JIAS

Journal of Intelligent Automation Systems
Published online: 24 May 2013

Evolvable Hardware Does Improve Design of Digital Combinational Circuits Performance

Mehdi Bagheri [a], Seyed Mohammad Reza Mosavi[b,*] and Karim Mohammadi[c]

Abstract: Evolvable Hardware (EH) is a new approach for designing digital circuits. In this paper, the effect of evolution variables for evolving digital combinational circuits has been considered. For this purpose, an extrinsic EH at gate level has been used. Computational effort and number of successful designs are used for evaluating the performance of evolved circuits. The effect of changing mutation rate, population size and circuit geometry on the performance of evolution has been analyzed. A 3-bit multiplier, a 7-bit even parity generator, a 2-bit full adder and an 8 to 1 multiplexer are used as benchmark circuits. Finally, we proposed a new method for changing mutation rate. Simulation results demonstrate that using dynamic mutation rate can increase the acceptable range for mutation rate value. The increase in the acceptable range by using dynamic mutation rate is about 40 times for 3-bit multiplier, about 10 times for even parity generator, about 40 times for a 2-bit adder and about 16 times for an 8 to 1 multiplexer.

Keywords: Evolvable Hardware, Combinational Circuits, Extrinsic, Gate Level, Dynamic Mutation Rate

a M.Sc. Department of Electrical Engineering, Iran University of Science and Technology, Narmak, Tehran 16846-13114, Iran.
**Corresponding author. b Department of Electrical Engineering, Iran University of Science and Technology, Narmak, Tehran 16846-13114, Iran, Professor, Email: m_mosavi@iust.ac.ir*
c Full Professor, Department of Electrical Engineering, Iran University of Science and Technology, Narmak, Tehran 16846-13114, Iran

1. Introduction

The complexity of electronic and computer systems is increasing at a rapid rate. This increase of complexity enables the designers to come up with effectively engineered systems such as airplanes, modern vehicles, mobile phones, internet, and intelligent homes. However, this level of complexity also causes problems in the design procedure, and in the management and functioning of such systems. Unfortunately, with the increasing degree of complexity, the chance occurrence of errors and mistakes in these systems becomes more prevalent. In certain remote systems like the airplanes and satellites, these errors may lead to a complete destruction of the system.

The Evolvable Hardware (EH) is a new branch of science and technology that incorporates the capabilities of four technologic areas of reconfigurable hardware, artificial intelligence, error-tolerant systems, and automatic systems. In general, an EH has two major applications. Its first application is in the instantaneous adaptation of the systems that are located in sensitive and out-of-reach places. A typical example of such an application is the designing of electronic systems related to space vehicles. The second major application of the EH is in the designing of circuits. In this application, the characteristic of the considered circuit is submitted to the evolving system, and the desired circuit is first designed, and then is optimized. The authors of this paper have concentrated on the second application.

The EH had been designed, at first, for real applications. However, problems, like the strong dependence of the performance of evolution on the parameters that could be changed by users, and the scalability problem made this method inapplicable to complicated systems [1]. To do away with these problems, different methods have been proposed so far, which we shall discuss in the second section of the paper.

This paper has been organized as follows. Section 2 refers to the relevant research works carried out on the subject. Section 3 deals with the method of evolution of combinational digital circuits in the extrinsic way. In section 4, different evaluation parameters of the performance of circuit evolutions are discussed. In section 5, the effects of various evolution parameters on the performance of circuit evolutions are reviewed. In this section, the new method presented for making the mutation rate dynamic has been introduced as well. The conclusion and future works have been reported in section 6.

2. Related works

The first and foremost problem in designing combinational digital circuits through EHs is the problem of scalability. The second problem is the dependence of the evolution performance parameters on the user-defined parameters. In this section, first, a summary of the most important research works on the subject of scalability improvement is presented. Then, several evolution methods for reducing the dependency of evolution performance on mutation rate are discussed.

2.1. Scalability

The first suggestion for solving the scalability problem was presented by Higuchi et al. in 1997 [2]. This method known as the EH at function level, instead of applying the EH at the gate levels (like AND, NAND and OR), uses functions such as adders, subtractions, and multipliers. Using the EHs at the function level enables us to apply the EHs for the larger circuits. However, the EHs at the function level suffer from three major flaws. The first flaw is that more gates are used in designing circuits. The second drawback involves the lesser potential of obtaining novel and hard-to-imagine circuits. Finally, the third flaw is that the primary functions should be designed by human beings.

The second solution to the scalability issue, known as the context switching was proposed in the year 2000, by Seok et al. [3]. This method designed for the automatic division of a complicated process, for the purpose of applying the EHs, consisted of two main parts, titled the "evolver" and the "hardware library". The main task of the evolver is to evolve a decomposition strategy by using the function nodes and terminal nodes. The function of the hardware library is to provide information for the process of hardware implementation, such as the gate, and routing.

