

# Synthesis of Time-to-Amplitude Converter by Mean Coevolution with Adaptive Parameters

Yerbol A. Sapargaliyev, Tatiana G. Kalganova

Brunel University, London, UK.  
Email: {yerbol.sapar, tatiana.kalganova}@brunel.ac.uk

Received June 20<sup>th</sup>, 2011, revised July 17<sup>th</sup>, 2011, accepted July 26<sup>th</sup>, 2011.

## ABSTRACT

*The challenging task to synthesize automatically a time-to-amplitude converter, which unites by its functionality several digital circuits, has been successfully solved with the help of a novel methodology. The proposed approach is based on a paradigm according to which the substructures are regarded as additional mutation types and when ranged with other mutations form a new adaptive individual-level mutation technique. This mutation approach led to the discovery of an original coevolution strategy that is characterized by very low selection rates. Parallel island-model evolution has been running in a hybrid competitive-cooperative interaction throughout two incremental stages. The adaptive population size is applied for synchronization of the parallel evolutions.*

**Keywords:** Analog Circuit Synthesis, Evolutionary Electronics, Computational Circuits, SPICE

## 1. Introduction

The analog circuit is much more difficult to design than the digital one due to the complex and knowledge-intensive nature of the analog electronics. Without an automated synthesis methodology, analog circuit design has suffered from long design time, high complexity and cost, and requires large experience. Therefore, automated synthesis methodologies for analog circuits have received much attention.

In recent years, evolutionary strategy (ES) has been found to be one of the most powerful evolutionary algorithms (EA) when applied toward the synthesis of electronic circuits [1-4]. Despite ES proving itself effective in solving the scalability problem and reaching great experimental results, the existing ES-based circuit synthesizers by their technique are far behind the state-of-the-art EA theoretical developments. Moreover, in the area of analog circuit synthesis, the number of publications on ES applications converges are few in comparison with those of genetic algorithms and genetic programming.

In this paper, the power of ES-based EHW system is verified by a challenging task to design the *time interval meter circuit* (TIMC), which is the core of up-to-date laser rangefinder. The targeted analog TIMC belongs to a class of devices that are known as “time to amplitude converters” (TAC). “TAC generates a rectangular output pulse whose peak amplitude is linearly proportional to

the time interval between a START and STOP input pulse pair” [5]. To our knowledge, this is the first attempt toward automatic synthesis of a TAC circuit.

To solve the problem, a recently developed ES-based system [1] is upgraded with a combination of the novel adaptive individual-level *differentiated mutation* (DM) and the *winner-dominates-winner-cooperates* (WDWC) coevolution strategy formed by means of parallel island-model evolution.

These two techniques are combined into a “very narrow focused search tool” that we name for simplicity as *very narrow focused evolution* (VNFE). The literature review gives on the subject of “optimal selection rate” “an idea that the selection is a mechanism, which increases the mean fitness of a population while “having the least deleterious effect” [6] on the genotypes and thus is directly proportional to the size of a population. In this work, we use relatively large populations (from 15,000 to 35,000 individuals evolving in parallel), each of which uses very low selection rates (SR) from 0.2% to 2% and aggregately the SR of the system reaches 0.048%. Furthermore, the proposed technique theoretically enables all the parallel evolutions “focus” on a single chromosome at one generation, which in our experimental case can bring up to 0.0006% SR<sup>1</sup>.

<sup>1</sup>While during an experiment we have achieved a selection rate equal to 0.048%, theoretically the value 0.0006% is reachable.

The idea of VNFE rather comes from experiments than from theory of EA. In earlier works, we already had reported about some aspects of DM and adaptation [1,3]. In later works, we found that the system performs better at SRs 10 times lower than before (from 10% to 1%) [3]. Despite proving our recent technique to be a powerful tool in comparison with others [7], we have failed with it when evolving TIMC. Within research in the frame of this paper, we have applied incremental approach, decomposing TIMC into two subcircuits. However, the second stage has failed to converge. Moving to strengthening the evolution power by increasing the population size, we have applied parallel computing, which finally has brought us to the WDWC strategy. Thus, the reported methodology is figuratively a “success story” of where our endeavors in searching for suitable technique have brought us.

The rest of this paper proceeds as follows. Section 2 provides a background on adaptation, incremental evolution, and parallel evolution. Section 3 describes the system, task description, and experimental setup. Section 4 discusses results of an experiment: the circuit designed and the behavior of an evolution. Section 5 provides a comparison of the successful results with the failed ones. And, finally, in the last section we draw some conclusions.

## 2. Background

### 2.1. Adaptation

To get successful results using an EA, one needs good parameters such as the mutation rate (MR), SR, and the crossover rate. They predetermine how the EA will solve the particular task. Often parameters have to be predefined or tuned manually and are only optimal for a specific problem [5]. To find optimized parameters for a certain application, researchers generally base their choices on tuning them by hand, that is, experimenting with a multitude of values and selecting the one that exhibits the best performance. For instance, different values of MR are desired at different stages of the evolutionary process to achieve balance between global and local searches. Tuning the rates manually is very time consuming where the tuned result is only efficient for some particular instance. The space of operators and parameters is large. Therefore, hand-designed adaptive mechanisms have had relatively less success, and there has been natural interest in the application of adaptive techniques. The important feature of adaptation is that the algorithm can be adjusted to the particular task while solving that task. In general, there are three levels where the adaptation may take place inside evolution:

- Population-level adaptation adjusts control parameters

that apply for the entire population. This approach is most presented in the literature;

- Individual-level adaptation is focused on adapting parameters for every chromosome. For instance, each chromosome has its own crossover and MR [8]. They may be varied depending on the convergence state of the population, the fitness value of the chromosome, the average fitness value of the population, and whether the population tends to get stuck at a local optimum or is scattered in the solution space. This approach looks most perspective from our point of view. The convincing example here is the work [9], where the automatically defined function is an individual-level adaptive genetic program where each individual adapts its definitions for a predetermined set of subroutines.
- Component-level adaptation dynamically alters how the particular gene of each chromosome will be manipulated independently from each other [10].

In this paper, we propose a novel and feasible individual-level adaptive mutation and adaptive population size schemes for our analog circuit synthesis system and apply it toward TIMC. The mutation operator has been suggested to be the most sensitive parameter in the theory of EA [10], as well as in a digital electronic circuit synthesis domain [4]. In our approach, adaptation is based on the concept of defining the particular mutation type for every chromosome using the current and past features of the chromosome and the population it comes from. In our method, the parameters that control the MR of a chromosome are not encoded into their corresponding chromosome as additional genes [8], but are represented by such chromosome and population characteristics as the chromosome length story, the chromosome mutation story, and the chromosome and population fitness stories. For example, the mutations that caused the creation of better chromosomes are used more frequently in further generations. On the other hand, operators who produced chromosomes with a lower fitness should be used more rarely.

There are two general features of the adaptive DM technique that make it distinctive from others. Firstly, as each chromosome has its own “personality,” described by fitness, length, past behavior, etc., it requires a “personal approach” when choosing and applying a mutation to it. In an evolutionary analog circuit synthesis, the mutation is not just specified by rate, but could be classified by types, for example, the mutation of a node connection, parameter, component name, etc. This approach is more individual specific and excludes additional randomizations, which are involved in the “evolution-of-mutation-rate” approach. Secondly, the novel approach suggests an economy of computing efforts. Because there is no need

of additional genes in the chromosome for coding MR, the shorter chromosome requires less time for evaluation, operation, memorization, etc.

## 2.2. Incremental Evolution

The *incremental evolution* is regarded as one of the main techniques to tackle the scalability problem. One of the first attempts to apply it was undertaken in [11] toward digital circuits. Since then, many approaches have been developed in the digital domain [2,12,13]. In the analog area, few works have distinctly utilized these approaches [14,15]. Furthermore, the targeted circuits were not complex enough to exploit the potential of the technique. The essence of the *incremental evolution* lies in “divide and conquer” method, when the task is decomposed into subtasks, and then the subtasks are solved step by step. In the evolutionary electronics, the division can be made based on parallel or series connection of the subcircuits. The whole process of multistep evolution can be arranged in an automatic regime. However, we distinguish the incremental approach from the “divide and conquer” in how the subcircuits are united.

In the analog electronics, the subcircuits cannot be easily connected to get the proper functioning solution. That is, two perfectly working circuits, when connected to a common input, are not guaranteed to perform the same way; more likely each circuit will disturb the function of its neighbor. This comes from the physical nature of the electronic components that get influenced from each other by potentials and currents. This situation differs from digital circuits, where the *Boolean algebra* and the complex task could be decomposed by *Shannon’s expansion theorem* or *output decomposition* [2]. For instance, in [13], the digital circuit with multiple outputs was broken down into many smaller subcircuits (each encoded by a single chromosome) with a single output. Due to the digital nature of the target, the “divide and conquer” approach enabled the parallel evolution per each subcircuit. And the final solution was built by simple joining “bricks into a wall.”

In this context, when we mention *incremental evolution* we mean, first of all, not independent evolution of the targets, but rather the evolution of the current target together with all targets evolved previously. That is, if one has the already evolved target, when evolving the second one the first solution must participate in that evolution, being encoded in the chromosome. This fact decreases the benefit of the “divide and conquer” approach for analog circuit synthesis in comparison to other applications.

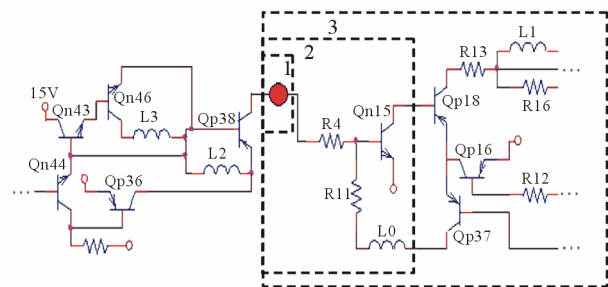
However, despite significant increase of the chromosome length with each stage, the need to involve every gene of the previously evolved solution into every evolu-

tion operation is not necessary. What is obligatory is just participation in the *evaluation* process, that is, in getting the adequate fitness value of the whole chromosome. As we use ES, where the recombination is not used, the main evolution operation is a mutation. We regard three options by degree of involving of genotypes of the previous solutions into mutation:

1) When the fragments of the chromosome that belong to previous solutions do not participate in all kinds of mutation. On one hand, this option keeps the solution space constrained and saves computing efforts. On the other hand, removing the opportunity of the previous solutions to adapt their structures and parameters to a new more general solution may obstruct finding a current subcircuit and even leave the process out of any solution.

