits of up to 12 inputs. The use of SO-BIE allows the generation of fully functional circuits of 14 inputs and GD-BIE is estimated to be able to evolve circuits of 21 inputs.

1. Introduction

Evolvable hardware (EHW) [1] is a technique introduced to automatically design circuits, where the circuit configuration is under the control of an evolutionary algorithm (EA) [6]. Initially, evolvable hardware was introduced to be applied to real-world applications, but to date no relatively large applications have been developed. This is mainly due to the fact that EHW is not scalable to larger problems [1], [7], [8], [9]. Let us focus on the investigation of scalability issues applied to the design of combinational logic circuits. The existing EHW systems introduced to evolve combinational logic circuits are generally not scalable by the following factors:

- the length of chromosome representation of logic circuits [10]
- the number of input-output combinations in the truth table
- The computational complexity of EA [2].

The length of the chromosome depends on the number of logic gates used and the connectivity between logic gates. The number of input-output combinations increases exponentially with the increase of the number of inputs in the evolved logic circuit.

The computational complexity of evolutionary algorithms appears mainly due to "stalling" effect that emerges in evolutionary processes for complex problems. Recently these issues have been tackled predominately in two directions: the improvement of evolutionary processes and the development of multi-evolutionary processes using the principles of problem decomposition. Previously, the performance of EHW on the evolution of 3-bit multiplier has been studied. Both PLA-based and FPGA-based circuits have been considered. For example, the 3-bit multiplier containing 26 logic gates has been evolved for FPGA structure after 3,000,000 generations using gate-level EHW approach [11]. Function-level EHW was first introduced by Higuchi et al. in [12] and further extended in [13] which the reduction of the number of generations required to evolve successfully the 3-bit multiplier to 30 generations. Although the proposed approach allowed the significant reduction of the number of generations required to obtain fully functional solution, the evolvability of logic circuits with a higher number of inputs remained to be the actual issue. For example, the analysis of the complexity of evolved logic circuits revealed that the most complex multiplier currently evolved is the 4 digit multiplier (8 inputs; 8 outputs) [14]. This circuit was evolved by using the logic gates as building blocks for

The authors thank EPSRC for financial support (Grant GR/S17178).

IEEE COMPUTER SOCIETY

FPGA target structure. The introduction of the dynamic mutation rate allowed the improvement of the achieved results by evolving 12-input 8-output logic circuits from MCNC benchmark library [5]. This circuit was generated for AND-OR PLA structure. Unfortunately it is difficult to compare the evolution of PLA- and FPGA-based logic circuits due to the fact that FPGA-based circuits have larger search space. Therefore they are more difficult to evolve. The main drawback of the last approach is that the dynamic mutation is specifically designed based on the behaviour of PLA-based logic circuits during the evolutionary process. Therefore, it is not applicable for the evolution of FPGA-based logic circuits. Based on decomposition strategies, several approaches to overcoming scalability problem have been introduced such as: divide-and-conquer [15]; bi-directional incremental evolution (SO-BIE) [2] and the *generalised disjunction decomposition*, a new decomposition strategy for evolvable hardware introduced by the author in [3]. Regarding the divide and conquer method, so called *increased incremental evolution* [16] has been introduced to reduce the search space. This method has been demonstrated complete evolve of logic circuits of 10 inputs (5-bit multiplier) introducing partioned training vector and partioned training set [17]. However a significant weakness is also present, that is the difficulties in defining the fitness function for the initial stages of the evolution, which makes it less suitable for completely automatic systems. SO-BIE evolution is a completely automatic system which does not require any knowledge from the designer and is not scalable to really large circuits due to the limitations of EHW-oriented output and Shannon decompositions [2]. The first attempt to use this approach in EHW was achieved by the evolution of 7-inputs 10 outputs logic function from MCNC benchmark and has been further improved by introduction a new assembling techniques [18]. Furthermore, the introduction of generalized disjunction decomposition into SO-BIE improved design and optimization of logic circuits to 16 inputs 1 output. The drawback is the imposition to the system to use multiplexers. In this paper bi-directional incremental evolution, the "generalised disjunction decomposition" and ES with variable mutation rate are evaluated in an attempt to establish the advantages and disadvantages of each of them.

This paper is organized as follow: the next section gives a brief description of these three methods together with the evolutionary algorithms, chromosomes structures and fitness functions used. Section 3 gives the experimental results, followed by the conclusions.