In the same year, Kalganova presented a new technique for overcoming the scalability problem. In the symmetrical routing method, a complicated process is divided into simpler subdivisions and the EHs are applied for these simpler subdivisions. In the new presented method, this task is performed in two directions. First, the complex problem is gradually converted into several simple sub-problems. In the next step, these sub-problems gradually become more general and approach the status of the complex problem. This method has been called the “bidirectional incremental evolution". In addition to making

the chromosome lengths shorter, the bi-directionally increasing evolution also helps reduce the number of generations needed for reaching the desired goal.

In 2003, Koza et al. emphasized the importance of using the "reuse" in the EHs [4]. The reuse method enables the designer to solve a particular problem once, and then use the solution repeatedly the next times. In 2004, Stomeo and Kalganova offered a new strategy for dividing a problem into smaller subsections through forming new output functions. The number of inputs into the circuit evolving is reduced [5].

Finally, in 2006, Stomeo et al. presented a general method for decomposing a complex problem into smaller portions, and called it "Generalized Disjunction Decomposition (GDD)" [6]. With this method, we are able to evolve, in a short amount of time, very complicated circuits that no other technique has been able to evolve. Moreover, this method can reduce the number of necessary generations for the evolution of digital logic circuits, and make the achievement of higher fitness possible. Therefore, this method is very appropriate for the optimization of digital logic circuits. The GDD method has been the last technique presented in recent years, which tries to solve the scalability problem associated with the evolvable circuits.

2.2. Types of evolution methods

In 1997, Hinterding et al. presented a general classification for various types of evolution, which will be briefly explained in this section [7] [8]. The first evolution type is the static evolution, in which all the parameters of Evolutionary Algorithm (EA) are considered stable. In this type of evolution, an external factor is needed for the optimization of the evolution performance. Also, the performance of this type of evolution strongly depends on the parameter values. The second type of evolution is the dynamic evolution, in which, without the interference of an external cause, one or several of the EA parameters change(s). With respect to the class of EA that is used, the dynamic evolution is divided into three subdivisions explained below.

The first type of dynamic evolution is the "deterministic dynamic evolution" in which, the change of one or several parameters of EA is based on an algebraic relationship, and no feedback from the EA is used for changing the parameters. This method can be used for changing the parameters of EA, as a function of time or the current iteration number. In 1989, Fogarty used the deterministic dynamic mutation rate [9]. He applied Equation (1), which he had obtained experimentally:

$$P_m(t) = \frac{1}{40} + \frac{0.11375}{2^t} \tag{1}$$

In this equation, t represents the elapsed time since the onset of the evolution, and as the evolution proceeds, the mutation rate decreases from 0.1179 to a value of 0.00416. In later years, newer algebraic equations were presented for this method, which were a function of the current generation number. For example, in 1996, Back and Schutz used Equation (2) for reducing the mutation rate [10]:

$$P_m(t) = (2 + \frac{l-2}{T-1} n)^{-1} \tag{2}$$

where n is the current generation number, l is the length of the string, and T represents the maximum generation number.

The second type of dynamic evolution is the "adaptive dynamic evolution" in which, a kind of feedback from the EA is used for changing one or several parameters of the EA. In 1973, Rechenberg applied this method for the adaptive evolution process, which became known as the Rechenberg's 1/5 law [11]. In this method, he considered the ratio of the number of successful mutations to the total number of mutations as 1/5 and if the above ratio was more than 1/5, he would increase the mutation rate, and if it was less than 1/5, he would decrease the mutation rate.

The third type of dynamic evolution is the "self-adaptive dynamic evolution". In this method, the idea of evolvable EA is used for the automatic adaptation of one or several parameters of the EA. In this technique, the parameters that are set to evolve, are entered into a chromosome as a gene, and evolve along with the evolution of the system. In 2004, Hartono devised a method known as the "labeled genetic algorithm", where the parameters set to change, enter the genes, instead of the chromosome.

In 2002, Krohling et al. designed an EA with adaptive dynamic mutation rate, which was suitable for controlling a robot via the FPGA [12]. In his procedure, for the society members with a higher fitness function values, they used a smaller mutation rate, and for the members having a lower fitness function values, they applied a larger mutation rate. By doing this, they were able to cut the number of generations necessary for reaching the target.

Although, many approaches have been presented, so far, for changing the evolutionary parameters, the

performance of these methods in the evolution of circuits, has not been evaluated. In this paper, the effects of different parameters on the evolution fitness of combinational digital circuits have been investigated. Also, a new method has been offered for making the mutation rate dynamic, and its fitness has been compared with that of other approaches.

3. Designing of combinational circuits by mean of evolutionary hardware

In this section, the common procedure for the evolution of a combinational circuit has been outlined completely. The evolution strategy, manner of chromosome encoding, fitness function and the genetic operators that have been employed are described in this section. Also in this paper, for evaluating the performance of the dynamic mutation rate, the strategy explained in this section for the evolution of circuits has been adopted.