2) To avoid problems in the first option, the second option suggests enabling some loci of the previous solutions to participate in evolution along with the rest of the chromosome. These loci represent the components’ names, nodes, and parameters of the evolved subcircuits that are located at the junctions between the currently evolving subcircuit and the previous one(s). These junctions are predefined when one is decomposing the task into subtasks at the start. One can expand the length of the chromosome to be participated in mutation by adding not only the genes coded for components at junctions, but also by adding genes coded for their neighbors, and the neighbors of neighbors, etc. (**Figure 1**).

3) And the third case is when the genotypes of the previously completed subtasks have the same rights to participate in all kinds of evolutionary operations as the genotypes related to the current subtask. In this case, the power of “divide and conquer” method drastically falls down due to extreme expansion of the search space.



**Figure 1. Incremental approach:** Two subcircuits that are joined in a point marked by a red circuit. On the right is a subcircuit evolved first, on the left is a currently evolving one. When evolving a subcircuit, there are three degrees of involving of parts of the previous solution(s) in the current process: non-involving (marked by dotted square 1), partially where there are components neighboring to a junction point (square 2) and full, when every component of a previous subcircuit(s) take(s) a part in the evolution along with components on the left side of the figure (square 3).

However, the positive side of such an approach, when it is applied to a deliberately easy problem, is an unconventional design and the foreseen component economy of the solution.

When one is utilizing the extrinsic evolution with the help of PSPICE, one could utilize the PSPICE built-in function for subcircuits that are instantiated using letter "X." This statement causes the referenced subcircuit to be inserted into the circuit "using the given nodes to replace the argument nodes in the definition" [16]. It allows a block of circuitry to be defined once and then used in several places. This build-in PSPICE coding can be handy when one would like to protect some fraction of a chromosome as well as to ease up the operations over the too lengthy chromosomes. We call it *X-coding* for short.

### 2.3. Parallel Evolution

EA are inspired by biological evolution, which is a massively parallel system, where every individual can perform independently. Thus, countless numbers of individuals can be evaluated simultaneously. This feature becomes increasingly important as grid and cluster computing becomes more powerful. There have essentially been three approaches to parallel EA: the master-slave approach, the diffusion approach, and the migration approach.

The master-slave approach is the simplest form of parallel EA. A master node implements all aspects of the EA itself, other than calculating the fitness; this has the advantage of introducing no new parameters. This approach is used when calculating the fitness function that is a very costly operation compared to ranking and mutation. The string representation of parameters makes mutation very simple and can thus be easily run on the master node. For instance, in [13], they used parallel evolution to design a multioutput digital circuit. Each slave node was designing a particular subcircuit with a single output selecting a particular part or a so-called multichromosome. Then, the master node joined the subchromosomes and made the evaluation of the multichromosome.

The fine-grained approach (also called the *diffusion approach*) concentrates on producing a large, interacting population over a number of nodes, often with one or few individuals per node.

The migration approach, often called the *coarse-grained approach* (or the *island-model approach*), involves running a number of largely independent EA, each on a separate processor, which occasionally exchanges information with each other. Whereas the diffusion approach has much in common with mainland population biology, this approach is inspired by island population biology, with populations connected together by migrations. In [17], the parallel island-model evolution was applied

toward a synthesis of analog circuits, namely, "hierarchical fair competition model." As the purpose of the work was a comparison of different techniques, the circuits evolved were not complex enough to probe the power of the parallel EA.

It has been noticed that ES would be particularly well suited to the migration approach. Work [18] noted that an island-ES could solve problems that a standard ES could not. In [19], there was a parallel ES for use in a protein structure determination, which behaved mostly like a master-slave ES, but with coarse-grained elements. Selection was done in parallel, with each slave node evaluating a subset of the population. However, no extensive study has been done on the implementation and the efficiency of a purely migration-based parallel ES [20].

Another technique called *coevolution* often accompanies a parallel evolution. The term *coevolution* is used for describing two or more independent evolving subsystems that are running in a parallel way with some kind of interconnection and cooperation [10]. It should be pointed out that coevolution has been considered as a promising way for producing an adaptation. There are two classes of coevolution that are conventionally recognized: *cooperative* and *competitive*. In brief, the competitive one is supposed when individuals are rewarded if they defeat the individuals with which they compete. These interactions can support "arms races" in which the individuals force each other to become increasingly competent [21]. The instances are the predator-prey kind of relations. In [22], they performed an analog circuit sizing coevolving two parallel competing evolutions. After each generation, they compared the fitness of the fittest chromosomes from each node and automatically corrected the evolution parameters.

In the cooperative coevolution, conventionally, a population represents a part of the potential solution, and when different populations altogether cooperatively produce a complete solution to the problem. By partitioning the problem in this manner, the search space that each population has to cover would significantly reduce [23]. In [8], two evolutions coevolved in parallel one of which was an evolution of fuzzy rule bases that produced suitable control parameter values for a second allowing the genetic operator to show an adequate performance.

In the following section, we give details of the methodology we propose, the basics of which is stated previously in [3].

## 3. The System Description and Experimental Setup

### 3.1. Evolutionary Strategy

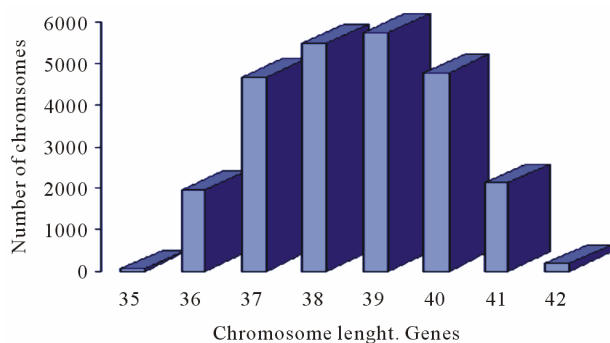
We use seven PCs of different power to run seven ESs.

The populations consist of a different number of chromosomes depending on a power of a particular PC varying from 15,000 to 35,000 individuals. The following basic parameters, operators, and features rule the every evolution.

**1) Representation (Coding).** The linear (direct) circuit representation is proposed for use, similar to that one exploited in [3,24], where each component in a circuit is coded in a gene. Whether it is a transistor or a resistor or a capacitor, the component's features (nodes, parameter, and name) are coded into four loci. The set of genes composes a chromosome. A set of chromosomes compose a population. The direct coding simplifies the terminology. We mean, for example, "circuit" when we mention "chromosome," we mean "component" when we mention "gene," and vice versa. When evolution passes to a second incremental stage, it applies *X-coding* of the best chromosome. As we detailed in the second option of the section "Incremental Evolution," some fragments of the first subcircuit become the *ad hoc* initial genes for the next stage solution. For resistors and capacitors, we set 84 and 96 values of E-12 series, *i.e.*, there are 7 and 8 decades correspondent with 12 parameters for each available evolution.

**2) Oscillating length genotype strategy (OLG).** The OLG strategy has been utilized similar to the one described in [24] where different genotype varying strategies have been compared. In OLG, the chromosomes are enabled to increase as well to decrease their lengths. However, in a long-term perspective, we enable genotypes permanently to grow up. Due to OLG, during our experiments, the difference in growth results in a difference of chromosome lengths that reaches 7 - 8 genes in one population (Figure 2).

**3) Selection rate.** The roulette-wheel selection scheme is used with a selection strength of  $\beta = \infty$  [5]. From each population, 0.2% - 2.0% of the best individuals are chosen as parents to the next generation. The single best



**Figure 2. Distribution of chromosomes after generation No46 among different lengths in a 25,000-population located in the 4-st PC.**

chromosome with all its properties always stays as a reference for individuals of future generations until a new one appeared with better features. The selection rates 0.2% - 2.0% are set based on previous experience [7] and initial tests before the experiments.

**4) Chromosome replication prevention.** During ranking procedure, when comparing two or more chromosomes with identical fitness values (with precision until 5 decimal digits) and genotype length, only one goes to a next generation. This is because we suppose that chromosomes with the exact properties must have identical genotypes and thus replicate each other.

**5) History inheritance.** The best 0.2% - 2.0% individuals are cloned to create a complete population, that is, every parent does clone from 30 to 700 descendants (for populations from 15,000 to 35,000). Each descendant inherits the mutation and fitness story of its ancestors from previous generations. These stories are helpful at mutating operation.

**6) Adaptive Mutation.** Each individual of a new population yet consisting of clones is mutated according to an individual adaptive mutation scheme (see next section).

**7) Fitness function.** The fitness function similar to one that used in [3] is scheduled that is calculated by the following static fitness function set to a sum over  $p$  fitness cases of the absolute weighted deviation between the target value  $V_{ideal}^i$  and the actual output value Voltage produced by the circuit  $V_{measured}^i$ :

$$F = \sum_{i=0}^p |V_{ideal}^i - V_{measured}^i|. \quad (1)$$

The  $p$  equals 11 time points for TIMC. The smaller the fitness value is, the closer the circuit is to the target. The fitness penalizes the output Voltage by 10 if it is not within the specified percentage range of the target Voltage value. For TIMC, where the output from the circuit is supposed to be a constant Voltage, all 11 measured points are equidistant within the range from 1ms to 10ms (which is quite a long period of time for ADC to catch up the signal for further coding).

The fitness threshold is set to 0.3%, that is, the evolution ranks the fitness of a new chromosome as the best of current one if the relative fitness difference between the best previous and pretending chromosome is more than 0.3%. This barrier enables pressure to be applied on during ranking that stimulates an application of more radical mutations (see next chapter). Furthermore, it prevents appearance of chromosomes with negligible differences that any simulation software like xSPICE is inherent to detect.