2. Extrinsic EHW approaches

Bi-directional incremental evolution, applied to design of combinational logic circuits, combines the evolutionary processes carried out by extrinsic EHW with EHW-oriented circuit decomposition that identifies the sub-tasks to be evolved. Let us consider the main features of BIE with Shannon and output decompositions (SO-BIE), extended BIE with generalised disjunction decomposition (GD-BIE) and ES with dynamic mutation rates applied to the evolution of combinational logic circuits. Each circuit and sub-circuit, defined by decomposition, is consequently evolved using extrinsic EHW.

2.1 Extrinsic EHW

In this section the evolutionary algorithm used to evolve logic circuits, together with the fitness function and chromosome representations are presented.

2.1.1 Evolutionary algorithm. The evolutionary algorithm used is the (1+λ) rudimentary evolutionary strategy with cell and circuit geometry mutation, where λ represents the population size [19], [20]. Once the fitness function of each individual is calculated, the fittest individual is selected and duplicated for the population of the next generation and it is brought up to date by using both cell and circuit geometry mutation operators.

2.1.2 Encoding. 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. 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, XOR, NOT and multiplexer. 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 and the 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. The third level represents the structures of each cell in the circuit [19].

2.1.3 Dynamic fitness function. The fitness function evaluates the evolved circuits in terms of their functionality. In our experiment a dynamic fitness function has been considered. It has two main criteria:

IEEE COMPUTER SOCIETY

first design and second, once the circuit is fully functional evolved, optimization which leads to reduced numbers 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. $|y_i-d_i|$ is the absolute difference between the actual and the required outputs. The fitness function for the optimization stage is calculated as:

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

where N_{lg} is the total number of logic gates, N_{plg} is the number of primitive logic gates and N_{alg} is the number of active logic gates.

2.2 Bi-directional incremental evolution

Bi-directional incremental evolution [2] operates by gradually decomposing a complex system into a series of simpler ones when the evolution does not bring any improvement in terms of fitness function value, see Figure 1. These simpler blocks are evolved separately, and then merged together once completely developed. If, during the evolution of each single subsystem, the stalling effect occurs again, the single sub-circuits will be decomposed another time, until all the sub-circuits are simple enough to be evolved. The systems are decomposed by using Shannon and output decomposition [2].

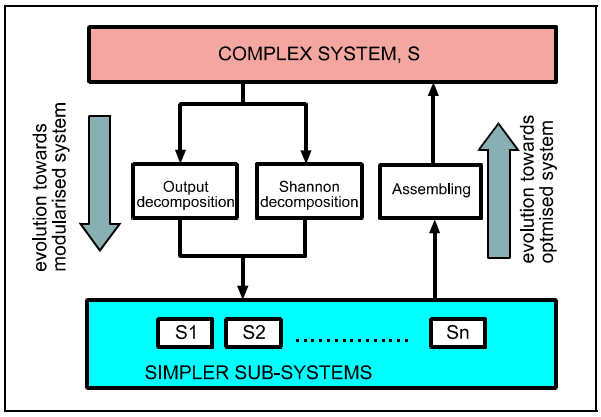

Figure 1. Bi-directional Incremental Evolution (BIE) approach

As can be seen the evolution is in both sides: firstly towards modularization (having simpler and smaller logic circuits) and secondly towards an optimized system, by assembling the simpler sub-circuits together. For example, the output decomposition guarantees that each sub-system is synthesized separately and is completely independent. In the case of functional decomposition, the corresponding outputs generated for various input combinations in different sub-systems have to be connected together using one-control multiplexer. Analysis of experimental results show that it is reasonable to assemble the subsystems decomposed by functional decomposition first and then the sub-systems separated using output decomposition [18].

2.3 Generalised disjunction decomposition

The "generalised disjunction decomposition" proposed in [3] is based on the statement that:

- the number of generations required to completely evolve logic circuits is mainly dependant on the number of inputs instead of the number of outputs, which is shown in [3].
- The decomposition of a complex system into smaller ones in BIE is done by using output decomposition.