3.1. Evolutionary algorithm

The EA that is employed here is the EA known as $\lambda+1$, in which λ indicates the size of the society. Figure 1 clearly illustrates the various steps of the process. First, all the chromosomes are produced randomly. In the second step, the fitness of each chromosome is estimated. The third step involves the selection of the chromosome that has the best fitness value. In the fourth step, the chromosome selected in the third step is evaluated to see if it meets the conditions of the evolution completion. These conditions are: (1) the chromosome's fitness value is 100%, or (2) the number of produced generations is larger than the permitted number of generations that has been set by the user for that test. If one of these conditions is fulfilled, the evolution is complete. Otherwise, λ through time mutations of the best chromosome selected in the third step, a new population is formed, making the number of new members equal to $\lambda+1$. In the next step, all the newly created members are evaluated and the fitness of these members is compared to the fitness of the best chromosome of the previous step. Thus, the best chromosome of the new generation is selected. This process continues until the conditions of the evolution completion are fulfilled.

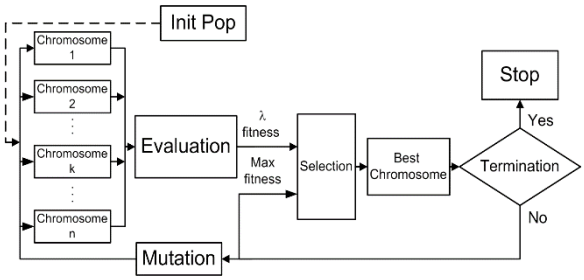

Fig. 1: Block diagram of used evolutionary method

3.2. Chromosome encoding

The chromosome specifies the structure of a logic circuit, and expresses the communications between the gates. In this approach, the logic circuit is considered as a rectangular array of logic cells. The type of each logic cell is selected randomly from the set of primary functions AND, OR, NOR, and NAND. The chromosome is defined by a tri-layer structure.

In Figure 2, an example of chromosome encoding has been shown for a sample circuit with three inputs and two outputs. The overall design of the circuit is depicted by an array with two rows and three columns, and the level of internal connections parameter has been considered as 3. Therefore, the chromosome is represented as 2, 3, and 3 in the geometric layer. In the functional layer, the chromosome determines the array of logic cells from a functional perspective. For example, the logic gate identified with the output number 7 is an OR gate. An OR gate has been encoded by the number 8. Thus, the first two data related to the functional layer chromosome are 7 and 8. The last two data in the functional chromosome specify the final outputs of the circuit. In this particular case, for example, the final outputs of the circuit have been taken from the outputs of logic gates 4 and 5. The logic gate with the output number 4 is an AND gate and the AND gate has been encoded by number 7. Therefore, the next two data associated with the chromosome in the functional layer are 4 and 7.

The chromosome in the connections layer specifies the logic gate number, the number of inputs to the gate, and the outputs of the other gates from which these inputs have been taken. The chromosome of the first logic gate, in the first column and first row, includes the number 3, which denotes the number of the logic gate. The following number, 2, indicates that this gate has two inputs. The next numbers (0 and 1) indicate that, the inputs to this gate have been taken from the X_0 and X_1. The

descriptions of the rest of the logic gates are similar to this gate.

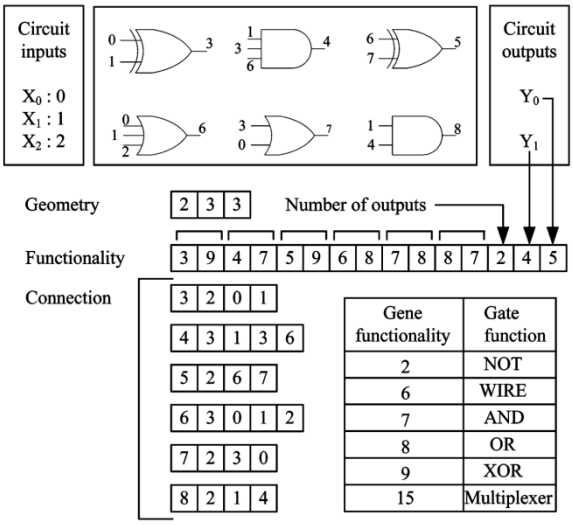

Fig. 2: An example of chromosome encoding [6]

3.3. Fitness Function

The fitness function evaluates the performance of the evolved circuit. In the process of the evolution of combinational circuits, usually two fitness functions are used; one at the designing stage, and the other at the optimization stage. At the designing stage, the goal is to obtain a circuit that has a fitness of 100%. At this stage, the number of gates used, and the other performance parameters are not considered. The fitness function for the design criteria is obtained from Equation (3):

$$f_1 = \sum_{f_c=0}^{2^n-1} \sum_{i=0}^{m-1} \left| y_i - d_i \right| \tag{3}$$

where m and n are the number of outputs and inputs of the given logic function, respectively, y_i is $i\,th$ digit of the output combination that has been obtained through the evaluation of the evolved circuit, d_i is the desired output for the considered f_c, and finally, $\left| y_i - d_i \right|$ is the absolute difference between the output value generated by the evolved logic circuit and the desired value. The maximum value of the fitness function is equal to $2^n \times m$. After obtaining a 100% efficient circuit, it is time to optimize the circuit. At this stage, the circuit can be optimized in terms of various fitness parameters. One of these parameters is the number of gates used in the circuit. The fitness function for the reduction of gate quantities is as follows:

$$f_2 = f_1 + Rows \times Cols - A_{Cells} \tag{4}$$

In this equation, *Rows* and *Cols* indicate the number of rows and columns of the array used for the evolution procedure, respectively. A_{Cells} is the number of gates used in the circuit.