We meet the problem of generalization during the experiment. The problem of generalization appears when



the validity of the circuit functioning is limited only by a case of source signals used during evolution and is not extending to arbitrary signals. We suggested seven cases of coupled signals corresponding to distances 0.4, 2, 10, 30, 45, 65, and 95 km to tackle the problem of generalization. This number means that every chromosome of a population at each generation is tested seven times for seven different incoming signals, and the final fitness value for that particular chromosome is created by a simple sum of seven normalized fitness values:

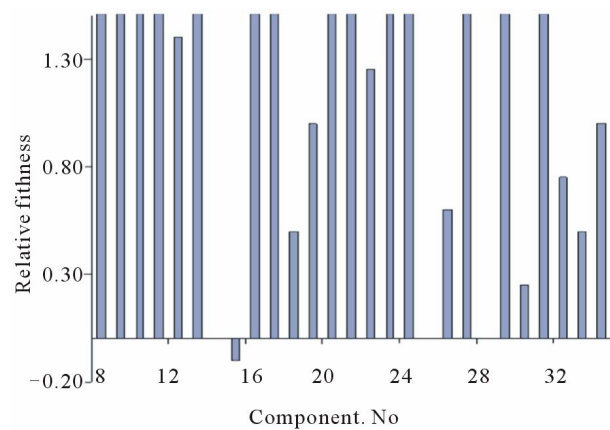
$$F = \sum_{j=1}^{j=7} w_j F^j, \text{ where } F^j \text{ is defined by expression (1)}$$

and  $w_j$  is a weight that normalizes the contribution of the each case. For the mentioned distances, the weights are: 237.5, 47.5, 9.5, 3.167, 2.11, 1.462, and 1.

**1) Ranking.** First of all, we should distinguish a chromosome with the best fitness from the best chromosome. For the first case, an individual attains the best fitness value according to the fitness function (described earlier). In the second case, an individual has gone through ranking and been ranked as number one among the population. In most of the cases, these two are represented by the same chromosome, though not always, because the chromosome length is taken as the second objective during ranking. Thus, if one looks at the graph of fitness function of the best chromosome, it will not be always slowing down (improving), but there may appear some ridges.

By ranking, we have got an opportunity to apply the pressure: along with the functionality of the circuit, we also prefer shorter chromosomes to longer ones. The fine-grained open-ended evolution of analog circuits with dynamic encoding has a side effect when a resulted circuit integrates in its structure along with functional components, the ones that had no effect to circuit's behavior [25]. **Figure 3** shows a fragment of the experiment where a circuit resulted in 34 components has four of them (Nos. 14, 15, 25, and 28) with no influence to a circuit's functionality. As an experiment displayed, the hint with pressure at ranking reduces the number of redundant components from about 20% - 40% to 5% - 10%.

It should be mentioned that including the parameter of a genotype length in the ranking operator represents the case of multiobjective evolution, where to each objective corresponds its own weight, what in its turn could be evolved [26]. However, in our case, we use the second objective as an *ad hoc* parameter, whose purpose is to handle the evolution and chromosome behavior in the right way. Thus, in the ranking procedure at each subsystem, we uniformly utilize a simple *pressure constant* that behaves adaptively, that is, depending on the progress



**Figure 3. The relative fitness sensitivity of every component in a 34-element circuit. It could be noticed that the earlier designed components (on the left) are most important for the circuit functionality than others. There are also 4 components (No14, 15, 25, 28) that do not deteriorate the circuit. Furthermore, the presence of a component No15 in the circuit structure makes the functioning worse, pruning this component will lead to a fitness improvement.**

of the evolution it varies by maximum 60%.

When comparing two chromosomes, the procedure of choosing the higher ranked is as follows:

- The shorter length at better (smaller) fitness, the higher is the ranking;
- The chromosome's rank is higher if its fitness is less at a longer length such that the following inequality is true:

$$f_{best} > f^k + f_{best} \times (l^k - l_{best}) / c \quad (1)$$

Here  $f_{best}$  and  $l_{best}$  are the fitness and the length of the best chromosome,  $f^k$  and  $l^k$  are the fitness and the length of the current one. The  $c$  is a *pressure constant*, the meaning of which is a predicted number of genes (components) in the target; the smaller the number, the higher the pressure is applied. The adaptive features are described further.

The same formula (1) is applicable in case the chromosome rank is higher if its fitness is higher (worse) at a shorter chromosome length.

The advantage of the described ranking scheme is that it enables the comparison of different length genotypes, which will be useful also during a migration operation.

**2) Pruning.** The idea under the pruning procedure is to prune the components that have no influence to the circuit's functionality. As the procedure is time consuming, it has not been applied toward every chromosome of a population and even toward the 0.2% - 2.0% of the best ones (after ranking). In more detail, after evaluation and ranking, each chromosome gets in the special subsystem, which tries to eliminate one by one each gene from a

chromosome. The dangling nodes after elimination are connected to each other in all possible variations and each time a new variant of a chromosome is tested. Finally, the gene is eliminated, and the new chromosome is adopted if any variant is evaluated with a fitness better than that was initially before the procedure. However, experiments show that applying this procedure at each, one of two, one of three, and one of four generations toward the best single as well as toward the 0.2% - 2.0% best individuals does not guarantee an improvement. After getting these results, and taking into account the crucial role that some works [25] provide for the neutral mutation, we decided to apply this procedure only twice toward a final chromosome after each stage.

**3) Initial circuit buildup.** At the beginning of each stage, the initial circuit is built up from the embryo circuit. We start evolution from the first evaluation when an embryo consists of at least 5 components. The lower number results in a large amount of identical chromosomes in a population.

**4) Termination criteria.** First of all, we should determine that the whole system terminates only if every subsystem terminates. We should distinguish the termination criteria for the first stage and for the second one, that is, a complete experiment. Two most probable events may happen that could cause the evolution to be terminated. The first one is reaching a goal in the form of exceeding the preset threshold fitness. If this happens during the initial stage, two main events are automatically triggered: the best chromosome that attained the threshold is *X-coded* and all other parallel subsystems are forced to stop searches and activate a migrant operator, that is, to receive that individual as a migrant with all its data, initiate a new population and all other standard procedures after migration. If the same happens by the end of the second stage, it means “happy end” for the whole experiment. For both cases, the threshold is set as reaching the fitness less than 1.0, which is equivalent to an average deviation from the ideal reply function per point 0.031 V for the first subcircuit and 0.044 V for the whole circuit.

The second reason of the most probable termination is when a subsystem is not able to update the best chromosome for over  $L$  consecutive generations. During  $L$  generations, every evolution, in case of its fitness stuck, receives some number of migrants, and if every migrant is worse or equal to the subsystem’s best individual (what means all other subsystems did not improve too), the experiment stops. However, if some subsystem finds a better solution, it is able to revive others. In last case, the subsystems that stopped the search can join the process under the same conditions, as if they just have activated the migrant operator.

If the first term terminates all subsystems simultaneously, the second one acts independently for per subsystem.

## 3.2. Adaptation

### 3.2.1 Differentiated Mutation

Every chromosome at each generation may go through mutations of different values (except the several initial generations). The essence of our individual-level adaptive mutation approach is based on the following rules.

**1) Rate.** The different MRs are associated with different *types* of mutations. The reference rate of mutation is set by us to 4%. That means this rate is a minimum level that could take place in a subsystem.

**2) Types and ways of mutations.** The available *types* of mutation to be applied to chromosomes are in **Table 1**.

Combined each other, the mutation types suggest the variety of mutation *ways*. In **Table 2** there are 5 examples of chromosomes against the list of different *ways* of mutations that may be applied to a corresponding individual. For instance, the 100-gene chromosome can be mutated in 13 different ways, each of which is a combination of 7 different mutation types. The choice of the particular *way* is set as a random procedure, if it is not specified elsewhere.

Thus, it is suggested to *differentiate* mutation in analog circuit evolution not just by rates, but also by *types* and *ways*. In the following are described the aspects of the

**Table 1. The types of mutation.**

No	Mutation Type
1	<i>Node number</i> , <i>Parameter</i> or <i>Component name</i> mutation phenotypically means reducing, adding or replacement of only 1 locus. There are no limitations on where to use it. In most of the cases it is applied in combinations with <i>Component mutation</i> and <i>Substructure X mutation</i> .
2	<i>Component mutation</i> phenotypically means reducing, adding or replacement of a component by another component. It concerns 4 loci at once.
3	<i>Substructure 1 mutation</i> concerns 8 loci. It adds/reduces 2 genes at once to/from a chromosome. These two genes compose the first substructure.
4	<i>Substructure 2 mutation</i> concerns 12 loci. It adds/reduces 3 genes at once to/from a chromosome. These three genes compose the second substructure.
5	<i>Substructure 3 mutation</i> concerns 16 loci. It adds/reduces 4 genes at once to/from a chromosome. These four genes compose the third substructure.
6	<i>Substructure 4 mutation</i> concerns 20 loci. It adds/reduces 5 genes at once to/from a chromosome. These five genes compose the fourth substructure.
7*	<i>Substructure 5 mutation</i> concerns 24 loci. It adds/reduces 6 genes at once to/from a chromosome. These six genes compose the fifth substructure.

\*This type of mutation was not applied in the following experiment. However, the substructure was stored and prepared to be applied.

**Table 2. Examples of the ways of mutation for 5 different chromosomes.**

Chromosome size, genes	No of loci mutated at 4%	The mutation way list applied to different chromosomes (by combining mutation types No1-No7)
10	2	1) No1×2;
20	4	1) No1×4; 2) No2;
50	8	1) No1×8; 2) No1×4+No2; 3) No3;
80	13	1) No1×13; 2) No1×9+No2; 3) No1×5+No2×2; 4) No1×5+No3; 5) No1×1+No2×3; 6) No1×1+No2+No3;
100	16	1) No1×16; 2) No1×12+No2; 3) No1×8+No2×2; 4) No1×8+No3; 5) No1×4+No2×3; 6) No1×4+No2+No3; 7) No2×4; 8) No2×2 +No3; 9) No3×2; 10) No1×4+No4; 11) No1×4+No4; 12) No2+No4; 13) No5;

mutation strategy.

**3) Substructures' sources.** The best chromosomes of sizes 5 and 6 genes are stored for use as substructures and to be applied as mutations of type Nos. 6 and No. 7 correspondingly. The substructures of size 2, 3, and 4 genes are obtained by decomposing the best chromosomes of size 5 and 6 genes. They are memorized to be applied as mutation Nos. 3, 4, and 5 correspondingly.

**4) Substructure database.** We enable the system to memorize per one substructure of size 4, 5, and 6 genes and, per two substructures of size 2 and 3 genes for evolution to have a choice. For each stage, the substructure database is built independently. The substructures to the second stage are taken aside from the *X-coded* part of a chromosome.

**5) Diversification of a mutation history.** Each individual carries its own history about mutations its ancestors had gone through. If the chromosome is ranked in 10% worst of the best 0.2% - 2%, the random choice of mutation is replaced by the following rule: the most seldom mutation type from the individual's history should be applied in the first place at the current generation.