So, supposing that a complex system F with n inputs and m outputs, see Figure 2, requires numerous generations to be evolved. This could be decomposed into two sub-systems as reported in Figure 3; where the subsystem G with r inputs and s outputs represents the evolvable part of the newly created system. The number of input-output combinations is:

$$q = 2^r \tag{4}$$

and the number of output is:

$$s = m \cdot 2^{n-r} \tag{5}$$

IEEE COMPUTER SOCIETY

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. By using this decomposition strategy the number of generations required to evolve the circuits is much smaller; furthermore this method allows the evolution of larger circuits [3]. This sub-system G, which has fewer inputs and more outputs than the original ones, can be evolved using either the traditional EHW approach or any other scalable approach such as divide-and-conquer, bi-directional incremental evolution, etc.

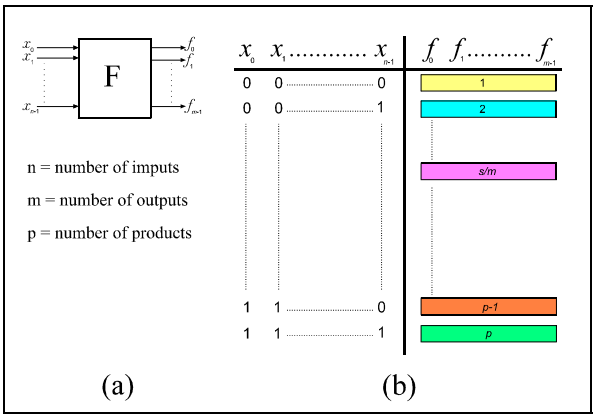

Figure 2. General description of a system with n inputs and m outputs (a); truth table of the system (b), where p is equal to all the possible input-output combinations.

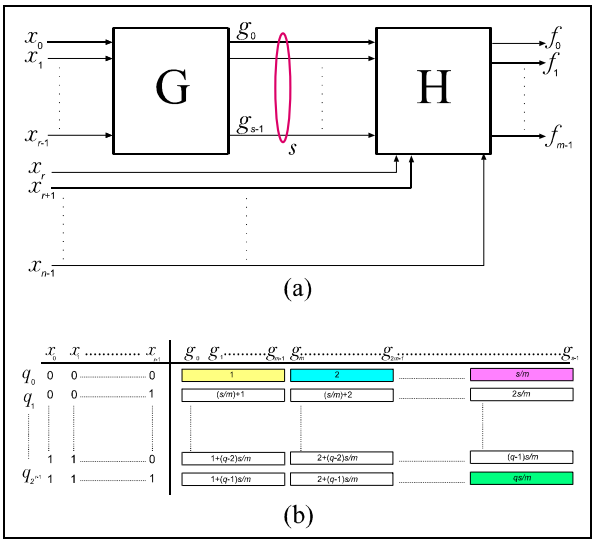

Figure 3. Generalized disjunction decomposition of the initial logic circuit. (a) Schemata *r* and *g* refer to the number of inputs and outputs respectively. (b) Truth table of the evolved part of the proposed sub-system

The complexity of the evolutionary process will depend on the type of method used.

2.4 Evolving PLA structures using ES with dynamic mutation rate

This approach is based on the idea of evolving logic circuits using a dynamic mutation rate that adapts to the evolved circuit structure [4]. This technique uses evolutionary strategy with uniform mutation, roulette wheel selection and binary chromosome representation to generate the AND-OR PLA structure. The mutation rate is changed according how good the evolved solutions are. The chromosome encodes the structure of Programmable Logic Array (PLA) by describing the connections between lines in AND and OR planes. Therefore, the PLA structure is encoded using 2 arrays of genes as shown in Figure 4.

The chromosome is composed of three genes: connection genes in AND plane, input line genes in AND plane and connection in OR plane. The evolutionary process is divided into 2 sub-processes, where different fitness functions are activated. The functionality of the evolved logic function is used during the PLA design process. The number of product lines in the PLA structure is minimized during the PLA optimization process.

Dynamic fitness function similar to one introduced earlier in the extrinsic EHW approach, is used to evaluate the quality of the evolved circuits. The difference is that the quality of the evolved fully functional circuits is defined by the number of product lines actually used in the obtained solution.