3.4. Genetic Operator

In this sample system, two genetic operators (mutation and elitism) have been used. Elitism insures the transfer of the best chromosome from the current generation to the next generation. The mutation operator is used for the alteration of some existing data in the chromosomes. The main goal of this operator is the creation of change in the society. With the increase of the mutation rate, the genetic search turns into a random search; but it helps us discover the lost solutions once again. Here, the cross over operator has not been used.

As mentioned above, a new method has been offered for making the mutation rate dynamic. In this section, we explain this method. As stated in the related works section, there are three different approaches for making the mutation rate dynamic. In this paper, the method of algebraic dynamic mutation rate has been used. In this paper, the mutation rate changes as a function of the value and maximum value of the fitness function. Equation (5) is used for the change of the mutation rate:

$$P_m = \begin{cases} P_{m,start} \times (1 - \dfrac{F}{F_{\max}}) & ;\ P_m > P_{m,\min} \\ P_{m,\min} & ;\ P_m \le P_{m,\min} \end{cases} \tag{5}$$

where $P_{m,start}$ and $P_{m,\min}$ are the initial and minimum values specified for the mutation rate, respectively. $F_{\max}$ and F represent the maximum value of the general fitness function and the maximum value of the fitness function in the current generation, respectively. The mutation rate decreases linearly as a function of time by applying Equation (5). Since the use of very small mutation rates in the EHs leads to the diminishing of the evolution fitness, a low limit has been set for the mutation rate [13]. In the simulation results section, this method will be compared to the other methods.

4. Performance parameters for the evolution of combinational digital circuits

The performance parameters are some criteria for the evaluation of the quality of circuit evolutions. The parameters that have been cited by numerous researchers are presented in this section, along with the manner of their calculations. Due to the random nature of circuit evolution by the genetic algorithm method, to decide on the quality of an evolved circuit, that circuit should go through several evolutions. Then, the obtained raw data should be subjected to statistical analysis. In this research, each circuit has been evolved 50 times.

4.1. Number of successful evolutions (NSEs)

This parameter indicates the NSEs in the course of 50 different executions of the evolutionary program. A "successful evolution" means an evolution that has resulted in the production of a circuit with a 100% accurate performance. Some parameters, like the computational effort, can only be defined for circuits. This parameter value is equal to 50.

4.2. Computational effort (CE)

This parameter is used for the calculation of the required number of generations so that with a specified probability (z), we can achieve a circuit with 100% performance. In most of the works related to the evolution of combinational circuits, this parameter is usually referred to as the main performance parameter. Ordinarily, in the existing references, the z parameter has a value of 0.99. Therefore, in this paper also, a value of 0.99 has been considered for the z parameter. This means that, if we want to succeed (with a 99% probability) in evolving a circuit, it is expected that the number of required evaluations be equal to the CE number. For the calculation of this parameter, the following equations are used [14]:

$$P(M,i) = \frac{N_S(i)}{N_{total}} \tag{6}$$

$$R(z) = ceil\left(\frac{\log(1-z)}{\log(1-P(M,i))}\right) \tag{7}$$

$$I(M,i,z) = M \times R(z) \times i + 1 \tag{8}$$

$$CE = \min_i I(M,i,z) \tag{9}$$

In the above equations, $N_S(i)$ expresses the number of successful executions in the total number of executions, at the generation number i. For example, it is possible that, in the course of 50 evolutions, 8 evolutions attain the answer, exactly at the generation number 3678. In this case, $N_S(3678) = 8$. N_{total} denotes the total number of various executions of the EA. In this study, to simplify the calculations, it is considered that $N_S(i) = 1$. The reason is that, because of the diversity of different executions of evolution, it seems improbable that the generation numbers for different executions of the algorithm be identical. $P(M,i)$ is the cumulative probability function of success, with the number of population M and the number of generations i. For example, if the NSEs, for which the number of required generations is less than 3678, is 20, then, $P(M,3678) = 20/50$. $R(z)$ indicates the number of independent executions necessary for the evolution of a circuit, with the probability z, and the number of generations (i). The $I(M,i,z)$ parameter denotes the number of evaluations required for achieving a 100% efficient circuit at the number of generations (i) and with the probability z. To estimate the CE in this research, we have first calculated the $P(M,i)$. To calculate this parameter, after arranging the data in the primary table, Equation (10) has been used:

$$P(M,i) = \frac{\text{Row index}}{\text{Total number of runs}} \tag{10}$$

After this step, the following equation has been used to calculate the value of $I(M,i,z)$ for each row:

$$I(M,i,z) = M \times ceil\left(\frac{\log(1-z)}{\log(1-P(M,i))}\right) \times i + 1 \tag{11}$$

Smallest value of $I(M,i,z)$ that is obtained from the above equation represents the CE parameter.

