16

Evolvable Metaheuristics on Circuit Design

Felipe Padilla[1,2], Aurora Torres[1], Julio Ponce[1],
María Dolores Torres[1], Sylvie Ratté[2] and Eunice Ponce-de-León[1]
[1]*Aguascalientes University,*
[2]*École de Technologie Supérieure*
[1]*México*
[2]*Canada*

1. Introduction

Evolutionary computation algorithms are stochastic optimization methods; they are conveniently presented using the metaphor of natural evolution: a randomly initialized population of individuals evolves following a simulation of the Darwinian principle. New individuals are generated using genetic operations such as mutation and crossover. The probability of survival of the newly generated solutions depends on their fitness (Michalewicz et al., 1995). Evolutionary algorithms (EAs) have been successfully used to solve different types of optimization problems (Back, 1996). In the most general terms, evolution can be described as a two-step iterative process, consisting of random variation followed by selection.
The structure of any evolutionary computation algorithm is shown in the figure 1.

```
procedure evolutionary algorithm
t ←0
initialize P(t)
evaluate P(t)
while (not termination-condition) do
begin
        t←t + 1
        select P(t) from P(t - 1)
        alter P(t)
        evaluate P(t)
end
```

Fig. 1. Structure of any evolutionary algorithm

The term evolutionary computation is used to describe techniques such as genetic algorithms, evolution strategies, evolutionary programming and genetic programming. The different approaches are distinguished by the genetic structures under adaption and the genetic operators that generate new candidate solutions (Cordon et al., 2001).
Evolvable hardware (EHW) is an exquisite combination of evolutionary computation and electronic hardware. While the most common techniques of evolutionary computation are

genetic algorithms and genetic programming, electronic hardware implies not only digital but analog circuits also. This field has earned importance since the early 1990´s because of the advent of reconfigurable hardware.

The ultimate objective of this field is to design and construct intelligent hardware, capable of online adaptation (Yao and Higuchi, 1999).

The first classification of evolvable hardware can be found in (De Garis, 1993). In this work De Garis established there are extrinsic and intrinsic EHW. While Extrinsic EHW simulates evolution by software and downloads to hardware only the best configuration; intrinsic EHW simulates evolution directly in hardware.

Nowadays the scope of this discipline has grown vastly. According to Zebulum (Zebulum, 1996), evolvable hardware can be classified by several criterion like hardware evaluation, evolvable computation approach, application area and evolvable platform. In regard to its application area EHW in divided in: Circuit design, robotics and control, pattern recognition, fault tolerance and very large scale integration (VLSI). We are interested in discuss about the first one.

Circuit design is the art of constructing a sized circuit from user specifications (Das and Vemuri, 2009). This task is divided according to the kind of circuits that are handled in digital and analog circuit design.

Nowadays there are different algorithms that can be used to solve problems of optimization of circuits like: Genetic Programming, Genetic Algorithm, Estimation of the Distribution Algorithms, Ant Colony Optimizations, Others.

The more amenable nature of digital circuits made researchers like Louis (Louis, 1993) and Koza (Koza, 1992) to focus first on the production of functional logic circuits. Afterwards, the goal was not only to obtain functional circuits, but optimum ones. The work of Louis (Louis, 1993) was pioneer on the use of genetic algorithms on the design of combinational circuits; Thompson et al (Thompson et al., 1996) were the first in coding logic gates and its connections. Other outstanding researches on digital design are Higuchi et al. (Higuchi et al., 1996) specially focused on intrinsic evolution based on neural networks; Hernández and Coello (Hernández and Coello, 2003) first worked with genetic algorithms and later with genetic programming and Information Theory. A very interesting case is the use of ACO on the optimization of combinatorial circuits (Mendoza, 2001).

The analog synthesis world also has numerous successful implementations of different metaheuristics like genetic algorithms (Lohn and Colombano, 1998), (Zebulum et al., 2000), (Goh and Li, 2001), (Das and Vemuri, 2007), (Khalifa et al., 2008), (Torres et al., 2010); genetic programming (Koza et al., 1997), (Hu et al., 2005)(Chang et al., 2006) and estimation of the distribution algorithms (Torres et al., 2009). Analog circuit synthesis is a process composed of two phases: the selection of a suitable topology and the sizing of all its components (Torres et al., 2010). While topology consists on the determination of the type of components and its connections; sizing refers to the selection of the components values. Further on this document, will be discuss some of the mentioned approaches.

Others types of evolutionary algorithms are based in biological systems in which complex collective behaviour emerges from the local interaction of simple components. Some examples of these algorithms are Swarm Intelligence, Ant Colony, Bees Algorithm, etc. We will speak of an ant colony, this algorithm is based in the foraging behaviour of some species of ants. Ant colonies are capable of finding the shortest paths between their nest and food sources, through a substance denominated pheromone.

2. Optimization algorithm

Actual trends in VLSI technology are towards integration of mixed analog-digital circuits as a complete system-on-a-chip. Most of the knowledge intensive and challenging design effort spent in such systems design is due to the analog building blocks (Balkir et al., 2004). Analog design has been traditionally a difficult discipline of integrated circuits (IC) design. In circuit design optimization, a circuit and its performance specifications are given and the goal is to automatically determine the device sizes in order to meet the given performance specifications while minimizing a cost function, such as a weighted sum of the active area or power dissipation (Baghini et al., 2007). This is a difficult and critical step for several reasons: 1) most analog circuits require a custom optimized design; 2) the design problem is typically under constrained with many degrees of freedom; and 3) it is common that many (often conflicting) performance requirements must to be taken into account, and tradeoffs must be made that satisfy the designer (Rutenbar et al., 2007).

Fuzzy techniques have been successfully applied in a variety of fields such as automatic control data classification, decision analysis, expert systems, computer vision, multi-criteria evaluation, genetic algorithms, ant colony systems, optimization, etc.

Works showing the possibility of application of fuzzy logic in computer aided design (CAD) of electronic circuits started to appear in late 1980s and early 1990s. An argument for fuzzy logic application in CAD is derived from the nature of the algorithm used for solving design problems. The majority of algorithms for synthesis use heuristics that are based on human knowledge acquired through experience and understanding of problems. Another important source of knowledge is numerical data. Fuzzy logic systems are appropriate in such situations because they are able to deal simultaneously with both types of information: linguistic and numerical.

Also, fuzzy systems being universal appoximators can model any nonlinear functions of arbitrary complexity. This is very useful in modelling complex circuit functions of high accuracy at low cost, necessary in performance evaluation.

Design optimization of an electronic circuit is a technique used to find the design parameter values (length and width of MOS transistors, bias current, capacitor values, etc.) in such a way that the final circuit performances (de gain, gain-bandwidth, slew rate, phase margin, etc.) meet as close as possible the design requirements.

There is no general design procedure independent of the circuit; also, there is no formal representation to connect the circuit functions on its structure in a consistent manner. The major obstacle consists in the peculiarity of the analog signals: the continuous domain of the signals` amplitude and their continuous time dependency. Hereby the analog circuit design is known like an iterative, multi-phase task that necessitates a large spectrum of knowledge and abilities of designers.

3. Genetic algorithms

Genetic algorithms originally were called "reproductive plans" by John Holland (Holland, 1975), and were the first emulators of the genetic evolution that produced practical results. In 1989, when Goldberg (Goldberg, 1989) published his book, mentioned more than 70 successful applications of this paradigm that continues winning popularity nowadays.