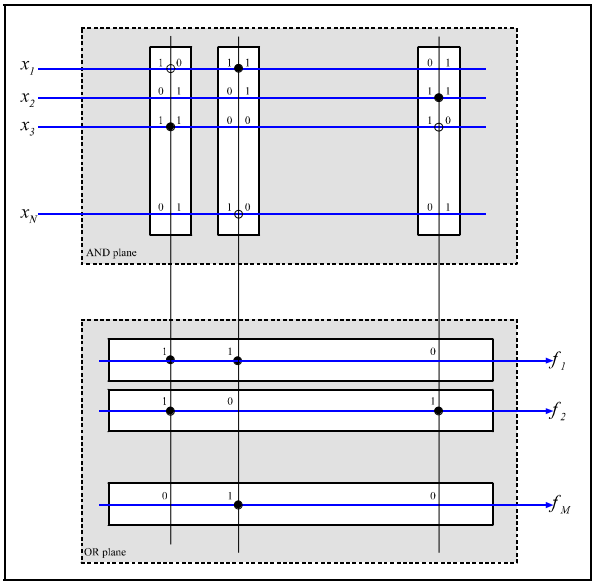

Figure 4. The chromosome encoding of a PLA structure

IEEE COMPUTER SOCIETY

3. Experimental results

Evolvable hardware and Digital Logic Design are two competitive areas that have the common goal: to design of logic circuits. Evolvable hardware attempts to introduce completely automated circuit design processes in contrast to traditional Digital Logic Design where even today the human intervention plays a vital role in the design of logic circuits. Although both areas have the same goal, the algorithms proposed in these two areas are analyzed using different libraries. For example, the approaches proposed in the area of Digital Logic Design are validated using MCNC benchmark library [5], [21] in contrast to Evolvable hardware, where validation is mainly based on the evolution of multipliers with different complexity and randomly generated logic circuits [8][11][15]. Through our experimental work we have attempted to merge a validation process used in Evolvable Hardware and Digital Logic Design. Therefore, the evolvability of logic circuits randomly generated, as well as circuits taken from MCNC benchmark library and multipliers of different complexity are analyzed. This provides an indication on how EHW-based approaches perform in general for the evolution of combinational logic circuits. In this work, only the logic circuits given on complete set of input-output combinations have been considered. For example a 3-bit multiplier has 6 inputs and 6 outputs and it is described with 64 input-output combinations. Similarly a 6-bit multiplier contains 12 inputs and 12 outputs and it is described by 4096 input-output combinations. The presented results are obtained based on the analysis of the truth table of completely specified switching functions. The aim of these experiments is to illustrate:

- the maximum possible size of evolvable logic circuits for each method discussed earlier;
- the performance of methods discussed earlier during optimization process;

The experiments have been carried out separately for methods evolved FPGA- and PLA-based circuits.

3.1 Experimental results: BIE and generalized disjunction decomposition

In this section the experimental results obtained with the use of BIE and the generalised disjunction decomposition are presented. The initial data used for those experiments are given in Table 1. The system used for evolving circuits with SO-BIE is shown in Figure 1, while the schema shown in Figure 5 is used for the generalized disjunction decomposition.

Table 1. Initial data for the experiments carried out using BIE and the generalized disjunction decomposition

Evolutionary algorithm	(1+λ) rudimentary ES
Population size	5
Number of Generations	500000
Number of runs for each experiments	≥100
Elitism is applied	
Cell mutation rate	0.05
Geometry Mutation Rate	0.05
Termination criteria for evolutionary process	2000 generations without any improvement in fitness function

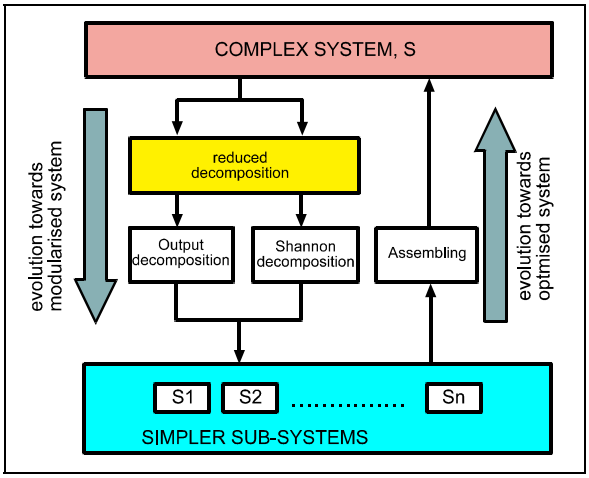

Figure 5. System used for evolving logic circuits

The experimental results obtained by using BIE and the generalised disjunction decomposition are shown in Table 2. In that table all the characteristics of the circuit are given. For example, by looking at the logic circuit 9sym.pla, it has 9 inputs, 1 output and 512 input-output combinations. Then, the number of generations (average out of 100 experiments and "best solution") required to evolve the logic circuits, is reported. The next two columns give the average and best time (values are expressed in seconds) for each experiment. The next two columns provide the value of fitness function for the final optimized solutions. The last three columns give information on the circuit layout used to evolve the logic circuits, such as number of rows, columns and level's back [22]. For each circuit different results are given, this is because two different methodologies are used.