5. Simulation results

In this section, the effects of the parameters used in the evolution on the evolution performance have been examined. For this purpose, the parameters discussed in the previous section have been used, as performance parameters, for the sake of comparison. A 3-bit multiplier circuit along with a 7-bit even parity generating circuit, a 2-bit adder circuit, and an 8 to 1 multiplexer circuit have been selected to undergo the evolution process. The use of these circuits is very common in the area of circuit design by

means of EH. Due to the random nature of the EAs, each of the circuits has been evolved 50 times and their performance parameter values have been averaged.

Table 1 shows that the presumed parameters are used in the evolution. In the performed simulations, when we want to investigate the impact of one parameter on the evolution performance, we consider the rest of the parameters as stable and use the parameters listed in Table I. Also, in all the evolutions, only the AND, OR, NAND, and NOR gates have been used.

Table 1. Presumed Parameters Used in the Evolution

Parameter	Value
Population size	5
Mutation rate	3%
Mutation rate minimum limit	1%
Max number of generations	3000000
Max generation for stalling	1000000
Number of rows	1
Number of columns	100
Level back	100
Mutation rate type	Constant

5.1. Study of the effect of mutation rate

In this section, the effect of the mutation rate on the performance parameters of circuit evolution is reviewed. For this purpose, two types of mutation rates have been used.

The first type is the fixed mutation rate, and the second type is the dynamic mutation rate introduced in section 3.4. Figures 3 and 4 show the variation of the CE and the NSEs for the four noted circuits that are evolved by applying the fixed mutation rate.

As these figures clearly illustrate, when using the fixed mutation rate, the CE parameter and the NSEs become strongly dependent on the selected value of the fixed mutation rate. For example, for the evolution of the 3-bit multiplying circuit, if the chosen fixed mutation rate is larger than 5, the CE becomes very large and the NSEs become zero. This condition is true for the evolution of the simpler 2-bit adding circuit, although with less intensity. For example, for the evolution of the 2-bit adding circuit by means of the constant mutation rate, the mutation rate value should be approximately less than 15. Therefore, there is a higher dependency of the evolution parameters on the fixed mutation rate in the more complicated circuits such as the 3-bit multipliers, compared to simpler circuits like the 2-bit adders. Figures 5 and 6 show the variation of the CE and the NSEs for the four mentioned circuits that are evolved by using the dynamic mutation rate introduced in section 3.4. As is obvious from these Figures, by using the dynamic mutation rate, the dependency of the CE parameter and the NSEs on the initially selected mutation rate becomes less. For example, for the evolution of the 3-bit multiplying circuit, if the initially selected mutation rate is less than 30, the CE remains almost stable. This situation is also true for the evolution of the simple 2-bit adding circuit. For example, in the evolution of the 2-bit adding circuit using the dynamic mutation rate, the initial value of the mutation rate has a little influence on the value of the CE.

The more important point is that, through the use of the dynamic mutation rate, the NSEs for the four considered circuits have been almost equal to 50. This shows that the application of the dynamic mutation rate has improved the process of circuit evolutions. Thus, it can be concluded that, by using the dynamic mutation rate, the dependency of the evolution parameters on the initially chosen mutation rate decreases.