According to Coello (Coello, 1996), a good definition of genetic algorithm was established by Koza in his book of 1992 (Koza, 1992), he says the following: "The genetic algorithm is a

highly parallel mathematical algorithm that transforms a group (population) of individual mathematical objects (that usually have the form of chains of characters of fixed longitude), each one with an associate aptitude value, in new populations (for example the following generation) using modelling of operations under the Darwinian principle of the reproduction and survival of the "most capable", naturally, after the occurrence of the genetic operators (sexual recombination)".

Ponce de León (Ponce de León, 1997) summarizes the mechanism of operation of the simple genetic algorithm in the following way; "it is generated a population of n structures aleatorily (chains, chromosomes or individuals) and then, some operators act transforming the population. The transformation is carried out by means of the application of three operators; once this culminates, it is said that a generational cycle has finished". The three operators Ponce references are: selection, crossover and mutation.

The genetic algorithm in the form like Holland illustrates it (Holland, 1975) has the following characteristic elements:

1. Representation of binary chains.
2. Proportional selection.
3. Crossover like the main method to produce new individuals.

After the Holland's proposal, have been carried out different modifications; either by means of the use of different representation outlines, or until certain modifications to the selection operators, crossover, mutation and elitism.

The diagram shown in the following figure presents the simplest version in the genetic algorithm, well-known as SGA (for the initials in English of "Simple Genetic Algorithm").

Simple Genetic Algorithm

1. Let t=0 generations counter
2. Initialization P(t)
3. Evaluate P(t)
4. While stop criterion do not be reached, do:
   for i=1,…,N/2 do
      i. Select 2 individuals from P(t)
      ii. Apply crossover with probability p_c
      iii. Apply mutation to the offspring with probability p_m
      iv. Introduce the 2 new individuals to P(t+1)
   end-for

Fig. 2. Pseudocode of SGA

Although the general mechanism of this algorithm is extremely simple, it can be demonstrated by means of Markov's chains that the evolutionary algorithms that use elitist selection mechanisms, will converge to a good global solution of certain functions whose domain can be an arbitrary space (Torres, 2010). Günter Rudolph in 1996, generalized the previous developments in theory of the convergence for binary search spaces and Euclidian ones to general search spaces (Rudolph, 1996).

3.1 Genetic algorithms in automated analog design

Due to the high level of complexity that implies the task of designing and also to the strong dependence that this task has with the knowledge and experience experts; the automatic design of analogical circuits is a challenge and a necessity. Some researchers of the area believe that the automation of the design should be preceded by a change in the process of current design, for example, governed by the execution of the restrictions (Jerke, 2009). The fact is that nowadays, it has not still been possible to automate this process in a complete way.

One of the metaheuristics that have shown better benefits in the realization of this task are the genetic algorithm and the genetic programming; this space belongs to the genetic algorithm.

Lohn and Colombado (Lohn and Colombado, 1998) used the genetic algorithm to design two analog filters, one of low complexity and one of medium complexity. The contribution of these researchers resides in that they demonstrated that it was feasible to use a very simple lineal representation. They proposed a code outline in which each element was represented by a fixed number of bytes called bytecodes in which they included an operation code that dictated the connection of each element and three bytes more they used to code its value.

Koza on the other hand, continued making use of the genetic programming in the synthesis of computational circuits (Koza, 1997b) and controllers, filters and other kind of circuits (Koza, 1997).

According to Ricardo Zebulum and his collaborators (Zebulum et al., 1998), the Evolutionary Electronics is an area that seeks to find new techniques of automatic design based on Darwinian concepts. The authors of the mentioned work, made the comparison of three different methodologies in the design of electronic filters. Their work was put on approval with two cases of study: A low-pass filter discussed in (Koza, 1996) and a filter pass-band with band in passing between 2000 and 3000 Hz and the bands of rejection above 4000 and below 1000Hz. The methodologies on approval were the following:

"Outline of representation of variable longitude in combination with an evolutionary algorithm that restricts the topology of the filter (parallel meshes of two elements each one). For the simulation, an own tool was used in C, based on Laplace´s analysis.

"Outline of representation of fixed longitude in combination with an evolutionary algorithm that doesn't restrict the topology of the circuit. To analyse the circuits they used Smash and SPICE, obtaining the same results.

"Outline of representation of variable longitude in combination with an evolutionary algorithm that doesn't restrict the topology of the circuit. For the simulation of the circuits they used as much Smash as SPICE, obtaining the same results.

In this work, Zebulum and his collaborators demonstrated that making use of an evolutionary algorithm based on the "Genetic Algorithm of Adaptation of Species (SAGA) of Harvey (Harvey, 1993), they could be obtained results comparable with those obtained using genetic programming, as for the answer in frequency of the obtained circuits using much smaller populations. This work concludes settling down that as for time, the first methodology was better, however this can explain to you for the rigidity of the used topology that allowed the use of a tool of quicker simulation. In spite of the success of this work, all the methodologies had inducer circuits whose values were so big as a result (2.2H for example) that are not very practical. On the other hand, investigators as Grimbleby and their collaborators (Grimbleby et al., 1995) they were working with mechanisms of numeric

optimization in combination with genetic algorithms for the synthesis of analogical circuits using a chromosome of fixed length and a type of null component to fight with the variable size of the real circuits.

The XXI century has also been witness of numerous efforts made toward the automation of the synthesis of the analogical circuits, for example, in the year 2000, Zebulum et al. (Zebulum et al., 2000), established some advantages of variable length representation systems. Among other things, they argued that when using a fixed size, it is not only required expert knowledge of the problem, but the potential of the evolutionary algorithms is also limiting. That same year, they also proved an outline of representation of variable longitude that they understood passive elements, connected nodes and disconnected nodes. The authors emphasize the use of resistances and capacitors with programmable values in their architecture. These investigators intend to work the two phases of the evolution of an electric circuit (topology and adjustment of the parameters) in a sequential way, instead of making it simultaneously.

In the year 2001, the investigating Goh and Li (Goh and Li, 2001) they began to outline some of the weaknesses that persisted in the process of design of analogical circuits that they were commented later by investigators as Khalifa and their collaborators (Khalifa et al., 2008), (Das, 2008) among others.

The weaknesses that these investigators declare that they should be assisted, the reduction of the enormous computational effort that implies the evaluation of big generations of circuits

Year	Author	Application
1993	Horrocks and Spittle	Active low-pass filter
1994	Horrocks and Khalifa	Low-pass filter
1995	Grimbleby	High-pass filter
1996	Horrocks and Khalifa	Low-pass filter
1998	Lohn and Colombano	Low-pass filter
1998	Zebulum et al.	Low-pass filter Band- pass filter
1999	Krasnicki et al.	OP-AMP
2000	Ando and Iba	Passive filters
2000	Zebulum et al.	Passive filters
2001	Goh and Li	Low-pass filter High-pass filter
2007	Das and Vemuri	Low-pass filter
2008	Khalifa et al.	Low-pass filter High-pass filter
2008	Das and Vemuri	OP-AMP
2010	Torres et al.	Low-pass filter

Table 1. Relevant research on analog circuit synthesis using Genetic Algorithms (Torres, 2010).

that they don't always produce results and the reduction of the breach between the evolved circuits and those that finally are taken to the physical implementation, due to the restrictions of commercial physical devices. Other equally important aspects are related with the elaboration of tools that due to their complexity, they require expert personnel's manipulation or with a considerable level of knowledge (Krasnicki, 2001); as well as the execution in teams whose level of sophistication is outside of the reach of a great number of people.

4. Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDA's) constitute a relatively new field of the Evolutionary Computation (Larrañaga, 2002) that replaces genetic operators (crossover and mutation) for the estimation of the distribution of the selected individuals and the sampling from the distribution to obtain the new population.
The objective of this paradigm is to avoid the use of arbitrary operators as crossover and mutation, to modeling explicitly the most promising solutions for sampling solutions from its distribution.

Pseudocode of the algorithm EDA:

Step 1: Random generation of M individuals (initial Population)
Step 2: Repeat the steps 3-5 for the generation l=1, 2,… until an stop criterion is reached
Step 3: Select N <= M individuals from D_l-1 according to a selection method
Step 4: Estimate the distribution of probability $p_l(x)$ from the group of selected individuals
Step 5: Sample M individuals (new population) from $p_l(x)$

EDAs can be classified according to two fundamental approaches. The first is the level of interdependences of variables, and the second is the type of involved variables. With regard to the level of interdependences EDAs are divided in 3, when the variables are independent, when there are bivaluated dependences and when there are multiple dependences. With regard to the type of involved variables, they can be discrete, continuous or mixed.
The easiest version of an EDA is the "Univariate Marginal Distribution Algorithm" (UMDA) introduced by Mühlenbein (Mühlenbein and Paaβ, 1996). This algorithm works on the supposition of complete independence among variables. Pseudocode of this algorithm in presented in figure 3.

UMDA_AC

1. Begin
2. $D_0 \leftarrow$ Generate *M* individuals at random
3. Repeat for *l*= 1,2,... until the stopping criteria met
   a) $D_{l-1}^{Se} \leftarrow$ Select N ≤ M individuals from D_{l-1} according to the selection method
   b) $p_l(x) = p(x \mid D_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i) = \prod_{i=1}^{n} \frac{\sum_{j=1}^{N} \delta_j(X_i = xi \mid D_{l-1}^{Se})}{N} \leftarrow$ Estimate the joint probability distribution
   c) $D_{l-1}^{Se} \leftarrow$ Sample M individuals from $p_l(x)$

Fig. 3. Pseudocode for UMDA (Larrañaga, 2002).

Another very common approach for the estimation of the distribution supposing independence among the variables is the algorithm PBIL ("Population-based incremental learning") (Baluja, 1994) that contrary to UMDA, doesn't estimate a new model in each generation, but refines it.
The main problem of the distribution of the estimation algorithms, is to estimate the model; because as it gets more complicated, the dependences among the variables are captured in a better way, however, its estimation becomes more expensive (Larrañaga, 2002). Regarding models that consider bivariated dependences (dependences among pairs of variables), the most outstanding methods according its use in the literature are those that use chains like the "MIMIC" algorithm (Mutual Information Maximizing Input Clustering Algorithm) (De Bonet et al., 1996), those that use trees, as the case of the COMIT (Baluja and Davies, 1997) that uses the method of Chow and Liu [Chow 1968] based on the concept of mutual information and the BMDA (Pelikan, 1999), in which Pelikan and Mühlenbein propose a factoring of the distribution of joint probability. This algorithm is based on the construction of an acyclic directed graph of dependences that is not necessarily connected.
Finally, the most common n-varied models are those that allow estimating a model in a Bayesian-net form. This approach has originated a great variety of algorithms according to the learning method, according to the nature of the variables (discrete or continuous), according to the imposed restrictions, etc. (Larrañaga, 2002).
The great success genetic algorithms (GAs) have shown on several synthesis problems, has motivated some researches to explore the EDA´s world in analog circuit synthesis. Next table show some examples.

Year	Author	Application	Used metaheuristic
2002	Mühlenbein et al.	Low-pass	UMDA
2007	Zinchenko et al.	Mixed circuit	UMDA
2009	Torres et al.	Filters	UMDA
2010	Torres et al.	Filters	MITEDA

Table 2. Relevant works on analog circuit synthesis by means of Estimation of the Distribution Algorithms

From table 2 it can be seen UMDA is the most common approach implemented on the analog circuit synthesis, nevertheless, MITEDA represents an effort on exploring the behavior of more complex EDAS. This algorithm was developed inspired by the COMIT and it uses the concept of mutual information used by Baluja and Davies (Baluja, 1997) to build the tree of dependences. Later this tree is sampling in order to create new generations. This algorithm represents the first tool that considers bi-valuated dependencies used in the design of analogical circuits we know until this moment.

5. Ant Colony Optimization

The Ant Colony Optimization Algorithm is a meta-heuristic bio-inspired in the behavior of real ant colonies. The first algorithm which can be classified within this framework was presented in 1991 by Marco Dorigo. In his PHD thesis with Title: "Optimization, learning, and Natural Algorithms", modeling the way real ants solve problems using pheromones. Real ants are capable of finding the shortest path from a food source to their nest. The ants

deposit a concentration of pheromone in theirs paths, and they follows with more probability the way with more concentration of pheromone that it was previously deposited by other ants, the essential trait of ACO algorithms is the combination of a priori information about the structure of a promising solution with a posteriori information about the structure of previously obtained good solutions. In the Ant Colony Algorithms a number of artificial ants (agents) build solutions for an optimization problem and exchange information on their quality via a scheme of global communication that is reminiscent of the one adopted by real ants.

When exist paths without any amount of pheromone, the ants explore the neighbourhood area in a totally random way. In presence of an amount of pheromone, the ants follow a path with a probability based in the pheromone concentration. The ants deposit additional pheromone concentrations during his travels. Since the pheromone evaporates, the pheromone concentration in non-used paths tends to disappear slowly.

To find the shortest path, a moving ants lay some pheromone on the ground, so an ant encountering a previously trail can detect it and decide with high probability to follow it. As a result, the collective behavior that emerges is a form of a positive feedback loop where the probability with which each ant choose the next path increases with the number of ants that previously chose the same path.

The Ant Colony System (ACS) models the behavior of ants, which are able to find the shortest path from their nest to a food source. Although individual ants move in a quasi-random form, performing relatively simple tasks, the entire colony of ants can collectively accomplish sophisticated movement patterns. Ants accomplish this by depositing a substance called a pheromone as they move. This chemical trail can be detected by other ants, which are probabilistically more likely to follow a path rich in pheromone. This trail information can be utilized to adapt to sudden unexpected changes to the terrain, such as when an obstruction blocks a previously used part of the path.

5.1 Application of ant colony to the design of combinatory logic circuits

To apply Ant Colony Algorithm to the design of logic circuits, in (Mendoza, 2001) is shown as the design of logic circuits with ACO. In the case of the logic circuits, the treatment of the problem does not seem to be so immediate.

5.2 Circuit representation

The circuits are represent used a bidimensional matrix. Where each element of the matrix is a triplet of the type [Entrance 1, Entrance 2, Type of floodgate] (see figure 5). Was used five types of floodgates: AND, OR, NOT, XOR and WIRE, although this last one is not a floodgate, but rather it is a connection (a wire) that unites an element of certain column with another one of the previous column. Each element of the matrix receives its entrances solely of the exits of the previous column.

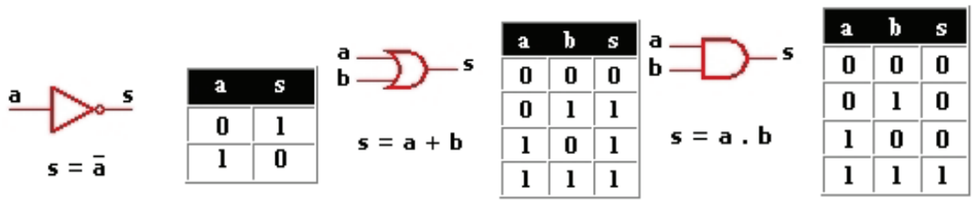


Fig. 4. Basic floodgate Not, Or, And

The first column directly receives its entrances of the table really of the given circuit. The last column provides the exits of the circuit. The first N rows corresponds to the N exits of the circuit. This form to represent a circuit has been used successfully.
In the following figure are shown the basic floodgate.