**6) Mutation.** If the chromosome does not improve its fitness for the last two generations, the following rule is activated: the lowest mutation way number temporarily leaves out the potential mutation way list (Table 2) increasing the probability of the others to be chosen. That brings more radical changes to a genotype by joining bigger substructures. The MR is yet staying in the frame of the initial 4%. The mutation may continue until there is only one mutation way left. This kind of pressure disappears once a chromosome improved its fitness.

**7) Radical mutations.** If the chromosome has not improved within the last 4 generations, the next should be

applied is the mutation of an upper type number than it is allowed within the standard 4% (3rd column of Table 3). Say, 10-gene chromosome at 4% MR is allowed only a single No. 1 (*Node number*, *Parameter* or *Component name*) mutation type. However, now due to the fitness stuck, it should go through No. 2 (*Component mutation*), which is equivalent to a 10% rate mutation. Furthermore, if the chromosome has not improved within the last 6 generations, the next to be applied is the mutation of an even higher type number (5th column of Table 3). The same instantiated 10-gene chromosome now goes through mutation No. 3 (*Substructure\_1\_mutation*), and the rate will be 50%.

The *radical* MR-2 depends on the length of the chromosome to which it should be applied and may vary from 80% for a 5-gene individual to lower than 6% for ones that contain more than 100 genes. The *radical mutation* is a very important part of VNFE; it provides the essential modifications to uncommonly homogeneous individuals peculiar to VNFE, especially during stuck periods.

### 3.2.2. Other Adaptive Parameters

Besides the mutation, two more parameters are enabled for each evolution to adapt. We set different initial populations and SRs.

The strategy for population size adaptation is to keep equal times per generation of different populations. If initially the weaker processors keep pace with others, later, starting from the average chromosome length of 10 - 15 genes, they need their cycle periods to be reduced. To enable this synchronization, we set the evolving period of the first population (at 1-st PC) as a reference for others. Thus, during the migrant operator active, each subsystem, except the 1st one, adjusts its population size  $P$  to a new one  $P_1$  to keep pace with the population on the first processor according to the following simple formula:

**Table 3. Examples of radical mutations for 5 different chromosomes.**

Chromosome size, genes	Standard mutation type within 4%, No(gene size)	Radical mutation 1		Radical mutation 2	
		Mutation type, No(gene size)	%	Mutation type, No(gene size)	%
10	1(0)	2(1)	10	3(2)	50
20	2(1)	3(2)	10	4(3)	15
50	3(2)	4(3)	6	5(4)	8
80	4(3)	5(4)	5	6(5)	6.25
100	5(4)	6(5)	5	7(6)	6



$P_1 = t_1 \times P/t$ , where  $t_1$  and  $t$  are generation times at the 1st subsystem and the synchronizing subsystem.

Seven SRs from 0.2% to 2% (**Table 4**) are enabled to migrate from a “winner” evolution to a “loser” evolution along with a “winner’s” genotype and its history. This is going to enable some statistics to be accumulated at the end of an experiment on: which SR is become the most frequent “winner” and finally has dominated others?

### 3.3. Evolutionary Strategy

#### 3.3.1. Parallel Subsystems

Like in a typical island-model algorithm, we have different populations, each of which acts as an independent ES, with each one separately initializing, ranking, selection, cloning, and mutation performing only within populations. Each population runs on a separate processor; thus, all these operations can be performed locally, keeping down communication time. Totally, seven PCs have been running in parallel connected to a hub.

#### 3.3.2. Migrant Strategy

The populations are connected together by a migration operation, which is performed by communication between processors. There is no centralized (“master-slave” mode) schedule set for communication frequency and magnitude of migration. On the contrary, each evolution “decides itself” when to start migration and what it needs for migration. There is only one condition set when a communication among the parallel subsystems can take place. Each evolution is allowed to run without communication as long as the best chromosome improves (for the term the “best chromosome improvement,” see “Ranking” in chapter “Evolutionary Strategy”). As soon as any population does not improve for at least  $N$  generations, the built-in migration operator activates and makes

the subsystem search for help from other subsystems. First of all, the stagnated subsystem collects the data files (**Figure 4**) from all the parallel nodes, analyzes them, and decides what the most successful evolution among all until now is. It applies the ranking rules, including formulas (1) and (2), to rank out the best chromosome among the latest of each evolution. So, the subsystem gets a ranking list consisting of 6 members, a top member of which becomes a “winner” chromosome from a “winner” subsystem. Then, the subsystem gets a clone of the “winner” with all his history, checks the substructures (see paragraph below), updates its SR and population size. It clones a single individual to the total population and continues with its further isolated evolution procedures. Here, our approach differs from the others. Usually, the migrant strategy implies the highest-ranking individual to replace the lowest-ranking individual without dumping out the rest of the genotype material. An experiment revealed the last approach did not bring the valuable results at least within the time twice longer than taken by the proposed technique.

The advanced feature of the migrant strategy described is an ability of the whole system to adapt the activity of the migrant operator varying it from 0-power, when no one subsystem met any problem during evolution, till the full 7-power regime, when every processor gets a  $N$ -generation stuck period and 7 migrations happened.

In the previous paragraph, we described the general migrant operation that is liable during all the evolution. However, there is a migration of the substructures. The

**Table 4. Initial conditions at 7 parallel PCs.**

1	PC description	Initial pop. size*, individ.	Initial selection rate, %
1	Intel Core2Quad, 2.4 Ghz, 4 GB	35,000	0.2
2	Intel Core2Quad, 2.4 Ghz, 4 GB	35,000	0.5
3	Intel Core2Duo, 2.2 Ghz, 2 GB	25,000	0.8
4	Intel Core2Duo, 2.2 Ghz, 2 GB	25,000	1.1
5	Pentium4, 2.8 Ghz, 2.0 GB	18,000	1.4
6	Pentium4, 2.5 Ghz, 1.0 GB	16,000	1.7
7	Pentium4, 2.8 Ghz, 0.5 GB	15,000	2.0

\* The population sizes are chosen in accordance with operational powers of each PC, so that times to be taken by initial generations are approximately equal.

GenNo	BstNo	BstFit	Length	Crip	Substr
...	425	1.72e+001	5	27	0
5	21999	1.72e+001	5	59	0
6	658	1.64e+001	7	46	0
...					
17	21815	1.28e+001	13	175	1,1
18	19857	1.18e+001	13	136	1,2
19	21859	1.14e+001	12	128	1,2,1
...					
33	17925	8.98e+000	16	111	1,1,2
34	21995	8.98e+000	15	139	1,2
35	21999	8.98e+000	16	153	1,1,2
...					
45	873	7.61e+000	25	143	2,2,3,2,3
46	20476	6.63e+000	23	154	1,1,1,2,2,3
47	21999	6.63e+000	23	182	1,1,1,2,2,3
48	9059	5.41e+000	30	219	1,1,1,2,2,3,4
...					
54	1268	4.29e+000	35		2,2,1,2,2,3,3,3,1
55	1488	4.11e+000	37	265	1,1,1,2,2,3,2,3,2,2
56	871	3.96e+000	37	273	1,2,1,1,2,2,2,2,4,2
57	21945	3.69e+000	35	299	1,1,1,2,2,3,3,3,2
...					

**Figure 4. The fragment of a data-file. By columns: 1-“Generation number,” 2- “The best chromosome number,” 3- “The best chromosome’s fitness value,” 4- “Gene (component) number,” 5- “The number of cripple chromosomes,” 6- “The substructure reuse story,” where the numbers are the substructure numbers: “1” is Substructure\_1\_mutation, etc.**

substructures are limited to 6 genes in our approach. When a stagnated subsystem analyzes the data files (see paragraph earlier), it makes two independent rankings for individuals of length 5 and 6 genes. The best are decomposed and stored in a substructure database. The subsystems that donate the best chromosomes may differ. The procedure of substructure checks is performed every time before the start of a new generation until all the subsystems past the 11-gene (or higher) chromosome in their data file cause, as the experiment has shown, the oscillation to never come back from 11 to 6-gene length at this stage of evolution.

### 3.3.3. Coevolution

In general, our approach represents a competitive coevolution, where with more time some evolution produces “winners” as more dominating than others. Notice that while migrating to a new subsystem the best chromosome destroys all the previous genotype material of the subsystem as unwarranted, enforcing to adopt its own. Despite different initial conditions set for separate evolutions, due to migration, it may happen that the descendants of one ancestor compete with each other. Furthermore, the approach theoretically enables that all 7 subsystems during our experiment (that is totally about 169,000 individuals) evolve the same one chromosome at the same time. Together with the extremely narrow SR (0.2% - 2.0%), it seems quite contradictory to a conventional opinion in EA theory hailing the diversity of genotypes. However, the experiment proved that the technique is able successfully to solve this problem; moreover, it appears as the only way to find a solution. The feature that plays the main role is the DM technique, where a single individual requires quite numerous clones to enable the variety of mutation ways to take their chances.

Thus, the competition aspect consists of a constant threat of destruction of the whole population when a system “defeats.” Even the subsystem that imported in past the genotypes from a “winner” subsystem is regarded as a competitor to the “winner,” because due to the radicalism of the DM strategy, these two become different much faster than during a conventional evolution with a standard MR.

The competition aspect also plays the important role when the termination criteria appear during the first stage: the most competitive subsystem at that moment becomes a “winner” and only “he” who is allowed to save and transfer its population into the next stage. All others must obey a migrant operation.

On the other hand, the competitive coevolution we present has some features of cooperation. The apparent one is when a “winner” shares the successful SR and

genotypes with a “loser.” A “winner” does not allow a stagnated subsystem to stop. It shares with last one the best what it has, in spite of latter, the “loser,” becoming its competitor. Another aspect that makes all subsystems cooperative is a substructure database that may consist of genotypes (totally 7 substructures from two 5- and 6-gene individuals) from different evolutions and is accessible by everyone. Moreover, all the subsystems focus in their work toward the same target and have the same fitness function, which is unconventional for competitive coevolution. If one could watch the coevolving subsystems at work, it would find in common with *rugby game*, where everyone in a team at an attack tries to get the same target, and if any player has a trouble with going forward, he passes a ball to one who has the most favorable position. In this kind of sense, the ball plays the role of a migrant operator activated by a “problematic player.” Further, we call our parallel coevolution approach as *winner-dominates-others-winner-cooperates-loser* or *winner-dominates-winner-cooperates* (WDWC) strategy.