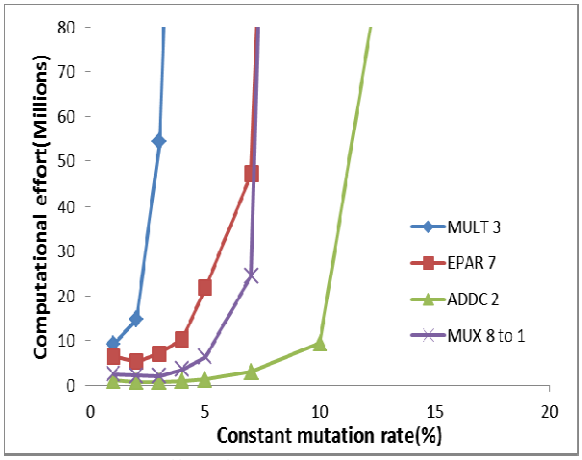

Fig. 3: Effect of constant mutation rate on CE

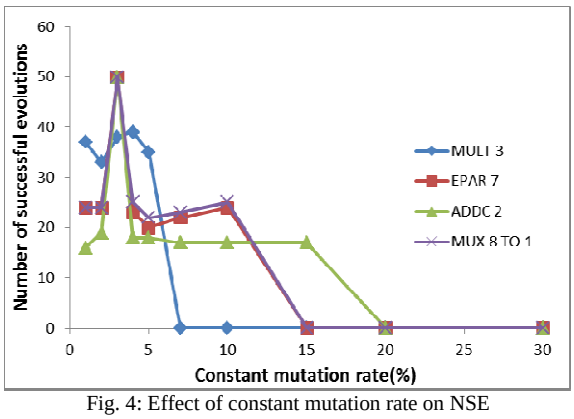

Fig. 4: Effect of constant mutation rate on NSE

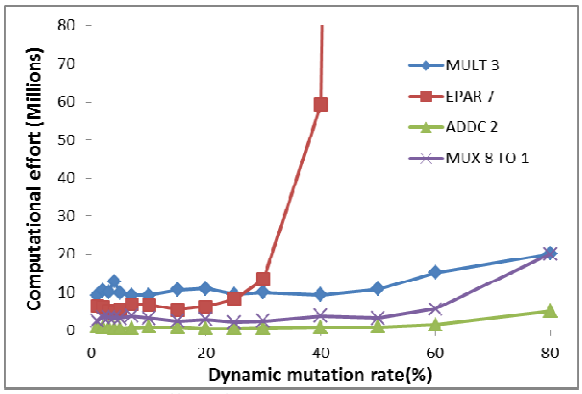

Fig. 5: Effect of dynamic mutation rate on CE

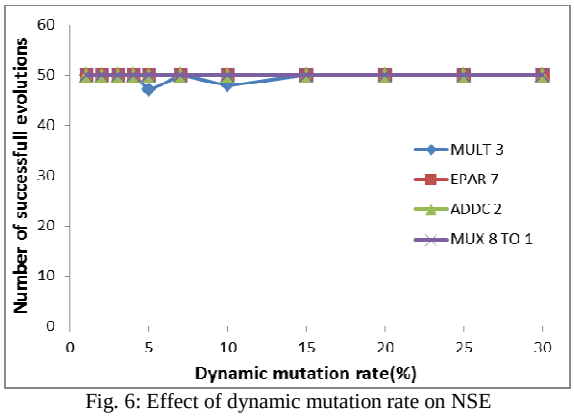

Fig. 6: Effect of dynamic mutation rate on NSE

5.2. Study of the effect of the population size

In this section, the effect of the population size on the performance parameters of circuit evolution is examined. Figures 7 and 8 show the variation of the CE and the NSEs for the four noted circuits that are evolved by using the parameters listed in Table I. In view of Figure 7, we find out that with the increase in the number of population members, contrary to expectation, the performance parameter of the CE gets worse. The reason is that, as mentioned before, the CE parameter is a criterion of the number of evaluations conducted for the evolution of the circuit; meaning that the average number of generations required for evolution is multiplied by the number of society members. In societies with few members, the average number of necessary generations exceeds the number of society members. However, although the use of societies with a large number of members, leads to an average reduction in the number of generations necessary for evolution, but when this number is multiplied by the number of society members, the outcome is a larger number compared to the evolution with fewer generations. Therefore, the use of the average number of generations required for evolution cannot be an appropriate parameter for the evaluation of the performance of circuit evolutions; because, as the number of society members increases, the average number of generations necessary for evolution decreases. Thus, the CE parameter can be used to obtain an optimum value for the number of society members.

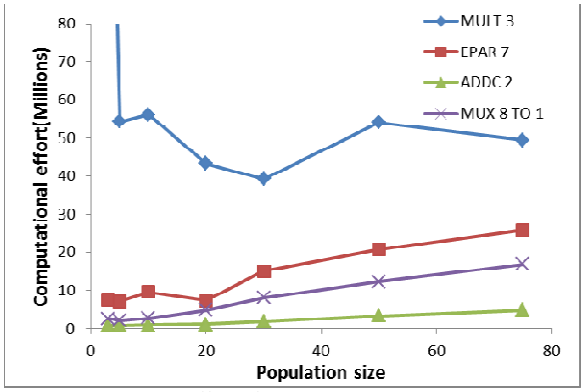

Fig. 7: Effect of population size on CE

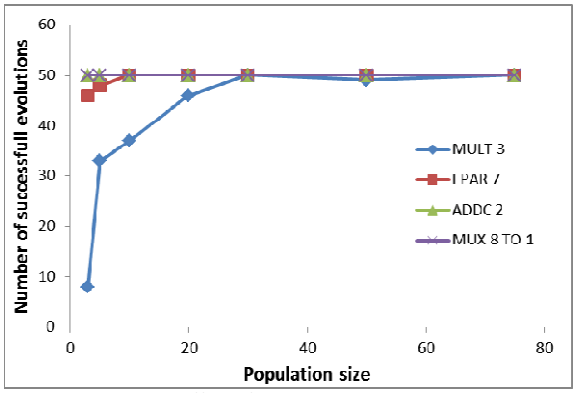

Fig. 8: Effect of population size on NSE

The other point is that, the NSE is independent of the number of society members. As can be observed from the diagram of Figure 8, for a society with more than 10 members, almost all the circuits have had 50 successful evolutions out of 50 intended evolution operations. The only exception relates to the evolution of the 3-bit multiplying circuit, for which, for the number of society members less than 5, the NSEs reduces drastically. Thus, the diagram of the NSEs can present a lower limit for the number of society members. However, for obtaining an optimum quantity for the number of society members, the best parameter is the CE.