For the circuits 9sym it can be observed that it is evolved using BIE (first row, 9 input and 1 output) and the generalised disjunction decomposition (second and third rows respectively with 6 input and 8 outputs and 4 inputs and 32 outputs).

IEEE COMPUTER SOCIETY

Table 2. Experimental results from SO-BIE and generalized disjunction decomposition (GD-BIE), where in, out and p are the number of inputs, outputs and products (input-output combinations) in the given logic function. Each logic circuit (except for Mult6) has been evolved 100 times with a success rate of 100%. The last three columns give dimension size: number of rows (R), columns (C) and level back (L) [22] of the circuit layout used during simulations

Info circuit					Experimental results								
					Number of generations performed		Total time spent per each experiment in seconds		Final fitness function		Circuit layout parameter:		
name	method	in	out	p	average	best	average	best	average	best	R	C	L
Randomly generated logic circuits													
6-5	SO-BIE	6	5	64	40,425	26,381	1,194	624	15,658	25,088	3	80	80
	GD-BIE	4	20	16	16,121	9,446	410	257	13,815	17,931			
6-4	SO-BIE	6	4	64	30,095	14,587	1,099	442	11,866	20,456	3	80	80
	GD-BIE	4	16	16	10,507	6,744	321	159	23,937	34,030			
6-3	SO-BIE	6	3	64	30,754	16,099	366	229	3,498	6,404	10	10	10
	GD-BIE	4	12	16	8,251	4,489	129	64	7,458	10,972			
6-2	SO-BIE	6	2	64	12,886	4,160	289	94	1,598	3,486	10	10	10
	GD-BIE	4	8	16	3,500	564	54	12	2,152	5,614			
6-1	SO-BIE	6	1	64	10,784	4,406	136	59	1,483	3,887	10	10	10
	GD-BIE	3	8	8	3,684	1,575	62	25	3,049	5,102			
Logic circuits taken from MCNC benchmark library													
majority	SO-BIE	5	1	32	1,323	17	17	0.5	608	1,385	10	10	10
	GD-BIE	3	4	8	237	19	8	1	1,139	1,957			
9sym	SO-BIE	9	1	512	67,041	44,261	2,852	2,204	15,976	32,790	3	80	80
	GD-BIE	6	8	64	28,741	13,771	745	412	15,971	26,448			
		4	32	16	10,142	5,540	323	185	28,034	39,128			
add2_7	SO-BIE	7	4	128	28,121	10,535	269	112	3,036	5,268	10	10	10
	GD-BIE	5	16	32	11,665	4,358	90	29	7,465	14,190			
		4	32	16	7,448	4,541	52	32	13,248	20,455			
5xp1	SO-BIE	7	10	128	43,643	22,623	1,878	1,003	16,994	30,008	3	80	80
	GD-BIE	5	40	32	24,560	13,116	884	518	51,659	77,001			
addm4	SO-BIE	9	8	512	168,053	127,206	10,713	8,039	40,847	68,204	3	80	80
	GD-BIE	7	32	128	132,414	103,563	4,908	3,753	73,027	94,339			
co14	SO-BIE	14	1	16,384	184,476	150,838	70,877	64,222	5,024	7,531	3	80	80
	GD-BIE	10	16	10,24	50,733	14,139	6,240	3,479	13,179	33,075			
con1	SO-BIE	7	2	128	6,584	2,177	286	126	3,015	8,881	3	80	80
	GD-BIE	5	8	32	7,092	2,307	212	79	10,036	17,760			
		3	32	8	4,893	2,553	136	71	22,358	30,441			
rd84	SO-BIE	8	4	256	87,752	58,640	781	568	13,571	24,545	3	80	80
	GD-BIE	6	16	64	56,764	35,698	410	256	16,473	22,388			
		5	32	32	38,533	15,808	250	126	23,701	35,104			
t841	SO-BIE	16	1	65,536	Not evolved								
	GD-BIE	9	128	512	597,469	463,396	20,250	13,482	396,399	445,554	4	100	100
Multiplier circuits													
Mult3	SO-BIE	6	6	64	21,948	9,030	288	126	4,279	2,373	10	10	10
	GD-BIE	4	24	16	9,156	4,434	123	67	8,820	14,219			
Mult4	SO-BIE	8	8	256	146,663	117,495	1,718	1,468	13,019	20,592	10	10	10
	GD-BIE	6	32	64	87,411	70,999	1,040	594	23,554	30,926			
Mult5	SO-BIE	10	10	1,024	740,164	685,372	16,033	15,338	48,786	52,860	3	80	80
	GD-BIE	8	40	256	506,347	482,789	24,088	17,458	152,372	171,368			
Mult6	SO-BIE	12	12	4,096	2,582,678	2,582,678	190,450	190,450	537,322	537,322	3	80	80