### 3.3.4. Incremental Procedure

Upon the end of the first subcircuit, the second one starts to be evolved. We enable evolution to involve two levels of components from the junction point (see the second option in chapter “Incremental Evolution”), which is totally three components that belong to a first subcircuit take part in further evolution toward the second one. The rest of the subchromosome is frozen up with the help of *X-coding*.

## 3.4. The Problem Description: TIMC for the Laser Rangefinder

A laser rangefinder is a device that uses a laser beam to determine the distance to an object. The most common laser rangefinder operates on the time-of-flight principle by sending a laser pulse in a narrow beam toward the object and measuring the time taken by the pulse to be reflected off the target and returned to the sender. The distance is given by:

$$S = \frac{ct}{2}, \quad (2)$$

where  $c$  is the speed of light and  $t$  is the amount of time for the round trip between the device and the target. The typical laser rangefinder has two main parts: optical and electrical. The optical block sends the laser beam and receives the reflection, providing the electrical block with two Voltage pulses, upon which the electrical block calculates the distance.

As a prototype, we took the artillery quantum rangefinder “DAQ-2” [27] with the following data:

- working at range is 0.2 - 100 km,
- measurement accuracy is 6 - 30 m,
- the width of both pulses is 50ns; the fall/rise time of the pulse is up to 5 ns; the first pulse has 9-V amplitude, the reflected one has 6 V.
- power supply required is 29 V.

The core part of the electrical block of the device is a time interval meter sub-block (TIMSB). The working principle of a conventional TIMSB consists of three functional stages.

1) At the first stage, two electrical pulses received from an optical block should be reshaped into the Voltage gate pulse, where the first incoming pulse is caused by the laser beam sent toward a target, and the second one is caused by the beam reflected off the target. The gate pulse is a pulse of some constant potential that should have the same time width as the interval between two narrow pulses caused by a laser beam.

2) At the second stage, the gate pulse (*i.e.*, the time interval of the gate pulse) is filled up by the clock signals from crystal oscillator. According to (1), the gate pulse width varies from about 0.667  $\mu$ s for the minimum measured distance (0.1 km) to 0.667 ms for the maximum measured distance (100 km).

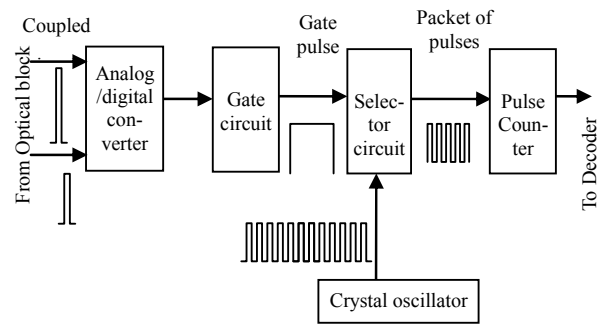
3) And finally, to count the number of pulses contained in the packet. The result of counting in binary code should be sent to a decoder for further conversion into a decimal code.

**Figure 5** is a general schematic of the TIMSB of the up-to-date laser rangefinder. Based on a description available for public, we set as a goal to synthesize the analog circuit that is able in its functioning to unite stages 1), 2), and 3) described earlier, and replace 5 digital units from **Figure 5**. A new circuit receives two pulses from an optical block and produces the particular constant Voltage. The linear correlation between time gap, and the Voltage produced is set, ranging between the maximum 5V (against the maximum 100 km) and 5mV (for a distance 0.1 km). The proposed TIMSB based on an analog circuit is shown in **Figure 6(a)**. The decomposition method is shown in **Figure 6(b)**, where we suggest two subcircuits to be evolved during two incremental stages.

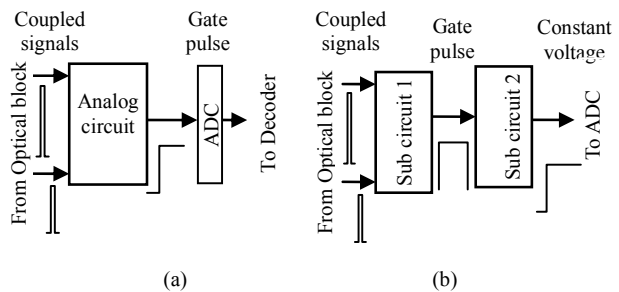
## 4. Experimental Results

### 4.1. The Circuit

For TIMC, initially we have tried to evolve the whole circuit at once without exploiting the task decomposition, but the evolution has failed to converge toward the acceptable solution. Then, the problem was decomposed into two subtasks (**Figure 6(a)**). The initial stage is the evolution of two-input-one-output gate pulse producing



**Figure 5.** The TIMSB of an up-to-date laser rangefinder made of digital logic. The shapes of the signals are shown under each pin. From left to right: there are two pulses coming in from an optical block, 9 V and of 6 V, separated by a time taken for the beam to be reflected and returned; they are converted to a digital form by ADC. Then, they are transformed to a gate pulse by gate circuit; a selector circuit fills up the gate with clock pulses generated by a crystal oscillator; a pulse counter circuit gets the packet of pulses and counts the clock pulses; a decoder converts that count to a decimal form.



**Figure 6.** (a) The proposed TIMSB with the targeted analog circuit. The shapes of the signals are shown under each pin. From left to right: two pulses are converted into constant voltage; the voltage level is in linear proportion to the time interval between two pulses; the ADC converts the voltage into the binary code for further decoding. Due to the resolution of the circuit is preferred to be at least 50  $\mu$ V (corresponds to 1 meter), that is totally  $1e+5$  discrete values, the 18-bit ADC with 262,144 quantization levels will meet the requirement. (b) The proposed decomposition of the targeted analog circuit. The first subcircuit's task is to form the gate pulse based on a coupled signal. The aim of the second is to produce a constant voltage.

subcircuit and the next one is the evolution of an one-input-one-output subcircuit, which is in series with the first one.

The experiment has been running nonstop throughout all stages. To design the whole circuit it took about a week, where 17% of time was spent for the first subcircuit and the rest 83% for the second one. The first subcircuit with 2 inputs and 1 output, with a primary task to provide a gate pulse, consisted of 34 components. Before the next evolution start, the fitness of the best first sub-

circuit is 0.906. The pruning procedure has eliminated 3 components that have no influence to a circuit's behavior (until 31). The second subcircuit, with a task to accept a gate pulse and produce the required constant Voltage, consists of 61 elements. Five components are pruned. The final whole design consists of 95 components before pruning and 87 components after pruning, among which are 29 resistors, 26 p-n-p transistors, 17 n-p-n transistors, and 15 capacitors. The second termination criteria stop an experiment at the total fitness 1.137 at generation 105.

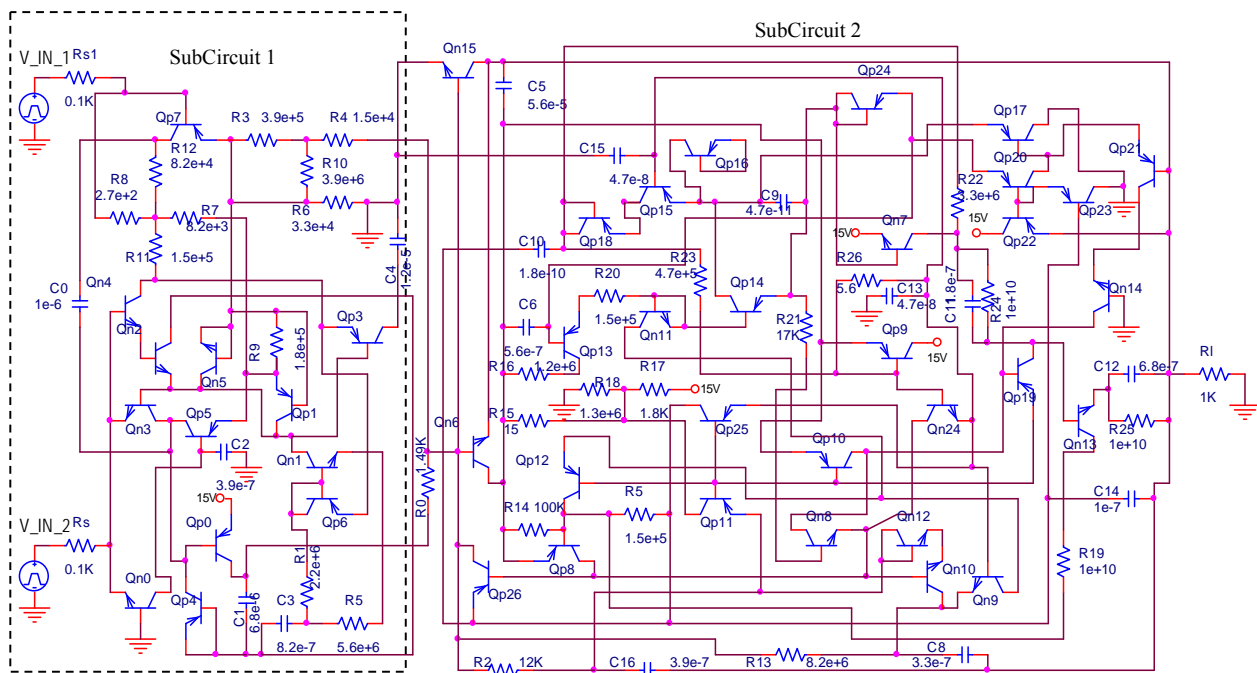
The resulted device is presented in **Figure 7**, and its functionality is shown on six arbitrary instances in **Figure 8(a)**. To one of the features of the circuit can be added the lower Voltage supply in comparison with the "DAQ-2," 15 V against 29 V.

The PSpice's performance analysis enables to measure the generalization ability of the circuit by tracing the dependence of circuit replies on a swept parameter. If we take as a swept parameter the absolute average deviation from the ideal circuit response and apply it to a family of waveforms, we produce a trace that is a function of the variable that changed within the family. As seen from **Figure 8(b)**, which represents the absolute average deviation along 1000 equidistant circuit replies, the measurement accuracy of TIMC could be approximately split into three groups: 2.4 meters for distance range  $0.1 \div 3$  m, 16 meters for  $3 \div 14$  km, and 54 meters for  $16 \div 100$  km.

In comparison with conventional digital TIMC, where the measurement accuracy varies within the range  $6 \div 30$  m, it should be mentioned that for shorter distances the analog TIMC does much more accurate measurements, and in general looks relatively competitive. Furthermore, the resulting device is able to work out measurements within distances from 0.1 to 0.2 km, for which the "DAQ-2" cannot. Moreover, while solving the *generalization problem*<sup>2</sup>, we noticed the tendency during which the accuracy of the measurements directly depends on a number of input cases during evolution. Thus, it is logical to conclude that reaching the same accuracy (30 m) for longer distances and even exceeding it is just a matter of computing time.