5.3. Study of the effect of the number of circuit columns

In this section, the effect of the number of circuit columns on the performance parameters of circuit evolution is reviewed. Figures 9 and 10 illustrate the variation of the CE and the NSEs for the four mentioned circuits that are evolved by using the parameters listed in Table 1. With regards to Figure 9, we realized that with the increase of the number of circuit columns, the performance parameter of the CE improves. In this Figure, the dependence of the CE on the number of columns used in the circuit structure has been demonstrated. Since the CE parameter depends more on the average number of generations required for evolution and on the number of society members, this parameter, therefore, does not express the time required for one repetition of the EA. By increasing the number of columns used in the circuit structure, the required time for one repetition of the EA increases.

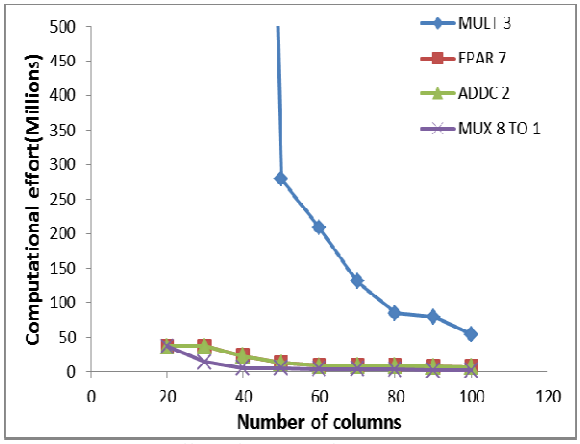

Fig. 9: Effect of number of columns on CE

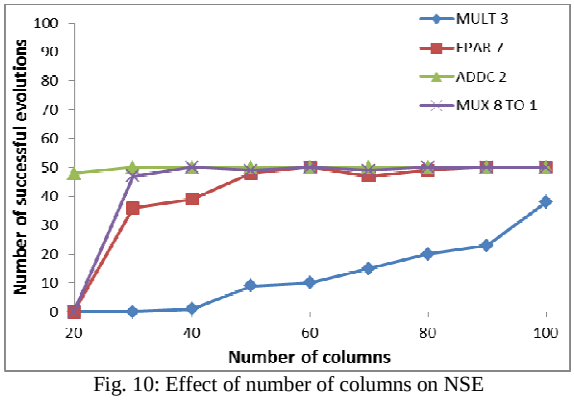

Fig. 10: Effect of number of columns on NSE

The more important point is the strong dependence of the NSEs on the number of columns used in the 3-bit multiplying circuit structure. The major difference between the 3-bit multiplying circuit and the other evolved circuits is the higher complexity of this circuit relative to the other ones. The conclusion from the diagram of Figure 10 is that for more complicated circuits, we need more columns. For example, in the case of the 2-bit adding circuit, it is obvious that even with fewer columns, the NSE is close to 50. The reason for this is the lower complexity of this circuit. As can be observed in Figure 10, for the number of columns higher than 30, almost all the circuits have had 50 successful evolutions out of 50 intended ones. The sole exception belongs to the evolution of the 3-bit multiplying circuit, in which, for less than 80 members, the NSEs decreases considerably. Therefore, the diagram depicting the NSEs versus the number of columns used for evolution can be regarded as a criterion for the complexity of a circuit.

5.4. Results obtained from the evolution of various circuits

Through the use of simulations, we reach the conclusion that in the evolution of combinational digital circuits, the effects of different evolution parameters on the performance of evolved circuits are much more significant for complicated circuits than for the simpler circuits. For example, the choice of initial parameters for the evolution of a 3-bit multiplying circuit is much more sensitive than the choice of the initial parameters for the evolution of a simple 2-bit adding circuit. Therefore, it is useful to obtain a method that reduces the sensitivity of the evolution performance parameters on the initial parameters. As was obvious from the simulation results in section5.1, making the mutation rate parameter dynamic, substantially reduces the dependency of the circuit's evolution performance parameters on the initially selected mutation rate. Table II shows the range of the obtained parameters for achieving the highest performance value. As this Table demonstrates, the use of the dynamic mutation rate has led to the expansion of the acceptable range for the mutation rate parameter. In consideration of Table II, it can be concluded that the use of the dynamic mutation rate, has improved the acceptable range of the mutation rate almost 40 times for the 3-bit multiplying circuit, almost 10 times for the 7-bit even parity generating circuit, almost 40 times for the 2-bit adding circuit, and around 16 times for the 8 to 1 multiplexer circuit.