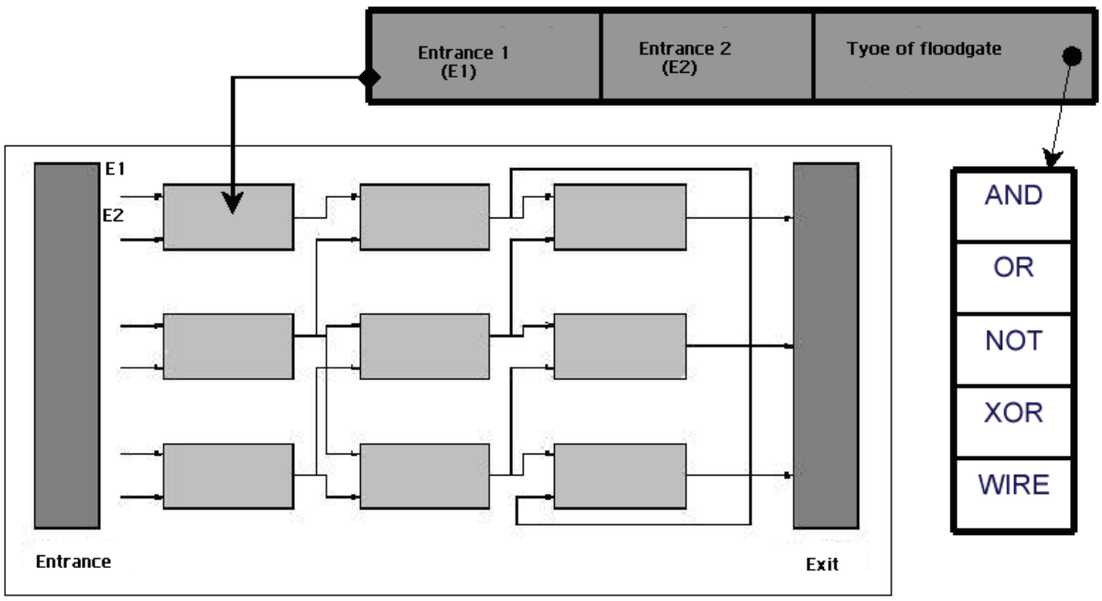


Fig. 5. Matrix used to circuit representation

5.3 Implementation

The route of an ant or agent will be a complete circuit. While each ant crosses a route, it constructs a circuit. In the TSP the ants find the route in terms of distance, do it here in terms of the number of floodgates.
A state or city is a column, which is made up of several elements to which it is called substates to them, being these each one of the floodgates of a column and the number of combinations of possible entrances of each floodgate of this column. The first N substates (N is the number of exits in the circuit) is chosen with a selection factor P, and the others are chosen randomly.
The distance between cities or states is measured as the increase or diminution from the successes to the exits of the circuit when changing from a level to another one.
Unlike the problem of the TSP, in a same route (circuit), they do not have to visit all the states.
The pheromones keep in a matrix called Trails. The length of this matrix corresponds to the number of exits of the circuit. Each element of Trails is a three-dimensional matrix as well. Next it is explained what they represent each one of the dimensions of the element. The first dimension of this matrix corresponds to the combination of possible entrances to the floodgate and goes from 0 to 6. The possible combinations of entrances, independent of the incoming number of the table really.
The second dimension corresponds to the number of floodgate, that is to say, goes of 0 to the number of floodgates except one (NumGates-1). The third dimension corresponds to the

number of successes that take until the level (column) previous and really goes of 0 to the number of lines in the table, because the number of successes that can be had in any level is between 0 and the number of lines of the true table.

5.3.1 The construction of a solution (route)

As it was already mentioned before, a state is a column of the matrix, each element of the column is a floodgate with its respective entrances and their exit. Because of that, the election of a state is a process that becomes by parts (floodgate by floodgate), reason why we will call to each floodgate (element of the column) a substate. A state a combination of three elements (floodgate, IN1, IN2). In order to choose a substate of anyone of the first N rows, a value is assigned to him to each one of the possible combinations, call selection factor P, with which it will compete remaining in that position.

The distance is a heuristic value and is given by the number of successes that the portion of the circuit constructed until the moment produces with respect to exit l of the True table. This is analogous to the distance in the TSP.

Once it has assigned a factor of selection to all the combinations, is chosen what of them remains in the position in game. This is repeated with all the substates that belong to one of the rows that represent an exit of the circuit. The other substates, are chosen randomly. This is repeated until arriving at the last state from the circuit or column of the matrix. When all the ants finish their route, the pheromone signs are updated. This becomes in two steps:

1. First the amount is due to update pheromone in the ways, simulating the pheromone evaporation of the ways by the artificial ants to the passage of time.
2. The ways are due to update or to increase according to the routes constructed by each ant in the algorithm. This becomes of the following form: If the circuit result of the route is not valid (that it does not produce all the exits).

6. Multiobjective optimization

A population based evolutionary multiobjective optimization approach (Coello, 2009) to design combinatorial circuit was proposed for first time by Coello and Hernández in 2000 (Coello and Hernández, 2000). This approach reduced the computational effort required by genetic algorithm to design circuit at gate level. The main motivation was the reduction of fitness function evaluations while keeping the capabilities of the GA to generate novel designs. The main ideas behind MGA algorithm are:

1. Circuit representation as a matrix (originally proposed by Louis in 1991 (Louis and Rawlins, 1991)) and an n-cardinality alphabet.
2. Incremental method to resized of matrix used to fit a circuit.
3. Fitness function in two stages. At the beginning only validity of the circuit outputs is taken into account, and at the ending the fitness function is modified such that any valid designs produced are rewarded for each WIRE gate that they include. (WIRE gate indicates a null operation, that is, the absence of gate)
4. Use a multi-objective optimization technique (Fonseca and Fleming, 1995) (Coello, 1999). In general, it redefines the single-objective optimization of as a multiobjective optimization problem in which we will have m + 1 objectives, where mm is the number of constraints. There is a new vector, $\bar{v} = (f, f_1, \ldots, f_n))$, where f is the objective function. $f_1, \ldots, f_n$ are the original constraints of the problem. An ideal solution X X

would thus have $f_i(X) = 0$ for $i = 1, \ldots, m$, and $f(X) \leq f(Y)$ for all feasible Y (assuming minimization). For combinatorial logic circuit design this technique consists on using a population based multiobjective optimization technique such as VEGA (Schaffer, 1984) to handle each of the outputs of the circuit as an objective. At each generation, the population is split in to $m + 1$ sub-populations, $m = 2^n$ (outputs), n: inputs of the circuit. The main mission of each sub-population is to match its corresponding output with the value indicated by the user in the truth table. After one of these objectives is satisfied, its corresponding sub-population is merged with the rest of the individuals in what becomes a joint effort to minimize the total amount of mismatches produced (between the encoded circuit and the truth table). Once a feasible individual is found, all individuals cooperate to minimize its number of gates (Coello and Hernández, 2002).

The MGA algorithm outperformance the GA algorithm in quality of solution and decreased the evaluation amount of fitness function. This approach made a path in solving evolutionary design of combinational logic circuits.