## 4.2. The Evolution

The general view of the experiment, consisting of seven evolving subsystems with the details of each migration, is shown in **Figure 9**. Totally, 33 times the migrant operator has taken place, where only once it has happened during the first stage. The first subtask is significantly easier than the second one: only one subsystem has turned to a migrant operation while six others have finished the task on their own. During the second stage, no one eVolution has come directly to a global solution. Due to the complexity of the problem, all of them have got into the local optimum and have required "assis-



**Figure 7.** The evolved TIMC is consisted of 2 subcircuits: the first subcircuit (dashed) passes the gate pulse to the second one.

<sup>2</sup>The problem of generalization appears when the validity of the circuit function is limited only to a case of source signals used during evolution and does not extend to arbitrary signals.

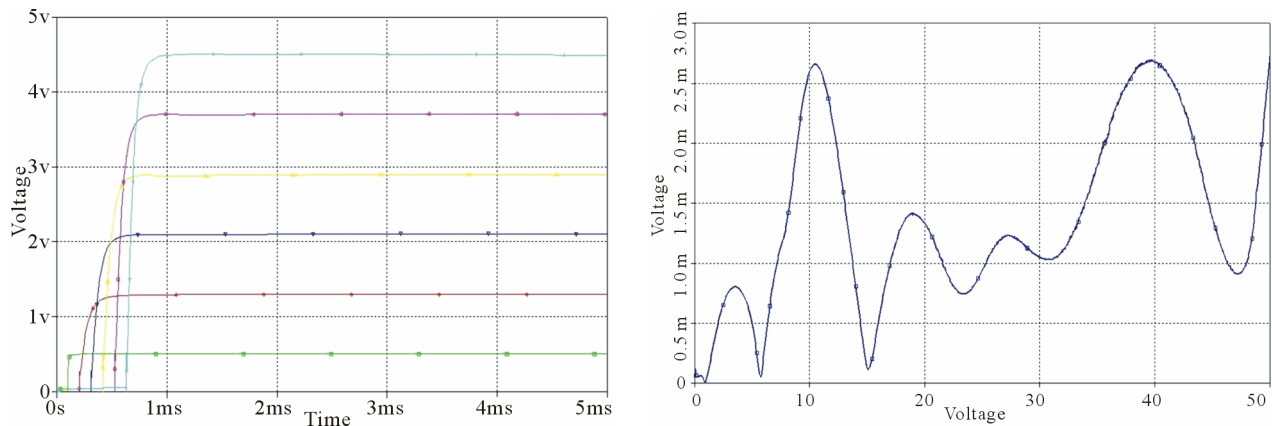


Figure 8. (a) The voltage replies of the evolved TIMC to six arbitrary incoming signals corresponding to 10, 26, 42, 58, 74 and 90 km. (b) The function of the integrated absolute average deviation from the ideal circuit response along 1000 equidistant circuit replies.

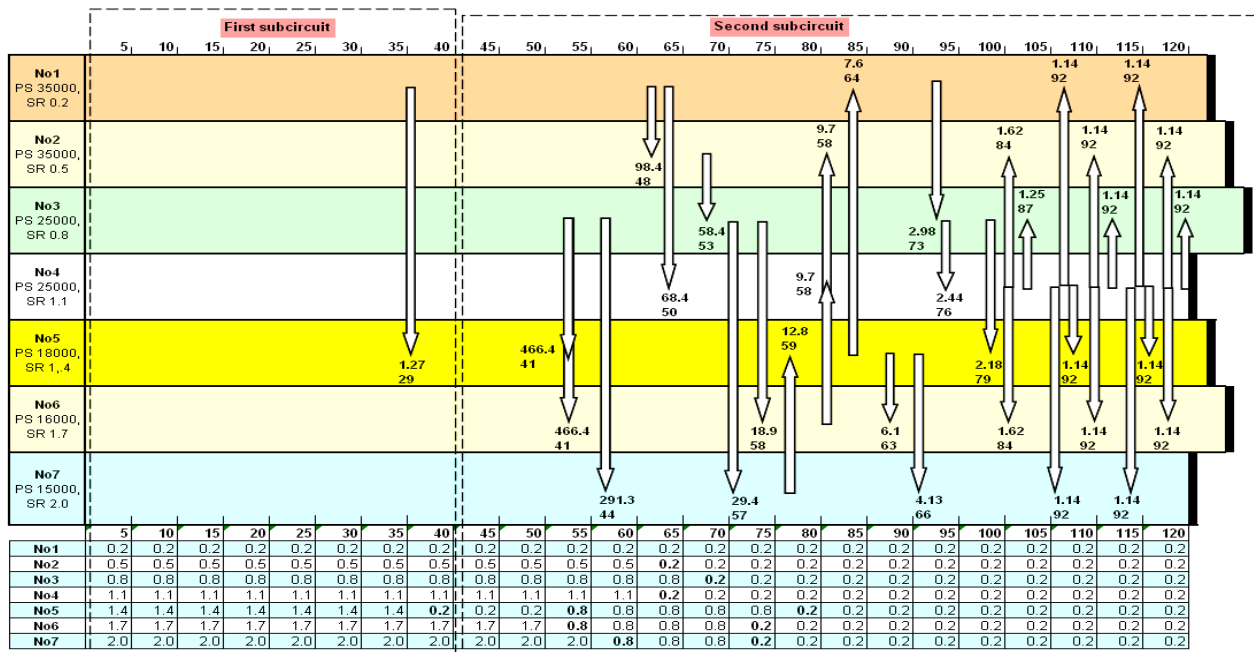


Figure 9. The migrant schedule. The diagram shows when and how the migrant takes place along a horizontal axis representing generation numbers. Seven numbered subsystems with initial population size (PS) and SR evolve in parallel from left to right. The arrow indicates which subsystem is a receiver, from where and at which generation. Totally, 33 migrants are shown where only one is happened during the first substage. Each migrant is described by the fitness value of the migrant individual and its length in genes. The table below carries additional information on the process of how selection rates migrate along the same axis. The rates just imported are bolded. Since generation 77 the selection rate 0.2 has dominated over the last fifth evolution.

tance” from others many times (Table 5).

From Table 5 and Figure 9, it can be noticed that the most influential aspect during the experiment is a subsystem No1, whose SR (0.2%) has dominated others and whose genotype have spread to every subsystem. Surely, it is not the case where we could declare the principal dominance of lower SRs over the higher ones; more statistics is needed for that. However, two facts show a

general tendency on SR. First is statistics on how many times systems with a particular SR become a winner. Systems with the minimum SR 0.2 have become a winner 7 times, systems with 0.8 3 times and no winners with any other SR. Second, it takes 77 generations for SR 0.2 to occupy every system.

What is interesting is that during the process each subsystem, powerful ones as well as weak, has been a

**Table 5. Initial vs. final parameters and migrant import/export numbers against the properties of 7 parallel subsystems.**

PC No		Initial population size, individ.		Initial SR, %		Migrants, No	
		Initially	Finally	Initially	Finally	Import	Export
1	PC1	35000	35000	0.2	0.2	3	3
2	PC2	35000	36800	0.5	0.2	5	1
3	PC3	25000	19400	0.8	0.2	5	7
4	PC4	25000	19900	1.1	0.2	3	15
5	PC5	18000	11800	1.4	0.2	5	3
6	PC6	16000	6600	1.7	0.2	6	2
7	PC7	15000	7200	2.0	0.2	5	1

“winner” for several times. This fact assures that the WDWC strategy has uniformly distributed the probability of finding successful solutions among all processors.

The case of tough competition is expressed at the moment in between the stages, when only one chromosome is enabled to be bred by every subsystem except a “winner.” This extreme migrating act is implemented by us, instead of release of a system to transfer gradually to the next stage, due to the proven fact of the superiority of the first approach over the second.

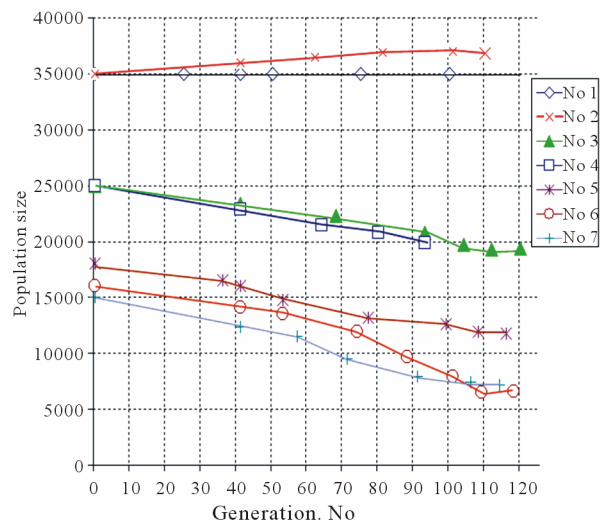
From generation 91 to 105, the 4th subsystem dominates the rest, spreading out its best genotype to every PC. But from generation 106 none showed improvement, and after 15 generations the experiment has stopped.

**Figure 10** shows the result of the population size adaptation. It should be noticed from there a tendency according to which the difference in computing properties of the subsystems brings as more diverging population sizes as longer the average length of the chromosome. On average, the productivity of PC Nos. 6 and No. 7 has fallen down more than twice of an initial population size.

The best chromosomes of length 5 and 6 genes from subsystem Nos. 1 and No. 4 and their fragments were stored in and used as substructures.

**Figure 11** demonstrates seven fitness cases (the fitness of the best individuals) during the experiment and in between the stages. The high complexity of the second subcircuit makes the fitness value to scale for longer than 3 decades. The “waves” of migrations are distinctly visible when straight vertical lines connect one function with another.