Table 2: Range of the obtained parameters for achieving the highest performance value

Performance parameter	Computational effort				Number of successful evolutions			
Evolution parameter	Constant mutation rate	Dynamic mutation rate	Population size	Columns	Constant mutation rate	Dynamic mutation rate	Population size	Columns
3-bit multiplier	1	1-50	20-30	> 80	1	1-40	> 20	> 80
7-bit even parity	1-3	1-25	5-20	> 60	1-3	1-30	> 5	> 50
2-bit adder	2-4	2-40	3-10	> 40	1-10	1-80	> 3	> 40
8 to 1 multiplexer	1-3	1-50	5-10	> 70	1-5	1-80	> 3	> 70

The other conclusion drawn from the results of Table II is that, with the rise in the number of society members, contrary to expectation, the CE performance parameter does not improve. The CE parameter represents a criterion for the number of performed evaluations for the evolution of a circuit. So, the assumption is that by increasing the society members and thus and by reducing the average number of generations required for evolution improving the evolution performance of circuits are wrong. However, increasing the number of society members causes an improvement in the NSEs parameter.

Also, the use of the diagrams associated with the number of columns used in the circuit can be a criterion for the complexity of the evolved circuit. The less is the minimum number of columns necessary for the evolution of a circuit, the simpler that circuit is for the purpose of evolution. For example, looking at Table II, the minimum number of columns required for the evolution of the simple 2-bit adding circuit is 40; while for the more complicated 3-bit multiplying circuit, this same number of columns is 80. Therefore, the results indicate that the 3-bit multiplying circuit is more complex than the 2-bit adding circuit that is compatible with the operation.

6. Conclusion

In this paper, EH was used to evaluate the impact of effective parameters on the evolution performance of combinational digital circuits. For this purpose, four circuits consisting of a 3-bit multiplying circuit, a 7-bit even parity detection circuit, a 2-bit adding circuit, and an 8 to 1 multiplexer circuit were evolved by applying different parameters. Then, the efficiencies of the evolved circuits were compared against the selected parameters. The manner of change of the CE versus the number of columns used for circuit evolution was introduced as a criterion for the assessment of circuit complexity. Furthermore, with the evolution of various circuits through the use of the dynamic and fixed types of mutation rates, it was shown that the use of the dynamic mutation rate leads to a very significant improvement in the performance of the EA for the more complicated circuits. In addition, the use of the dynamic mutation rate considerably reduces the dependency of the evolution performance on the initially selected mutation rate. Also, the use of the dynamic mutation rate has improved the acceptable range of the mutation rate almost 40 times for the 3-bit multiplying circuit, almost 10 times for the 7-bit even parity generating circuit, almost 40 times for the 2-bit adding circuit, and around 16 times for the 8 to 1 multiplexer circuit.

References

[1] C.W. Greenwood and A.M. Tyrrell, Introduction to evolvable hardware, a practical guide for designing self-adaptive systems, IEEE Press Series on Computational Intelligence, 2007.

[2] T. Higuchi, M. Murakava, M. Iwata, I. Kajitani, W. Liu and M. Salami, Evolvable hardware at function level, in Proc. IEEE Evolutionary Computation Conf., 187–192, 1997.

[3] H.S. Seok, K.J. Lee, B.T. Zhang, D.W. Lee and K.B. Sim, Genetic programming of process decomposition strategies for evolvable hardware, The Second NASA/DoD Workshop on Evolvable Hardware, 25–34, 2000.

[4] J. R. Koza, M.A. Keane and M.J. Streeter, The importance of reuse and development in evolvable hardware, In proc. NASA/DoD Evolvable Hardware Conf., 33–42, 2003.

[5] J. Lee and J. Sitte, A gate-level model for morphogenetic evolvable hardware, In Proc. IEEE Field-Programmable Technology Conf., 113–119, 2004.

[6] E. Stomeo, T. Kalganova and C. Lambert, Generalized disjunction decomposition for evolvable hardware, IEEE Trans. System, **36**(5), 1024–1043 (2006).

[7] R. Hinterding, Z. Michalewicz and A.E. Eiben, Adaption in evolutionary computation: A survey, In Proc. IEEE Evolutionary Computation Conf., 65–69, 1997

[8] H.G. Beyer, The theory of evolution strategies, Natural Computing Series, Springer, 2001.

[9] T. Fogarty, Varying the probability of mutation in the genetic algorithm, In Proc. Third Genetic Algorithms conf., 104–109, 1989.

[10] T. Back and M. Schutz, Intelligent mutation rate control in canonical genetic algorithms, In Proc. International Symposium on Methodologies for Intelligent Systems, 158–167, 1996.

[11] I. Rechenberg, Evolutions strategie: optimierung technischer systeme nach prinzipien der biologischen evolution, Frommann, 1973.

[12] A.R. Krohling, Y. Zhou and M. Tyrrell, Evolving FPGA-based robot controllers using an evolutionary algorithm, Presented at the 1st Int. Conf. Artificial Immune, 2002.

[13] E. Stomeo, T. Kalganova and C. Lambert, Mutation rate for evolvable hardware, World Academy of Science, Engineering and Technology, 2005.

[14] J.A. Walker and J.F. Miller, The automatic acquisition, evolution and reuse of modules in cartesian genetic programming, IEEE Trans. Evolutionary Computation, **12**(4), 397–413 (2008).