6.1 Formulation of multiobjective optimization problem

The multiobjective optimization problem can be formulated as follows (Coello and Hernández, 2000):

A General Multiobjective Optimization Problem ($\mathcal{MOP}$): Find the vector $\vec{x}^* = [x_1^*, \ldots, x_n^*]^T$ which will satisfy the m inequality constraints:

$$g_i(\vec{x}) \geq 0, i = 1, \ldots, m \tag{1}$$

the p equality constraints

$$h_i(\vec{x}) = 0, i = 1, \ldots, p \tag{2}$$

and optimizes the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{3}$$

where $\vec{x} = [x_1, \ldots, x_n]^T$ is the vector of decision variables.

That is, we wish to determine from among all $\vec{x} = [x_1, \ldots, x_n]^T$, which satisfy the inequality and equality constraints above, the particular $\vec{x}^* = [x_1^*, \ldots, x_n^*]^T$ which yields the optimum values of all the k objective functions of the problem. Let be Ω the set defined as all vectors $\vec{x} = [x_1, \ldots, x_n]^T$, that do not violate the constraints.

Pareto Optimality Definition: We say that $\vec{x}^* = [x_1^*, \ldots, x_n^*]^T \in \Omega, \Omega \subseteq \mathbb{R}^n, f_i: \mathbb{R}^n \to \mathbb{R}$, is *Pareto optimal* if for every $\vec{x} = [x_1, \ldots, x_n]^T$, and $I = \{1, \ldots, k\}$ either,

$$\bigwedge_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{4}$$

Or, there is at least one $i \in I$ such that

$$`f_i(\vec{x}) > f_i(\vec{x}^*) \tag{5}$$

$\vec{x}^* = [x_1^*, \ldots, x_n^*]^T$ is Pareto optimal if there exists no feasible vector $\vec{x} = [x_1, \ldots, x_n]^T$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion.

Pareto Dominance Definition: A vector $\vec{u} = (u_1, \ldots, u_n)$ is said to dominate $\vec{v} = (v_1, \ldots, v_n)$ (denoted by $\vec{u} \preccurlyeq \vec{v}$) if and only if $\vec{u}$ is partially less than $\vec{v}$, i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\}: u_i < v_i$.

Pareto Optimal Set Definition: For a given $\boldsymbol{\mathcal{MOP}}$, $\vec{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_k(\vec{x})]^T$, the *Pareto optimal set* ($\mathcal{P}^*$) is defined as:

$$\mathcal{P}^* = \{\vec{x} \in \Omega \,/\, \nexists \vec{x}' = [x_1', \ldots, x_n']^T \in \Omega(\vec{f}(\vec{x}') \preccurlyeq \vec{f}(\vec{x}))\} \tag{6}$$

Pareto Front Definition: For a given $\boldsymbol{\mathcal{MOP}}$, $\vec{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_k(\vec{x})]^T$, and Pareto optimal set $\mathcal{P}^*$, the *Pareto front* ($\mathcal{PF}^*$) is defined as:

$$\mathcal{PF}^* = \{\vec{u} = \vec{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_k(\vec{x})]^T \,/\, \vec{x} \in \mathcal{P}^*\} \tag{7}$$

7. Application

Due to the enormous success genetic algorithms has proved on the field of circuit design, this section has the purpose of show how this metaheuristic could be used for the synthesis of analog circuits.

In order to implement a genetic algorithm for the artificial evolution of any kind of process, is indispensable to find a way to represent a solution of the given problem, to find the way to generate possible solutions, to be able to evaluate the quality of the solutions and to have a group of operators that let transform one solution into another. Figure 6 shows the general flow used to implement a genetic algorithm in the analog circuit design according to Azizi (Azizi, 2001).

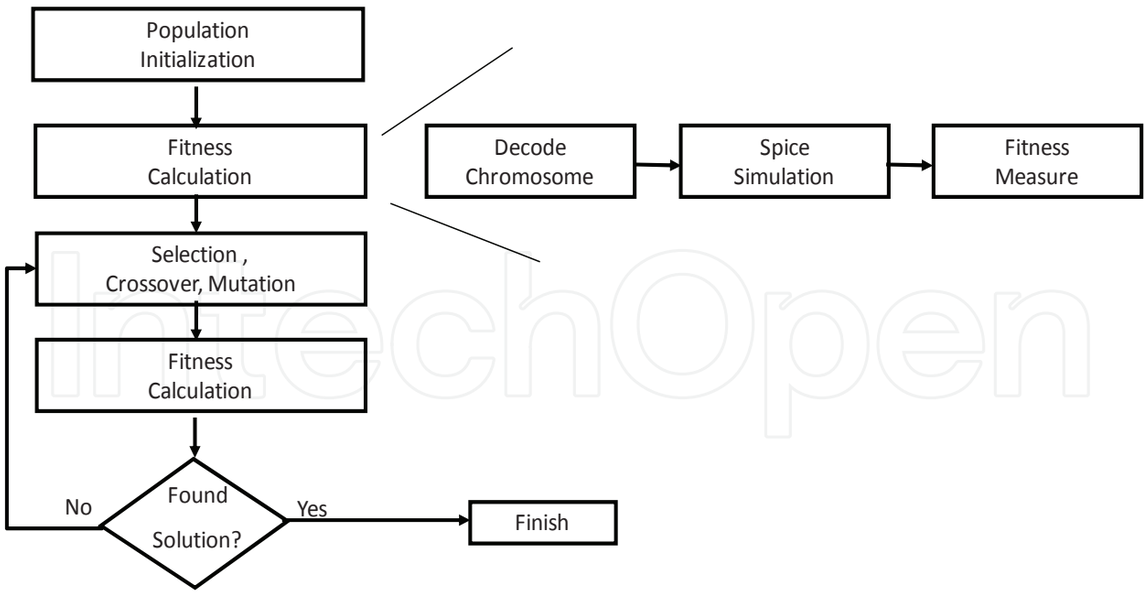


Fig. 6. Genetic Algorithm Flow for Analog Circuit Synthesis.

Representation mechanism

In order to initialize the first population, the programmer has to establish how each solution is going to be represented and how the population can be generated.

A genetic encoding for artificial evolution of analog networks must be capable of representing both; the topology and the sizing of the network (Mattiussi and Floreano, 2007). While topology refers to the way each element is going to be connected to each other; sizing refers to the type and dimension of each element on a net. Other important aspects of the representation mechanism are its ability to capture any kind of circuit and the chance to reduce the process and time inverted in translate the circuit into a netlist (net description list). The representation mechanism has also to be flexible enough to be used with a wide range of components values but sufficiently short to be computational handling. (Torres et al., 2009).
Torres et al (2009), reported a representation mechanism for passive elements of two terminals. This mechanism uses a gene of six parts to represent an analog element as figure 7 shows. Each circuit is a linked list of several genes.

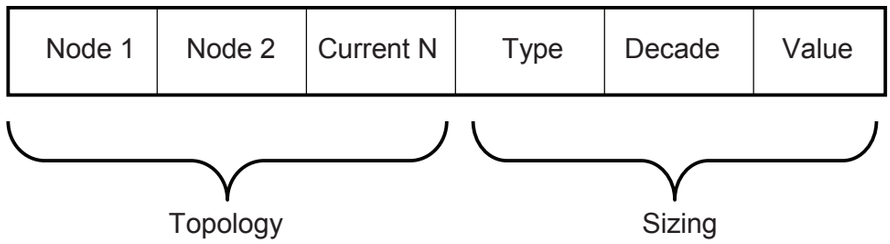


Fig. 7. Gene description.

While node 1 and node 2 refers to the terminals of an electrical device; current N is a pointer that is going to be used to build the network. Type, decade and value are the parameters that completely characterize a specific element (Torres, 2009). These parameters use integer coding according to table 3.