Another very important notion is about an *aggregated selection rate* (ASR), which is a correlation between the total number of selected individuals of the whole system at each generation and the current total number of



**Figure 10. Adaptation schedule of population size. Seven curves correspond to seven populations along the horizontal axis representing the generation number. The curve No1 plays as an etalon for which all others must synchronize their generation cycles by varying the number of individuals during each migrant operation.**

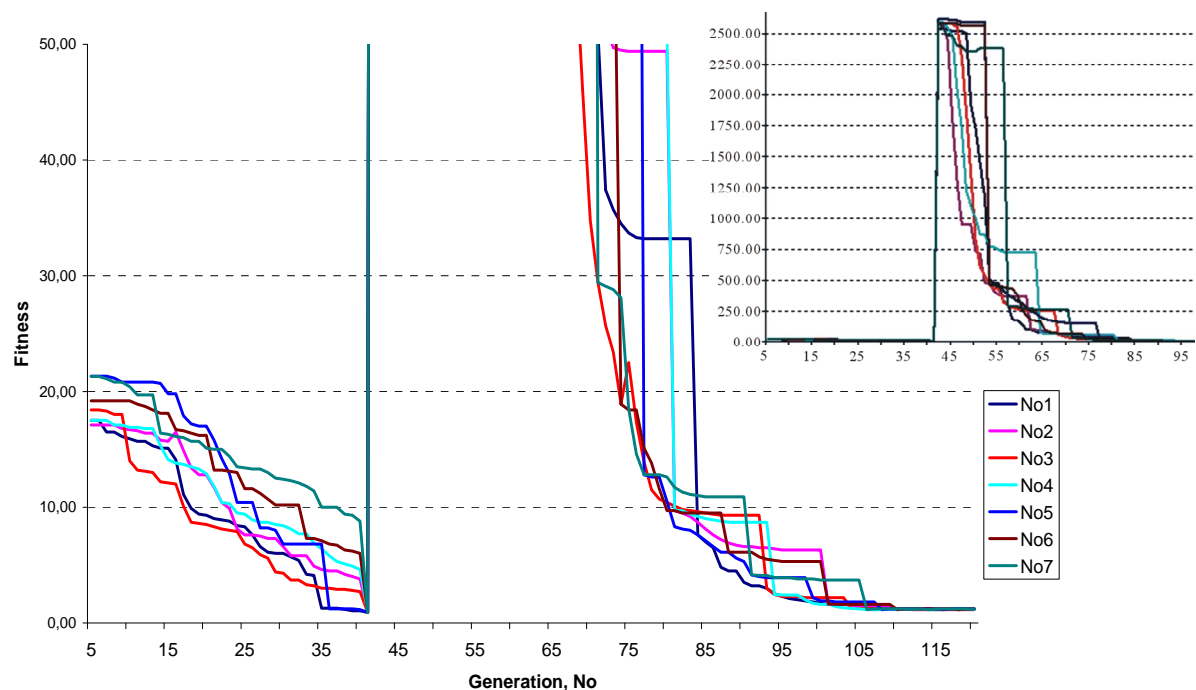
individuals inside all populations. It is introduced by us to differentiate local SRs and the global one of the system. At starting conditions when SRs are fixed from 0.2% to 2.0% and the total number of individuals is 169,000, ASR equals to 0.914%. However, during the experiment, there are two events that lead to change in ASR:

1) When some subsystem activates a migrant operator. According to our methodology, only one chromosome is migrating to the stuck subsystem; in other words, one is selected, and ASR and SR of the subsystem fall down. Furthermore, together with an individual, a new SR comes from a “winner”. Because the subsystem No. 1 has been dominating others, ASR finally has converged to a rate 0.2. Migrations bring spikes in ASR happening more frequently and longer to the end of an experiment (**Figure 12**) leading ASR to 0.124%.

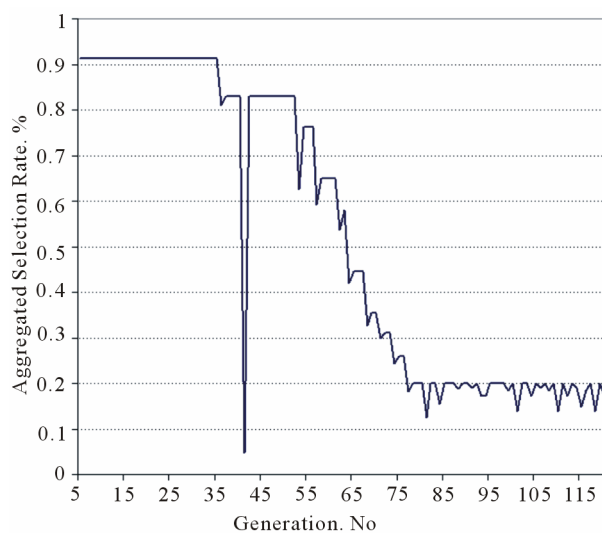
2) When evolution is incrementing to the next stage. At this moment, only one chromosome among all populations (except the “winner” population) is selected to next generations, and ASR falls down drastically. In **Figure 13**, this moment happens at generation 41 where ASR reaches 0.048%.

Watching the adaptive behavior of ASR is useful because one can notice the rule according to which the evolution moves forward: as the harder it is for evolution to continue as lower the average ASR becomes. Otherwise, during the successful periods, the system tries to expand its gene pool, but at “crisis” periods it sharply reduces a gene pool focusing on breeding the populations



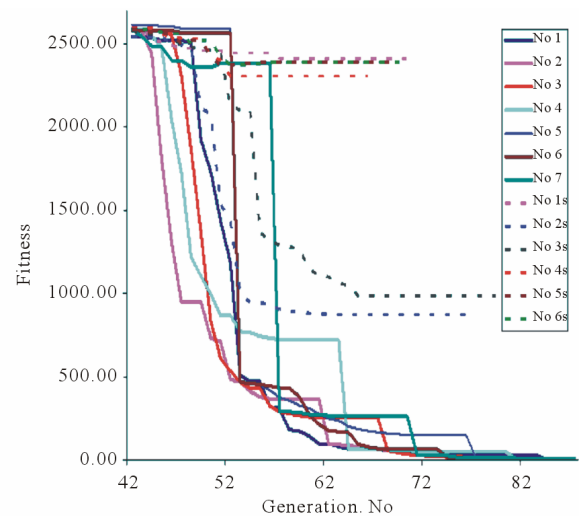


**Figure 11.** Seven fitness cases (the fitness of the best individuals) during both incremental substages is in the upper right corner. It shows how different are they scaled to each other due to unlike levels of complexity. The central picture is focused on a fragment between the stages. At generation 41 there is a transition to a second substage. The frequent migrating “waves” are distinctly visible at the end of the second substage (lower right corner).



**Figure 12.** The ASR along the generation number. The ASR is converging to 0.2 selection-rate initially set at PC No. 1. The spikes are representing the moments at migrations. The ASR narrowings are visible where the largest is reaching 0.048% at the first generation of the second substage.

from fewer individuals. We find this methodology similar to natural evolution, when such kinds of crisis like natural disasters, pandemics, and wars force a few survivals to regenerate the rest of the kind.



**Figure 13.** By dotted lines there are six fitness cases No. 1s – No. 6s along the generation No corresponding to SR from 0.5% to 5.0% and 7 cases No. 1 – No. 7 of VNFs with selection rates from 0.2% to 2.0%. All six “conventional” evolutions have stuck far away from the targeted fitness.

## 5. Comparison

The methodology that has been presented so far was discovered after a number of failed attempts to evolve TIMC by means of standard nonparallel ES-based evolu-

tion with more conventional SRs. In this chapter, we present some of these failed attempts to enable a comparison.

When applied toward less complicated circuits [1,3,7], our previous approach was able to solve successfully all the tasks set including low-pass filters, computational circuits, and 4-output Voltage distributor with superior functional and physical features. If earlier experiments [1] utilized 10%, the latter ones [3,7] applied 1% - 10% SRs.

In this work, being inspired by previous 1%-rate approach success, we have tried to run three independent evolutions with an SR 1%. Then, we have tried to apply SRs 5.0%, 3%, and 0.5% running another three evolutions. Here, we present six independent nonparallel evolutions with SRs 5%, 3%, 1%, and 0.5% applied toward TIMC (**Table 6**). All of them have been applied at the second incremental stage of the experiment, as the first one is too easy to play a role of a challenging task. All the evolutionary parameters and operators have been the same as described in this paper except SRs and the absence of communication among subsystems.

We have used the *X-coded* chromosome from the 1st stage that differs from that one presented earlier, because it is obtained during the nonparallel evolution, but due to ease of the first subtask it is very similar by length 32 against 31 and by a fitness value 0.96 against 0.91.

In **Figure 13**, there are 6 fitness cases of evolutions that are superposed with 7 cases of VNFES. It should be noticed that there is a very high fitness barrier at about 40 - 43 chromosome length for both evolutions. That represents the specific feature that is inherent to the functioning of TIMC. Once an evolution gets over this barrier, the fitness improves from about 2200 till 1000. If in coevolution case 3 subsystems that stuck at this barrier continued eVolving after the migrant operation, 3 subsystems of the standard evolution left the experiment

after 15 futile generations. The second barrier appears at length 48 - 50 and fitness about 100. However, the conventional evolution has met it much earlier at a fitness higher than 500. We can suppose that this happens due to relatively high SRs. That is, as the DM is applied to both cases, it requires the same large amount of clones per individual for manifesting itself, what evolutions with higher SRs provide worse. So, the rest of conventional evolutions have been stopped in there.

## 6. Conclusions

In this paper, we have described a novel methodology of coevolution of parallel island model subsystems with adaptive parameters. The methodology is named *very narrow focused evolution* (VNFE) as it possesses very small SRs. It has been described why authors are obliged to apply narrow SRs and have enabled the subsystems to destroy the genotypes of competitors during the coevolving strategy called *winner-dominates-winner-cooperates* (WDWC). The novel *differentiated mutation* (DM) strategy that we “blame” is built up based on a paradigm where substructures are regarded as mutation types. Together with such the operating tools as *diversification of mutation history*, *mutation*, and *radical mutation*, the DM strategy represents a quite aggressive operator, whose stimulating “credo” is increasing an MR each time a chromosome does not bring an improvement. These two relatively extreme approaches, DM and WDWC, work in a balance with each other at “crisis” and “wealth” periods: whenever the mutation adapts to operate more aggressively without success, the more actively WDWC operates bringing the aggregated selection rate (ASR) to lower values. And vice versa, if evolutions altogether gradually move forward, both operators let the system work alone in the frames of standard parameters. Thus, the proposed system is able to apply two adaptive regimes, one is a “standard evolution” in the frames of conventional mutation (4%) and selection (0.2% - 2.0%) parameters. And another, VNFE, that is triggered whenever the first regime meets any problem, and which involves DM (up to 80%) and WDWC (down till 0.124%).