Based on the results found, one may conclude that the main advantages of using the generalized disjunction decomposition are:

- a smaller amount of generations are required during evolution
- a better values of fitness functions are achieved, therefore the circuits are better optimized
- it solves the tasks quicker than by using BIE

All the circuits (except of the multiplier 6x6, which has been evolved only once, because of the high computational time required) have been evolved 100 times with an achievement rate of 100%.

3.2 Experimental results: ES with dynamic mutation rate

In this section the results obtained with the use of the evolutionary strategy with variable mutation rate are presented. In Table 3 the initial data together with the experimental results are shown. In that table I_{max} is the initial given number of products lines in PLA; N^{max} refers to the maximum number of generations given for each evolutionary process: PLA design and optimization; N^{design} and N^{opt} are the average number of generations for design and optimization processes, respectively; I^{design} and I^{opt} are the average number of product lines in PLA obtained after the completion of

IEEE COMPUTER SOCIETY

design and optimization processes respectively; I^{best_design} and I^{best_opt} are the minimum number of product lines over 100 runs obtained for design and optimization processes respectively. I^{imp} gives the value in percentage of improvements in terms of fitness function. The experimental results have been obtained based on the analysis of 100 runs for each logic function, except of the circuits with the number of inputs higher than 10. Those functions have been evolved 5 times each. This is because the computational requirements needed to complete the evolution are too high for a desktop PC. It should be observed that this method was not able to evolve logic circuits with 14 inputs and higher, so the most difficult task solved was the 6-digit multiplier. In several cases, no significant improvement during optimization process has been noticed, see Table 3 last column (which gives the improvements in terms of fitness function during optimization). This can be explained by the use of a low number of generations during the evolutionary process.

4 Conclusion

In this paper a comparison of evolving logic circuits using three different methodologies has been presented. The performance of these three different techniques has been tested on the evolution of logic circuits taken from different sources: some of them were randomly generated, others were taken from MCNC benchmark and others describe the behaviour of multipliers of different complexities. The experimental results show that, the generalised disjunction decomposition used together with BIE:

- requires fewer of generations
- the evolved circuits are better optimized
- speeds up the evolutionary algorithm
- gives the possibility to completely evolve circuits of 16 inputs (which means 65536 input-output combinations), which is the biggest logic circuits completely evolved until now, by using a desktop PC.

The most complex logic circuit evolved for SO-BIE has 14-inputs. This may indicate the current limitations of SO-BIE. Since the evolution of evolvable part in GD-BIE is carried out by SO-BIE, the limitations implied to SO-BIE also are implied to GD-BIE. Therefore, GD-BIE can successfully perform evolution while the evolvable part of the circuit G remains no more complex than 14-inputs. Considering that currently we have managed to reduce the number of inputs in the evolvable part by 7, than one can predict that the most complex logic circuit that GD-BIE is capable to evolve should have no more than 21 input. ES, with a dynamic mutation rate, performs far better when compared with a BIE-based approach. This is because the statistical method evolves logic circuits using the AND and OR planes, which are simpler than the FPGA based logic circuits evolved with the BIE approach. The largest circuit evolved with this method is a 6-digit multiplier. The method was not able to evolve more complex circuits. Both approaches discussed in the paper have demonstrated the capability to evolve more complex logic functions than the ones reported earlier. The analysis of experimental results demonstrated that there is a potential for improvements in these algorithms.