Type	Decade	Value
C(0)	10^{-6} – 10^{-9} (0-3)	E6 (0-5)
R(1)	10^{+3} – 10^{+6} (0-3)	E12 (0-11)
L(2)	10^{-3} – 10^{-6} (0-3)	E12 (0-11)

Table 3. Sizing encoding system

Next figure shows an element and its corresponding gene. We refers to initial node, that represents the beginning of an analog circuit.

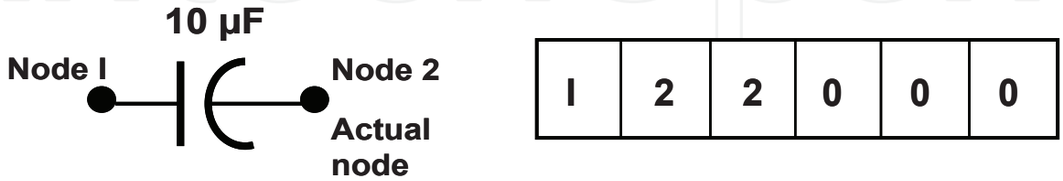


Fig. 8. An element of circuit and its corresponding gene.

Generation mechanism

Once, a representation mechanism has been selected, the generation routine need to be established. The generation mechanism proposed by Torres et al. (2009) is based on an

operation code randomly generated. The operation code establishes the connection that has to be done in the construction process of an admitted topology. The process begins in "Initial node" and ends when certain termination criterion is reached. This criterion could be one of two possibilities: the connection is done with the "Final Node" or the circuit reaches a preset amount of elements.
Next figure describes how the generation mechanism works (CNode refers to the current node, and INode corresponds to Initial Node) (Torres et al., 2009).

Generation mechanism

1. begin
2. CNode <- INode
3. while(Not meet termination criterion)
   - Node1 = Cnode
   - Generate OP-Code
   - Execute_connection (Update Node2 and Cnode)
   - Generate Type, Decade and Value
4. end_while
5. end

Fig. 9. Algorithm for the generation of each solution.

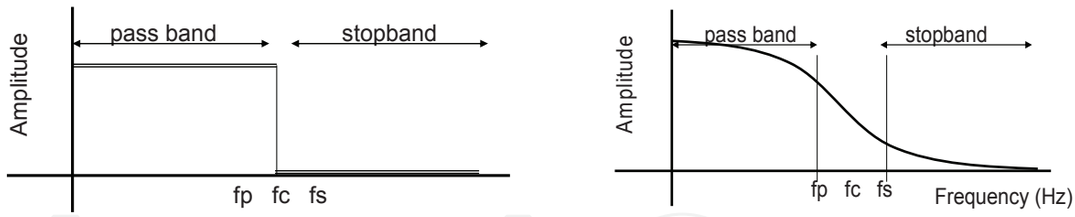
The circuit creation process performed by the former algorithm is very flexible. Once the operation code has been chosen and the connection has been done, type, decade and value of each element are generated. All operation codes used and their meaning are depicted in table 4.

Op code	Instruction
0	Connect to grown
1	Connect to final node
2	Connect to x node
3	Connect to new node

Table 4. The operation code of the generation mechanism (Torres et al. 2010)

Evaluation function

Evolvable process depends on the ability to distinguish good and bad solutions, because it consists in continuously improve solutions from one generation to another. Therefore, a fitness function that describes how close a circuit is from the target is needed.
Within the scope of analog circuit design, filters and amplifiers are the most frequently discussed. Fitness function used on the synthesis of low-pass filter will be presented below.
Filters are circuits that block certain frequencies or bands of frequencies (Curtis, 2003). A low pass filter is the one that let pass low frequencies while blocks high frequencies. Next figure, illustrates the frequency response of an ideal and a real filter.



Ideal low-pass filter. Real low-pass filter.

Fig. 10. Frequency response of an ideal and a real low-pass filter.

The fitness function used by Torres et al, is based on the measurement of the distance between the ideal and the real (evolved) filter. This function is an adaptation of the one used by Koza (Koza et al., 1997) and Hilder and Tyrrell (Hilder and Tyrrell, 2007) among other researchers. This function is the sum of errors between the ideal frequency response and the actual candidate, along N sampling points. Equation 8 describes the fitness measure for filters.

$$F = \frac{1}{1+\xi} \tag{8}$$

Where :

$$\xi = \sum_{i=1}^{N} \lambda(\varepsilon_i) * \varepsilon_i \tag{9}$$

$$\varepsilon_i = \left| M(f_i)_{Target} - M(f_i)_{Actual} \right| \tag{10}$$

"ξ" represents the error over the N points of frequency. If the deviation from target magnitude is inacceptable according to the frequency band, then a penalty factor "λ" has to be assigned to the error function.

A sample error function "ε" give us the absolute deviation between the actual output response and the target response over the "i" sampling point. $M(f_i)$Target denotes target magnitude at a f_i frequency, $M(f_i)$Actual is the magnitude of the actual evolved circuit at a f_i frequency and f_i is the sampling frequency.

Transformation of a solution

Finally, when representation, generation and evaluation of candidate solutions have been solved, the programmer needs to find a group of operators to transform one solution into another. Starting from two parents chosen by any selection routine, an offspring is produce through two possible operators: crossover and mutation.

There are several selection algorithms; one of the more popular is the roulette-wheel. Roulette-wheel selection is an operator used for selecting potentially useful solutions for recombination. The fitness level of each solution is used to associate a probability of selection. If fi is the fitness of individual i in the population, its probability of being selected is $pi = \frac{fi}{\sum_{j=1}^{n} fj}$, where n is the number of individuals in the population.

Crossover operation, introduces new solutions into the genetic algorithm starting from previous circuits; this operator is the responsible of changing some parts of a circuit by parts

from another one. According to Dastidar et al., (Dastidar et al.,2005) and Das and Vemuri (Das and Vemuri, 2007), the use of some suitable connectivity rules, can reduce the unwanted search space not only for active, but for passive circuit synthesis. The crossover operator proposed by Torres et al., generates topological modifications because it alters the connection order of the offspring. This operator can be applied to one or two crossover points.

Next figure shows how this operator can be executed on the condense chromosome of two progenitors, using the representation mechanisms proposed by Torres et al. In the figure "T" refers to ground connection and "F" represents the final node of the analog circuit. This condense representation of each solution only has connection nodes and type of each element.

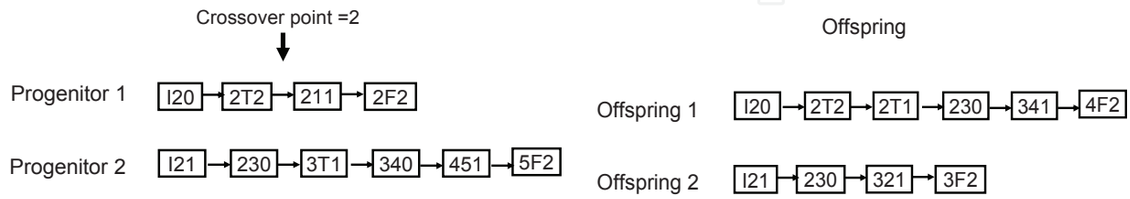


Fig. 11. Crossover operator (Torres et al., 2010)

Mutation is an operator that traditionally introduces new solutions modifying only one chromosome. There are several ways to implement mutation, Goh and Li (Goh and Li, 2001) show a nice group on operators. The mutation exhibit in this section was proposed by Torres et al. This mutation operator is executed at gene level; and it works by altering a randomly chosen gene with another randomly generated. A mutated gene corresponds to a different type of element with different value, but connected to the same pair of nodes (Torres et al., 2010). Next figure shows an example of the use of this operator.