Indeed, this technique has expressed itself in full during an experiment. At the first stage, WDWC has brought little help in solving an initial subtask with only one migration. That is, it is the “standard evolution” that has been “running the show” and is able to tackle the problem by itself. On the contrary, as the generation 53 of the second stage during the rest 67 generations there are 32 migrations, which means it is VNFE that mostly pulled the process through.

It is obvious that if we made the migrant operator activation term as adaptive to the complexity level of subtasks, the first stage would be involved in the coevolution

**Table 6. The population size and initial conditions of the 3 non-parallel PCs.**

	PC description	Initial population size, individ.	SR, %	Best fitness achieved
1s	Intel Core2Quad, 2.4 Ghz, 4 GB	35,000	1.0	2410.8
2s	Intel Core2Quad, 2.4 Ghz, 4 GB	35,000	1.0	873.4
3s	Intel Core2Duo, 2.2 Ghz, 2 GB	25,000	1.0	985.6
4s	Intel Core2Quad, 2.4 Ghz, 4 GB	35,000	5.0	2303.5
5s	Intel Core2Duo, 2.2 Ghz, 2 GB	25,000	3.0	2389.5
6s	Intel Core2Quad, 2.4 Ghz, 4 GB	35,000	0.5	507.4

more actively and would converge in much quicker with a better result, in terms of component economy and functionality. The question that arises here on the effectiveness of VNFE toward such easy problems is the tradeoff between “standard evolution” and VNFE, that is, what is better: 7 independently made solutions to choose among or only 1 solution made in a team by 7? The last one is solvable to run tests for both approaches on relatively easy tasks.

The described approach has some features in common with *Niche GA* [28,29]. The similarities are the low selection rates, where a single individual is selected as a parent for the whole subpopulation and niches that in our case are represented by subpopulations. However, the differences that do not enable call the approach as *Niche ES* are much more numerous: *the niches* in our case interact each other via a migrant operator, they are not synchronized, there are no clearing and sharing procedures, etc.

Though the DM technique has been derived from the properties that are inherent particularly to analog circuits, it may be extended to other real-world instances that enable to mean under “mutation” term not just the *numerical* change of genotype fragments, but first of all, the change of *phenotypical* features of the instance.

The initial target of research described in this paper is the application aspect of how to succeed in evolution of such the complex tasks as TIMC. The VNFE has come to us and has been explored as an *ad hoc* problem. Therefore, as the research does not pretend to be deep enough from the EA theory point of view, future work is necessary on accumulating statistics.

## REFERENCES

- [1] Y. Sapargaliyev and T. Kalganova, “On Comparison of Constrained and Unconstrained Evolutions in Analogue Electronics on the Example of LC Low-Pass Filters,” *IEICE Transactions on Electronics*, Vol. E89-C, No. 12, 2006, pp. 1920-1927.
- [2] T. Kalganova, “Bidirectional Incremental Evolution in Evolvable Hardware,” *Proceedings of 2nd NASA/DoD Workshop Evolvable Hardware*, 13-15 July 2000, pp. 65-74.
- [3] Y. Sapargaliyev and T. Kalganova, “Challenging the Evolutionary Strategy to Synthesis Analogue Computational Circuits,” *Journal of Software Engineering and Applications*, Vol. 3, No. 11, November 2010, pp. 1032-1039.
- [4] E. Stomeo, T. Kalganova and C. Lambert, “Mutation Rate for Evolvable Hardware,” *International Conference on Computational Intelligence—ICCI 2005*, Prague, Czech Republic, 26-28 August 2005, pp. 117-124.
- [5] “Model 2145 Time to Amplitude Converter/Single Channel Analyzer,” Datasheet, *Canberra Industries*, Inc. <http://www.canberra.com/pdf/Products/Model-2145-SS-0226.pdf>
- [6] J. Shapiro, A. Prügel-Bennett and M. Rattray, “A Statistical Mechanical Formulation of the Dynamics of Genetic Algorithms,” *Lecture Notes in Computer Science*, Vol. 865, 1994, pp. 17-27.
- [7] Y. Sapargaliyev and T. Kalganova, “Automated Synthesis of 8-Output Voltage Distributor using Incremental Evolution,” *Proceedings of 2010 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE*, 15-18 June 2010, pp. 186-193.
- [8] F. Herrera and M. Lozano, “Adaptive Genetic Operators Based on Coevolution with Fuzzy Behaviors,” *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 2, 2001, pp. 149-165. [doi:10.1109/4235.918435](https://doi.org/10.1109/4235.918435)
- [9] J. R. Koza and D. Andre, “Evolution of Both the Architecture and the Sequence of Workperforming Steps of a Computer Program Using Genetic Programming with Architecture-Altering Operations,” In: P. Angeline and K. Kinneer, Eds., *Advances in Genetic Programming*, Vol. 2, 1996.
- [10] T. Bäck, “The Interaction of Mutation Rate, Selection, and Adaptation within Genetic Algorithm,” In: R. Männer and B. Manderick, Eds., *Parallel Problem Solving from Nature 2*, Elsevier, Amsterdam, The Netherlands, 1992, pp. 85-94.
- [11] J. Torresen, “A Divide-and-Conquer Approach to Evolvable Hardware,” In: M. Sipper, D. Mange and A. Pérez-Uribe, Eds., *ICES 1998*, LNCS, Springer, Heidelberg, Vol. 1478, 1998, pp. 57-65.
- [12] J. Wang, K. Je, Y. Lee and H. Chong, “Using Reconfigurable Architecture-Based Intrinsic Incremental Evolution to Evolve a Character Classification System,” In: Y. Hao, J. Liu, Y.-P. Wang, Y.-M. Cheung, H. Yin, L. Jiao, J. Ma and Y.-C. Jiao, Eds., *CIS 2005. LNCS (LNAI)*, Springer, Heidelberg, Vol. 3801, 2005, pp. 216-223.
- [13] J. Walker, K. Völck, S. Smith and J. F. Miller, “Parallel Evolution Using Multi-Chromosome Cartesian Genetic Programming,” *Genetic Programming and Evolvable Machines*, Vol. 10, No. 4, 2009, pp. 417-445. [doi:10.1007/s10710-009-9093-2](https://doi.org/10.1007/s10710-009-9093-2)
- [14] C. Mattiussi and D. Floreano, “Analog Genetic Encoding for the Evolution of Circuits and Networks,” *IEEE Transactions on Evolutionary Computation*, Vol. 11, 2007, pp. 596-607. [doi:10.1109/TEVC.2006.886801](https://doi.org/10.1109/TEVC.2006.886801)
- [15] W. Feng, L. Yuanxiang, L. Kangshun and L. Zhiyi, “A New Circuit Representation Method for Analog Circuit Design Automation,” *IEEE World Congress on Computational Intelligence*, IEEE Press, Hong Kong, 2008. [doi:10.1109/CEC.2008.4631059](https://doi.org/10.1109/CEC.2008.4631059)
- [16] PSpice A/D Reference Guide, (includes PSpice A/D, PSpice A/D Basics, and Pspice. Product Version 15.7), Cadence, July 2006.
- [17] J. Hu, E. D. Goodman, K. Seo and M. Pei, “Adaptive Hierarchical Fair Competition (AHFC) Model for Parallel Evolutionary Algorithms,” *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*,

- New York, 2002, pp. 772-779.
- [18] R. Lohmann, "Application of Evolution Strategy in Parallel Populations," In: H. P. Schwefel and R. Mäanner, *Parallel Problem Solving from Nature—Proceedings of 1st Workshop, (PPSN 1)*, Vol. 496 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1991, pp. 198-208.
  - [19] A. Ngom, "Parallel Evolution Strategy on Grids for the Protein Threading Problem," *Journal of Parallel and Distributed Computing*, Vol. 66, No. 12, 2006, pp.1489-1502. [doi:10.1016/j.jpdc.2006.08.005](https://doi.org/10.1016/j.jpdc.2006.08.005)
  - [20] L. Jostins, "A Comparison of Parallel Global Optimisation Algorithms for Reverse Engineering Gene Networks," MPhil Thesis, University of Cambridge, Cambridge, 21 August 2008.
  - [21] J. Lohn, W. Kraus and G. Haith, "Comparing a Coevolutionary Genetic Algorithm for Multiobjective Optimization," *Proceedings of the Evolutionary Computation, CEC'02. Proceedings of the 2002 Congress*, 12-17 May 2002, pp. 1157-1162.
  - [22] B. Liu, Y. Wang, Z. Yu, L. Liu, M. Li, Z. Wang, J. Lu and F. Fernandez, "Analog Circuit Optimization System Based on Hybrid Evolutionary Algorithms," *Integration, the VLSI Journal*, Vol. 42, No. 2, February 2009, pp. 137-148.
  - [23] K. Maneeratana, K. Boonlong and N. Chaiyaratana, "Multi-Objective Optimisation by Co-Operative Co-Evolution," In: X. Yao, *et al.*, Eds, *Parallel Problem Solving from Nature—PPSN VIII*, Lecture Notes in Computer Science, Vol. 3242, September 2004, Springer-Verlag, Birmingham, pp. 772-781. [doi:10.1007/978-3-540-30217-9\\_78](https://doi.org/10.1007/978-3-540-30217-9_78)
  - [24] R. Zebulum, M. Vellasco and M. Pacheco, "Variable Length Representation in Evolutionary Electronics," *Evolutionary Computation*, Vol. 8, No. 1, March 2000, pp. 93-120. [doi:10.1162/106365600568112](https://doi.org/10.1162/106365600568112)
  - [25] A. Thompson, "Artificial Evolution in the Physical World," In: Gomi, Ed., *Evolutionary Robotics*, AAI Books, 1997, pp. 101-125.
  - [26] I. Guerra-Gomez, E. Tlelo-Cuautle, Luis G. de la Fraga, T. McConaghy and G. Gielen, "Sizing Mixed-Mode Circuits by Multi-Objective Evolutionary Algorithms," *Proceedings of IEEE Midwest Symposium on Circuits and Systems (MWSCAS)*, Seattle, 2010, pp. 813-817.
  - [27] Feodosia State Optical Factory, "The Artillery Quantum Rangefinder (Online)," February 2011. <http://fkoz.feodosia.com.ua/main3.phtml?link=23>
  - [28] J. Hu, E. D. Goodman, K. Seo and M. Pei, "Adaptive Hierarchical Fair Competition (AHFC) Model for Parallel Evolutionary Algorithms," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, New York, 2002, pp. 772-779.
  - [29] A. Petrowski, "A Clearing Procedure as a Niching Method for Genetic Algorithms," *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1996, pp. 798-803.