Table 3. Experimental results obtained by making use of statistical model

Name	in	out	p	Experimental results									I^{imp}
				Initial parameters		*Success rate (%)*	PLA design			PLA optimization			
				I_{max}	N^{max}		N^{design}	I^{design}	I^{best_design}	N^{opt}	I^{opt}	I^{best_opt}	
Logic circuits randomly generated													
6-5	6	5	64	64	10,000	12	1232.4	63.7	62	10,000	62.8	59	1.4
6-4	6	4	64	64	10,000	61	101.6	61.9	56	10,000	56.8	50	8.2
6-3	6	3	64	64	10,000	100	5.7	57.	47	10,000	47.4	40	16.8
6-2	6	2	64	64	10,000	100	2.7	48.9	34	10,000	30.9	26	36.8
6-1	6	1	64	64	10,000	100	1.7	37.5	23	10,000	21.5	18	42.7
Logic circuits taken from MCNC benchmark													
Majority	5	1	32	32	10,000	100	1.3	24.7	15	10,000	12.3	8	50.2
con1	7	2	128	96	10,000	100	5.5	86.7	60	10,000	60.9	49	29.8
Add2_7	7	4	128	128	10,000	100	10.8	117	96	10,000	101.2	91	13.5
5xp1	7	10	128	128	10,000	62	108.3	127.7	126	10,000	124.2	118	2.8
rd84	8	4	256	256	10,000	0	-	-	-	-	-	-	-
9sym	9	1	512	512	10,000	100	8.5	301.0	252	10,000	253.8	239	15.7
addm4	9	8	512	384	10,000	89	30.8	379.6	365	10,000	369.4	350	2.7
alu1	12	8	4,096	4,096	10,000	100	16.6	3506.0	3417	10,000	3,203.2	3,169	8.6
co14	14	1	16,384	16,384	10,000	0	-	-	-	-	-	-	-
rd84	16	1	65,536	65,536	10,000	0	-	-	-	-	-	-	-
Multiplier circuits													
Mult3	6	6	64	64	10,000	100	7.1	56.9	47	10,000	49.4	42	13.2
Mult4	8	8	256	256	10,000	100	15.5	240.1	222	10,000	225.1	210	6.2
Mult5	10	10	512	512	10,000	100	26.4	990.5	954	10,000	965.7	936	2.5
Mult6	12	12	4,096	4,096	10,000	100	53.0	3997.2	3991	10,000	3,983.2	3,979	0.4

IEEE COMPUTER SOCIETY

6. References

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

[2]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.

[3]E. Stomeo and T. Kalganova. "Improving EHW performance introducing a new decomposition strategy". *2004 IEEE Conference on Cybernetics and Intelligent Systems*. Pp. 439-444. Singapore, 1-3 December 2004.

[4]T. Kalganova, N. Lipnitsakya, Y. Yatskevich. "Evolving PLA structures using evolutionary strategy with dynamic mutation rate". *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom December 2004. pp. 466 - 471. ISBN: 1-84233-110-8

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

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

[7]J. Dinerstein, N. Dinerstein, H. de Garis. "Automatic Multi-Module Neural Network Evolution in an Artificial Brain". *NASA/DoD Conf. on Evolvable Hardware, EH-2003,* USA, 2003.

[8]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

[9]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

[10]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, pp. 136–165.

[11]C. A. Coello, A. D. Christiansen and A. A. Hernández. "Use of evolutionary techniques to automate the design of combinational circuits" *International Journal of Smart Engineering System Design,* 1999

[12]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

[13]T. Kalganova. "An Extrinsic Function-Level Evolvable Hardware Approach". *Proc. of the Third European Conference on Genetic Programming, EuroGP2000*, Edinburgh, UK. Eds. R. Poli, W. Banzhaf. Springer-Verlag.

[14]D. Job V. Vassilev and J. Miller. "Towards the automatic design of more e_cient digital circuits". *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware,* pp. 151-160. IEEE Computer Society, Silicon Valley, USA.

[15]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.

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

[17]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

[18]I. Baradavka and T. Kalganova. "Assembling Strategies in Extrinsic Evolvable Hardware with Bi-directional Incremental Evolution". *Proc. of the 6th European Conference on Genetic Programming, EuroGP2003*, Essex, UK. Published by Springer-Verlag. Vol. 2610. pp. 276-285.

[19]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

[20]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.

[21]P.K, Samudrala, J. Ramos, S. Katkoori, S.; "Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs".*IEEE Transactions on Nuclear Science*, Volume: 51 , Issue: 5 , Oct. 2004 Pages:2957 – 2969

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

IEEE COMPUTER SOCIETY