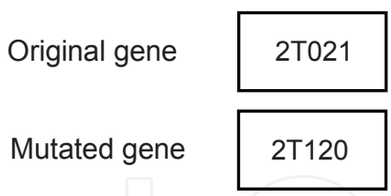


Fig. 12. Mutation operator.

Using all elements discuss in this section, the interested reader can implement an effective genetic algorithm for the automated synthesis of a passive filter.

8. Conclusion and the future research

Nowadays exist applications in real life problems, where is possible used evolvable metaheuristics based on populations to the circuit design process, in this chapter was present some algorithms and applications, like Genetic Algorithms, Estimation of the distribution algorithms, Ant colony optimization.

As shown there are multiple metaheuristics that can be used to circuit design trough different representations. We describe how is the representation with Genetic Algorithms.

Since avoiding non valid topologies and non simulable networks, implies a very high reduction on time and computational resources in our problem; mainly three algorithms

were compared at designing a low pass filter; a genetic algorithm (GA-AC), Ant Colony Systems (ACO-AC) and an estimation of distribution algorithm (UMDA-AC). Experimental results demonstrated that the group of mechanisms used in theses algorithms, worked better with GA-AC than with UMDA-AC and ACO-AC, according to the Pearson's Chi-squared tests with respect to the generation of low rate of non spice-simulable circuits.

Although UMDA-AC and ACO-AC performed faster the execution, and found a better individual on 200 generations' execution; statistically it cannot be said, the time difference is significant.

With respect to the number of fitness evaluations, it can be said with statistical base, that UMDA-AC performs less evaluations than GA-AC per execution. In order to improve the performance of this algorithms, next step is the creation of a tool that blends the strengths of each metaheuristic. The work team is already working on the design of some new operators to be inserted on the EDA-AC and ACO-AC.

GA-AC could be improved by enhancing the algorithm with some mechanisms of diversity control, like other kind of operators and another type of selection, in order to improve its exploration and delays its convergence.

As future work is to continue working with various tools and algorithms that allow us to improve new circuit design.

A new Artificial Intelligence that can be in charge of these systems, continues being distant into the horizon, in the same way that we still lack of methods to understand the original and peculiar things of each form to represent circuits.

9. References

Back, T.: (1996). Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York.

Baghini, M.; Kanphade, R.; Wakade, P.; Gawande, M.; Changani, M.; Patil, M. (2007). GP-based Design and Optimization of a Floating Voltage Source for Low-Power and Highly Tunable OTA Applications, WSEAS Transactions on Circuits and Systems, Issue 10, Volume 6, October 2007, pp. 588-582.

Balkir, S.; Dundar, G.; Alpaydin, G. (2004). Evolution Based Synthesis of Analog Integrated Circuits and Systems, IEEE NASA/DoD Conference on Evolution Hardware, EH`04. Pp 26-29.

Baluja, S. (1994). Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report TR CMU-CS 94-163, Carnegie Mellon University.

Baluja, S. and Davies, S. (1997) Combining Multiple Optimization Runs with Optimal Dependency Trees. Technical Report TR CMU-CS-97-157, Carnegie Mellon University.

Chang, S.; Hou, H. and Su, Y. (2006). Automated Passive Filter Synthesis Using an Novel Tree Representation and Genetic Programming. IEEE ransactions on Evolutionary Computation, Vol. 10. No.1, February 2006. Pp. 93-100.

Coello, C. (1996). An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design, PhD thesis, Department of Computer Science, Tulane University, New Orleans, Louisiana, USA..

Coello, C. A. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. Knowledge and Information Systems. An International Journal, 1(3):269–308, August 1999.

Coello, C. A. and Hernández, A. (2002) Design of combinational logic circuits through an evolutionary multi-objective optimization approach. Artificial Intelligence for Engineering, Design, Analysis and Manufacture, 16(1): 39-53.

Coello, C. A. Hernández, A. and Buckles, B. P. (2000) Evolutionary Multiobjective Design of Combinational Logic Circuits, eh, pp.161-172, The Second NASA/DoD Workshop on Evolvable Hardware (EH'00).

Coello, C. A., Lamont and, G. B., and Van Veldhuizen, D. A. (2007) Evolutionary Algorithms for Solving Multi-Objective Problems, Second Edition, Springer, New York, ISBN 978-0-387-33254-3.

Cordon, O.; Herrera, F.; Homann, F. and Magdalena, L. (2001). Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, World Scientic.

Das A. and Vemuri R. (2007). An Automated Passive Analog Circuit Synthesis Framework using Genetic Algorithms. IEEE Computer Society Annual Symposium on VLSI ISVL '07. pp. 145-152.

Das, A. (2008) Algorithms for Topology Synthesis of Analog Circuits. Doctoral thesis. University of Cincinnati. November.

Das, A. and Vemuri, R. (2009). A Graph Grammar Based Approach to Automated Multi-Objective Analog Circuit Design. Design, Automation and Test in Europe Conference and Exhibition 2009. IEEE Conferences., pp. 700-705.

De Bonet, J.; Isbell, C. and Viola, C. (1996) MIMIC: Finding Optima by Estimating Probability Densities. Proceeding of Neural Information Processing Systems. Pp. 424-430.

De Garis, H. (1993). Evolvable Hardware: Genetic Programming of a Darwin Machine, in Artificial Neural Nets and Genetic Algorithms, Albretch, R.F., Reeves, C.R., and Steele, N.C., Eds., Springer-Verlag, New York.

Dorigo, M. (1991). Positive Feedback as a Search Strategy. *Technical Report*. No. 91-016. Politecnico Di Milano, Italy.

Fonseca, C. M. and Fleming, P. J. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In S. Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

Goh, C. and . Li (2001). GA Automated Design and Synthesis of Analog Circuits with Practical Constraints. Proc. IEEE Int. Conf. Evol. Computation. pp. 170-177.

Goldberg, D. (1989) Genetic Algorithms in Search Optimization & Machine Learning. Addison-Wesley .

Grimbleby, J. (1995) Automatic Analogue Network Synthesis Using Genetic Algorithms, Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems (GALESIAS-95), pp.53-58, UK.

Harvey, I. (1993). The Artificial Evolution of Adaptive Behaviour, PhD Thesis, University of Sussex, School of Cognitive and Computing Sciences, September, 1993.

Hernandez A. and Coello C. (2003). Evolutionary Synthesis of Logic Circuits Using Information Theory. Artificial Intelligence Review 20: 445–471, Kluwer Academic Publishers. Printed in the Netherlands.

Higuchi, T.; Iwata, M.; Kajitani, I.; Murakawa, M.; Yoshizawa, S. and Furuya, T. (1996). Hardware Evolution at Gate and Function Levels. In Proceedings of the International Conference on Biologically Inspired Autonomous Systems: Computation, Cognition and Action, Durham, North Carolina.

Holland, J. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press. Ann Arbor, MI, 1975; MIT Press, Cambridge, MA 1992.

Hu, J.; Zhong, X. and Goodman, E. (2005). Open-ended Robust Design of Analog Filters Using Genetic Programming Proceedings of the 2005 conference on Genetic and evolutionary computation, June 25-29, Washington, DC, USA, pp. 1619-1626.

Jerke, G. and Lienig, J. (2009) "Constraint-driven Design — The Next Step Towards Analog Design Automation". Proceedings of the 2009 international symposium on Physical design. San Diego, California, USA. Pp. 75-82. 2009.

Khalifa, Y.; Khan, B. and Taha, F. (2008). Multi-objective Optimization Tool for A Free Structure Analog Circuits Design Using Genetic Algorithms and Incorporating Parasitics. Hindawi Publishing Corporation. Journal of Artificial Evolution and Applications. Volume 2008, PP. 0-9.

Koza, J. (1992) Genetic Programming. On the Programming of Computers by Means of Natural Selection. MIT Press. Cambridge, Massachussetts, 1992.

Koza, J.; Bennett, F.; Andre, D. and Keane, M.. (1996) Toward Evolution of Electronic Animals Using Genetic Programming. Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems. Cambridge, MA: The MIT Press.

Koza, J.; Bennethh, F.; Lohn, J.; Dunlap, F.; Keane M. and Andre D. (1997b) Automated synthesis of computacional circuits using genetic programming" in Proc. 1997 IEEE Conf. Evolutionary Computation. Piscataway, NJ: IEEE Press, pp. 447–452, 1997.

Koza, J.; Bennethh, F.; Andre, D. and Keane, M. (1997). Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. IEEE Transactions on Evolutionary Computation, Vol 1, No.2, pp. 109-128.

Krasnicki, M.; Phelps, R.; Hellums, J.; McClung, M.; Rutenbar, R. and Carley, L. (2001). ASF: a practical simulation based methodology for the synthesis of custom analog circuits, In Proceedings of ICCAD 2001, pp. 350–357.

Larrañaga P. and Lozano, J. (2002) Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation". Kluwer Academic Publishers.

Lohn, J. and Colombano, S. (1998). Automated Analog Circuit Synthesis using a Linear Representation. Proc. of the Second Int'l Conf on Evolvable Systems: From Biology to Hardware, Springer-Verlag, Berlin, pp. 125-133.

Louis, S. (1993). Genetic Algorithms as a Computational Tool for Design. PhD Thesis, Department of Computer Science, Indiana University.

Louis, S. J. and Rawlins, G. J. E. (1991) Using Genetic Algorithm to Design Structures. Technical Report 326. Computer Science Department, Indiana University, Bloomington, Indiana.

Mendoza, B. (2001). Uso de del Sistema de la Colonia de Hormigas para Optimizar Circuitos Lógicos Combinatorios. Tesis de Maestría en Inteligencia Artificial de la Universidad Veracruzana. México.

Michalewicz, Z.; Dasgupta, D.; Le Riche, R. and Schoenauer M. (1995). Evolutionary Algorithms for Constrained Engineering Problems.

Muhlenbein, H. and Paaβ G. (1996). From Recombination of Genes to the Estimation of Distributions I. Binary Parameters, in H.M.Voigt, et al., eds., Lecture Notes in Computer Science 1411: Parallel Problem Solvingfrom Nature - PPSN IV, pp. 178-187.

Pelikan M. and Mühlenbein, H. (1999). The Bivariate Marginal Distribution Algorithm. Advances in Soft Computing-Engineering Design and Manufacturing. Pp. 521-535.

Ponce de León, E., (1997) Algoritmos Genéticos y su Aplicación a Problemas de Secuenciación". PhD. Tesis. Centro de Inteligencia Artificial. Instituto de Cibernética, Matemática y Física.

Rudolph, G. (1996) Convergence of Evolutionary Algorithms in General Search Spaces, In Proceedings of the Third IEEE Conference on Evolutionary Computation .

Rutenbar, R.; Gielen, G.; Roychowdhury, J. (2007). Hierarchical Modeling, Optmization, and Synthesis for System-level Anlog and RF Designs, Proc. of the IEEE, Vol. 95, Issue 3, March 2007, pp. 640-669.

Schaffer, J. D. (1984) Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. PhD thesis, Vanderbilt University.

Shragowitz, E.; Lee,J.; Kang, Q. (1998). Application of Fuzzy Logic in Computer-Aided VLSI Design, IEEE Trans. on Fuzzy Systems, Vol. 6, No 1, February 1998, pp. 163-172.

Thompson, A.; Harvey, I. and Husbands, P. (1996). Unconstrained evolution and hard consequences. In E. Sanchez and M. Tomassini, editors, Toward Evolvable Hardware: The Evolutionary Engineering Approach (Lecture Notes in Computer Science, Vol. 1062), pages 136--165, Heidelberg, Germany, Springer-Verlag.

Torres, A. (2010). Metaheurísticas Evolutivas en el Diseño de Circuitos Analógicos. Tesis Doctoral. Universidad Autónoma de Aguascalientes.

Torres, A.; Ponce de León, E.; Hernández, A.; Torres, M.D. and Díaz, E. (2010). A Robust Evolvable System for the Synthesis of Analog Circuits. Computación y Sistemas, Revista Iberoamericana de Computación. April-June, Vol. 13, No.4, pp. 295-312.

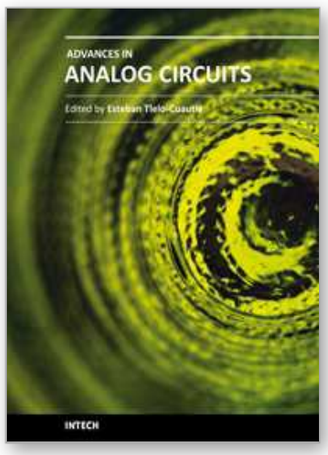
Torres, A.; Ponce de León, E.; Torres, M. D.; Díaz E. and Padilla, F. (2009). "Comparison of Two Evolvable Systems in the Automated Analog Circuit Synthesis". Artificial Intelligence, MICAI 2009. Eighth Mexican International Conference on, vol.1, pp 3-8, 8-13.

Yao, X. and Higuchi, T. (1999) "Promises and Challenges of Evolvable Hardware", IEEE Transactions on Systems, Man and Cybernetics_Part C. Applications and Reviews, Vol 29, No.1.

Zebulum, R.; Pacheco M. and Vellasco M. (1996). Evolvable Systems in Hardware Design: Taxonomy, Survey and Applications, Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware (ICES'96), Lecture Notes in Computer Science 1259, pp. 344-358, Tsukuba, Japan, October 7-8.

Zebulum, R.; Pacheco M. and Vellasco, M. (1998). Comparison of different evolutionary methodologies applied to electronic filter design. In Proc. Of IEEE. Inttl. Conf. On Evolutionary Computation. May.

Zebulum, R.; Vellasco, M. and Pacheco, M. (2000) Variable length representation in evolutionary electronics. Evol. Comput., vol. 8, no. 1, pp. 93–120.



Advances in Analog Circuits

Edited by Prof. Esteban Tlelo-Cuautle

ISBN 978-953-307-323-1
Hard cover, 368 pages
Publisher InTech
Published online 02, February, 2011
Published in print edition February, 2011

This book highlights key design issues and challenges to guarantee the development of successful applications of analog circuits. Researchers around the world share acquired experience and insights to develop advances in analog circuit design, modeling and simulation. The key contributions of the sixteen chapters focus on recent advances in analog circuits to accomplish academic or industrial target specifications.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Felipe Padilla, Aurora Torres, Julio Ponce, María Dolores Torres, Sylvie Ratté and Eunice Ponce-de-León (2011). Evolvable Metaheuristics on Circuit Design, Advances in Analog Circuits, Prof. Esteban Tlelo-Cuautle (Ed.), ISBN: 978-953-307-323-1, InTech, Available from: http://www.intechopen.com/books/advances-in-analog-circuits/evolvable-metaheuristics-on-circuit-design

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen